Semantically Enhanced, Multifaceted, Collaborative Access to Cultural Heritage

Project IST-034984

D2.2 Semantic Annotation

Deliverable

Editor:	ORT
Contributors:	ORT, XLab, Idea, CyberCultus, Technion
Dissemination Level:	Public
Version:	2.0
Date:	15-Jun-08

Executive Summary

Document Objective

This deliverable presents a conceptual framework, the ontologies and a variety of software tools and solutions adopted, adapted and developed in order to enable semantic annotation using OWL-formatted ontologies of Internet content resources, as well as semantic search and retrieval using these annotations.

Targeted Audience

MOSAICA technological partners

Prerequisite

IST-034984 MOSAICA Technical Annex
D5.1 MOSAICA System & Application Specifications
D2.1 Conceptual Alignment of Multiple, Distributed Content Source

Summary

The emergence of the Web 2.0 demonstrated the power of collaborative tagging. Semantic annotation using description logics enables application of machine reasoning to the information retrieval – this is the advantage of ontologies over the so-called *folksonomies*. By empowering semantic annotation with the level of usability encountered today in the Web 2.0, MOSAICA took the challenge of demonstrating that social tagging could be also performed using ontologies.

Several difficulties are encountered in application of ontologies to annotation: by blurring the distinction between classes and instances decidability in logical reasoning is put at risk, the size of the knowledgebase required for annotation makes its use for navigation problematic, there is no clear vision of a semantic search and retrieval product that could guide the future design, etc. MOSAICA provided solutions to these problems – partly by building on the existing research outcomes, and partly by introducing innovations.

Special attention was paid to the usability of both semantic annotator and semantic search and retrieval application. In this deliverable we report in detail the design of the semantic annotator, and provide a glimpse into the semantic search and retrieval solution. This focus on the usability is necessitated by the drive for commercial exploitation of the project's outcomes. Web 2.0 set very high standards for the usability. If there is a market for the semantic technologies, there usability must match the existing standards.

© 2008 MOSAICA

This document contains proprietary information of MOSAICA and shall not be reproduced or transferred to other documents, disclosed to others, or used for any purpose other than that for which it is furnished, without the prior written consent of MOSAICA.

The trademark and service marks of MOSAICA, including the MOSAICA mark and logo, are the exclusive property of MOSAICA, and may not be used without permission. All other marks mentioned in this material are the property of their respective owners

Contents

1. Introduction 5
1.1. Why Ontologies? 5
1.2. Why OWL? 6
1.3. The Price Tag of the Design Decisions 6
1.4. Structure of This Document 9

2. Ontology Design for Annotation 10
2.1. Design Principles 10
2.1.1. Requirements 10
2.1.2. Description Logic 13
2.1.3. Web 2.0 18
2.1.4. DAAT Framework 20
2.2. Cultural Heritage Ontology 37
2.2.1. Evaluation of the Multicultural Character ..... **Erreur ! Signet non défini.**
2.2.2. Jewish Cultural Heritage Ontology ..... **Erreur ! Signet non défini.**
2.3. Specific Ontologies 38
2.3.1. GIS/TIS 38
2.3.2. Multimedia 50
2.4. Deployment 51
2.4.1. DAAT Framework 51
2.4.2. MOSAICA Deployment 52

3. Semantic Annotator 54
3.1. Major Constituents 55
3.1.1. Metadata 55
3.1.2. Ranking 56
3.1.3. Tags 56
3.1.4. Relationships 57
3.2. Map Annotation 57
3.3. Multimedia Annotation 58

4. Appendix A: DAAT Core Ontology Specification 59
4.1. Concepts 59
4.2. Properties and Relations 59

5. Appendix B: DAAT Annotation Ontology Specification 61
5.1. Concepts 61
5.2. Properties and Relations 62

6. Appendix C: GIS/TIS Ontology Specification 70
6.1. XML Schema 70
6.2. Concepts 72
6.3. Properties and Relations 73

7. Appendix D: OWL DL Reasoning Capabilities 78

8. References 81

1. INTRODUCTION

A massive body of literature concerning the use of ontologies for annotation was published over the past 15 years. Nevertheless, very little attention was paid to the fundamental question: **Are ontologies suitable for annotation at all?** – Somehow, an affirmative answer is always automatically assumed.
As demonstrated in [Ide et al. 1997] and [Welty 1998], the use of Description Logics for annotation of resources raises more problems than what it may seem at the first glance. In this introductory chapter, we shall present these problems, as well as outline our strategy for handling them. However, prior to the discussion of these problems, it is necessary to provide a clear argument why the use of ontologies for annotation should be desired, and why ontologies should be preferred over vocabularies, classification schemas, thesauri, taxonomies, "folksonomies", etc.

1.1. Why Ontologies?

Typical criticism of the so-called *social tagging* (or *collaborative tagging*) is that "folksonomies" – in contrast to the ontologies – cannot be processed by machines ([Gruber 2006], [Bateman et al. 2006]). This is, however, only a partial argument, since it leaves unclear what machines can actually do, in order to facilitate a use of annotations that goes beyond a mere keyword search. Actually, the term "ontology" may be hereby deceptive, since a decisive argument regarding the need for machine-processability can be provided only for the ontologies that are Description Logics: **Description Logics enable the application of machine reasoning** [Welty 1998]. – Machine reasoning is something that "folksonomies", vocabularies, classification schemas, thesauri, taxonomies, etc. simply do not support.
Machine reasoning is indeed a feature that could significantly improve the existing search and retrieval methodologies ([Bontcheva et al. 2006]) – in both of their evaluation aspects:

1. **Precision** – semantic reasoning can ensure that only resources matching the intentions of the query will be retrieved. For example[1], consider the following query:
   "*Display: email address of John Smith, director of a telecom company in Europe.*"
   Submitted to a regular search engine this query would retrieve many documents, including those containing email addresses of other people who are not John Smith, as well as of people called John Smith, but who are not directors of a European telecom, i.e. who are not "the" John Smith of interest, etc. – It is sufficient that query keywords occur somewhere in the document text, in order to retrieve that document in response to a query. Machine reasoning, on the other hand, can ensure that only those documents in which the requested email is provided are retrieved.
2. **Recall** – machine reasoning can provide even more impressive results for recall. Consider for example a document including the following sentence:
   "At its meeting on the 10th of May, the board of London-based O2 appointed John Smith as CTO."
   In order to retrieve this document in response to the query above, a search engine would need to know the following:
   "O2 is a mobile operator, which is a kind of telecom."
   "London is located in the UK, which is a part of Europe."
   "A CTO is a kind of director."

[1] This and the following example were adopted (and slightly adapted) from [Bontcheva et al. 2006].

This kind of knowledge is exactly what Description Logics can encode in such way that a machine can infer that the document containing the sentence above is relevant, and should be included in the search results. What cannot be resolved at the terminological level, machine reasoning can resolve at the semantic level.

1.2. Why OWL?

Web Ontology Language (OWL) was selected due to the fact that it is W3C standard. This implies:

1. OWL is widely used for conceptual modeling, which ensures interoperability and reusability. OWL is an extension of the Resource Description Framework (RDF), which is simpler, and also widely used in the ontology engineering. However, while OWL parsers can also operate on RDF/RDFS-formatted ontologies, the other way around is not possible. Also, RDF/RDFS support only a fraction of the machine reasoning capabilities of OWL (see: Appendix D: OWL DL Reasoning Capabilities).
2. OWL is XML-formatted. In comparison to other Description Logics, parsing and handling of OWL is simple and efficient, and allows use of the existing XML parsers and XML document handlers.
3. There are many good reasoners available for OWL. Reasoning procedures are well understood and reasoning algorithms are fully developed ([Horrocks et al. 2007]). Actually, development of OWL Lite and OWL DL was driven by the early development of reasoners[2], such as FaCT[3], Pellet[4] and Racer[5].

OWL is a family of three well defined (and maybe few more undefined) formalisms. The difference between these formalisms is the amount of restrictions imposed on each. OWL Full is hereby the least restricted language, while OWL Lite is the most restricted. In terms of the Description Logics, OWL Lite is equivalent to ***SHIF(D)***, and OWL DL to ***SHOIN(D)***. OWL Full seems to be undecidable ([Horrocks et al. 2007][6]), at least for the current state of the art reasoners, and therefore cannot be identified with any known Description Logic. Nevertheless, OWL Full enables the most complete integration with the RDF/RDFS, which was a desirable feature, since at the time of OWL standardization efforts, significant amounts of semantic resources and applications using RDF/RDFS were already available.
OWL DL is a very expressive Description Logic, and enables formulation of logically complex definitions. Prima faciae, this seems like a significant advantage, since it provides the strongest reasoning capabilities. Nevertheless, as we shall argue in the next chapter, it also causes serious problems in the context of semantic annotation. OWL Lite is a more restricted version of the OWL DL, and the commitment to OWL DL implies the fallback to the simpler OWL Lite.

1.3. The Price Tag of the Design Decisions

The design decision to adopt OWL DL as the formalism for expressing ontologies comes with a price tag: as first discussed in [Welty 1998], Description Logics are not designed for the annotation of resources, and their application in that area opens grave problems that cannot be always overcome, at least not elegantly. We shall argue here that OWL Full would be (conceptually) more suitable to

[2] http://www.w3.org/TR/owl-ref/#OWLDL
[3] http://www.cs.man.ac.uk/~horrocks/FaCT/
[4] http://pellet.owldl.com/
[5] http://www.racer-systems.com/
[6] The undecidability of OWL Full was pointed by Ian Horrock during early discussions leading to the standardization of OWL, and then repeated in several papers. However, we are providing here only the reference to last paper in which the comparison between various OWL flavors was presented in the most extensive and clear manner.

semantic annotation (due to its closeness to RDF), but due to its undecidability, it does not provide the major advantage required from the ontologies: it does not support well machine reasoning. On the other hand, OWL DL (and OWL Lite) effectively support machine reasoning, but open conceptual problems associated with the application of Description Logics to annotation.

First, we shall present the argument from [Welty 1998], but in a slightly modernized terminology[7]. A resource can be conceptualized as an individual concept – for example, as an instance of a class *resource*. This is a very intuitive approach, since we expect resources to be real-world entities (= electronic files) identified by their URIs. An annotation of a resource would be, accordingly, expressed as an assertion in which the reference to a specific resource will serve as subject. In the case of a URI, for example, such assertion would use a data-type property as predicate, and a datum of the type string (or xsd:anyURI) as its object. – So far, everything seems right.

However, the problem occurs when assertions using object properties need to be formulated. By definition, assertions can be formulated only by using individual concepts as their subjects and (for assertions using object properties also as their) objects. This implies that **only** individual concepts could be used for semantic annotation. A domain ontology that would contain these individual concepts should be expected to be similar to a hierarchical vocabulary of the so-called *subjects* or *keywords* used for annotation and classification by the librarians[8]. The major difference is that in ontology the items are classes organized into a hierarchical structure by applying the subsumption mechanism. – And this is where the problem lies: individual concepts are instances of classes, and cannot be organized into a hierarchical structure based on subsumption[9]. Accordingly, only two (basic) design choices are available:

1. Either the hierarchical structure of the domain ontology must replaced by a flat list of individual concepts,
2. Or both classes and instances (somehow) need to be used for annotation.

Both of these solutions apparently eliminate the application of the machine reasoning from the annotation mechanism:

1. A flat list of individual concepts eliminates reasoning by subsumption. Actually, such a list cannot be even viewed any more as ontology – at least not as a Description Logic ontology.
2. Allowing assertions to be formulated using classes eradicates the distinction between classes and instances, which raises the problem of decidability. This is the so-called *meta-modeling*[10] that was forbidden in OWL DL because of the computational difficulties that it may cause. Accordingly, if such assertions are introduced in OWL, they transform OWL DL (or OWL Lite) into OWL Full, for which no complete reasoner is available. In addition, it seems like a serious logical inconsistency if assertions would be sometimes formulated using classes and sometimes individual concepts.

Regarding the latter solution, few additional remarks must be made here:

1. It is highly desirable to use also classes in annotations. For example, an image depicting a Hanukiah (8-branch candelabrum) could be annotated using an individual concept referencing the specific Hanukiah in the image. At the first glance, this seems logical, since Hanukiah is a physical entity that can be referenced by an individual concept (for example, an instance of the class *Hanukiah*). This is also appropriate for some prominent entities of that kind, such as the one depicted in Figure 1.

---

[7] Chris Welty presents his argument based on the experience of applying the CLASSIC framework (developed at MIT and described in [Brachman et al. 1989]) in the Text Encoding Initiative (TEI) described in [Ide et al. 1997]. We shall present his argument in the today accepted OWL terminology.

[8] Such as, for example, Dewey Decimal Classification (http://www.oclc.org/dewey/).

[9] One individual concept cannot be an instance of another individual concept.

[10] Meta-modeling is based on the principle that the same object can be seen as a class, an individual or a property, depending on the point of view.

Figure 1 Bronze Pal-Bell oil-burning Hanukkah menorah from Israel circa 1948, by Maurice Ascalon

However, it may be difficult to reference every single entity of Hanukiah (in particular if they are, for example, mass-produced with no distinguishable differences). In such case, it may be easier to simply say that the annotated image is "an image of Hanukiah", without specifying reference to the specific entity in the image. To that purpose the use of the class *Hanukiah* in the annotation would be more appropriate.

2. The example above is clearly meta-logical: "the main problem is to stick the desired information into an ontology; the applications then use this information in some programmatic way."[11] Accordingly, there is no expectation that the annotation assertion formulated using a class as its value would actually modify anything in respect to the logical reasoning about that class. If we abstract from these specific assertions (i.e. assertions which introduce OWL Full by violating the OWL DL restrictions) – the rest of the ontology remains a "legal" Description Logic.
3. Many reasoners are not restricted to OWL DL and OWL Lite, and operate efficiently[12] on OWL Full[13]. The extension of logical reasoning beyond OWL DL is certainly motivated by the well-known shortcomings of the OWL DL ([de Bruijn et al. 2005]), among which the lack of the meta-modeling plays a prominent role. The frequent demands on behalf of the RDF/OWL users led to the introduction of a basic form of meta-modeling, called punning into OWL 1.1 ([Grau et al. 2006]). Punning[14] is the meta-modeling practice of using a single name to refer to any or all of an individual, a class, or a property. Complications of this practice are managed by restricting its use such that, for a given use of a name, context sufficiently disambiguates the entities to which it may refer[15].

Considering all these pros and contras, the decision was taken to adopt the OWL Full in MOSAICA, and thus allow the use of classes as values in the annotation assertions. The detailed analysis of the reasoning capabilities required in MOSAICA (detailed in 2.1.2 Description Logic) revealed: 1) that

[11] http://www.w3.org/2007/OWL/wiki/PropertyPunning
[12] http://harry.hchen1.com/2005/12/22/206
[13] For example: http://www.w3.org/2001/sw/WebOnt/impls, although since December 2003 many reasoners have further evolved. – For example: http://pellet.owldl.org/faq/owl-full/.
[14] The term originates from "pun (or paronomasia) […] a phrase that deliberately exploits confusion between similar-sounding words for humorous or rhetorical effect" (http://en.wikipedia.org/wiki/Pun).
[15] http://webont.org/owl/1.1/overview.html#2.4

none of the required OWL DL reasoning capabilities seem to be affected by the introduction of punning (our use of punning was clearly meta-logical as pointed out previously), and 2) that some of the required reasoning capabilities in any case cannot be provided by a description logic, and must be obtained through proprietary mechanisms. Accordingly, the adoption of OWL Full seemed as a reasonable decision that will not harm the reasoning capabilities of MOSAICA, and even allow utilization of available logical reasoners, given that they support punning (OWL 1.1) or OWL Full.

1.4. Structure of This Document

In the section 2 Ontology Design for Annotation we shall present the system of MOSAICA ontologies that were developed for semantic annotation. First, we shall detail the design principles for development of the MOSAICA domain ontologies used for semantic annotation in 2.1 Design Principles. The focus will be hereby on the strategy that allowed the application of logical reasoning to annotation. However, in addition to providing the machine reasoning capabilities, ontologies must also provide display capabilities enabling efficient navigation. The complexity of the display-related requirements necessitated development of a generic framework for display of resources called *DAAT* that will be detailed in 2.1.4 DAAT Framework.
Next, we shall present the developed domain ontologies in 2.2 Cultural Heritage Ontology. In 2.3 Specific Ontologies we shall present two specific ontologies developed for particular functionalities required in MOSAICA: MOSAICA GIS/TIS Ontology and MOSAICA Multimedia Ontology. Finally, we shall discuss the way we deployed the MOSAICA system of ontologies in 2.4 Deployment.
In the section 3 Semantic Annotator we shall provide an outlook of the MOSAICA Semantic Annotator and detail the practical solutions that it provides.

2. ONTOLOGY DESIGN FOR ANNOTATION

2.1. Design Principles

2.1.1. Requirements

The following is not a presentation of the formal MOSAICA knowledgebase requirements specifications, since such specifications were never produced. Instead, this is a rather holistic description of the issues and difficulties that were discovered during the ontology engineering[16] and that were solved in various ways, as reported in the rest of this deliverable.

2.1.1.1. Logical Reasoning

The eradication of the distinction between classes and individual concepts seems in principle to be a desirable property of annotation vocabularies in general: an annotator (person) does not need to be aware of the logical definitions required for machine reasoning (and constraints implied therein), in order to describe the content of a resource. – Whether "Hanukiah" is an instance of the class *artifactFromDailyLife* or a class of individual entities does not modify the simple fact that an annotated photograph depicts a Hanukiah. Actually, one of the basic requirements from an ontology-based annotation vocabulary must be **to completely conceal the underlying logical intricacies from the annotator**.
The DAAT Framework adopted in MOSAICA (and detailed in 2.1.4 DAAT Framework below) presents concepts used for annotation as folders (such as MS Windows file system directories called *folders*), which further transforms the annotation procedure into storing a resource in desired folders. This is a very intuitive way to approach semantic annotation, since it assumes that the annotated resource will be in the future found "inside" the selected concept-folder. The only distinction from a classic file system is that in DAAT a resource can be "located"[17] in multiple concept-folders.
Accordingly, the principle guiding the design of the MOSAICA ontologies was to define as class everything that does not have a physical existence as an entity (such as, individual persons, countries, languages, etc.). The only exceptions were concepts that cannot be considered types of any "real" entities (i.e. that are classes with no identifiable instances, although they cannot be considered empty classes). Such a concept is, for example, a weekly Torah portion[18]. Although a

[16] To be exact: ontology **re-engineering**, since the foundation for the MOSAICA knowledgebase were the eJewish.org annotation vocabulary and the Jewish Encyclopedia online, as reported in D2.1 Conceptual Alignment of Multiple, Distributed Content Source.
[17] The original location of a resource defined by the URL schema of the URI is, of course, not modified.
[18] "(Hebrew: Parashat ha-Shavua, popularly just parashah or parshah and also known as a Sidra) is a section of the Torah (Hebrew Bible) read in Jewish services. In Judaism, the Torah is read publicly over the course of a year, with one major portion read each week in the Shabbat morning service" (http://en.wikipedia.org/wiki/Weekly_Torah_portion).

weekly Torah portion cannot be considered as a physical entity, it was defined as an individual, because it can also not be considered to be a class of a set of instances[19]. The same reasoning could be applied, for example, on all constituents of the Sefer Torah (defined thus as instances of the class *bookInTheHebrewBible*), but not to the Sefer Torah itself, since many prominent Torah scrolls existed throughout the history[20].
Although the eradication of the difference between classes and individual concepts led to the adoption of the OWL Full in MOSAICA, it was nevertheless clear that the ontology design has to preserve the OWL DL reasoning capabilities (detailed in 7 Appendix D: OWL DL Reasoning Capabilities). The logical inference is an important requirement from the annotation systems, since it enables:

1. **Intelligent querying** – guarantees *precision* in search, since only the search results that logically correspond to the query are retrieved (for example, the query "all images depicting Marc Chagall" will retrieve only the images depicting Marc Chagall, but not those depicting one of his paintings – which can be expected in a keyword search[21]);
2. **Inference in querying** – guarantees *recall* in search, since all results that can be inferred from the underlying knowledgebase are retrieved (for example, the query from the previous example will retrieve all images, including photographs, paintings, etc.).

The specific inference mechanisms of relevance specifically to MOSAICA are detailed in 2.1.2 Description Logic.
Hereby, the use of the annotation relations is of significant importance, and probably provides the most visible difference from the social tagging systems. Unlike social tagging (as well as librarian classification), semantic annotation formulates complete annotation assertions consisting of subject (reference to the resource), predicate and object (value). Predicates are hereby various annotation relations. Consider, for example, annotation of a (digital reproduction of a) painting using the (individual) concept "Marc Chagall": it makes a significant difference whether the relation *createdBy* or the relation *depicts* is used in the annotation!

2.1.1.2. Navigation

In addition to the logical reasoning, another critical aspect of the MOSAICA knowledgebase is its ability to facilitate effective navigation. There are several aspects of the navigation-related requirements:

1. **Human readability**. – Description logics in general are designed for machines to operate on them and typically ignore the human readability. OWL introduces *rdfs:label* (with *xml:lang* attribute) annotation property, allowing thus multilingual labeling of concepts and properties. However, this does not improve the readability of assertions, and does not simplify convoluted query formulation (see Figure 2 below), which is a critical usability success factor.
2. **Resolving clutter**. – Ontologies designed for logical reasoning often become soon too complex for display. A typical problem is the large number of classes displayed at every level in the hierarchical structure. This is even more emphasized by the fact that MOSAICA knowledgebase is a significantly large set of annotation concepts. Displaying

[19] A printed Torah or a Torah scroll does not identify weekly portions as entities, which could be in turn recognized as instances of the class "weekly Torah portion".
[20] The necessary distinction was introduced by differentiating between Torah (referring to the content) and Sefer Torah (literary: "the book of Torah", referring to the physical entity, typically appearing as hand-written scrolls).
[21] The query http://images.google.com/images?hl=en&q=Marc+Chagall&btnG=Search+Images&gbv=2 retrieves only two out of the first 18 images depicting the famous painter.

dozens of concepts in a single expanded tree node would simply preclude any navigation.

3. **Alternative divisions**. – A special case of the previous problem are alternative divisions that could be applied to some classes. People (instances of the class *person*), for example, can be divided by the period in which they lived, by region in which they lived, by their life achievements, etc. If subclasses resulting in all such divisions are displayed together in the same list, the navigation becomes cumbersome, since it is difficult to understand what is common to all those subclasses. However, OWL does not provide a mechanism to define alternative divisions that could be toggled in the display.
4. **Rich information** – One of the problems often encountered in the social tagging is that some tags are difficult to understand. This may be even more problematic in an ontology that supports logical reasoning, which is not always intuitive. For example, the meaning of a class such as "Book in the Hebrew Bible" may not be evident to every annotator, and the question may arise whether the section "Prophets" is to be considered an instance of that class or not[22]. Therefore, it would be certainly desirable to display the assertions describing a selected class, so that the user can get some idea regarding its meaning. The immediate implication of this is that assertions in the MOSAICA ontology should not only be formulated to facilitate the reasoning, but also to provide useful information.

In order to meet these navigation-related requirements, DAAT framework was adopted. DAAT Framework consists of two complementary conceptual models and two corresponding ontologies:

1. DAAT content management model and its corresponding DAAT Core Ontology, and
2. DAAT annotation model and its corresponding DAAT Annotation Ontology.

In addition, DAAT Framework provides a methodology for conversion of OWL axioms (both class and property definitions and individual assertions) into natural language statements. – This methodology will however not be detailed in this deliverable.
DAAT Framework was developed by one of the participants in the project for the purpose of empowering knowledge representation systems in general with display and navigational capabilities[23], and is detailed in 2.1.4 DAAT Framework.

2.1.1.3. Simplicity of Use

As discussed in 2.1.3 Web 2.0, semantic annotation was considered to be unreasonable demand from the WWW users, until the social tagging appeared, and became overnight a common practice. Although this opens new horizons for the (semi-) manual semantic annotation of the WWW, there are some preconditions that must be met, if semantic annotation intends to replace the tagging. – The most important of them is the usability: semantic annotation must be simple enough to become comparable to tagging. This is, however, easier said than done, especially in an extensive knowledgebase, such as the MOSAICA knowledgebase. The immediate implication for the ontology design is that the resulting knowledgebase must be intuitive enough, in order to ensure that the desired concept will be detected by navigation with a minimal effort. This requirement also has impact on the design of the annotation procedures as detailed in 3 Semantic Annotator.

[22] "Prophets" is actually an instance of the class "Parts of the Hebrew Bible". Accordingly, the division is hereby counterintuitive, since it is not based on meronymy (which is defined using assertions).
[23] DAAT Framework makes extensive use of annotation properties, which is another reason of preferring the OWL Full. OWL DL sets strict restrictions on the use of the annotations, which DAAT Framework clearly breaches.

2.1.2. Description Logic

OWL DL is ***SHOIN*(*D*)** Description Logic, as depicted in Table 1 below.
***SHOIN*(*D*)** provides reasoning capabilities detailed in Appendix D: OWL DL Reasoning Capabilities. Although these reasoning capabilities are already[24] impressive, the factual implementation is problematic. This is due not only to the fact that none of the available reasoners actually implements all of the reasoning capabilities ([Vrandecic 2005]), and that moreover these reasoners typically focus primarily on the validation and classification features (and not entailment), but it is rather related to the lack of a semantic search and retrieval product vision. Considering the overall amount of research effort invested in almost a decade, the applicability issues of the ontology engineering seems to be systematically ignored, which may be one of the reasons why the fast commercial uptake of the Web 2.0 somehow overtook the Semantic Web initiative, and de facto marginalized the contributions of the semantic research by applying simple, but significantly more user-oriented solutions[25].

S	***AL*** (Attributive Language)	Atomic negation (negation of concepts that do not appear on the left hand side of axioms) Concept intersection Universal restrictions Limited existential quantification
	C	Complex concept negation
		Transitive properties
H		Role hierarchy (sub-properties: rdfs:subPropertyOf)
O		Nominals (enumerated classes of object value restrictions - owl:one of, owl:hasValue)
I		Inverse properties
N		Cardinality restrictions (owl:Cardinality, owl:MaxCardinality)
(D)		Use of datatype properties, data values or data types

Table 1 Definition of OWL DL logic expressivity

It is rather evident that the competitiveness of the highly sophisticated experimental systems will be very low in comparison to the current standard for Web applications, as long as they relay on interfaces such as those proposed, for example, in [Staab et al. 2000] (depicted in Figure 2) or [Abasolo and Gomez 2000] (depicted in Figure 3).

(1) FORALL *Pub* ←
EXISTS ResID ResID:*Researcher* [NAME →
"Steffen Staab"; PUBLICATION →→ Pub].

Figure 2 Query formulated in F-Logic that would be an equivalent of the natural language question: "What are all publications by a researcher called 'Steffen Staab'?"

[24] Nevertheless, the current initiative is to replace OWL DL with the more expressive OWL 1.1 (http://www.w3.org/Submission/owl11-overview/), which is the even more expressive SROIQ Description Logic.
[25] Tom Gruber in [Gruber 2008] presents RealTravel (http://realtravel.com/), a Web 2.0 website that provides "Contextual Browsing: Combining tags, location, and rating data". The effect of intelligent software was accomplished without complex semantic technologies.

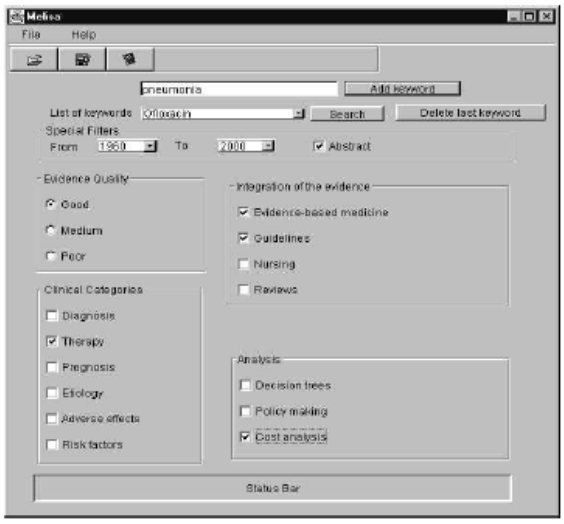

Figure 3 The so-called *Consultation Interface* for an ontology-based medical search engine

Research in MOSAICA focused primarily on the effective use of logical reasoning in a user-oriented application. Accordingly, MOSAICA introduced the so-called precision-search technology[26], depicted in Figure 4.

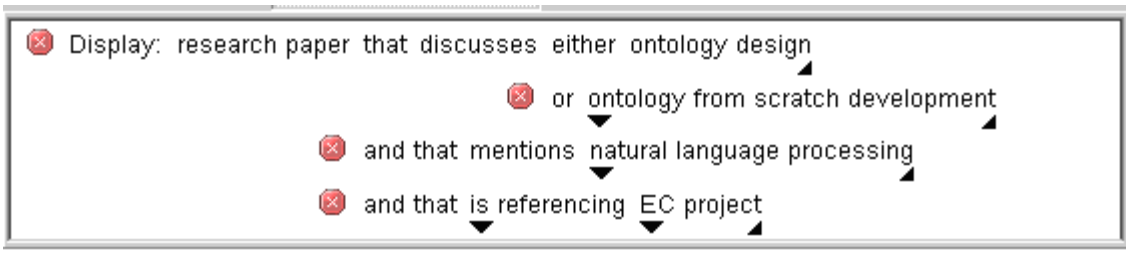

Figure 4 MOSAICA Precision-search graphic user interface

Precision-search technology enables formulation of queries in natural language that can be executed over an OWL-formatted knowledgebase while applying logical inference. The natural language formulation is made possible by introducing the controlled vocabulary drawn from the underlying ontology. User can formulate queries, or more precisely *query instructions* (always starting with the imperative form "Display …") by selecting either controlled vocabulary entries or predefined syntactic operators from dropdown lists. At every point in a sentence the dropdown list display only vocabulary that is applicable to that position in the sentence. The formulation is further simplified by introducing the smart auto-completion functionality, allowing user to start typing text, and then complete it by selecting an item from the list of displayed

[26] Precision-search technology is based on the DAAT Framework natural language conversion methodology.

matches. The smart auto-completion mechanism matches the suggestions not only from the beginning of an entry, but also from any other word occurring in multiple-word entries[27].
In addition to the controlled vocabulary, which originates in the labels defined in the underlying knowledgebase, precision-search introduces a series of syntactic operators[28] that provide parsing instructions, while preserving the natural language formulation. The following is an example of the resulting query instruction with syntactic operators marked in bold font (concept labels are marked in italics):

> **Display** *map* **that** presents *event* **that** took place in the *15th Century*, **and that** took place in *Spain*.

This query instruction is expected to retrieve, for example, the resource depicted in Figure 5, because:

[1] "Spanish expulsion and resulting Diaspora"[29] is a map.
[2] "Spanish expulsion and resulting Diaspora" presents Spanish expulsion.
[3] Spanish expulsion is an event.
[4] Spanish expulsion took place in the year 1492.
[5] 1492 is in the 15th Century.
[6] Spanish expulsion took place in Spain.

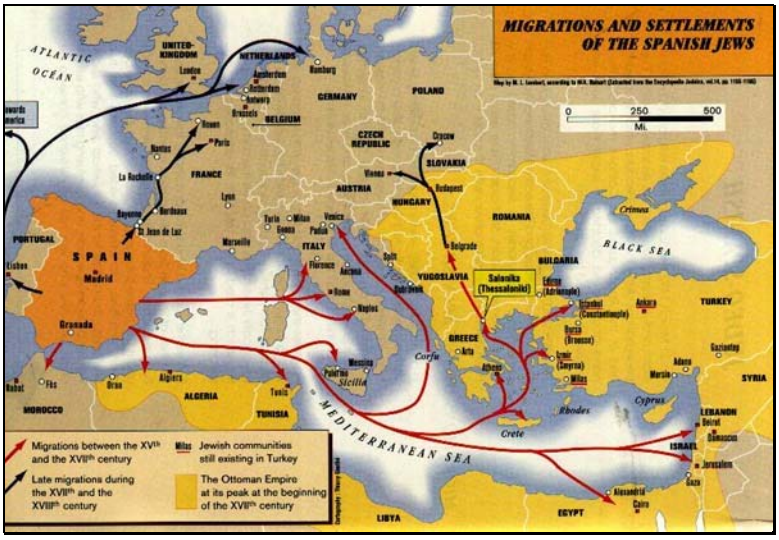

Figure 5 Spanish expulsion and resulting Diaspora

This kind of reasoning is typically referred to as the *ABox query*. ABox ("assertion box") is a set of statements that assert something about the real world. For example, the statements [1] through [6] above form an ABox. ABox is distinguished from the so-called *TBox* ("terminology box"), which is a set of axiom statements defining the vocabulary that will be used to formulate the assertion statements. In the Description Logics community the term 'ontology' is often

[27] Accordingly, if typing "Da…" in the auto-completion box, not only "David Copperfield" will be suggested, but also "King David" – although the typed text does not match the beginning of the entry, but rather the second word in it.
[28] For example, "that", "and", "or not", etc.
[29] "Spanish expulsion and resulting Diaspora" is the title given to the referenced resource, and is therefore enclosed in the quotes.

simply identified with the TBox (and factually viewed as a "schema", in contrast to the ABox "data"). ABox and TBox together are referred to as the *knowledgebase*.
In the example above the ABox querying proceeded by mostly applying the classification rules in order to infer by logical subsumption. Subsumption refers to the hyponym-hypernym relationship, established in the previous example between "Spanish expulsion" and an "event" that took place in the 15th Century Spain, or more formally.

$$spanishExpulsion \subseteq event.(tookPleceInThe(15thCentury) \cap tookPlaceIn(Spain))$$

Accordingly, the purpose of the ABox query above was to detect all instances that satisfy the definitions placed in the TBox. In the above example, the query instruction defines an (anonymous) class that does not exist in the ontology. The definition of that class is a complex logical statement expressed in a Description Logic:

$$event.(tookPleceInThe(15thCentury) \cap tookPlaceIn(Spain))$$

Querying proceeds so, that all assertions in the knowledgebase that satisfy the above definition are detected, and the subjects of the assertions are extracted as the satisfying instances.
The following are additional examples where logical reasoning based on the OWL mechanisms is applied to the querying in MOSAICA:

1. Location is detected within a larger geographical context by applying the logical inference through transitivity. For example, assuming that the role (property) $locatedIn$ was defined as *owl:transitiveProperty*:

   $$Madrid.locatedIn(Spain)$$
   $$Spain.locatedIn(Europe)$$

   Entails:

   $$Madrid.locatedIn(Europe)$$

2. Relations between resources are implied using inversion. For example, implying that the role (property) $isExtensionOf$ was defined as *owl:inverseOf* $isSummaryOf$:

   $$videoX.isExtensionOf(plotSummaryY)$$

   Entails:

   $$plotSummaryY.isSummaryOf(videoX)$$

3. Family (and intrapersonal) relationships are implied by a rather complex piece of reasoning. For example, if the role (property) $isParentOf$ was defined as *owl:inverseOf* $isChildOf$, then:

   $$Mary.isParentOf(John)$$

   Entails:

   $$John.isChildOf(Mary)$$

   However, when expressed in natural language ("John is a **child** of Mary") this assertion sounds somehow "unnatural", and seems to be better expressed as "John is **son** of Mary". Indeed, the roles (properties) $isMotherOf$, $isFatherOf$ and $isDaughterOf$, $isSonOf$ can be defined as *owl:subPropertyOf* of $isParentOf$ and $isChildOf$ respectively.
   Accordingly:

   $$John.isSonOf(Mary)$$

   Entails:

   $$Mary.isParentOf(John)$$

   Nevertheless, from here it cannot be concluded whether Mary is John's mother or father. This may be a serious problem when executing ABox queries requesting, for example, all

mothers of a certain group of persons. In order to correctly resolve such query instruction, it is necessary to introduce some kind of gender-based classes. For example:
$femalePerson \equiv person \cap (.isMotherOf(T) \cup isDaughterOf(T) \cup ... \cup isSisterOf)$ In addition, the domain of the roles (properties) specified in the logical union above must be defined as $femalePerson$.

Accordingly:

- In the first step $Mary$ is classified by subsumption as of the type $femalePerson$, by relying on an additional assertions, for example:
$$Mary.isSisterOf(Marc)$$
- In the next step, the assertion:
$$Mary.isMotherOf(John)$$
entails that $Mary$ is of the type $femalePerson$, which is indeed verified by the inference in the previous step.
- In the next step, the assertion:
$$John.isSonOf(Mary)$$
entails (by subsumption):
$$John.isChildOf(Mary)$$
Which further entails:
$$Mary.isParentOf(John)$$
- In the final step, the assertion:
$$Mary.isMotherOf(John)$$
Entails:
$$Mary.isParentOf(John)$$
Accordingly, it is inferred that $Mary$ is female parent of $John$ ($\equiv$ Mother).

Logical inference is of extreme importance for MOSAICA. We assume that the commercial uptake of MOSAICA depends directly on the system's usability and the impression of (machine) 'intelligence' that it provides. Therefore, it was one of the design goals to introduce logical inference wherever possible. Nevertheless, OWL proved sometimes unsuitable to that purpose, and we had to resort to other solutions. The following are few examples of such problems:

4. Placing a time specification within a broader timeline context requires simple calculations for which OWL does not provide specification mechanisms. In the example provided above for the ABox querying, the following assertion was stipulated:
[5] 1492 is in the 15th Century.
It is, however, unlikely that such assertions will be found in the ABox of a well-designed knowledgebase. The year "1492" is best provided as a datum, and there is not justification of its reification as an individual concept (which would require enumeration of all years of potential interest). Nevertheless, the "15th Century" could easily be reified as an instance of the class *century*, which can be defined, for example, by the properties *extendsFromTheYear* and *extendsToTheYear*. Accordingly, the assertions [5] above could be inferred by applying simple calculation over data. This calculation must be, nevertheless, defined in a proprietary algorithm, since OWL does not provide a mechanism to specify it.

5. Similar problem is encountered when the reasoning based on the proximity of location is required. Gruber in [Gruber 2008] provides example of an "intelligent" interface that can infer that New Jersey is closer to the New York City than Buffalo NY, although Buffalo NY is located in the same state, and New Jersey is not. Such inference requires either calculation

of distances based on the absolute position of the locations in a geographical grid, or utilization of some predefined distance matrices. In both cases, OWL is insufficient to provide this kind of reasoning, and proprietary algorithms must be developed.

2.1.3. Web 2.0

Since Tom Gruber's keynote at the 2006 Semantic Web conference ([Gruber 2006]), the question how to bridge the gap between the Semantic Web and the Web 2.0 seems to have become a research topic in its own right[30]. However, this question raises concerns only on the side of the Semantic Web researchers. – These concerns certainly do not seem to be shared by any of the major protagonist of the Web 2.0 phenomenon[31]. To the Semantic Web researchers the meteoric success of the Web 2.0 must seem like a fresh opportunity to eventually get the Semantic Web off the ground – in one way or another. Nevertheless, the old problems related to the so-called *semantic bottleneck* prevail: development and maintenance of high-quality real-life knowledgebases is costly, tedious and error-prone ([Zhang 2007]). Web 2.0 with its folksonomies did not solve the basic problem of how to develop a critical mass of the semantic resources that would enable development of applications that use them. Although folksonomies can be developed cheaply and quickly, they are not designed for machine processing ([Bateman et al. 2006]), and their manipulation by machines boils down to the complexity of the natural language processing [Gruber 2008].

Regardless of the lively drive on behalf of the Semantic Web community to somehow benefit from the sudden success of the Web 2.0, Web 2.0 in fact inflicted a serious blow to the vision of the Semantic Web (and to the significant resources invested into its research over the past decade or so): it demonstrated that "intelligent" (and enormously popular!) Web applications can be developed with very simple technologies and without the backup of a massive research. Web 2.0 marginalized the self-proclaimed achievements of the Semantic Web research by largely ignoring them. Web 2.0 accomplished a significant step towards the vision of an "intelligent" Web **without** complex inference algorithms (***SHIF***, ***SHOIN(D)*** or ***SROIQ***?), delicate terminological intricacies ("what is *ontology*?") and intercontinental standardization wars (RDF, DAML, OIL, DAML+OIL or OWL? OWL Lite, OWL DL, OWL Full, or OWL 1.1? OWL-S, WSMO or WSDL-S?). Web 2.0 met financial success comparable only to the infamous *dot-com bubble* **without** massive investment of public funding (at regional, national and international levels in parallel) into academic research that resulted primarily in indigestible amounts of low-quality research papers.

The real question is hereby if ontology engineering can learn anything from the surprising success of the Web 2.0? The following are few thoughts that seem relevant within the context of MOSAICA:

1. **Users are ready to contribute!** – Resources annotation that in the Semantic Web always seemed like a cliff too steep to climb became a common practice today. The enormous research effort (and funding) invested into experimental automatic and semi-automatic annotation technologies in the era of social tagging seem completely wasted. However, users' collaboration is not automatically granted. – There are two necessary conditions: 1) usability and 2) usefulness. Usability – tagging procedures must be simple. Users will tag resources as long as it requires only typing or selection of keywords from lists (or the so-called *tag clouds*). Users will not have patience for multi-

[30] Judging, form example, from: Bridging the Gap Between the Semantic Web and the Web 2.0, International Workshop located at the 4th European Semantic Web Conference ESWC 2007 June 7, 2007 - Innsbruck, Austria.

[31] For example: http://www.infoworld.com/article/08/06/04/Jimmy-Wales-unimpressed-Powerset-Wikipedia-search_1.html. Jimbo Wales (http://en.wikipedia.org/wiki/Jimmy_Wales) is de facto leader of the Wikipedia project.

step wizards, multiple list boxes and multi-node graphs offered by currently available semantic annotators. Usefulness – users must be motivated to tag. Tagging personal bookmarks, photos to share with friends and family, recipes, tips and tricks … – All this is useful, and user can easily see the benefits resulting from the effort. Delayed gratification is no gratification at all.

2. **However, user contribution has a price tag: low quality!** – There is a clear payoff between effort and the quality of the semantic resources. Semantic resources developed by a massive user base will never reach the quality of the semantic resources developed by professional knowledge engineers. Our anecdotic experience was repeated in MOSAICA: the question was raised concerning when to allow the consortium partners to start autonomously edit the MOSAICA knowledgebase, and the decision was constantly delayed, since the received contributions threatened to rapidly erode the delicately balanced MOSAICA knowledgebase. A typical error hereby (but corroborated in many other similar situations) was to introduce vast amounts of trivia information completely detached from the focus of the knowledgebase[32]. If the cooperation on behalf of the users is requested, the major attention needs to be paid to the quality assurance procedures.
3. **Description Logics may just be too expensive for the real-life applications!** – The expected benefits from the Description Logics may well be overrated. The inference capabilities of the Description Logics – even in theory – are much more limited than what is usually believed (even by knowledge engineers). Above, we pointed out two examples of simple, but useful calculations that required proprietary solutions outside the scope of the Description Logics. Moreover, Description Logics are not designed for annotation of resources ([Ide et al. 1997], [Welty 1998]). At some point the use of Description Logics may just become too expensive, and simpler, proprietary solutions may be introduced. – For example, rather than running complex algorithms in order to decide if somebody is mother or father of somebody else, maybe it would be cheaper to prompt user to specify that when entering data?
4. **What matter is application, not technology!** – Web 2.0 did not introduce any novel technology. Even AJAX is just a marginal development of the existing technologies, and Web 2.0 can even operate without it. What Web 2.0 introduced is a novel use of the Internet. It transformed the WWW from an online library into a meeting place for socialization. And the success was enormous. It is not technology that drives that success. – The innovative applications provide the thrust. Semantic technologies are a good solution that is still looking for the problem. The research effort should be therefore shifted from "how to better the already existing semantic technologies" to "how to use them". The recently released Powerset (http://www.powerset.com/) demo was received with mixed reactions. This should be rather worrying news for everybody involved in the Semantic Web research. Powerset claims to be a true semantic search engine, with most of the functionalities that the Semantic Web envisions. Nevertheless, when compared with Google, Powerset was found to be largely underperforming[33]. Maybe the semantic technologies should not compete in the category of search and retrieval? Web 2.0 succeeded without any revolutionary search and retrieval technology. Actually, the retrieval mechanisms that utilize tags are – technologically – probably the most primitive search engines around. Web 2.0 brought new applications that proved to be useful to millions of users. Maybe the semantic technologies should do the same?

[32] For example, a typical erroneous tendency was to introduce numerous educational institutions from all over the globe whose only connection to the Jewish cultural heritage was through the fact that a prominent Jew spent some time there.

[33] http://www.rojo.com/story/4hEEdtMHFrKf7ZJk

2.1.4. DAAT Framework

DAAT Framework consists of two complementary conceptual models and two corresponding ontologies:

1. DAAT content management model and its corresponding DAAT Core Ontology, and
2. DAAT annotation model and its corresponding DAAT Annotation Ontology.

2.1.4.1. DAAT Content Management Model

DAAT content management model handles aspects of the semantic infrastructure that are related to the **use** of annotations.

Requirements

1. The system *must* distinguish between three different conceptualization frameworks:
    a. **Domain conceptualization** – concepts specifying the conceptualization of the underlying content and their corresponding properties and relations enabling these concepts to be associated with the resources. These concepts, properties and relations *must* be visible to the user in various displays.
    b. **Resource conceptualization** – resource types enabling classification of resources in the system and their corresponding properties and relations enabling resources to be related to each other. These concepts, properties and relations *must* be also visible to the user.
    c. **Operational conceptualization** – auxiliary concepts, properties and relations used to provide system functionalities. These concepts, properties and relations *must* be invisible to the user, and are used only by the system.
2. The system *must* allow multiple sub-classing divisions of a class in the domain conceptualization framework. – A class could be often divided into subclasses by applying various division criteria. Each division must be displayed as a separate "view" of the super-class (rather than mixing all subclasses into a single display). A typical case is the subdivision of the top-level classes related to the **Domain conceptualization** into a smaller number of distinct "topics" through which domain ontology is accessed.
3. The system *must* enable classes related to the **Domain conceptualization** to be dynamically extended by the users. – It is assumed hereby that the elements of the **Domain conceptualization** cannot be modified by the user (but rather only by the ontology engineers and domain experts). Nevertheless, some classes in the **Domain conceptualization** are generic, and *must* be extended by specific subclasses[34]. However, such extension cannot be sub-classing, since 1) the extension's semantics is instantiation (not "a kind of" relationship – which should be always defined in the domain ontology, but rather "is a" relationship specifying that extension is an instance of

[34] For example, the **Domain conceptualization** may contain a class *Project* intended to contain all resources related to some projects in which the user is involved. However, the user may be involved in multiple projects, and the generic class *Project* would need to be further divided into specific projects in which the user is involved. This cannot be done in advance, since some of the projects may not be known at the time when the current version of the ontologies is developed, and such knowledge will become available only in the future. Accordingly, the user must be able to autonomously extend in the future the class *Project* with specific projects that may occur over the course of time. It must be noted here tat not every class related to the **Domain conceptualization** needs to be extended in such manner.

the class that is not generic, or even not known in advance at the time of ontology modeling), and 2) user defined extensions should be an item in the underlying knowledgebase, and not an element in the ontology.

4. When extending a class as required in 3 above, the system *must* enable automated subdivision of that extension. – The required extension may be required to be sub-structured automatically in a pre-defined way[35]. Accordingly, when generating a new extension, the entire sub-structure must be created automatically.
5. The system *must* enable creation of super-classes combining arbitrarily selected concepts from ontology. – User may decide to create an ad hoc collection of concepts (and resources) containing arbitrarily selected elements. Potential situations where such need may occur are primarily 1) "favorites" collection containing selected resources, but also complete sets of resources grouped by a particular concept (e.g., all resources annotated by the same concept), and 2) top-level view containing quick links to selected concepts.
6. The system *must* enable import of resources while applying the annotation mechanism to them. – The ultimate objective of the content management model is to enable import and semantic annotation of single resources provided as files (which may include URI or even script file dynamically generating HTML, Adobe Flash or similar documents, shortcuts and batch files running entire applications or accessing databases, etc.).

Solutions

1. *Markup Mechanism*

This mechanism enables marking the functionality of ontology elements (primarily concepts, but if necessary also properties and relations) in the terms of their belonging to a specific conceptual framework. It is specified in the following way:

- The annotation data type property *isDescriptor* is introduced in the DAAT Core Ontology (within the operational conceptualization framework). This property is devised to reference elements of the **Domain conceptualization**.
- The annotation data type property *isResourceType* is introduced in the DAAT Core Ontology (within the operational conceptualization framework). This property is devised to reference elements of the **Resource conceptualization**.

The *Markup Mechanism* works as follows:

- The ontology elements belonging to the **Operational conceptualization** are not annotated.
- All classes that will be considered as resource types are annotated using the annotation property isResourceType specifying value 'True' as object. If a class is annotated that way, all its subclasses will be automatically considered as resource types. The expected uses of this annotation are primarily as follows:
  - Assigning resource type to a resource (when importing it into the system, and annotating it),
  - Selecting the subject of an advanced search query statement, and

[35] Extending the example from the previous footnote: the class *Project* may be further structured into various "topics", such as, for example, *Administration*, *Reporting*, etc. Accordingly, when extending the class Project with a specific project, it is expected that these topics will be automatically created for that new, specific project.

- Displaying all available resource types for navigation through resources by resource type.

Classes annotated using the annotation property isResourceType do **not** need to be defined in the same ontology, and even do not need to be defined in a single ontology.

- All classes that will be considered as descriptors of the underlying domain of application are annotated using the annotation property isDescriptor specifying value 'True' as object. If a class is annotated that way, all its subclasses will be automatically considered as domain descriptors. The expected uses of this annotation are primarily as follows:
  - Annotating ("tagging") resources imported into the system,
  - Selecting object in an advanced search query statement, and
  - Displaying conceptualization of the underlying domain of application for navigation by domain semantics.

  Classes annotated using the annotation property isDescriptor do **not** need to be defined in the same ontology, and even do not need to be defined in a single ontology.

2. *Views Mechanism*

The class that can be divided into subclasses by applying various division criteria is called *Multifaceted Class*. Each subdivision is called *View*. It is specified in the following way:

- The annotation object property *canBeSubdividedBy* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).
- For each *Multifaceted Class* a corresponding enumeration class is created (within the domain conceptualization framework). The naming convention for these enumerated classes is the following: [name of the *Multifaceted Class*] + "View".
- *Multifaceted Class* is annotated using the annotation property canBeSubdividedBy referencing the corresponding enumerated class.
- The annotation object property *isVisibleUnderView* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).
- Each subclass of the *Multifaceted Class* is annotated by a single annotation using the annotation property *isVisibleUnderView* referencing a specific instance of the corresponding enumeration class.
- The annotation data type property *isDefaultView* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).
- One instance of the enumeration class is selected as the so-called *Default View* of the *Multifaceted Class*. The *Default View* is annotated using the annotation property *isDefaultView* specifying value 'True' as object.

The *Views Mechanism* operates as follows:

- When class is selected, it is checked for the annotation using *canBeSubdividedBy*. – If it exists, the selected class is multifaceted (and the procedure proceeds).
- Loop through the instances of the enumerated class referenced by the annotation, and display each instance for selection.
- While looping, check each instance for the annotation using *isDefaultView*. – If it exists, the current instance is the *Default View*.

- Select the *Default View* in the display of the *Views*. Loop through the subclasses of the *Multifaceted Class*. Check each subclass for the annotation using *isVisibleUnderView*. Check if the *Default View* is referenced by the annotation. If so, display it as the subclass of the *Multifaceted Class*.
- When user selects a *View* from the display of the *Views*, loop through the subclasses of the *Multifaceted Class*. Check each subclass for the annotation using *isVisibleUnderView*. Check if the selected *View* is referenced by the annotation. If so, display it as the subclass of the *Multifaceted Class*.

The top-level class in a domain ontology (*owl:Thing*) can be also specified as multifaceted, using the *Views Mechanism*.

3. *Flex Mechanism*
   The class that can be extended by the user with specific instances is called *Flexible Class*. It is specified in the following way:
   - The annotation data type property *isExtensible* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).
   - *Flexible Class* is annotated using the annotation property isExtensible specifying value 'True' as object.
   - Class customElement is defined in the DAAT Core Ontology (within the operational conceptualization framework). This class does not have subclasses. customElement is a generic class specifying a table in the knowledge base[36]. *Extension*s of a *Flexible Class* will be all recorded as instances of the customElement class (and consequently as entries in the underlying knowledgebase).
   - The data type property *isExtensible* is introduced in the DAAT Core Ontology (within the operational conceptualization framework). Every *Flexible Class* is annotated using the annotation property isExtensible specifying value 'True' as object.
   - Class elementType is defined as an enumeration class. The following are enumerated instances of this class:
     - etClassExtension
     - etTemplateExtension
     - etTemplateImplementation
     - etBasket
   - The object property *isElementOfType* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).
   - The data type property *isExtensionOf* is introduced in the DAAT Core Ontology (within the operational conceptualization framework)[37].

[36] The conversion of ontologies into the relational database schema and knowledgebase implementation is handled by a separate ontology, and not specified in this document.

[37] Although it references other concepts in the ontology, isExtensionOf is defined as a **data type** property, rather than object property (relation). The reason for it is that referenced concepts are classes, rather than their instances, and if converted into a relational database, they would become tables, and not data entries that could be referenced. Moreover, the domain ontology, which consists mostly of (a large number of) classes, is not to be converted into a relational database (which would feature with a large number of tables), since no data (instances) are to

When creating an *Extension* of a *Flexible Class*, the *Flex Mechanism* operates as follows:

- Application enables user to create an *Extension* of a *Flexible Class*. When user decides to create such extension, a new individual concept is created as an instance of the class customElement. – This individual concept is called *Extension*.
- An assertion is formulated for the new *Extension* using the property *isElementOfType* specifying etClassExtension as object.
- An assertion is formulated for the new *Extension* using the property isExtensionOf specifying the QName of the *Flexible Class* as object[38].

When selecting a class, the *Flex Mechanism* operates as follows:

- When class is selected, it is checked for the annotation using the annotation property isExtensible. – If it exists, the selected class is flexible (and the procedure proceeds).
- Loop through assertions formulated using the property isExtensionOf and referencing the selected class. The referencing concept (i.e. annotation subject) is an *Extension* of the selected *Flexible Class*. Display it as a subclass of the selected class.

4. *Template Mechanism*
This mechanism extends, and operates only in conjunction with the *Flex Mechanism*. A subclass of a *Flexible Class* is called *Template*. Subclasses of *Template*s are also *Template*s. It is specified in the following way:
   - *Flexible Class* is annotated using the annotation property isExtensible specifying value 'True' as object. Its subclasses are not annotated.
   - The object property *isSuubclassOf* is introduced in the DAAT Core Ontology (within the operational conceptualization framework).

   When creating an *Extension* of a *Flexible Class* using the *Flex Mechanism*, the *Template Mechanism* operates as follows:
   - When an *Extension* of a *Flexible Class* is created loop through the subclasses of the selected *Flexible Class* – these are *Template*s.
   - For each subclass, a new individual concept is created as an instance of the class customElement. – This individual concept is called *Template Implementation*.
   - An assertion is created using the property *isElementOfType* specifying etTemplateImplementation as object.
   - An assertion is created using the property isSubclassOf referencing the super-class (initially this is the *Extension* of the *Flexible Class*, but for the subclasses of a *Template*, this is the implementation of that *Template*).
   - *Template* is checked for the annotation using isExtensible. – If it exists, the *Template* is flexible (see below). *Template Implementation* is annotated using

be recorded in such database. Even if such conversion would be undertaken, isExtensionOf cannot be defined as an object property (i.e. a relation in the relational database), since it references classes (tables), and not their instances (data that could be referenced in a relational database).

[38] Accordingly, a mechanism must be implemented that converts the QName of a class into a datum of string data type.

isExtensible and specifying valuie 'True' as object.

- *Template* is checked for the annotation using canBeSubdividedBy. – If it exists, the *Template* is multifaceted (see below). *Template Implementation* is annotated using canBeSubdividedBy and copying the reference from the *Template*'s annotation.
- *Template* is checked for the annotation using isVisibleUnderView. – If it exists, *Template Implementation* is annotated using isVisibleUnderView and copying the reference from the *Template*'s annotation.
- Repeat procedure by recursion for subclasses of each subclass.

When selecting an *Extension* of a *Flexible Class*, the *Template Mechanism* operates as follows:

- When *Extension* is selected, loop through assertions formulated using the property isSubclassOf and referencing the selected *Extension*, and display the referencing concept (annotation subject) as a subclass of the selected *Extension*.

A *Template* can be defined as flexible. In this case, when creating an *Extension* of a flexible *Template Implementation* the *Template Mechanism* operates as follows:

- When user decides to create *Extension* of a *Template*, a new individual concept is created as an instance of the class customElement. – This concept is called *Template Extension*.
- An assertion is created using the property *isElementOfType* specifying etTemplateExtension as object.
- An assertion is created using the property isExtensionOf specifying the QName of the *Flexible Template* as object.

When selecting a *Template Implementation* of a *Flexible Template*, the *Flex Mechanism* operates as follows:

- When *Template Implementation* is selected, it is checked for the annotation using the annotation property isExtensible. – If it exists, the selected *Template Implementation* is flexible (and the procedure proceeds).
- Loop through assertions formulated using the property isExtensionOf and referencing the selected *Template Implementation*. The referencing concept (i.e. annotation subject) is a *Template Extension* of the selected *Template Implementation*. Display it as a subclass of the selected *Template Implementation*.

Subclasses of a *Flexible Template* are not considered when a *Template Implementation* is created, and will not be used to automatically create their corresponding *Template Implementation*s. They will be considered **only** when a *Template Extension* of the *Flexible Template* will be created (manually) by the user. In that case, the subclasses of the *Flexible Template* will be used as the *Template*s for the newly created *Template Implementation*.

A *Template* can be defined as multifaceted. In this case the *Template Mechanism* operates as follows:

- When a *Template Implementation* is selected, it is checked for the annotation using canBeSubdividedBy. – If it exists, the selected *Template Implementation* is implementation of a *Multifaceted Template* (and the procedure proceeds).
- Loop through the instances of the enumerated class referenced by the annotation, and display each instance for selection.

- While looping, check each instance for the annotation using isDefaultView. – If it exists, the current instance is the *Default View*.
- Select the *Default View* in the display of the *View*s. Loop through assertions formulated using the property isExtensionOf and referencing the selected *Template Implementation*. Check each referencing concept (i.e. annotation subject) for the annotation using isVisibleUnderView. Check if the *Default View* is referenced by the annotation. If so, display the referencing concept (i.e. annotation subject) as the subclass of the selected *Template Implementation*.
- When user selects a *View* from the display of the *View*s, loop through assertions formulated using the property isExtensionOf and referencing the selected *Template Implementation*. Check each referencing concept (i.e. annotation subject) for the annotation using isVisibleUnderView. Check if the selected *View* is referenced by the annotation. If so, display the referencing concept (i.e. annotation subject) as the subclass of the selected *Template Implementation*.

5. *Basket Mechanism*
   A collection of arbitrarily selected concepts is called *Basket*. It is specified in the following way:
   - When user decides to create a *Basket*, a new individual concept is created as an instance of the class customElement.
   - An assertion is formulated for the new *Extension* using the property *isElementOfType* specifying etBasket as object.
   - The annotation object property *contains* is introduced in the DAAT Core Ontology (within the operational conceptualization framework)[39].

   The *Basket Mechanism* operates in the following way:
   - When user adds a new concept to the *Basket*, an assertion is formulated for the *Basket* using the property contains referencing the added concept.
   - When user selects a *Basket*, loop through annotations formulated for the selected *Basket* and using the property contains. Display the referenced concept (i.e. annotation object) as the subclass of the selected *Basket*.
6. In the DAAT system a resource is considered to be simply an individual concept identified by a unique identifier (QName). It is specified as follows[40]:
   - Class importedResource is defined in the DAAT Core Ontology (within the operational conceptualization framework). This class does not have subclasses. importedResource is a generic class specifying a table in the knowledge base.
   - When user selects a resource for importing, a new individual concept is created as an instance of the class importedResource.

Glossary

[39] The data type property isExtensionOf cannot be used in this mechanism, since 1) basket can contain instances and classes (therefore an annotation property is required that can reference classes), and 2) the operating mood requires to specify basket's content as a collection, rather than to specify for each contained item is reference to the containing basket.

[40] This approach practically introduces OWL Full (by allowing assertions to be formed using not only individual concepts, but also classes).

Term	Explanation
Basket	Collection of arbitrarily selected concepts and resources. Baskets are created ad hoc manually by the user. Their content is determined by user selection, and it can be dynamically modified over a period of time.
Default View	One of the Views specified for a *Multifaceted Class* selected to be displayed automatically, prior to the manual selection of a View by a user.
Extension	Individual concept created as a specific instance of a *Flexible Class*.
Flexible Class	Class that can be extended by the user with a specific instance of it.
Flexible Template	*Template* defined as a *Flexible Class*. Flexible template enables *Extension* of its corresponding Template Implementations.
Multifaceted Class	Any class that can be divided into subclasses by applying various division criteria.
Multifaceted Template	*Template* defined as a *Multifaceted Class*. Multifaceted template can be divided into subclasses by applying various division criteria.
Template	Subclass of a *Flexible Class* used in the *Template* Mechanism to automatically create hierarchical structure for the *Extension* of a *Flexible Class*. Subclasses of a template are also templates.
Template Extension	Individual concept created as a specific instance of a *Flexible Template*.
Template Implementation	Individual concept automatically created from a *Template*. There will be a template implementation of each *Template* for each *Extension*.
View	A specific criterion applied to divide *Multifaceted Class*. Views are defined as instances of an enumeration class corresponding to the *Multifaceted Class*.

2.1.4.2. DAAT Annotation Model

DAAT Annotation model handles aspects of the semantic infrastructure that are related to the creation of annotations.

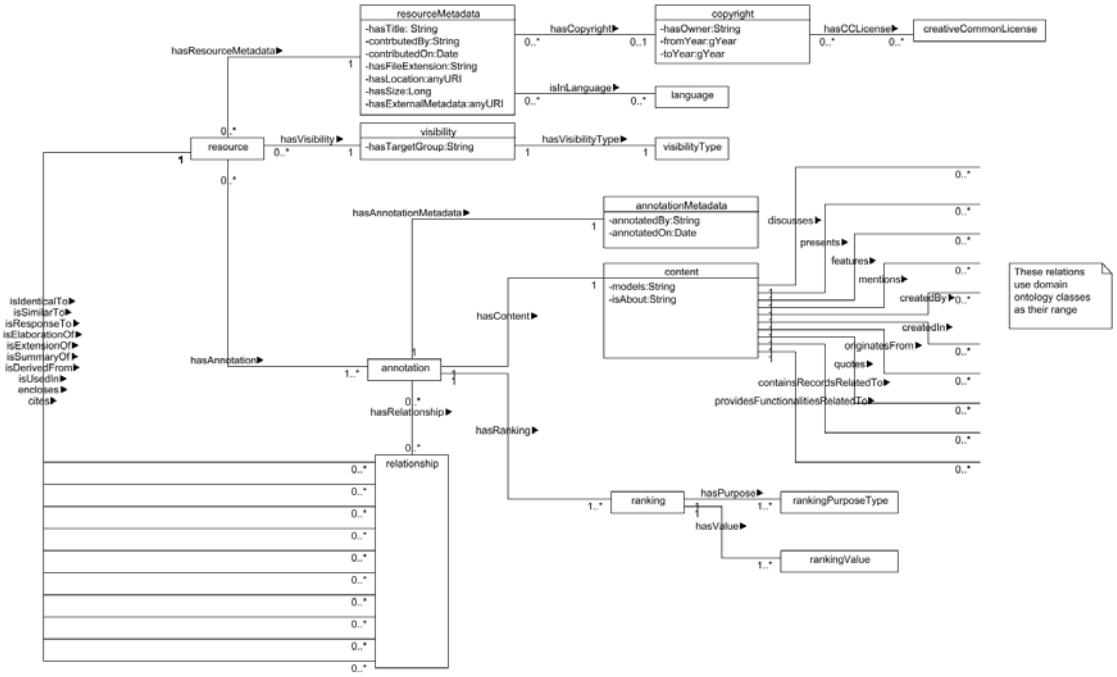

Figure 6 The Annotation Ontology UML diagram

Semantics

Resource is the annotated object. Its precise definition depends on the underlying application. Resource could be any file, or just specific types of file. In any case, resource is externally identified by its Unique Resource Identifier (URI).
From the point of view of annotation, all annotations reference a resource as their subject. Resource is defined by three relations and their corresponding objects:

Relation /Property	Cardinality	Object/Data type
hasResourceMetadata	One-to-one	***resourceMetadata***
hasVisibility	One-to-one	***Visibility***
hasAnnotation	One-to-many[41]	***Annotation***

resourceMetadata holds all basic metadata of a resource as a *file* – a digital entity with physical existence. It is defined by the following properties and relations:

Relation /Property	Cardinality	Object/Data type
hasTitle	One-to-one	String
contributedBy	One-to-one	String (holding an external identifier[42])
contributedOn	One-to-one	Date
hasFileExtension	One-to-one	String

[41] Multiple annotations created by one or multiple users could be associated with a single resource.
[42] This identifier is provided by the authentication and security layer of the underlying application. Since it is not stipulated that this layer uses formal ontology (and therefore user may not be defined as an instance of a class), the reference is of data type.

Relation /Property	Cardinality	Object/Data type
hasLocation	One-to-one	String (holding an internal identifier, such as hash key)
hasOriginalLocation	One-to-one	anyURI
hasSize	One-to-one	Long
hasExternalMetadata[43]	One-to-many	anyURI[44]
hasCopyright	One-to-one	***Copyright***
isInLanguage	One-to-many[45]	Language Types

Copyright holds copyright data. It is defined by the following relations and properties:

Relation /Property	Cardinality	Object/Data type
hasOwner	One-to-one	String
fromYear	One-to-one	gYear
toYear	One-to-one	gYear
hasCCLicense	One-to-many[46]	***creativeCommonsLicense***

Visibility defines visibility of a resource within the authentication and security mechanisms of the underlying application. It is defined by the following relations and properties:

Relation /Property	Cardinality	Object/Data type
hasVisibilityType	One-to-one	***visibilityType***
hasTargetGroup	One-to-many	String (holding an external identifier[47])

Annotation holds information about the content of a resource. This information falls into three main categories: association with concepts in a domain ontology, association with other resources in the system, and functionality ranking. It is defined by the following relations and properties:

Relation /Property	Cardinality	Object/Data type
hasAnnotationMetadata	One-to-one	***annotationMetadata***
hasContent	One-to-one	***Content***
hasRelationship	One-to-one	***Relationship***
hasRanking	One-to-many	***Ranking***

annotationMetadata holds all basic metadata of an annotation. It is defined by the following properties:

[43] Regarding the automatic transformation of DCMI into DAAT, see: Transformation of DCMI into DAAT.
[44] It is hereby assumed that external metadata are provided as a file (most probably XML-formatted) with a valid URI.
[45] A resource can combine several languages in its content (for example, parallel translations).
[46] Composite Creative Common licenses are composed of a combination of the elementary licenses.
[47] This identifier is provided by the authentication and security layer of the underlying application. Since it is not stipulated that this layer uses formal ontology (and therefore user may not be defined as an instance of a class), the reference is of data type.

Relation /Property	Cardinality	Object/Data type
annotatedBy	One-to-one	String (holding an external identifier)[48]
annotatedOn	One-to-many[49]	Date

Content holds associations between the currently annotated resource and various concepts in a domain ontology provided by the underlying application. To that purpose, a unifying core ontology provided by the application should import simultaneously both ontologies, and then use domain and range specification mechanisms to connect them. It is defined by the following relations and properties:

Relation /Property	Cardinality	Object/Data type
discusses	One-to-many	[domain ontology concept]
presents	One-to-many	[domain ontology concept]
features	One-to-many	[domain ontology concept]
mentions	One-to-many	[domain ontology concept]
createdBy	One-to-many	[domain ontology concept]
createdIn	One-to-many	[domain ontology concept]
originatesFrom	One-to-many	[domain ontology concept]
quotes	One-to-many	[domain ontology concept]
depictsEvent	One-to-many	[domain ontology concept]
depictsLocation	One-to-many	[domain ontology concept]
depictsPeriod	One-to-many	[domain ontology concept]
depictsSite	One-to-many	[domain ontology concept]
isArtefact	One-to-many	[domain ontology concept]
containsRecordsRelatedTo	One-to-many	[domain ontology concept]
providesFunctionalitiesRelatedTo	One-to-many	[domain ontology concept]
models	One-to-many	String
isAbout	One-to-many	String

[48] This identifier is provided by the authentication and security layer of the underlying application. Since it is not stipulated that this layer uses formal ontology (and therefore user may not be defined as an instance of a class), the reference is of data type.
[49] Annotation can be modified. Therefore the earliest date is the date of creation, and others are modification dates.

The above relations and properties are designed to provide a minimal meaningful set of annotation properties that satisfy the following requirements:

1. Provide annotation semantics sufficiently detailed to reflect the differences how topics could be handled in the content of various resources. Typical problem with the tagging is that all tags are attached to a resource without clarifying in which way they may be relevant to it. The same problem occurs in keyword annotation using a generic relation, such "related to". Topics can be related to resources in various ways, and a good semantic annotation system should provide a minimal, generic set of potential relations.
2. Ensure meaningful natural language expression of created semantic annotations. The annotations are designed to be displayed as fully grammatical and meaningful natural language sentences.
3. Cater for the potential diversity of the resource types, and how they handle content. For example, application handles content differently than a document.
4. Provide a minimal set of annotation properties and relations, in order to simplify the annotation process. Ideally, the annotation relation should be selected automatically by the system by simply selecting a concept in the domain ontology.

Following these requirements, the annotation relations and properties could be divided into the following groups:

- Generic
  - *isAbout*
- Resource type specific
  - Document[50] specific
    - Generic content
      - *discusses*
      - Specification of topic vs. feature
        - *features*
      - Specification of central topic vs. marginal
        - *mentions*
      - Specification of examination vs. presentation
        - *presents*
    - Specific content
      - Relation to a temporal reference
        - *createdIn*
      - Relation to a location reference
        - *originatesFrom*
      - Relation to a person
        - *createdBy*

[50] The intention hereby is all resource types, except application, i.e. graphic resource, sound resource, text resource, video resource, and in particular various composite resource types.

- *quotes*
- application specific
  - *providesFunctionalitiesRelatedTo*
  - map specific
    - *depictsEvent*
    - *depictsLocation*
    - *depictsPeriod*
    - *depictsSite*
    - *isArtifact*
  - database specific
    - *containsRecordsRelatedTo*
- knowledge representation system specific
  - *models*

Relationship creates an association between the currently annotated resource and other resources already available in the system. It is defined by the following relations:

Relation /Property	Cardinality	Object/Data type
isIdenticalTo	One-to-many	***Resource***
isSimilarTo	One-to-many	***Resource***
isResponseTo	One-to-many	***Resource***
isElaborationOf	One-to-many	***Resource***
isExtensionOf	One-to-many	***Resource***
isSummaryOf	One-to-many	***Resource***
isDerivedFrom	One-to-many	***Resource***
isUsedIn	One-to-many	***Resource***
encloses	One-to-many	***Resource***
cites	One-to-many	***Resource***

Ranking associates a particular functionality with a resource, and ranks the resources usefulness for that particular functionality. Functionalities are domain specific, and defined in an enumeration class that is extended through the underlying application. It is defined by the following relations:

Relation /Property	Cardinality	Object/Data type
hasPurpose	One-to-one	***rankingPurposeType***
hasValue	One-to-one	***rankingValue***

External Semantics

creativeCommonsLicense is an enumeration class enumerating all Creative Common licenses (http://creativecommons.org/about/license/):

- ccAttribution

- ccNoncommercial
- ccNoDerivativeWorks
- ccShareAlike

visibilityType is an enumeration class. It defines how a resource appears to the users of an underlying application:

- onlyOwner
- allUsers
- selectedGroups

rankingPurposeType is an enumeration class that is not defined in advance. The enumerated individuals are domain specific, and defined separately for each underlying application. Examples:

- forIllustration
- forDemonstration
- asExample
- forTeaching
- forFun
- …

rankingValue is an enumeration class that is not defined in advance. The enumerated individuals are domain specific, and defined separately for each underlying application. In principle, this enumeration class provides a Likert-type scale. Examples:

- notUseful
- somewhatUseful
- useful
- veryUseful
- excellent

It is defined by the following property:

Relation /Property	Cardinality	Object/Data type
hasNumericEquivalentOf	One-to-one	xsd:nonNegativeInteger

hasNumericEquivalentOf is used to associate a ranking value with a numeric value that can be used for ordering, graphic display (e.g., number of stars, or colored stars on a graphic scale, etc.), etc.

Supported Resource Types

The selection of the supported resource types is domain specific, and defined by the underlying application. In addition, the mechanism of the Selection of Content Annotation Relationships and Properties enables specific filtering of the resources recognized by the application.

Primitive Resource Types

- graphic resource
- sound resource
- text resource
- video resource
- application

Derivative and Composite Resource Types
Derivative resource types are considered to be specific cases of the primitive resource types. For example:

- map (graphic resource)
- database (application)
- knowledge representation system (text resource)
- lexical resource (text resource)
- semantic resource (text resource)
- software code (text resource)
- …

Composite resource types are composed of primitive and derivative resource types. For example:

- document
- web page
- …

Selection of Content Annotation Relationships and Properties

When a resource is annotated, the question comes up: which of the annotation relations and properties used to describe the content of a resource can be used with this particular resource? In principle, OWL provides the domain and range specification mechanism to that purpose. However, this mechanism is not applicable to the class ***Resource***, since its instances are not directly annotated. The class ***Annotation*** is introduced, in order to allow multiple annotations of the same resource provided by different users. Accordingly, a single resource will have one or more annotations. Further, the class ***Content*** was introduced, in order to group all content describing relations and properties together. Accordingly, the domain of the content describing properties and relations is the class ***Content*** which is related to the class ***Annotation*** via the relation hasContent (which in turn is related to the class ***Resource*** via the relation hasAnnotation).
Nevertheless, not every annotation relation and property describing content is suitable to every resource type. For example, the relation providesFunctionalitiesRelatedTo is applicable only to the resource type application. Since this cannot be specified via the domain mechanism, the annotation relation canBeUsedOnlyWith was introduced. This annotation relation is designed to be used for annotating the content describing relations and properties (see above). Each of these relations and properties is related via this relation to one or more specific resource types (see above).

Supported File Types

Supported File Types is a class comprising a number of enumeration subclasses equal to the number of the application-supported resource types. For each supported resource type a set of

recognized extensions can be defined, enabling the application to restrict the handling of resource files (for example, their upload) to a controlled subset.

Language Types

Language types are expected to be imported into the underlying application in one of the two ways:

1. As a reference from the IANA Tags for Identifying Languages (either RFC 3066 or RFC 4646/7 Language code assignments). In that case a language type is a literal in a field of the data type String.
2. As an enumeration class in ontology enumerating all language tags above as its instances. In the case a language type is an instance of that enumeration class.

Transformation of DCMI into DAAT

A useful feature of a semantic annotation system is the ability to import the existing metadata, and automatically transform them into the internal proprietary format. Since the widest used metadata standard is today the Dublin Core Metadata Initiative (DCMI), the following is a set of implementation recommendation how to maximize the information contain in a DCMI metadata file, in order to automatically transform it into semantic annotations following the DAAT conceptual model.

Contributor - The value of the field is extracted as the value of the object (data type String) in a statement formulated using the contributedBy data type property.

Coverage - DCMI defines this metadata field as follows: "Spatial topic and spatial applicability may be a named place or a location specified by its geographic coordinates. Temporal topic may be a named period, date, or date range. A jurisdiction may be a named administrative entity or a geographic place to which the resource applies."[51] Although the intentions of specific data in this field are ambiguous, the value of the field can be extracted, and suggested as the object in a statement formulated using the originatesFrom property. However, the property originatesFrom is an object property (relation) with the range specified as ***Location Reference*** (defined in domain ontology). Accordingly, the extracted DCMI value must be matched to the appropriate ***Location Reference*** concept.

Creator - DCMI defines this metadata field as follows: "An entity primarily responsible for making the resource."[52] Although the intentions of specific data in this field are ambiguous, the value of the field can be extracted, and suggested as the object in a statement formulated using the createdBy property. However, the property createdBy is an object property (relation) with the range specified as ***Person*** (defined in domain ontology). Accordingly, the extracted DCMI value must be matched to the appropriate ***Person*** concept.

Date - Date field of the DCMI cannot be utilized due to the underlying ambiguity. DCMI defines this metadata field as follows: "A point or period of time associated with an event in the lifecycle of the resource."[53] Accordingly, without additional interpretations it is unclear what could be the meaning of a specific date in this field.

[51] http://dublincore.org/documents/dces/
[52] http://dublincore.org/documents/dces/
[53] http://dublincore.org/documents/dces/

Description - Description is a DCMI free-text account of the resource, and there is no specific information that could be extracted from it. Description should be submitted for indexing (both keyword indexing and concept indexing).

Format - Format field of the DCMI cannot be utilized due to the underlying ambiguity. DCMI defines this metadata field as follows: "The file format, physical medium, or dimensions of the resource."[54] Accordingly, without additional interpretation it is unclear what could be the meaning of a specific value in this field.

Identifier - The DCMI metadata field Identifier is redundant with the internal DAAT identifiers.

Language - The value of the field is extracted as the object, or the value of the object in a statement formulated using the isInLanguage property. However, the property isInLanguage is an object property (relation) with the range defined as Language Types. Accordingly, the extracted DCMI value must be matched to the appropriate Language Types concept.

Publisher - DCMI metadata field Publisher is out of the scope of the DAAT model for semantic annotation. Nevertheless, under certain assumptions value of that metadata field could be extracted interpreted as the copyright owner (see below).

Relation - DCMI metadata field Relation corresponds to the entire set of properties of the DAAT concept ***Relationship***. Accordingly, the use of the value of this field requires an additional step besides extraction: user must be prompted to select the most appropriate property for the extracted value from the DAAT ***Relationship*** property set.

Rights - Assuming that the DCMI metadata field Rights holds as value a Creative Commons License[55], the value of the field can be extracted, and suggested as the object in a statement formulated using the hasCCLicense property. However, the property hasCCLicense is an object property (relation) with the range specified as ***creativeCommonsLicense***. Accordingly, the extracted DCMI value must be matched to the appropriate ***creativeCommonsLicense*** concept. In addition, assuming that the DCMI metadata field Publisher holds the information regarding the legal owner of the copyright, the value of the field Publisher can be also extracted, and suggested as the object value (data type String) in a statement formulated using the hasOwner data type property.

Source - DCMI metadata field Source corresponds to almost entire set of properties of the DAAT concept ***Relationship***. Accordingly, the use of the value of this field requires an additional step besides extraction: user must be prompted to select the most appropriate property for the extracted value from the DAAT ***Relationship*** property set.

Subject - DCMI defines this metadata field as follows: "The topic of the resource." Assuming that this field contains keywords (and further assuming that they are consistently delimited), the value of this field should be extracted, parsed, and each constituent should be matched to the concepts in the domain ontology. If match is found, the matched concept from the domain ontology could be suggested, in order to formulate annotation statements. Howevere, the use of the value of this field requires an additional step: user must be prompted to select the most appropriate property for the extracted value from the DAAT ***Content*** property set.
However, if the above assumptions do not hold, the value of the metadata field Subject should be submitted for indexing (both keyword indexing and concept indexing).

[54] http://dublincore.org/documents/dces/
[55] http://creativecommons.org/about/license/

Title - The value of the field is extracted as the value of the object (data type String) in a statement formulated using the hasTitle data type property.

Type - DCMI defines this metadata field as follows: "The nature or genre of the resource." In principle this metadata field is out of scope of the DAAT semantic annotation model. However, assuming that this field contains controlled vocabulary, the value of this field should be extracted, parsed, and each constituent should be matched to the concepts in the domain ontology. If match is found, the matched concept from the domain ontology could be suggested, in order to formulate annotation statements. However, the use of the value of this field requires an additional step: user must be prompted to select the most appropriate property for the extracted value from the DAAT ***Content*** property set.
However, if the above assumptions do not hold, the value of the metadata field Type should be submitted for indexing (both keyword indexing and concept indexing).

Domain Ontology Class Types

The following class types are expected in specific domain ontology:

Person is an entity that is a single human being (as contrasted with a social group or institution). Example: Alexander the Great.

Group is a number of individuals assembled together or having some unifying relationship. – The intention here is a social group, human community, movement, organization or institution. Example: the Luddites.

Object is something material and physical that may be perceived by the senses (thus excluding abstract objects). Object can be human-made artifact, or natural object. Example: PC.

Location Reference is any kind of spatial reference. – The intention here is position or site marked by some distinguishing feature, such as geographical name. Example: the Great Wall of China.

Time Reference is a point in time, or a portion of time, interval. The intention here is primarily period, a chronological division, typically a division of geologic time, or a stage of culture having a definable place in time and space. Example: Renaissance.

Event is something that happens, an occurrence. The intention here is primarily a noteworthy happening, a historic occasion. Example: the Battle of Hastings.

Abstract Concept is a transcendent entity that is a real pattern of which existing things are imperfect representations, an idea, something conceived in the mind, or an abstract or generic idea generalized from particular instances. Example: aleph-null.

2.2. Cultural Heritage Ontology

MOSAICA Ontology is a high-level ontology of the cultural heritage domain in general. Accordingly, it provides the basic concepts, relations and properties used to describe and specify cultural heritage content. These high-level concepts, relations and properties are then partially contextualized specifically to the Jewish cultural heritage (for example, Jewish religious professions, such as Rabbi, Cantor, etc. were included as subclasses of the generic class 'Clergyman'), and are expected to be further contextualized in other application domains. This

core ontology provides the semantics and the conceptual framework for development of large cultural heritage knowledgebases.
However, in addition to providing the infrastructure for a knowledgebase, MOSAICA Ontology must also provide the interface for navigation and browsing, and is thus designed to serve also as a directory for cataloguing cultural heritage content. MOSAICA Ontology is exposed to the user through the Semantic Browsing user interface as an alternative to searching by querying. Accordingly, it is expected to provide a simple and easily conceivable structure. MOSAICA thus exposes 7 general top-level topics:

- People
- Things
- Places
- Events
- Periods
- Notions
- Resources

It is evident that these general top-level topics roughly correspond to the domain ontology class types defined in the DAAT Framework (see Domain Ontology Class Types), as depicted in Table 2 below.

DAAT Framework	MOSAICA
Person	People
Group	People
Object	Things, Resources
Location Reference	Places
Time Reference	Periods
Event	Events
Abstract Concept	Notions, Resources

Table 2 Correspondence of MOSAICA top-level topics to DAAT Framework

Accordingly, range specifications for the DAAT annotation relations were provided in the Cultural Heritage Ontology (the domain specifications are provided in the DAAT Annotation Ontology using subclasses of the class *resource*). As detailed in 3 Semantic Annotator below, these domain and range specifications are utilized in the annotation process, in order to formulate appropriate annotation assertions.

Jewish Cultural Heritage Ontology is merely a light extension of the basic Cultural Heritage Ontology containing only additional, context specific concepts (for example, Jewish religious professions) and properties (for example, specifying dates in the Hebrew calendar).

2.3. Specific Ontologies

2.3.1. GIS/TIS

Different national and international organizations regulate the development of standards for Geographical Information (GI) metadata schemas. The most extended in USA is CSDGM[56] (Content Standards for Digital Geospatial Metadata) developed by the American FGDC[57] (Federal Geographic Data Committee). In Europe, Technical Committee 287 of CEN developed a voluntary norm ENV 12657[58] related to metadata. Technical Committee 211 of ISO is also in the

[56] http://geology.usgs.gov/tools/metadata/standard/metadata.html
[57] http://geology.usgs.gov/tools/metadata/standard/fgdc.html
[58] http://www.eurogeographics.org/eng/04_gdd.asp

process of defining a set of specifications for geographic information interoperability, where ISO 19115 is the approved standard concerning metadata[59]. ISO standard covers the main elements of CEN norm ([Gouveia et al. 2001]), which has almost completely been replaced in Europe. Finally, the Open GeoSpatial Consortium has completely adopted ISO 19115 in their specifications.

These standards support the definition of a set of keywords defining the main topic of a dataset. In the case of ISO/TC211 specifications, data schemas are included in other document, 19109, called *Rules for application schemas*[60]. In the case of CSDGM the description of the application schema (entities, attributes and values) is covered in the own standard.

Nevertheless, these schema definitions provided by ISO and FGDC standards do not consider semantics. In particular, they do not take into account possible relationships between different values used by different dataset providers. For instance, two land cover datasets may be structured according to different thematic categories; how they are related is not covered by the standards. Referencing these values to normalized vocabularies is an attempt to solve this problem. – A thesaurus is a hierarchically structured and controlled set of terms. Terms comprise one or more words and are organized through a set of standardized reciprocal relations among them. ISO 2788:1986 provides an international norm for the definition of mono-lingual thesauri[61], while ISO 5964:1985 covers multi-lingual thesauri[62]. These standards define a set of relations that can be applied to the GI datasets: broader term (BT), narrower term (NT), top term (TT), related term (RT) use-instead (USE) and used-for (UF). These relations are normalized by the ISO standard, and are present in all thesauri (although some include additional relations, for instance, the GEMET thesaurus for environmental terms[63] also includes *GROUP* and *THEME* relations, which provide two classifications of terms in 30 groups and 40 general themes, to facilitate defining sets of the thesaurus according to particular interests).

In the case of geographic information, there are no global shared vocabularies, but there are some initiatives that have defined different normalized hierarchical vocabularies of terms for specific domains. Hereby the land cover/land use is probably the most studied one. The CORINE project[64], which defines a vocabulary consisting of a hierarchy of three levels with 5, 15 and 44 items respectively ([Bossard et al. 2000]) was used in the production of a dataset for land-cover in the EU at a scale of 1:100,000, considering areas of at least 25 hectares. The terms included in the vocabulary are oriented to that scale and minimum size of area, and do not fit properly other datasets at more detailed scales. In USA, the equivalent to CORINE is the land cover/land use classification used by the United States Geology Survey[65], which is an adaptation of the Anderson vocabulary ([Anderson et al. 1976]). Other well-known vocabularies have been defined by other organizations, such as the vocabulary used in the framework of the International Geosphere Biosphere Programme (IGBP)[66].

Related to these vocabularies is the Land Cover Classification System (LCCS) ([Di Gregorio and Jansen 1998], [Di Gregorio 2005]) developed by UN's FAO. It permits the definition of land cover classes in terms of independent variables (or classifiers). LCCS has been used in the framework of developing land cover maps of several African countries. However, LCCS does not provide a normalized vocabulary of land cover classes, but a system to create new vocabularies, specifying how each vocabulary class is determined by different classifiers.

---

[59] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26020
[60] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39891
[61] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=7776
[62] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=12159
[63] http://www.eionet.europa.eu/gemet
[64] http://nfp-bg.eionet.eu.int/ncesd/eng/clc/agree1.html
[65] http://www.usgs.gov/
[66] http://www.igbp.kva.se/index.html

The Spatial Data Transfer Standard, SDTS[67] (ANSI 1998), is a format for the exchange of geographical data standardized by ANSI. Its *Spatial Features* subset includes a vocabulary with 200 entities and 1200 alternative terms, focusing on topographic and hydrographic maps. However, it covers a reduced domain that is mainly oriented to vector maps, and does not contain concepts for thematic maps. Related initiatives are two big thesauri of geographical terms in the context of world-wide gazetteers: Getty thesaurus of geographic names[68] and the Gazetteer Content Standard from the Alexandria Digital Library[69].
However, the integration of geographic information has a significant semantic component that cannot be addressed only with metadata standards and thesauri. A semantic framework representing the thematic component of geographic information is needed to facilitate sharing information. This semantic framework is usually expressed through ontologies since they provide explicit formal definitions of thematic concepts and their relations in datasets, and thus facilitates the definition of services for translating or integrating geographic information from different sources.
The importance of ontologies as a tool for semantic interoperability has motivated a long list of research activities in the last years, from very different perspectives. Frank affirms in [Frank 1997] that no single geographic ontology can capture all the aspects of reality. Instead, small ontologies can be built for specific aspects of physical, cognitive, administrative or legal reality. These "small theories" have to be formally described in a way that makes it possible to combine them with other similar "small theories".
Frank also suggests in [Frank 2001] and [Frank 2003] that GI should be structured in a 5-tier ontology. These five tiers are:

- Tier 0: human-independent reality (the physical world),
- Tier 1: observation of physical world,
- Tier 2: objects and properties,
- Tier 3: social reality, and
- Tier 4: subjective knowledge.

Tier 0 describes the physical world through four-dimensional (3D space + time) continuous fields of attribute values, assuming that only one single physical world exists. Tier 1 contains the results of observing tier 0, assuming that observations of reality are limited. This tier provides measured values of properties at concrete locations in time and space. Tier 2 divides the observed world into objects with properties. Tier 3 obtains, from these objects representing the physical world, new objects according to social constructs. For instance, country borders, a natural park or building zones are objects that are valid only within a social context. Finally, tier 4 is the subjective view of cognitive agents (persons and organizations). These agents have incomplete and partial knowledge of reality, but they use this knowledge to deduce other facts and take decisions.
From this analysis it is immediately evident that the tiers that have any relevance in the context of a cultural heritage GIS are only 3 and for 4, with maybe some aspects of the tier 2, which makes the adoption of the existing GIS ontologies very difficult, if feasible at all.
The problems arising from the adoption of the existing GIS ontologies for the domain of cultural heritage can be summarized into three groups:

1. Inaccurate (historical) maps,
2. Complex conceptual model, and
3. No clear reference.

The following is a brief overview of these problems.

[67] http://mcmcweb.er.usgs.gov/sdts/
[68] http://www.getty.edu/research/conducting_research/vocabularies/tgn
[69] http://www.alexandria.ucsb.edu/gazetteer/gaz_content_standard.html

2.3.1.1. Inaccurate (Historical) Maps

Cultural heritage domain must make use of the historical maps that are notoriously inaccurate, as graphically depicted in Figure 7.

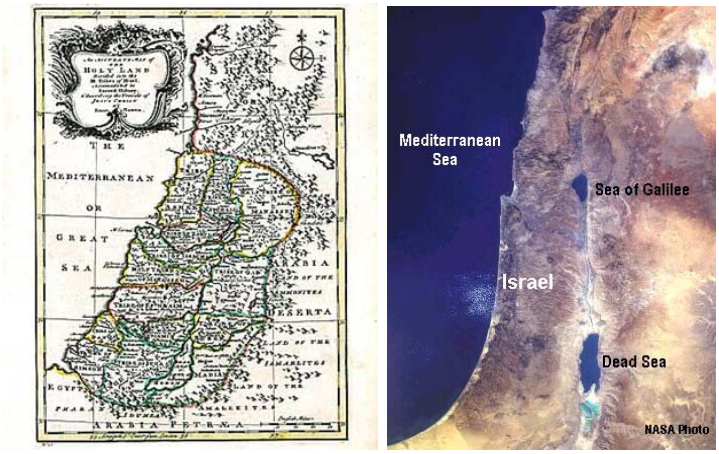

Figure 7 Example of inaccuracy: 1767 Map of the Holy Land Palestine (by Emanuel Bowen) and a satellite image of Israel

The coordinate system in use today cannot be accurately superimposed on such maps, even if they are not completely imaginary.

2.3.1.2. Complex Conceptual Model

Mapping of the cultural heritage concepts – even if those with clear spatial and geographical references – may be more complex than what it appears at the first glance. The following are some problems encountered when attempting such mapping:

- **Physical ambiguity** – often the references to physical entities vary significantly from one source to another. For example, the borders of the "Land of Israel" are defined in different books of the Hebrew Bible with significant differences, as depicted in Figure 8.

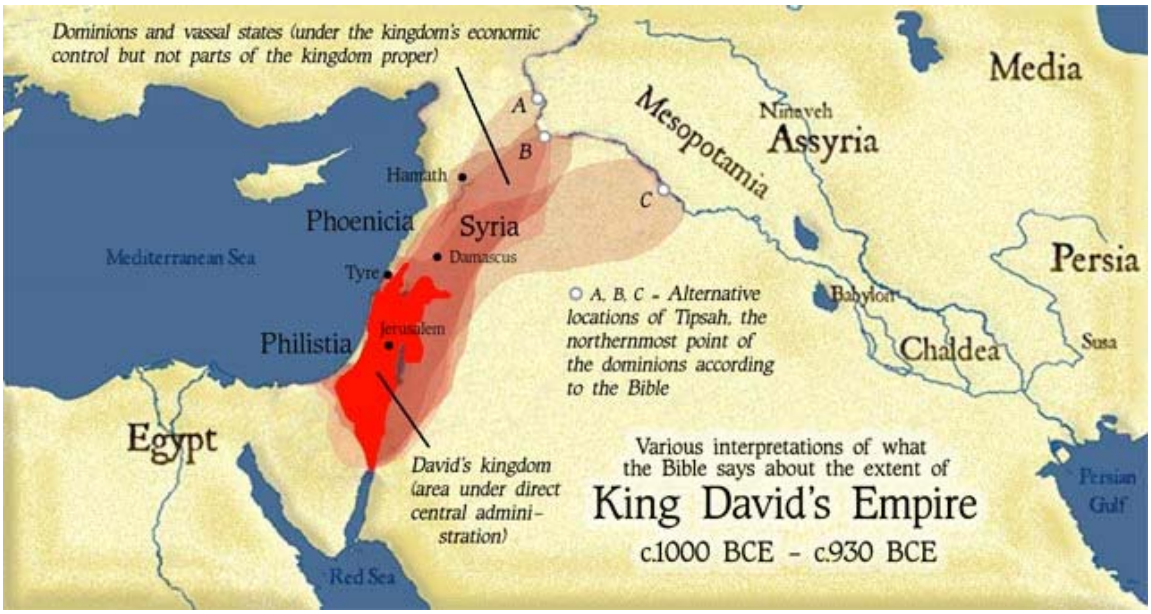

Figure 8 Various definitions of the "Land of Israel"

These differences are in particular stressed when comparing, for example, archeological vs. Biblical resources: the area defined as Canaan during the period of the so-called "Tribal Union" covers significantly larger areas than what the current archeological findings confirm. However, in the future this may change due to possible new archeological findings.

- **Historical ambiguity** – the political borders – and considering the time span of cultural heritage traditions such as Judaism, even the geophysical properties - of geographical entities vary significantly, as graphically depicted on the example of historical "Poland" in Figure 9.

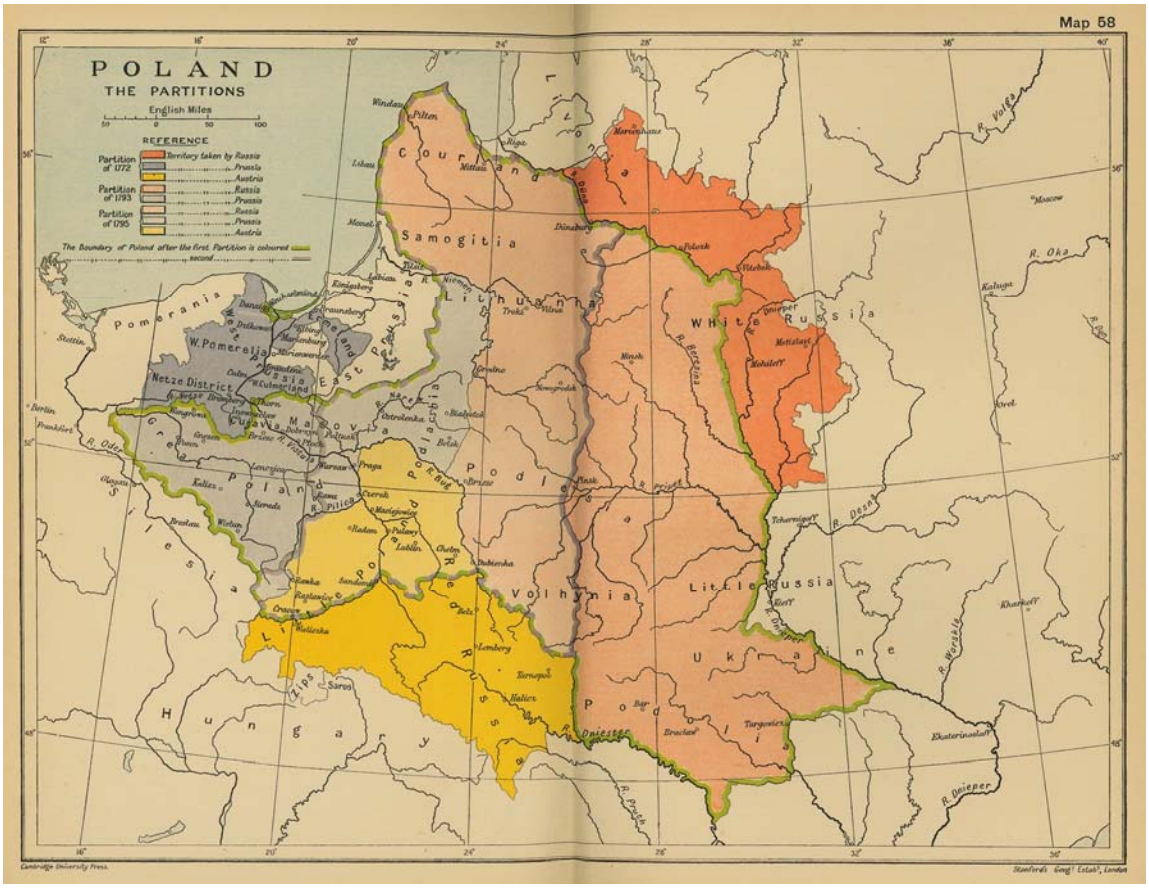

Figure 9 Poland through history – a changing geographical reference

- **Conceptual ambiguity** – often the geographical terminology is used in the context of the cultural heritage in an ambiguous way. The terms such as "Assyria" or "Babylonia" may refer to a geographical region (the northern and the southern part of Mesopotamia respectively), corresponding empires ("Assyria" from the 10th to the 7th Century BCE and "Babylonia" fro the 7th to the 6th Century BCE), or even the epochs in the corresponding historical timeline. Accordingly, it would be extremely difficult to define these terms in a gazetteer – abstracting from the specific context of discourse.
- **Abstract notions** – cultural heritage often applies abstract notions without clear physical reference. For example, the so called "Fertile Crescent" is defined as an area of a crescent shape in the Middle East encompassing all geographical areas considered as the "cradle of the civilization", as graphically depicted in Figure 10. Although from the definition it is clear that this concept includes Ancient Egypt, Canaan, Phoenicia, Mesopotamia, etc., it is less clear what should be the physical coordinates of that concept, since the borders of its constituent areas are simply not clearly defined.

Figure 10 "Fertile Crescent"

- **Political sensitivity** – often cultural heritage geographical concepts are subject to political discussions and struggles. The actual reference of the terms like "Palestine" and "Israel" is long disputed due to its political sensitivity in the context of the Arab-Israeli conflict. Extreme views on both sides deny even the existence of one of these concepts, while moderate views on both sides tend to identify them as partially or completely overlapping.

2.3.1.3. No Clear Reference

The above conceptual problems are further severed by the implied semantics:

- **No naming standards** – the domain of cultural heritage is notorious for the variety of alternative names applied to references. For example, both "Nablus" and "Shechem" refer to the same ancient city[70]. This is further complicated by different transliterations, pronunciation in different ancient languages, and the Ancient Greek equivalents introduced by the historians of the classic age.
- **Unknown reference** – Geographical locations found in the sources are often disputed by the modern scholars. For example, "Elkosh" mentioned only once in the Hebrew Bible is identified either with a town in the modern-day Iraq, or with two alternative locations in the Galilee in the modern-day Israel (one of them being Capernaum).
- **Differing historical references** – geographical locations may change in the course of history despite the continuity of events. The ancient Jerusalem only partially overlaps the modern-day Jerusalem, and Mosul in the modern-day Iraq that is considered to be the successor of the ancient Nineveh is in fact on the opposite bank of the Tigris river.
- **Uncertain references** – in the domain of the cultural heritage the reference to the physical locations may significantly differ even when there is a general consensus regarding the geographical contextualization. For example, there is a clear consensus regarding the location of the so-called "the First Temple" and "the Second Temple" in Jerusalem, an even on the Mount Moriah. Nevertheless, the exact location of the temples is disputed, and in terms of topographical location significantly differing speculations have been raised[71].

2.3.1.4. Summary

Considering the above complexity of the conceptual GIS modeling in general, and in the domain of cultural heritage in particular, it is not surprising that significant theoretical work was dedicated to the knowledge representation aspects, analyzing the problems of modeling geographic information through concepts and relations in ontologies. This often includes philosophical discussions on how space and geographic objects are perceived, and thus better represented. For example, [Agarwal 2005] provides an exhaustive review of some of the different viewpoints and objectives of GI ontology modeling. However, of particular interest is the so-called *Naïve Geography* [Egenhofer and Mark 1995], which studies the common-sense vision of the geographic world that is how non-expert people think and reason about geographic space and time. This has motivated a line of research that tries to investigate ontological considerations in how people understand and categorize concepts (geographic kinds in their terminology) related to geography ([Smith and Mark 1998]). In this work, they also discuss the particularities of the geographical domain that make it ontologically distinct from non-geographical domains. In [Smith and Mark 1999], [Mark et al. 2001] and [Smith and Mark 2001] the authors conduct different experiments with non-expert subjects to observe how they classify different concepts, and how these concepts are related to the top-level concepts as for instance "geographic features" or "something that could be portrayed on a map". This research clearly points out that if a sound GIS semantic framework is needed in the domain of the cultural heritage, an approach significantly different from the available GI ontologies is required. It is evident that the concept of Semantic Reference System as defined in [Kuhn 2003] cannot serve as the conceptual foundation for the GIS systems in the domain of cultural heritage. The spatial component of geographic data is represented there through geographic coordinates, which are referred to a spatial reference system that specifies the geodetic datum, map projection and coordinate system. Even if the methods for transforming data from one spatial reference system to another have been largely studied, and the transformation methods between temporal

[70] Shechem (Sichem, Shkhem or Shachmu, Hebrew: שְׁכֶם/ שְׁכֶם, Standard Šəḫem Tiberian Šəḵem; "Shoulder", modern Tell Balatah Israel, 2 km east of present-day Nablus).
[71] http://en.wikipedia.org/wiki/Solomon's_Temple#Location

reference systems exist (for instance for transforming from one calendar to another), most of the problems related to the cultural heritage domain still prevail.
A significant impact on the design of the MOSAICA GIS Ontology was taken from the BUSTER ([Visser et al. 2001]) – a complete system developed at the University of Bremen to integrate and query heterogeneous information from different geospatial datasets. It supports queries of type concept@location and provides integration services at the syntactic, structural and semantic levels. Focusing on the semantic level, they define thematic and spatial knowledge in a dataset through what they call the Comprehensive Source Description, CSD ([Visser and Schuster 2002], [Visser et al. 2002a] and [Visser et al. 2002b]). The spatial description consists of references to a gazetteer, such as Getty Thesaurus of Geographic Names. The thematic information is described through a terminological ontology based on references to some standardized vocabularies as WordNet, UpperCyc Ontology that includes about 3,000 terms of the most general concepts, GEMET that includes more than 5,000 terms on environmental disciplines, as well as other taxonomies on scientific knowledge, like for instance the Google Web Directory that comprises a classification of plants. This way, the merging phase ([Schuster and Stuckenschmidt 2001]) consists of an ontology expert that relates, in several steps, each thematic concept in a dataset with one or more terms of the standard vocabularies. As the usual result, a new class is created for the thematic concept in the terminological ontology, and is related to the standard taxonomies through axioms of type subclass or property value restriction.
Nevertheless, there are two significant theoretical differences introduced in MOSAICA, in order to adapt the idea behind BUSTER to the domain of the cultural heritage:

1. The spatial description cannot consist of references to a gazetteer, since to the best of our knowledge no suitable gazetteer exists, and considering the difficulties detailed above, it is doubtful if a standard gazetteer is feasible at all in the domain of the cultural heritage.
2. Furthermore, no standardized vocabularies are available in the domain of the cultural heritage, and it is arguable if such vocabularies are feasible at all[72]. Consequently, MOSAICA GIS ontology utilizes the MOSAICA knowledgebase as such standard vocabulary.

Actually, MOSAICA's Cultural Heritage Ontology, Jewish Cultural Heritage Ontology, and the corresponding knowledgebase are designed to serve as such specific gazetteers and vocabularies. However, the spatial referencing systems typically used in gazetteers were replaced by location references specified for each particular map separately (by transforming these maps into a simple two-dimensional coordinate system). This approach to the spatial referencing was incorporated into the MOSAICA's semantic annotation framework, and MOSAICA Semantic Annotator was extended with a specific user interface that enables map-specific spatial referencing. Accordingly, the reference to an identical geographical location would not be considered physical (i.e. identical coordinates), but rather semantic (i.e. identical cultural heritage notion).

[72] According to our anecdotic experience the introduction of the terminology generally accepted in the English-, German- and Hebrew-speaking academic circles, such as "The Northern Kingdom" (referring to the Kingdom of Israel after the split of the unified monarchy), and "The Southern Kingdom" (referring to the Kingdom of Judah), turned to be confusing in French (sic!). This terminology was originally introduced, in order to resolve the ambiguity arising form the terms "Israel" (which also refers to a person, confederation of tribes, unified monarchy, geographical area, Jewish nation and a modern state) and "Judah" (which also refers to several historical personalities, one of the Twelve Tribes, a Roman province, etc.) used in the Hebrew Bible and frequently occurring in the archeological findings.

2.3.1.5. MOSAICA GIS Ontology

Map is the top-level concept in the GIS framework. It is defined in the annotation ontology as a specific resource type (specifically in MOSAICA, it is a subclass of the class *graphic resource*). It describes a real map provided in a digital form, and its potential displays. Specific map is represented as an individual concept of the type Map, compiling all available information, and referencing the graphic file containing the actual map. It is specified by the following XML tags:

- <ID> – QName of the individual concept representing the map.
- <P2PRefToPicture> – ID used by the distributed content management system to identify the graphic file containing the actual map.
- <hasCaption> – Map's label in a desired language. It can be used as the caption when displaying the map.
- <depictsContent> – Description of map's content using references to the concepts in the domain ontology. In MOSAICA these references are to one or more of the following classes:
    - Location – Specifies that this is a map of a particular location identified by its geographical reference (for example, map of Poland).
    - Site – Specifies that this is a map of a particular site identified by its name (rather than its location – for example, map of The Western Wall).
    - Event – Specifies that this is a map of an event (for example, map of the Battle for Stalingrad).
    - Period – Specifies that this is a map depicting a period (for example, map of the Second Temple Period).
    - Artifact – Specifies that this map is a cultural artifact (for example, 1695 map of France by Nicolas Sanson d'Abbeville).
- <isDated> - Description of the temporal origin of the map. It can be expressed as one of the following:
    - Exact date (for example, 1695),
    - Approximate date (for example, around 1695),
    - Range of dates (for example, between 1660 and 1700), or
    - Reference to a historical period (for example, Middle Ages).
- <hasTemporalExtent> - Description of the temporal extent of the map's content. Designed to cater for the situation when the map originates from a later period (for example, a contemporary map), but depicts an earlier period (for example, a contemporary map of the Early Roman Empire). It can be expressed as one of the following:
    - Exact date (for example, presents France in 1695),
    - Approximate date (for example, presents France around 1695),
    - Range of dates (for example, presents France between 1660 and 1700), or
    - Reference to a historical period (for example, presents France in the Middle Ages).
- <content> - Description of the items displayed on the map. These items can be of two types:

- Point – Non-dimensional object on a map, specified by x and y coordinates, and marked by a symbol (a simple geometric form) or an icon (from a graphic file), and possibly also an accompanying label. Point is described in the following terms:
  - ID – QName of the corresponding individual concept.
  - isOnMap – ID of the map on which the point is to be displayed.
  - hasXAxisValue, hasYAxisValue – coordinates at which the point is to be displayed.
  - isOfGeometricFormType – a basic set of simple geometric forms to display point as a symbol on the given coordinates (include circle, square, cross and plus). It could be set to a custom value, in which case the property hasCustomForm holds ID used by the distributed content management system to identify the graphic file containing the icon that is to be displayed.
  - hasRadius – determines the size of the displayed symbol or icon. This property allows displaying symbols of various sizes on the same map (for example, in order to distinguish between different types of objects or locations – such as towns, villages, etc. – displayed on the map).
  - Color – allows specification of the symbol's color as red/blue/green values (it is ignored if an icon imported from a graphic file is used instead). This property allows displaying symbols of various colors on the same map (for example, in order to distinguish between different types of objects or locations – such as ancient and contemporary settlements – displayed on the map).
  - isRepresentedAsIcon – allows specification of an external graphic file that will be used to display the point on the map.
  - Label – displays the name of the object marked by the point in the desired language. In addition to the primary label, alternative labels can be also specified. This feature caters for various spellings, pronunciation varieties, modalities, etc. that are often encountered in the geographical context. Primary and alternative labels can be specified separately for each language. Nevertheless, whether labels will be specified is determined by the property labelRequiresActionOfType. This feature enables to hide labels on the maps that already contain marked and labeled points.
  - Text – free text description of the point.

  Point is associated with a **single** concept in the domain ontology by the relation Represents. Point can be associated only with a location or a site (i.e. with a concept denoting a physical location).
- Display – An area on a map defined by a centroid (specified by x and y coordinates) and a radius. Display has two functionalities: 1) to specify an area of the map by drawing borders around it, and 2) to specify zoom window coordinates (in which case the borders are not drawn, but the map is simply zoomed to the specified area). Point is described in the following terms:
  - ID – QName of the corresponding individual concept.
  - isOnMap – ID of the map on which the display is marked, or on which

the zooming will take place.

- hasXAxisValue, hasYAxisValue – coordinates of the centroid around which the display is centered.
- hasRadius – determines the size of the display. Radius is to be calculated as the diagonal distance from the centroid to the corners of either drawn rectangular (display as area marked by borders) or of the monitor (zooming operation).
- hasLineOfType – specifies the type of the display border (includes solid, dash, dot, dash-dot and dash-dot-dot), if display will be marked by borders.
- Color – allows specification of the borders' color as red/blue/green values.
- Label – displays the name of the area marked by the display in the desired language. In addition to the primary label, alternative labels can be also specified. This feature caters for various spellings, pronunciation varieties, modalities, etc. often encountered in the geographical context. Primary and alternative labels can be specified separately for each language. Nevertheless, whether labels will be specified is determined by the property labelRequiresActionOfType. This feature enables to hide labels on the maps that already contain marked and labeled points.
- Text – free text description of the display.

Display can be associated with a **single** concept in the domain ontology by the relation Represents. Display can be associated only with a location or a site (i.e. with a concept denoting a physical location). Nevertheless, a display can be created ad hoc for zooming purposes (for example, in order to zoom on a particular point on a map). In that case, display is not associated with any particular concept in the domain ontology, but is simply used as technical means.

In addition, both point and display can have album items. Album is a strip containing images – typically photographs – that is displayed along side a map. Whenever a point or a marked display is selected on a map, the associated images can be displayed in the strip along side the map. User can scroll the images, enlarge them, and read associated text. Album item is recorded as an individual concept in the knowledgebase. An album item is specified by the following:

- <ID> – the QName of the individual concept.
- <hasText> – free text description of the graphic resource associated with the album item.
- <displays> – ID used by the distributed content management system to identify the graphic resource that is displayed in the strip.
- <isAtPosition> – position of an album item in the strip.

Special role is assigned to the concept of menu item. Menus are created dynamically from the semantic description of the location or site that is associated with a point or a display. Accordingly, menu items are not elements permanently recorded in the knowledgebase, but merely XML constructs used to communicate data inferred from the semantic descriptions. Manu item is specified by the following:

- <ID> – an arbitrary identifier assigned to a menu item.
- <hasCaption> – text that is displayed as the caption.

- <isAtPosition> – position of a menu item in the menu list.
- <isSubItemOf> – enables structuring of menus in a hierarchical way (submenus).

2.3.2. Multimedia

Annotation of multimedia resources is considered useful for making these resources accessible through applying the standard information retrieval techniques developed primarily for the textual resources to which the automatic keyword indexing could be applied. This *symbolic-based* approach (as opposed to the *signal-based* approach which considers content of a multimedia file to be the raw digital information) considers the explicit semantic interpretation of images. This approach is, for example, implemented in the classic InfoHarness system ([Shklar et al. 1994]) providing integrated and rapid access to huge amounts of heterogeneous information independent of its type, representation, and location. The content description of MPEG-7 ([Martínez 2001]) and the Dublin Core Metadata Initiative ([Weibel et al. 1998]) also fall into this category. Such approaches take into account the fact that when people describe images they use symbols, and not signal features ([Jörgensen 1998], [Hollink et al. 2004]).
Symbolic-based descriptions are able to manage complex representations ([Gupta et al. 1991], [Mechkour 1995]), and are evidently the preferred approach. Previous concerns regarding 1) the lack of scalability of the symbolic-based approach, 2) the fact that manual annotation is tedious, and 3) that it is subject to inconsistencies due to human intervention in the indexing process ([Rasmussen 1997]) were originally countered by attempts to provide the so-called "automatic image annotation"[73] by learning symbols from image features. However, the occurrence of the so-called "collaborative tagging" and the Web 2.0 modified significantly current views on semantic annotation. Collaborative tagging proved that by harnessing the power of the community it is highly effective in creating large amounts of metadata quickly ([Bateman et al. 2006]). This was made obvious with the raising popularity of Flickr (photo sharing) and Del.icio.us (bookmark sharing), both based on the free text tagging.
Nevertheless, it was also pointed in [Bateman et al. 2006] that these metadata are limited to human consumption only, which can be quickly overwhelmed. This clearly provides the necessary argument for the use of ontologies in the annotation of multimedia resources. Consequently, the classic Annotea project ([Kahan et al. 2001]) for Web annotation and the early multimedia annotator called CREAM ([Handschuh and Staab 2002]) were soon followed by more sophisticated M-OntoMat-Annotizer ([Bloehdorn et al. 2005]) and AKTive Media ([Chakravarthy et al. 2006]).
Regarding the annotation features, MOSAICA is certainly comparable to the state of the art multimedia semantic annotators. The major strength shared by all of them is the ability to annotate both a multimedia file in its entirety as well as its constituting segments. However, in contrast to other annotators, no efforts where made in MOSAICA towards the automation of the annotation process. This is not accidental: the major conclusion drawn from the popularity of the Web 2.0 is that it is not based on introduced high levels of automation, not even on any novel technologies introduced, but rather on providing a high level of usability that minimized the costs of development and maintenance of collective intelligence. Accordingly, the major effort was focused in MOSAICA on the usability issue with the intentions to make the annotation procedures as similar as possible to the Web 2.0 experience (and thus as painless as possible).

[73] For extensive bibliography on automatic image annotation see: http://en.wikipedia.org/wiki/Automatic_image_annotation.

2.4. Deployment

The MOSAICA ontologies deployment is centered around the DAAT framework. Accordingly, we shall initially present the DAAT Framework deployment.

2.4.1. DAAT Framework

DAAT Framework consists of two ontologies: DAAT Core Ontology and DAAT Annotation ontology. These two ontologies are coordinated in the sense that they complement each other – one of them providing navigational and management mechanisms (DAAT Core Ontology) and the other providing annotation mechanisms (DAAT Annotations).
However, these ontologies are generic and designed to support any knowledge management application. Therefore, it is expected that each of these ontologies will be supplemented at the application level by imposing additional, domain-specific layers over them. These additional layers are the Application Core Ontology and the Application Annotation Ontologies respectively. – Each of them imports the corresponding DAAT Framework ontologies and introduces additional concepts, relations and properties, domain and rage specifications, labels, etc.

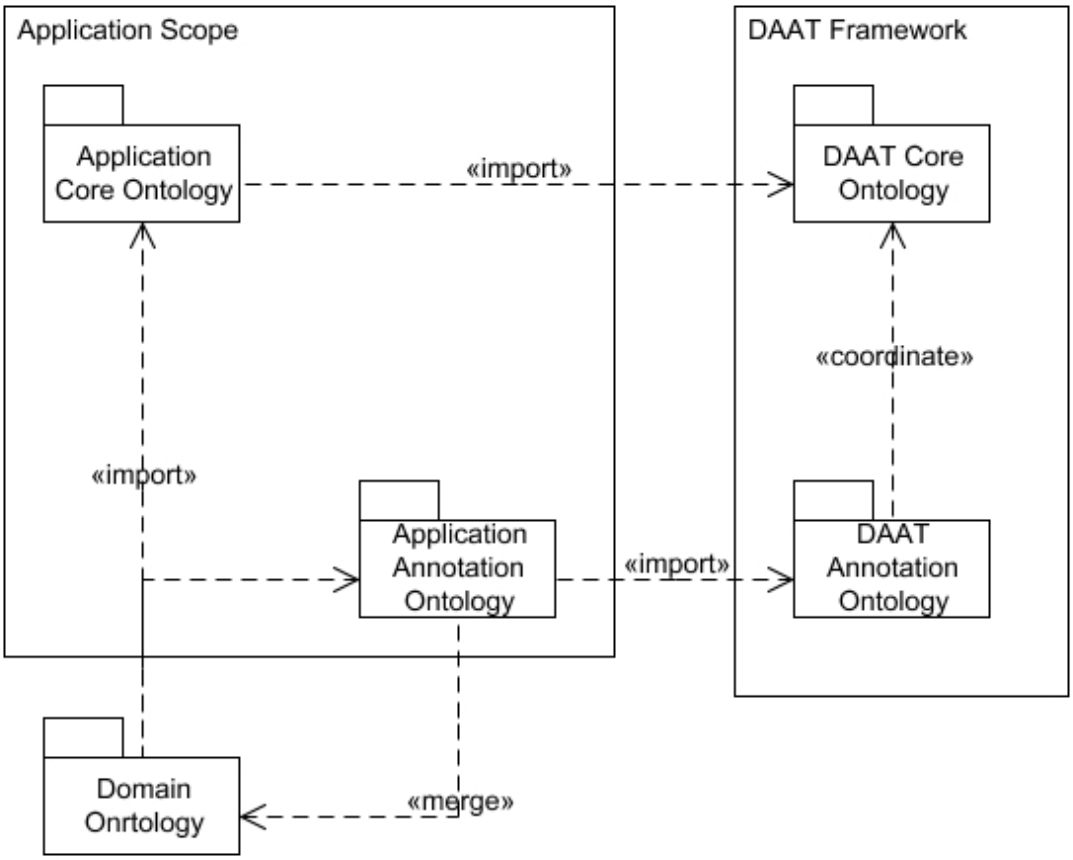

Figure 11 UML diagram depicting the intended DAAT Framework deployment

However, all these ontologies are utility systems providing the semantic infrastructure required for ontology-based navigation, search and annotation. The domain specific knowledge that will be used for the annotation of the content is expected to be provided in a separate system of

ontologies and knowledgebases. When deployed, such system is expected to import both the core ontologies and the annotation ontologies, as schematically depicted in Figure 11 above.

2.4.2. MOSAICA Deployment

Following the deployment structure detailed in the previous chapter, MOSAICA provided the MOSAICA Core Ontology and the MOSAICA Annotation Ontology supplementing the DAAT Framework. However, regarding the domain ontology holding the knowledge required for annotation, the situation in MOSAICA is more complex. MOSAICA is devised as a generic cultural heritage semantic framework. Accordingly, at the heart of MOSAICA is generic cultural heritage ontology, which can be then extended with specific cultural heritage domains. For example, in the MOSAICA project the Jewish cultural heritage was selected for the demonstration of the proof of concept.
Accordingly, it is the generic Cultural Heritage Ontology that imports both the application core and the application annotation ontology, but it is further imported by the more specific Jewish Cultural Heritage Ontology that further extends it, as schematically depicted in Figure 12 below.

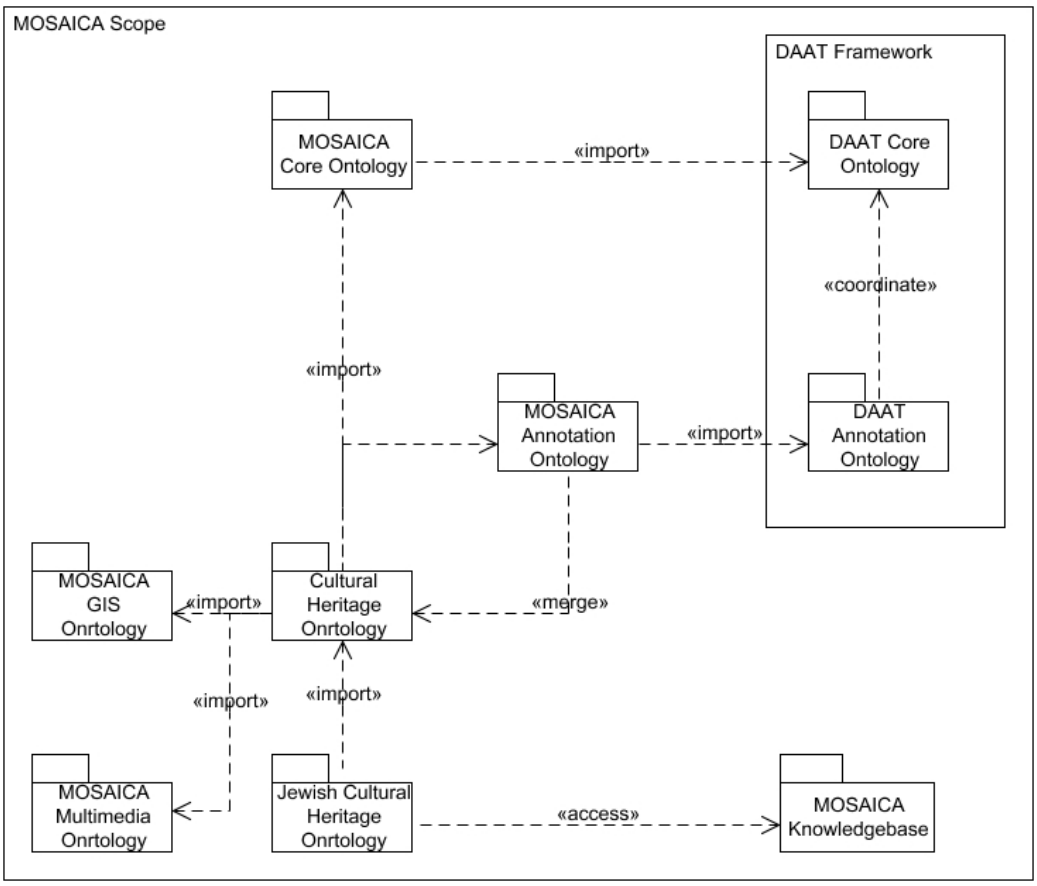

Figure 12 MOSAICA complete system ontologies deployment

It may be puzzling that the GIS/TIS and multimedia annotation ontologies were defined at the level of application, rather than at a more generic level. The reason for that design decision was provided in one of the previous chapters (2.3.1 GIS/TIS). There we argue that due to the specific problems encountered in the domain of cultural heritage it was impossible to adopt and

reuse existing gazetteers, GI ontologies and vocabularies. Accordingly, we were forced to develop a domain specific GI ontology that will allow us annotation of maps and construction of semantic layers on these maps. However, such situation is not necessarily expected in additional potential domains of the DAAT Framework, and it can be well envisioned that in other domains reuse of existing annotation resources will be feasible. Accordingly, the decision how to integrate additional, specific annotation ontologies into the system should be made at the application level, rather than at a more global level.

3. SEMANTIC ANNOTATOR

There are many semantic annotators available today (for a survey of the state of the art semantic annotators, consult [Uren et al. 2006]), some of them providing an impressive range of capabilities. However, MOSAICA undertook developing a proprietary Semantic Annotator. The reason for this decision was the experience gained in the use of the Web 2.0 in the past 3 years. Web 2.0 made tagging of resources enormously popular. As pointed out in 2.1.3 Web 2.0 above, the popularity of tagging was a result of the low effort required for it. This high usability encountered in social tagging served as the blueprint for the MOSAICA Semantic Annotator.
To that purpose, MOSAICA Annotator introduced several innovations:

1. **The minimal set of mandatory information** was defined, allowing it to be extended with additional information, if requested by a user. The minimal information set includes:
   - URL of the annotated resource (provided as the first step in annotation procedure),
   - Title (provided manually),
   - Author of annotation (automatically extracted from the user's identity),
   - Date of annotation (automatically added from the system's clock, but with an option to manually modify it),
   - Resource type (manually added in the second step of the annotation procedure),
   - At least one associated concept from the domain ontology (manually selected), and
   - Functionality ranking (also manually specified).
2. **Introduction of automation** wherever possible. From above it is evident that 3 out of 7 mandatory elements are provided automatically. In addition, specification of the resource type automatically determines which properties (and accordingly, also which concepts) can be used for annotation (by analyzing the domain and range of the annotation relations). Finally, depending on the selected annotation concept, the system will attempt to automatically select the matching relation, in order to complete the annotation assertion. Only in case that the selection was ambiguous, the user will be prompted to select the annotation relation manually.
3. **Introduction of smart auto-completion textboxes** that allow users to retrieve desired annotation concepts even if their location in the hierarchy is unknown. The auto-completion was designed that way that even the user cannot recall the exact formulation of the desired concept, a partial clue will be typically sufficient, in order to retrieve the intended concept.
4. **The design of the annotation procedure** followed the current tagging standards encountered in the Web 2.0. The annotation assertions are not formulated by selecting first predicate, and then object (depending on the selection of the predicate). Instead, the list of available concepts is displayed as if they were simple tags (also display of concepts in a tag cloud is provided, in order to make the experience even more similar to the Web 2.0). After the selection of a concept, the user is prompted to specify the annotation relation that will be used as a predicate in the assertion, only if there is an ambiguity detected in the analysis of the domain and range specifications of the selected annotation relation.
5. Moreover, the annotation procedure was not designed as a multi-step wizard. Instead, **annotation elements were grouped into four simple constituent groups**, allowing the user to select which of the elements will be populated with data. For some resource types, additional constituents were provided. For example, maps allow annotation of specific locations (defined as points or areas on a map). Nevertheless, a

user may decide to provide only the most basic annotation for a map (the minimal set of mandatory information), in order to speed up the annotation process.

The following is a brief description of the annotation constituents, and demonstration of the annotation user experience on screenshots.

3.1. Major Constituents

Major constituents are exposed in a menu displayed in the annotation main window, as depicted in Figure 13.

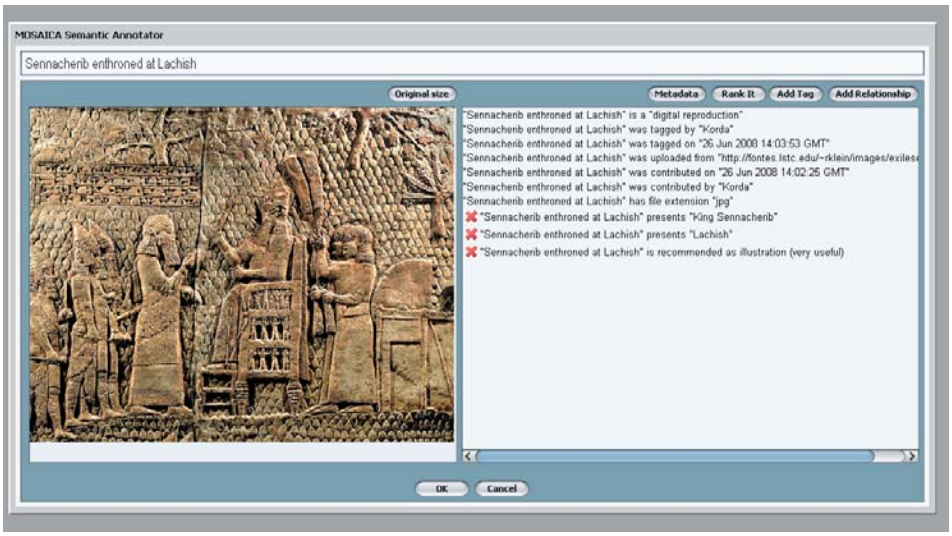

Figure 13 Main annotation window

3.1.1. Metadata

Metadata contains the information regarding (as depicted in Figure 14):

1. The resource as an entity (URL, Title, type), and
2. The annotation as an entity (author, annotation date).

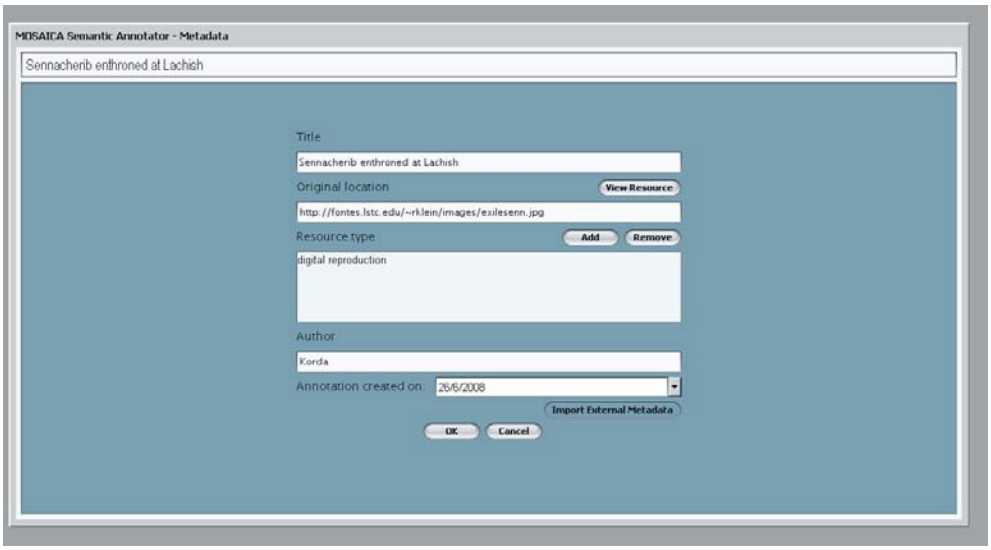

Figure 14 Metadata editing dialog box

3.1.2. Ranking

Ranking contains information regarding the user's assessment of the resource's potential use, and its usefulness for that particular use (as depicted in Figure 15):

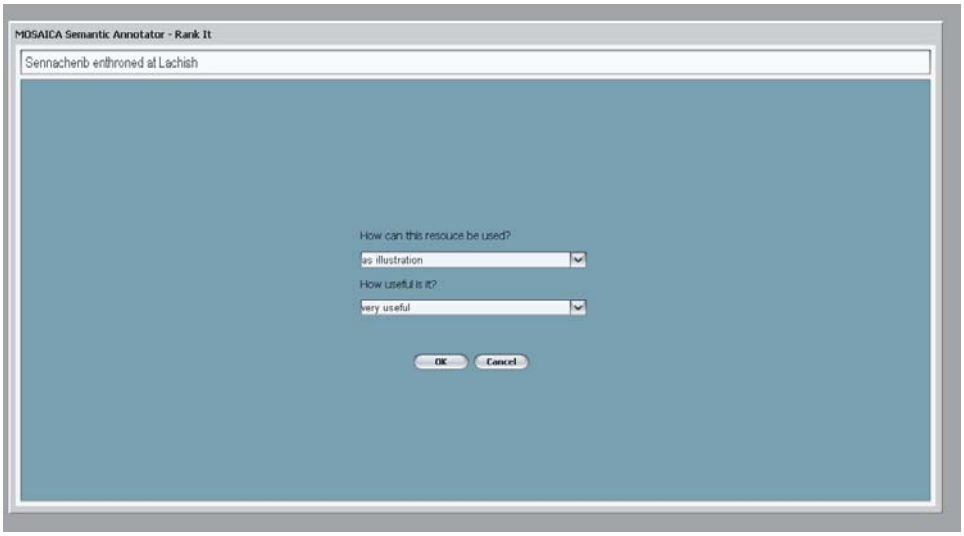

Figure 15 Dialog box for editing ranking of a resource

3.1.3. Tags

This constituent contains all semantic annotations associating the annotated resource with one or more concepts in the underlying domain ontologies, as depicted in Figure 16.

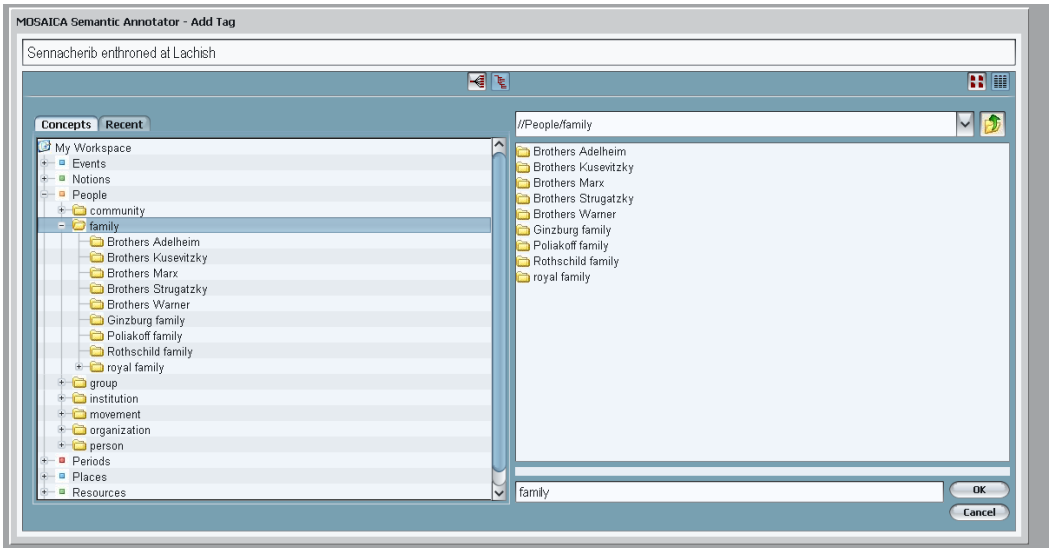

Figure 16 Dialog box for tagging resource with ontology concepts

3.1.4. Relationships

This constituent contains relationships established between the annotated resource and other resources (that were previously annotated), as depicted in Figure 17.

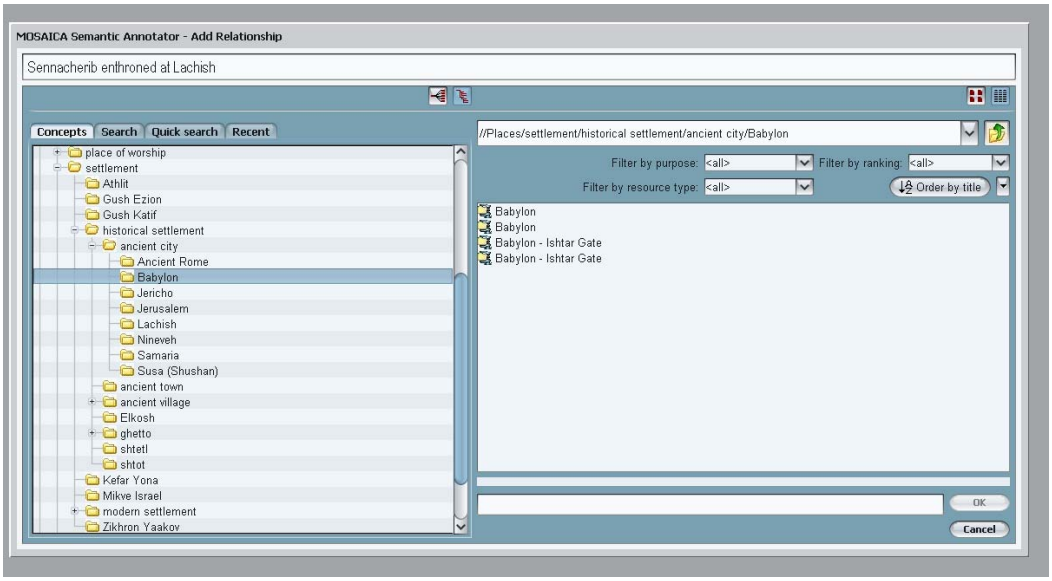

Figure 17 Dialog box for adding relationships between resources

3.2. Map Annotation

Semantic Annotator enables annotation of specific locations on a map specified either as points (for example, to mark settlements, sites, etc.) or areas (for example, to mark regions, districts, etc.), as depicted in Figure 18.

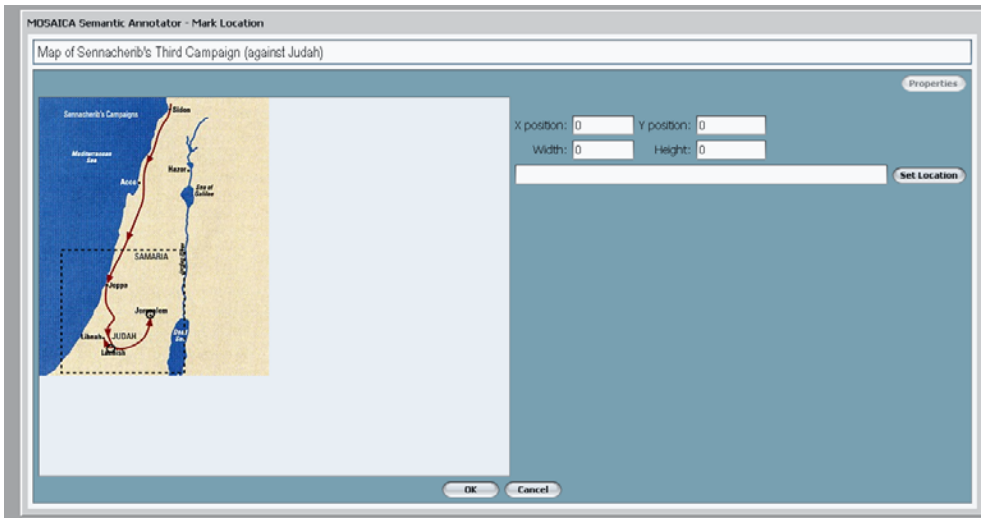

Figure 18 Dialog box for annotation of specific locations on a map

3.3. Multimedia Annotation

Semantic Annotator also annotation of segments within a multimedia (both sound and video) file defined either as frames (if the format supports frames) or time points, as depicted in Figure 19.

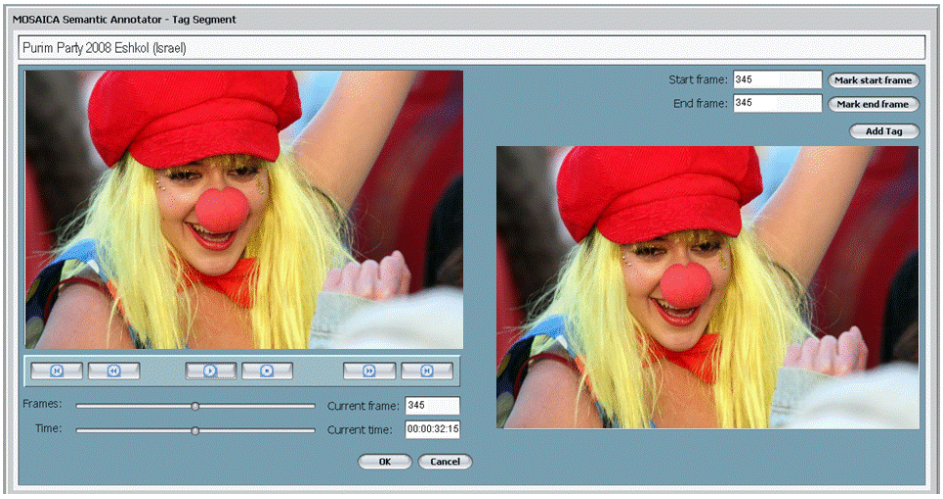

Figure 19 Dialog box for annotation of segments in multimedia files

4. APPENDIX A: DAAT CORE ONTOLOGY SPECIFICATION

4.1. Concepts

customElement	
Description:	Class containing all custom defined elements as its instances. Custom defined elements are of the following types: *Extension* *Template Extension* Template Implementation *Basket*

elementType		
Description:	Enumeration class enumerating types of custom defined elements:	
	etClassExtension	*Extension*
	etTemplateExtension	*Template Extension*
	etTemplateImplementation	Template Implementation
	etBasket	*Basket*
	Custom defined elements are instances of customElement.	

importedResource	
Description:	Class containing internal representations of imported and annotated resources. An external resource imported into the system is represented as an individual concept identified by a unique identifier.

4.2. Properties and Relations

isResourceType	
Type:	Data type property – annotation

isDescriptor	
Type:	Data type property – annotation

canBeSubdividedBy	
Type:	Object property (relation) – annotation

isVisibleUnderView	
Type:	Object property (relation) – annotation

isDefaultView	
Type:	Data type property – annotation

isExtensible

Type:	Data type property – annotation

isExtensible	
Type:	Data type property
Domain	**Range**
Owl:Thing	Boolean

isElementOfType	
Type:	Object property (relation)
Domain	**Range**
customElement	elementType

isExtensionOf	
Type:	Data type property
Domain	**Range**
customElement	String

isSubclassOf	
Type:	Object property (relation)
Domain	**Range**
customElement	customElement

contains	
Type:	Object property (relation) – annotation

5. APPENDIX B: DAAT ANNOTATION ONTOLOGY SPECIFICATION

5.1. Concepts

resource	
Description:	Resource is the annotated object. Its precise definition depends on the underlying application. Resource could be any file, or just specific types of file. In any case, resource is externally identified by its Unique Resource Identifier (URI).

resourceMetadata	
Description:	This concept holds all basic metadata of a resource as a *file* – a digital entity with physical existence.

copyright	
Description:	This concept holds copyright data.

Visibility	
Description:	This concept defines visibility of a resource within the authentication and security mechanisms of the underlying application.

annotation	
Description:	Annotation holds information about the content of a resource. This information falls into three main categories: association with concepts in a domain ontology, association with other resources in the system, and functionality ranking.

annotationMetadata	
Description:	This concept holds all basic metadata of an annotation.

content	
Description:	This concept holds associations between the currently annotated resource and various concepts in a domain ontology provided by the underlying application.

relationship	
Description:	This concept creates an association between the currently annotated resource and other resources already available in the system.

ranking	
Description:	This concept associates a particular functionality with a resource, and ranks the resources usefulness for that particular functionality.

5.2. Properties and Relations

hasResourceMetadata	
Type:	Object property (relation)
Domain	**Range**
Resource	***resourceMetadata***

hasVisibility	
Type:	Object property (relation)
Domain	**Range**
Resource	***Visibility***

hasAnnotation	
Type:	Object property (relation)
Domain	**Range**
Resource	***Annotation***

hasTitle	
Type:	Data type property
Domain	**Range**
resourceMetadata	String

contributedBy	
Type:	Data type property
Domain	**Range**
resourceMetadata	String (holding an external identifier)

contributedOn	
Type:	Data type property
Domain	**Range**
resourceMetadata	Date

hasFileExtension	
Type:	Data type property
Domain	**Range**
resourceMetadata	String

hasLocation	
Type:	Data type property
Domain	**Range**
resourceMetadata	String (holding an internal identifier, such as hash key)

hasOriginalLocation	
Type:	Data type property
Domain	**Range**
resourceMetadata	anyURI

hasSize	
Type:	Data type property

Domain	**Range**
resourceMetadata	Long

hasExternalMetadata	
Type:	Data type property
Domain	**Range**
resourceMetadata	anyURI

hasCopyright	
Type:	Object property (relation)
Domain	**Range**
resourceMetadata	***Copyright***

isInLanguage	
Type:	Object property (relation)
Domain	**Range**
resourceMetadata	Language Types

hasOwner	
Type:	Data type property
Domain	**Range**
Copyright	String

fromYear	
Type:	Data type property
Domain	**Range**
Copyright	gYear

toYear	
Type:	Data type property
Domain	**Range**
Copyright	gYear

hasCCLicense	
Type:	Object property (relation)
Domain	**Range**
Copyright	***creativeCommonsLicense***

hasVisibilityType	
Type:	Object property (relation)
Domain	**Range**
Visibility	***visibilityType***

hasTargetGroup	
Type:	Data type property
Domain	**Range**
Visibility	String (holding an external identifier)

annotatedBy

Type:	Data type property
Domain	**Range**
annotationMetadata	String (holding an external identifier)

annotatedOn	
Type:	Data type property
Domain	**Range**
annotationMetadata	Date

discusses	
To consider or examine in detail or by disputation, to reason upon by presenting favorable and adverse considerations, to debate, to investigate. Distinguished from: presents (considering how the topic is handled) Something is discussed when it is examined and scrutinized. Something is presented when it is merely shown, displayed or demonstrated. Distinguished from: features (considering functionality in the content) Something is discussed when it is the topic handled in the content (what content is about). Something is featured when it is utilized for handling a topic. Distinguished from: mentions (considering how central a topic is in the content) Something is discussed when it is one of the main topics in the content. Something is mentioned when it is only referenced "by the way". For example, a video *discussing* the Establishment of the State of Israel *presents* Declaration of Independence, *features* David Ben Gurion, and *mentions* British Mandate in Palestine.	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type graphic resource, sound resource, text resource or video resource – an their derivative and composite types)	[domain ontology concept] of the type ***Person***, ***Group***, ***Object***, ***Location Reference***, ***Time Reference***, ***Event***, or ***Abstract Concept***.

presents	
To show or demonstrate, to offer to view, to bring to one's attention. Distinguished from: discusses (considering how the topic is handled) Something is discussed when it is examined and scrutinized. Something is presented when it is merely shown, displayed or demonstrated. For example, a video *discussing* the Establishment of the State of Israel *presents* Declaration of Independence, *features* David Ben Gurion, and *mentions* British Mandate in Palestine.	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type graphic resource, sound resource, text resource or video resource – an their derivative and composite types)	[domain ontology concept] of the type ***Person***, ***Group***, ***Object***, ***Location Reference***, ***Time Reference***, ***Event***, or ***Abstract Concept***.

features		
To have as a feature, to picture or portray in the mind, to give special prominence to, to play an important part (intransitive verb). Distinguished from: discusses (considering functionality in the content) Something is discussed when it is the topic handled in the content (what content is about). Something is featured when it is utilized for handling a topic. For example, a video *discussing* the Establishment of the State of Israel *presents* Declaration of Independence, *features* David Ben Gurion, and *mentions* British Mandate in Palestine.		
Type:	Object property (relation)	
Domain		**Range**
Content (but only for the ***Resource*** of the type graphic resource, sound resource, text resource or video resource – an their derivative and composite types)		[domain ontology concept] of the type ***Person***, ***Group***, ***Object***, ***Location Reference***, ***Time Reference***, ***Event***, or ***Abstract Concept***.

mentions		
To make reference to, to refer to. Distinguished from: discusses (considering how central a topic is in the content) Something is discussed when it is one of the main topics in the content. Something is mentioned when it is only referenced "by the way". For example, a video *discussing* the Establishment of the State of Israel *presents* Declaration of Independence, *features* David Ben Gurion, and *mentions* British Mandate in Palestine.		
Type:	Object property (relation)	
Domain		**Range**
Content (but only for the ***Resource*** of the type graphic resource, sound resource, text resource or video resource – an their derivative and composite types)		[domain ontology concept] of the type ***Person***, ***Group***, ***Object***, ***Location Reference***, ***Time Reference***, ***Event***, or ***Abstract Concept***.

createdBy		
Used only when referring to a domain ontology class of the type ***Person***.		
Type:	Object property (relation)	
Domain		**Range**
Content		[domain ontology concept] of the type ***Person***.

createdIn		
Used only when referring to a domain ontology class of the type ***Time Reference***.		
Type:	Object property (relation)	
Domain		**Range**
Content		[domain ontology concept] of the type ***Time Reference***.

originatesFrom		
Used only when referring to a domain ontology class of the type ***Location Reference***.		
Type:	Object property (relation)	
Domain		**Range**
Content		[domain ontology concept] of the type ***Location Reference***.

quotes

Used only when referring to a domain ontology class of the type ***Person***.	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type graphic resource, sound resource, text resource or video resource – an their derivative and composite types)	[domain ontology concept] of the type ***Person***.

depictsEvent	
Used only if annotating resource of the specific type Map (a derivative of graphic resource).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type Map)	[domain ontology concept] of the type ***Event***.

depictsLocation	
Used only if annotating resource of the specific type Map (a derivative of graphic resource).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type Map)	[domain ontology concept] of the type ***Location Reference***.

depictsPeriod	
Used only if annotating resource of the specific type Map (a derivative of graphic resource).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type Map)	[domain ontology concept] of the type ***Time Reference***.

depictsSite	
Used only if annotating resource of the specific type Map (a derivative of graphic resource).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type Map)	[domain ontology concept] of the type ***Location Reference***.

isArtefact	
Used only if annotating resource of the specific type Map (a derivative of graphic resource).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type Map)	[domain ontology concept] of the type ***Object***.

containsRecordsRelatedTo	
Used only if annotating resource of the specific type database (a derivative of application).	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type database)	[domain ontology concept] of the type ***Person***, ***Group***, ***Object***, ***Location Reference***, ***Time Reference***, ***Event***, or ***Abstract Concept***.

providesFunctionalitiesRelatedTo	
Used only if annotating resource of the specific type application.	
Type:	Object property (relation)
Domain	**Range**
Content (but only for the ***Resource*** of the type application)	[domain ontology concept] of the type ***Abstract Concept***.

models	
Used only if annotating resource of the specific type knowledge representation system. String data type was preferred as the range, in order to enable unrestricted specification of the modeled domain.	
Type:	Data type property
Domain	**Range**
Content (but only for the ***Resource*** of the type knowledge representation system)	String

isAbout	
An optional content annotation property serving as a tag in the context of the collaborative tagging. Enables specification of the content beyond the selected domain application.	
Type:	Data type property
Domain	**Range**
Content	String

isIdenticalTo	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isSimilarTo	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isResponseTo	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isElaborationOf	
Type:	Object property (relation)
Domain	**Range**

Relationship	***Resource***

isExtensionOf	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isSummaryOf	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isDerivedFrom	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

isUsedIn	
Used to specify whether a resource was embedded in other resources (inverse of encloses).	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

encloses	
Used to specify embedded file (such as images, sound, etc.) – inverse of isUsedIn)	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

cites	
Used to specify bibliographic references (citations)	
Type:	Object property (relation)
Domain	**Range**
Relationship	***Resource***

hasPurpose	
Type:	Object property (relation)
Domain	**Range**
Ranking	***rankingPurposeType***

hasValue	
Type:	Object property (relation)
Domain	**Range**
Ranking	***rankingValue***

hasNumericEquivalentOf		
Type:	Data type property	
Domain		**Range**
rankingValue		xsd:nonNegativeInteger

6. APPENDIX C: GIS/TIS ONTOLOGY SPECIFICATION

6.1. XML Schema

Although MOSAICA GIS Ontology is formatted as OWL, the data format for exchanging information between various components (primarily, GIS and the Semantic Layer) is XML with the following XML schema:

Map		
<ID>	QName	
<hasCaption>	rdfs:label	
<P2PRefToPicture>	hasP2PCMSLocation	xsd:string
<depictsContent>	[Grouping element]	
<depictsLocation>	depictsLocation	Location
<depictsSite>	depictsSite	Site
<depictsEvent>	depictsEvent	Event
<depictsPeriod>	depictsPeriod	Period
<isArtefact>	isArtefact	Artifact
<isDated>	[Grouping element]	
<dated>	isDated	xsd:date
<datedAround>	isDatedAround	xsd:date
<originates>	originatesFromDate	xsd:date
	originatesToDate	xsd:date
<originatesFromPeriod>	originatesFromPeriod	Period
<hasTemporalExtent>	[Grouping element]	
<presentsDate>	presentsLocationOnDate	xsd:date
<presentsAroundDate>	presentsLocationAroundDate	xsd:date
<presents>	presentsLocationFromDate	xsd:date
	presentsLocationToDate	xsd:date
<presentsPeriod>	presentsLocationInPeriod	Period
<content>	[Grouping element]	
<point>	Point	
<display>	Display	

Point		
<ID>	QName	
<isOnMap>	isOnMap	Map
<hasXAxisValue>	hasXAxisValue	xsd:nonNegativeInteger
<hasYAxisValue>	hasYAxisValue	xsd:nonNegativeInteger
<hasGeometricForm>	isOfGeometricFormType	geometricFormType
<hasCustomForm>	hasCustomForm	xsd:string
<hasRadius>	hasRadius	xsd:float

<hasColor>	[Grouping element]	
<hasRedValue>	hasRedValue	xsd:nonNegativeInteger
<hasBlueValue>	hasBlueValue	xsd:nonNegativeInteger
<hasGreenValue>	hasGreenValue	xsd:nonNegativeInteger
<hasPrimaryLabel>	rdfs:label	
<hasAlternateLabel>	isAlsoCalled	xsd:string
<displayLabel>	labelRequiresActionOfType	labelDisplayActionType
<hasText>	rdfs:comment	
<hasMenu>		
<hasAlbum>		

Display		
<ID>	QName	
<isOnMap>	isOnMap	Map
<hasCentroid>	[Grouping element]	
<hasXAxisValue>	hasXAxisValue	xsd:nonNegativeInteger
<hasYAxisValue>	hasYAxisValue	xsd:nonNegativeInteger
<hasRadius>	hasRadius	xsd:float
<displayAction>	displayRequiresActionOfType	displayActionType
<hasLineOfType>	hasLineOfType	displayLineType
<hasColor>	[Grouping element]	
<hasRedValue>	hasRedValue	xsd:nonNegativeInteger
<hasBlueValue>	hasBlueValue	xsd:nonNegativeInteger
<hasGreenValue>	hasGreenValue	xsd:nonNegativeInteger
<hasPrimaryLabel>	rdfs:label	
<hasAlternateLabel>	isAlsoCalled	xsd:string
<displayLabel>	labelRequiresActionOfType	labelDisplayActionType
<hasText>	rdfs:comment	
<hasMenu>		
<hasAlbum>		

albumItem		
<ID>	QName	
<hasText>	rdfs:label	
<displays>	Displays	xsd:string
<isAtPosition>	isAtPosition	xsd:nonNegativeInteger

Location	
<ID>	QName
<label>	rdfs:label

Site	
<ID>	QName
<label>	rdfs:label

Event	
<ID>	QName
<label>	rdfs:label

Period

<ID>	QName
<label>	rdfs:label

Artifact	
<ID>	QName
<label>	rdfs:label

6.2. Concepts

Map	
MOSAICA Annotation Ontology	
Property / relation	**Value / object**
isDated	xsd:date
isDatedAround	xsd:date
originatesFromDate	xsd:date
originatesToDate	xsd:date
originatesFromPeriod	Period
presentsLocationOnDate	xsd:date
presentsLocationAroundDate	xsd:date
presentsLocationFromDate	xsd:date
presentsLocationToDate	xsd:date
presentsLocationInPeriod	Period

Location
MOSAICA Domain Ontology

Site
MOSAICA Domain Ontology

Event
MOSAICA Domain Ontology

Period
MOSAICA Domain Ontology

Artifact
MOSAICA Domain Ontology

Point
MOSAICA GIS Ontology

Display
MOSAICA GIS Ontology

geometricFormType
MOSAICA GIS Ontology

labelDisplayActionType
MOSAICA GIS Ontology

displayLineType
MOSAICA GIS Ontology

displayActionType
MOSAICA GIS Ontology

albumItem	
MOSAICA GIS Ontology	
Property / relation	**Value / object**
isIllustrationOf	Point Display
Displays	xsd:string
isAtPosition	xsd:nonNegativeInteger

6.3. Properties and Relations

Represents	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Point Display	Location Site

hasP2PCMSLocation	
MOSAICA Annotation Ontology	
Data type property	
Domain	**Range**
Resource	xsd:string

depictsLocation	
DAAT Annotation Ontology	
Object property (relation)	
Domain	**Range**
Content (DAAT Annotation Ontology)	Location

depictsSite	
DAAT Annotation Ontology	
Object property (relation)	
Domain	**Range**
Content (DAAT Annotation Ontology)	Site

depictsEvent	
DAAT Annotation Ontology	
Object property (relation)	
Domain	**Range**
Content (DAAT Annotation Ontology)	Event

depictsPeriod	
DAAT Annotation Ontology	
Object property (relation)	
Domain	**Range**
Content (DAAT Annotation Ontology)	Period

isArtefact	
DAAT Annotation Ontology	
Object property (relation)	
Domain	**Range**
Content (DAAT Annotation Ontology)	Artifact

isDated	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

isDatedAround	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

originatesFromDate	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

originatesToDate	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

originatesFromPeriod	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Map	Period

presentsLocationOnDate	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

presentsLocationAroundDate
MOSAICA GIS Ontology

Data type property	
Domain	**Range**
Map	xsd:date

presentsLocationFromDate	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

presentsLocationToDate	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Map	xsd:date

presentsLocationInPeriod	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Map	Period

isOnMap	
Object property (relation)	
MOSAICA GIS Ontology	
Domain	**Range**
Point Display	Map

hasXAxisValue	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point	xsd:nonNegativeInteger

hasYAxisValue	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point	xsd:nonNegativeInteger

isOfGeometricFormType	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Point	geometricFormType

hasCustomForm
MOSAICA GIS Ontology

Data type property	
Domain	**Range**
Point	xsd:string

hasRadius	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point Display	xsd:float

hasRedValue	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point	xsd:nonNegativeInteger

hasBlueValue	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point	xsd:nonNegativeInteger

hasGreenValue	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
Point	xsd:nonNegativeInteger

isAlsoCalled	
MOSAICA Domain Ontology	
Data type property – annotation	
Domain	**Range**
owl:Thing	xsd:string

labelRequiresActionOfType	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Point Display	labelDisplayActionType

hasLineOfType	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Display	displayLineType

displayRequiresActionOfType	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
Display	displayActionType

isIllustrationOf	
MOSAICA GIS Ontology	
Object property (relation)	
Domain	**Range**
albumItem	Point Display

Displays	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
albumItem	xsd:string

isAtPosition	
MOSAICA GIS Ontology	
Data type property	
Domain	**Range**
albumItem	xsd:nonNegativeInteger

7. APPENDIX D: OWL DL REASONING CAPABILITIES

Adopted from **http://www.ksl.stanford.edu/software/JTP/doc/owl-reasoning.html**.

Reasoning capabilities shared with RDF/RDFS	
Type inheritance through rdfs:subclassOf	(rdf:type Morris Cat) (rdfs:subClassOf Cat Mammal) Entails: (rdf:type Morris Mammal)
Reflexivity of rdfs:subPropertyOf and rdfs:subclassOf	(rdfs:subPropertyOf p p) (rdfs:subClassOf C C)
Type inference through rdfs:range and rdfs:domain constraints	(rdfs:domain teaches Teacher) (rdfs:range teaches Student) (teaches Bob Scooter) Entails: (rdf:type Bob Teacher) (rdf:type Scooter Student)
Transitivity of rdfs:subClassOf and rdfs:subPropertyOf	(rdfs:subClassOf Dog Mammal) (rdfs:subClassOf Mammal Animal) Entails: (rdfs:subClassOf Dog Animal) (rdfs:subPropertyOf parent ancestor) (rdfs:subPropertyOf ancestor relative) Entails: (rdfs:subPropertyOf parent relative)
OWL Reasoning Capabilities	
Enforcing transitivity of owl:TransitiveProperty	(rdf:type ancestor owl:TransitiveProperty) (ancestor Sue Mary) (ancestor Mary Anne) Entails: (ancestor Sue Anne)
Semantics of owl:SymmetricProperty is enforced	
Reasoning with owl:inverseOf	(owl:inverseOf parentOf hasParent) (parentOf Goldie Kate) Entails: (hasParent Kate Goldie)

Inheritance of disjointness constraints	(owl:disjointWith Plant Animal) (rdfs:subClassOf Mammal Animal) Entails: (owl:disjointWith Plant Mammal)
When an owl:sameAs relationship is asserted or inferred between two entities that are known to be classes, an owl:equivalentClass relationship is inferred between the classes. Similarly, when an owl:sameAs relationship is asserted or inferred between two entities that are known to be properties, an owl:equivalentProperty relationship is inferred between the classes.	(owl:sameAs Human Person) (rdf:type Human rdfs:Class) (rdf:type Person rdfs:Class) Entails: (owl:equivalentClass Human Person)
All the subclasses of a given class are disjoint with the class's complement.	(owl:complementOf Animal NonAnimals) (rdfs:subClassOf Mammal Animal) Entails: (owl:disjointWith Mammal NonAnimals)
(=> (and (owl:complementOf ?c1 ?c2) (rdfs:subClassOf ?c3 ?c1) (rdfs:subClassOf ?c4 ?c2) (owl:complementOf ?c4 ?c5)) (rdfs:subClassOf ?c3 c5))	
Inferring owl:sameAs relationships via owl:FunctionalProperty and owl:InverseFunctionalProperty.	(rdf:type mother owl:FunctionalProperty) (mother Joe Margaret) (mother Joe Maggie) Entails: (owl:sameAs Margaret Maggie)
If a class A is owl:oneOf a list of objects, say X, Y, and Z, then each of X, Y, and Z has rdf:type A.	
If an object is rdf:type an owl:hasValue owl:Restriction, then the object has the specified value for the specified property.	(owl:onProperty RestrictionOrangeSkin skinColor) (owl:hasValue RestrictionOrangeSkin Orange) (rdf:type MrOompaLoompa RestrictionOrangeSkin) Entails: (skinColor MrOompaLoompa Orange)
If an owl:hasValue owl:Restriction restricts a particular property to a particular value, and an object has that value for that property, then the object has the Restriction as a type.	(owl:onProperty RestrictionOrangeSkin skinColor) (owl:hasValue RestrictionOrangeSkin Orange) (skinColor MrOompaLoompa Orange) Entails: (rdf:type MrOompaLoompa RestrictionOrangeskin)
If an object is a rdf:type an owl:allValuesFrom owl:Restriction, and the object has values for the specified property, then the values are of the specified type.	(owl:onProperty RestrictionCatChildren child) (owl:allValuesFrom RestrictionCatChildren Cat) (rdf:type Fluffy RestrictionCatChildren) (child Fluffy Cupcake) Entails: (rdf:type Cupcake Cat)

If an owl:someValuesFrom owl:Restriction restricts a particular property to a particular type, and if an object has some values of the specificied type for the specified property, then that object has the Restriction as a type.	(owl:onProperty RestrictionIvyLeagueDegree degree) (owl:someValuesFrom RestrictionIvyLeagueDegree IvyLeagueSchool) (degree Mary Harvard) (rdf:type Harvard IvyLeagueSchool) Entails: (rdf:type Mary RestrictionIvyLeagueDegree)
If a property Q is owl:inverseOf of a property P, and P is an owl:TransitiveProperty, then Q is also an owl:TransitiveProperty.	
All of the elements of an owl:AllDifferent are owl:differentFrom each other.	
OWL Classification Inferences	
All classes are rdfs:subClassOf owl:Thing.	
If a class A is the owl:unionOf a list of classes, say B, C, and D, then each of B, C, and D, is rdfs:subClassOf A.	
If a class A is the owl:intersectionOf a list of classes, say B, C, and D, then A is rdfs:subClassOf of each of B, C, and D.	
If a class A is the owl:intersectionOf a list of constituent classes and a class B is the owl:intersectionOf a list of constituent classes, then A is a rdfs:subClassOf B if every constituent of A is a rdfs:subClassOf some constituent of B.	
If a class A is the owl:unionOf a list of constituent classes and a class B is the owl:unionOf a list of constituent classes, then A is a rdfs:subClassOf B if every constituent of B is a superclass of some constituent of A.	
If a class A is the owl:intersectionOf a list of constituent classes and a class B is the owl:unionOf a list of constituent classes, then A is a rdfs:subClassOf B if every constituent of A is a rdfs:subClassOf every constituent of B.	
An owl:hasValue owl:Restriction A subsumes another owl:hasValue restriction B if the property restricted by B is a rdfs:subPropertyOf the property restricted by A, and the restricted value is the same.	
An owl:allValuesFrom owl:Restriction A subsumes another owl:allValuesFrom restriction B if the property restricted by B is a rdfs:subPropertyOf the property restricted by A, and the restricted class of B is a rdfs:subClassOf the restricted class of A. Recall that any class is a subclass of itself, and any property is a subproperty of itself.	
If an object is a type of owl:maxCardinality owl:Restriction where the specified property is restricted to a maximum of 1 value, then all values of which the object has for the specified property are owl:sameAs one another. An owl:minCardinality owl:Restriction A is disjoint from an owl:maxCardinality owl:Restriction B if the property restricted by A is a rdfs:subPropertyOf the property restricted by B, and the number of values specified by A is greater than the number of values specified by B.	
An owl:minCardinality owl:Restriction A subsumes another owl:minCardinality restriction B if the property restricted by B is a rdfs:subPropertyOf the property restricted by A, and the number of values specified by A is less than or equal to the number of values specified by B.	
An owl:maxCardinality owl:Restriction A subsumes another owl:maxCardinality restriction B if the property restricted by B is a rdfs:subPropertyOf the property restricted by A, and the number of values specified by A is greater than or equal to the number of values specified by B.	

8. REFERENCES

[Abasolo and Gomez 2000] Abasolo, J.M. and M. Gomez (2000). "MELISA, an ontology-based agent for information retrieval in Medicine," ECDL 2000 Workshop on the Semantic Web.

[Anderson et al. 1976] Anderson, J. R., E. E. Hardy, J. T. Roach and R. E. Witmer (1976). A Land Use And Land Cover Classification System For Use With Remote Sensor Data. Geological Survey Professional Paper 964, Washington, DC, USA.

[Agarwal 2005] Agarwal, P. (2005). Ontological Considerations in GIScience. International Journal of Geographic Information Science 19(5): 501-536.

[Bateman et al. 2006] Bateman, S., Brooks and C., McCalla, G. (2006) Collaborative Tagging Approaches for Ontological Metadata in Adaptive ELearning Systems In the Proceedings of the Fourth International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL 2006) in conjunction with 2006 International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2006), June 20, 2006. Dublin, Ireland.

[Bloehdorn et al. 2005] Bloehdorn, S., K. Petridis, C. Saatho, N. Simou, V. Tzouvaras, Y.S. Avrithis, S. Handschuh, I. Kompatsiaris, S. Staab, and M.G. Strintzis. Semantic annotation of images and videos for multimedia analysis. In ESWC, pages 592-607, 2005.

[Bontcheva et al. 2006] Bontcheva, K., J. Davies, A. Duke, T. Glover, N. Kings and I. Thurlow (2006). Semantic Information Access. In Davies, J., R. Studer and P. Warren (Eds.) Semantic Web Technologies, John Wiley & Sons, Ltd, Published Online: 3 Jul 2006

[Bossard et al. 2000] Bossard, M., J. Feranec and J. Otahel (2000). CORINE land cover technical guide – Addendum 2000. European Environment Agency (EEA), Copenhagen, Denmark.

[Brachman et al. 1989] Brachman, R, Borgida, A., McGuinness, D., and Resnick, L. The CLASSIC Knowledge Representation System. In Proceedings of the 11th IJCAI. August, 1989.

[de Bruijn et al. 2005] de Bruijn, J., A. Polleres, R. Lara, and D. Fensel. OWL DL vs. OWL Flight: Conceptual modeling and reasoning on the semantic web. In Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan, 2005. ACM.

[Chakravarthy et al. 2006] Chakravarthy, A., F. Ciravegna, and V. Lanfranchi. Cross-media document annotation and enrichment. In Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW2006), 2006.

[Di Gregorio and Jansen 1998] Di Gregorio, A. and L. J. M. Jansen (1998). Land Cover Classification System (LCCS): Classification Concepts and User Manual. Food and Agriculture Organization of the United Nations (FAO), Rome, Italy.

[Di Gregorio 2005] Di Gregorio, A. (2005). Land Cover Classification System (LCCS), version 2: Classification Concepts and User Manual. Food and Agriculture Organization of the United Nations (FAO), Rome, Italy.

[Egenhofer and Mark 1995] Egenhofer, M. J. and D. M. Mark (1995). Naive Geography. Spatial Information Theory-A Theoretical Basis for GIS, International Conference COSIT'95, Semmering, Austria, Springer-Verlag.

[Frank 1997] Frank, A. U. (1997). Spatial Ontologies. A Geographical Point of View. Spatial and Temporal Reasoning. O. Stock. Dordrecht, The Netherlands, Kluwer Academic Publishers.

[Frank 2001] Frank, A. U. (2001). Tiers of ontology and consistency constraints in geographical information systems. International Journal of Geographic Information Science 15(7): 667-678.

[Frank 2003] Frank, A. U. (2003). Ontology for Spatio-Temporal Databases. Spatiotemporal Databases: The Chorochronos Approach. M. e. a. Koubarakis, Springer-Verlag.

[Gouveia et al. 2001] Gouveia, C., P. Henriques, R. Nicolau, J. Rocha and M. Santos (2001). Moving from CEN TC 287 to ISO/TC 211 - The approach of the Portuguese National Geographic Information Infrastructure. 4th AGILE Conference on Geographic Information Science, Brno, Czech Republic.

[Grau et al. 2006] Grau, B. C., I. Horrocks, B. Parsia, P. Patel-Schneider and U. Sattler (2006). Next Steps for OWL. In OWLEd 2006, Proceedings of the 2nd International Workshop: OWL Experiences and Directions, Athens, Georgia November 10-11, 2006 2006.

[Gruber 2006] Gruber T. (2006). Where the Social Web Meets the Semantic Web. Keynote presentation at ISWC, The 5th International Semantic Web Conference, November 7, 2006.

[Gruber 2008] Gruber, T. (2008). Collective knowledge systems: Where the Social Web meets the Semantic Web. Journal of Web Semantics 6(1): 4-13 (2008)

[Gupta et al. 1991] Gupta, A., Weymouth, T. E. and Jain, R. (1991). "Semantic Queries with Pictures: the VIMSYS model". In Proceedings of the Very Larage Database Systems Conference, pages 69-79, 1991.

[Handschuh and Staab 2002] Handschuh, S. and S. Staab. Authoring and annotation of web pages in cream. In Proceedings of the Eleventh International World Wide Web Conference, WWW2002, pages 462-473, 2002.

[Horrocks et al. 2007] Horrocks, I., P. F. Patel-Schneider, D. L. McGuinness, and Ch. A. Welty (2007). OWL: a Description Logic Based Ontology Language for the Semantic Web. Deborah L. McGuinness and Peter F. Patel-Schneider. From Description Logic Provers to Knowledge Representation Systems. In The Description Logic Handbook: Theory, Implementation and Applications, ed. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. Cambridge University Press, 2nd edition, 2007, pp. 458--486.

[Hollink et al. 2004] Hollink, L., Screiber, A. Th., Wielinga, B. J. and Worring, M. (2004). "Classification of user image descriptions". Intl. Journal of Human-Computer Studies, 61, 2004, pp. 601-626.

[Ide et al. 1997] Ide, N., McGraw, T. and Welty, Ch. (1997). Representing TEI Documents in the CLASSIC Knowledge Representation System. Proceedings of the Tenth workshop of the Text-Encoding Initiative. November, 1997.

[Jörgensen 1998] Jörgensen, C. (1998). "Attributes of Images in Describing Tasks". Information Processing and Management, 34(2/3): 161-174, 1998.

[Kahan et al. 2001] Kahan, J., M.R. Koivunen, E. Prud'hommeaux, and R.R. Swick. Annotea: an open rdf infrastructure for shared web annotations. In Proceedings of the 10th International World Wide Web Conference, pages 623-632, 2001.

[Kuhn 2003] Kuhn, W. (2003). Semantic reference systems. International Journal of Geographic Information Science 17(5): 405-409.

[Mark et al. 2001] Mark, D. M., A. Skupin and B. Smith (2001). Features, Objects, and other Things: Ontological Distinctions in the Geographic Domain. Spatial information theory: Foundations of Geographic Information Science. COSIT 2001, Springer.

[Martínez 2001] Martínez, J. M. (2001). "Overview of the MPEG-7 Standard (version 5.0)". ISO/IEC JTC1/SC29/WG11 N4031, March 2001.

[Mechkour 1995] Mechkour, M. (1995). "EMIR2: an extended model for image representation and retrieval". In Proceeding of Database and Expert Systems Applications Conference, London, pages 395-404, 1995.

[Rasmussen 1997] Rasmussen, E. (1997). "Indexing Images". ARIST, 32:169-196, 1997.

[Schuster and Stuckenschmidt 2001] Schuster, G. and H. Stuckenschmidt (2001). Building shared terminologies for ontology integration. Künstliche Intelligenz (KI), Vienna, Austria.

[Shklar et al. 1994] Shklar, L.. S. Thatte, H. Marcus, and A. Sheth, "The InfoHarness Information Integration Platform ", Advance Proceedings of the Second International WWW Conference `94, Chicago, October 17-20, pp. 809-819.

[Smith and Mark 1998] Smith, B. and D. M. Mark (1998). Ontology and Geographic Kinds. International Symposium on Spatial Data Handling, Vancouver, Canada.

[Smith and Mark 1999] Smith, B. and D. M. Mark (1999). Ontology with human subjects testing: An Empirical Investigation of Geographic Categories. American Journal of Economics and Sociology 58: 245-272.

[Smith and Mark 2001] Smith, B. and D. M. Mark (2001). Geographical categories: an ontological investigation. International Journal of Geographic Information Science 15(7): 591-612.

[Staab et al. 2000] Staab, S., Angele, J., Decker, S., Hotho, A., Maedche, A., Schnurr, H-P., Studer, S., Sure, Y.: AI for the Web --- Ontology-based Community Web Portals. In: AAAI 2000/IAAI 2000 - Proceedings of the 17th National Conference on Artificial Intelligence and 12th Innovative Applications of Artificial Intelligence Conference, Austin/TX, USA, July 30-August 3, 2000, Menlo Park/CA, Cambridge/MA, AAAI Press/MIT Press.

[Uren et al. 2006] Uren, V., Ph. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, F. Ciravegna (2006). Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics: Science, Services and Agents on the World Wide Web (4): 14-28. 2006.

[Visser et al. 2001] Visser, U., H. Stuckenschmidt, H. Wache and T. Vögele (2001). Using Environmental Information Efficiently: Sharing Data and Knowledge from Heterogeneous Sources. Environmental Information Systems in Industry and Public Administration. C. Rautenstrauch and S. Patig. Hershey, USA & London, UK, IDEA Group: 41-73.

[Visser and Schuster 2002] Visser, U. and G. Schuster (2002). Finding and Integration of Information - A Practical Solution for the Semantic Web -. ECAI Workshop on Ontologies and Semantic Interoperability, Lyon, France.

[Visser et al. 2002a] Visser, U., H. Stuckenschmidt, G. Schuster and T. Vögele (2002). Ontologies for Geographic Information Processing. Computers & Geosciences 28(1): 103-118.

[Visser et al. 2002b] Visser, U., T. Vögele and C. Schlieder (2002). Spatio-Terminological Information Retrieval using the BUSTER System. EnviroInfo, Vienna, Austria.

[Vrandecic 2005] Vrandecic, D. (2005) Explicit Knowledge Engineering Patterns with Macros. In Chris Welty and Aldo Gangemi, Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005. Galway, Ireland, November 2005.

[Weibel et al. 1998] Weibel, S., Kunze, J., Lagoze, C. and Wolf, M. (1998). "Dublin Core Metadata for Resource Discovery." IETF #2413. The Internet Society, September 1998.

[Welty 1998] Welty, C. (1998). The Ontological Nature of Subject Taxonomies. In N. Guarino, ed., Formal Ontology in Information Systems. IOS Press Frontiers in AI Applictions Series. 317-327. Trento, Italy. June, 1998.

[Zhang 2007] Zhang, J. (2007). Ontology and the Semantic Web. In Tennis, Joseph T., Eds. Proceedings North American Symposium on Knowledge Organization 2007 1, pages pp. 9-20, Toronto, Ontario.