



D4.1 Model and Specification for Distributed Knowledge Stores

Damyan Ognyanov (Ontotext Lab)

Atanas Kiryakov, Vassil Momchev, Ruslan Velkov (Ontotext Lab)

Abstract

EU-IST Specific targeted research project (STREP) IST-2004-026460 TAO

Deliverable D4.1 (WP 4)

Heterogeneous Knowledge Stores are specified which allow for efficient management of several types of knowledge: unstructured content, structured data, ontologies, and semantic annotations. There are three major contributions:

- Analysis of representation and reference models (task T4.1);
- Model for representation of the information types discussed above;
- Specification for heterogeneous knowledge stores, based on this model.

The basis of the model for management of heterogeneous information is a uniform data-model for management of structured information. It is defined as RDF, extended with support for Named Graphs and triplesets – groups of “contextualized” triples. A schema for representation of document-related metadata is layered on top of it.

The Knowledge Stores specification outlines middleware architecture, in the core of which is a set of interfaces, which define the interoperability of a repository which allows for management (storage, modification, retrieval) of RDF with the above mentioned extensions.

Keyword list: semantic repository, ontology management, distribution

WP4 Scalable, Heterogeneous Knowledge Stores

Type: Report

Contractual date: 31 August 06

Distribution: PU

Actual Delivery Date: 25 October 2006

TAO Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2004-026460.

University of Sheffield

Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1930
Fax: +44 114 222 1810
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Mondeca

3, cité Nollez
75018 Paris
France
Tel: +33 (0) 1 44 92 35 03
Fax: +33 (0) 1 44 92 02 59
Contact person: Jean Delahousse
E-mail: jean.delahousse@mondeca.com

University of Southampton

Southampton SO17 1BJ
UK
Tel: +44 23 8059 8343
Fax: +44 23 8059 2865
Contact person: Terry Payne
E-mail: trp@ecs.soton.ac.uk

Sirma Group Corp., Ontotext Lab

Office Express IT Centre, 5th Floor
135 Tsarigradsko Shosse Blvd.
Sofia 1784
Bulgaria
Tel: +359 2 9768 303
Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Atos Origin Sociedad Anonima Espanola

Dept Stream-Public Sector
Atos Origin Spain, C/Albarracin, 25, 28037
Madrid
Spain
Tel: +34 91 214 9321
Fax: +34 91 754 3252
Contact person: Nuria De Lama
E-mail: nuria.delama@atosorigin.com

Dassault Aviation SA

DGT/DPR
78, quai Marcel Dassault
92552 Saint-Cloud
Cedex 300
France
Tel: +33 1 47 11 53 00
Fax: +33 1 47 11 53 65
Contact person: Farid Cerbah
E-mail: Farid.Cerbah@dassault-aviation.com

Jozef Stefan Institute

Department of Knowledge Technologies
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 477 3778
Fax: +386 1 477 3131
Contact person: Marko Grobelnik
E-mail: Marko.Grobelnik@ijs.si

Executive Summary

This document specifies a Heterogeneous Knowledge Stores which allow for efficient management of several types of knowledge: unstructured content (documents), structured data (databases), ontologies, and semantic annotations which augment the content with links to machine-interpretable metadata. There are three major contributions:

- Analysis of representation and reference models (task T4.1);
- Model for representation of the information types discussed above;
- Specification for heterogeneous knowledge stores, based on this model.

An introductory discussion on the various sorts of information provides analysis with respect to three major qualities (named data qualia): structure, semantics, schema-usage. An unambiguous interpretation of several terms related to ontologies is provided with respect to the three data qualia. Then we continue with analysis of the possible reference and annotation models, most notably: stand-off vs. embedded annotations.

The model for representation of heterogeneous information is specified on two layers. In the basis, there is a uniform data-model for management of structured information. It is defined as RDF, extended with support for named graphs and triplesets – groups of “contextualized” triples. A schema for representation of document-related metadata is layered on top of this data-model.

The knowledge stores specification outlines middleware architecture, in the core of which is a set of interfaces, which determine the interoperability patterns of a repository, allowing management (storage, modification, retrieval) of RDF with the above mentioned extensions. The central interface there, named **store**, is designed to allow for multiple implementations of stores and extensions. Wrappers for popular data management systems (e.g. RDBMS) shall be developed as alternative implementations. Higher-levels of modelling – document annotations, concrete ontology languages (e.g. WSML) shall be implemented as extensions of ORDI’s interfaces.

The Knowledge Stores will be developed on the basis of the second generation of the ORDI framework, which is a joint development with project TripCom¹. The ORDI specification, [29], shall be considered complementary to this deliverable, as it provides further details and more extended discussion on two important aspects: the uniform model for representation of structured data and the store architecture.

This deliverable provides a specification for the development of a Heterogeneous Knowledge Store (D4.2) and their distributed variant (D4.3). It is also relevant to the overall architecture and integration specification (D5.2), as one of TAO’s key components is specified here. The developments in WP2 and WP3 are related to the knowledge stores, as long as they should store the extracted ontologies and the metadata on the legacy content there.

¹ <http://www.tripcom.org/>

Contents

TAO Consortium	2
Executive Summary	3
Contents	4
1 Introduction.....	5
1.1 Ontology-Related Terminology	5
1.2 The Borderline between Ontologies and Knowledge Bases.....	7
1.3 Analysis of the Different Types of Information	8
1.4 Metadata, Annotations, Semi-Structured Information.....	9
2 Annotations and Reference Models.....	11
2.1 Semantic Annotations and the Semantic Web	12
3 Modelling Structured Knowledge	15
3.1 Modelling Requirements for Structured Knowledge.....	15
3.2 Uniform Structured Data-Model.....	16
3.2.1 The RDF Model	16
3.2.2 Named Graphs and Datasets	16
3.2.3 Triplesets.....	17
3.2.4 Reification.....	17
3.2.5 Tripleset Usage Example	18
3.3 Interoperability with WSML.....	19
4 Modelling Annotations	20
4.1 Semantic Annotation of Named Entities.....	20
4.1.1 Named Entities.....	21
4.2 Modelling Requirements for Semantic Annotations.....	21
4.3 Document and Annotation Modelling	22
4.4 Annotations Encoding Shema.....	23
5 Knowledge Store Architecture.....	24
5.1 Architectural Requirements	24
5.2 Architecture Overview	24
6 Conclusion	27
7 Bibliography and references	29

1 Introduction

This document specifies a Heterogeneous Knowledge Store, which allow for efficient management of several types of knowledge: unstructured content (e.g. documents), structured data (e.g. databases), ontologies, and semantic annotations which augment the content with links to machine-interpretable metadata. Deliverable D4.1 also presents the result of “T4.1. Analysis of representation and reference models”, the scope of which is a model specification that allows for encoding of formal relations between concepts/instances and parts of heterogeneous document content, i.e. metadata at a sub-document level, as part of an ontology. Thus, the three major contributions are as follows:

- Analysis of representation and reference models;
- Model for representation of the information types discussed above;
- Specification for heterogeneous knowledge stores, based on this model.

The reminder of this section provides discussion on several terms related to ontologies and the major sorts of (digitalized) information, which form the intellectual assets of an enterprise. Analysis of various annotation models is presented in section 2. The 3rd section defines a uniform structured data-model, while section 4 layers a document annotation model on top of it. Section 5 provides outline for knowledge stores architecture, which support management of information with respect to this model. Finally, the sixth section concludes the deliverable and discusses the immediate implementation plans.

The Knowledge Stores will be developed on the basis of the second generation² of the Ontology Representation and Data Integration (ORDI) framework, which is a joint development with project TripCom³. While both projects will share the core of the architecture, they will extend it in different directions:

- TAO will put emphasize on distributed repositories, annotation management, and heterogeneous queries;
- TripCom will focus on integration with relational database integration and data federation (or consolidation) of independent data-sources.

ORDI is used (or foreseen to be used) as ontology management and data integration middleware in several projects and products, among which: OWLIM, KIM, SemanticGov, MediaCampaign, RASCALLI. The ORDI specification, [29], shall be considered complementary to this deliverable, as it provides further details and extended discussion on two important aspects: the uniform structured data-model and the store architecture.

1.1 Ontology-Related Terminology

This section provides definitions for terms related to ontologies and their usage in the Knowledge Management (KM), Semantic Web, and enterprise application integration (EAI) context. While these terms are fairly popular, each one of them is in possession of a variety of meanings, which could be a source of confusion. What is even more

² The specification of the first generation of ORDI is [24], an implementation is available in [25].

³ <http://www.tripcom.org/>

problematic – many of those meanings form a scenic example of inconsistency, when taken together. For instance, there are several interpretations of what should and what should not be considered ontology. Some of those are in conflict with the most popular definition, [4], as well as with the most widely accepted meanings of terms like schema and dataset. As a terminology roadmap we provide a consistent combination of interpretations of the basic terms, together with analysis and discussion on the distinctions between the different types of information.

Dataset is any set of structured data, as defined in Dublin Core, [11]: “A dataset is information encoded in a defined structure (for example, lists, tables, and databases), intended to be useful for direct machine processing.”

Ontology has different meaning in Philosophy and Computer Science (CS). It was originally defined by philosophers as a discipline concerned with the nature of being and existence. The term has become somewhat overloaded and ambiguous in recent years in the CS community. There are authors which use ontology as a place holder for any sort of KB and even any sort of conceptual model, including such without formal semantic. We find such interpretations ambiguous and confusing and stick to the “classical” definition here.

In the field of CS ontologies have become popular as a paradigm for knowledge representation (KR) in Artificial Intelligence (AI), by providing a methodology for easier development of interoperable and reusable knowledge bases. The first popular definition was given in [20], as follows: “An ontology is an explicit specification of a conceptualization”, where “a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.” The most widely used definition is the one provided by Borst in [4]: “An ontology is a formal, explicit specification of a shared conceptualization.” An extended discussion on the terminology used in the KR community in the early 1990s is provided by Guarino and Giaretta in [21]. Discussion on the different sorts of (upper-level, domain, application) ontologies is provided in [22].

Here we would like to add some comments related to the usage of ontologies in KM, Semantic Web and EAI context. Ontologies can be considered as conceptual schemata, intended to represent knowledge in the most formal and re-usable way possible. Formal ontologies are represented in logical formalisms, such as OWL, [15], which allow automatic inferencing over them. An important role of ontologies is to serve as schemata or “semantic” views over information resources⁴. A semantic annotation or mapping of the scheme of a dataset to an ontology, allows inference on top of the information in the dataset. This way ontologies can be used for indexing, querying, and reference purposes over non-ontological datasets and systems, such as databases, document and catalogue management systems.

Knowledge Base (KB) is a term with a wide usage and multiple meanings. Here we consider a KB as a dataset with some formal semantics. A KB, similarly to an ontology, is represented with respect to a KR formalism, which allows automatic inference. It could include multiple axioms, definitions, rules, facts, statements, and any other primitives. In contrast to ontologies, KBs are not intended to represent a (shared/consensual) schema, a basic theory, or a conceptualization of a domain⁵. As

⁴ Comments in the same spirit are made in [20], [21], and [22].

⁵ The distinction between KB and ontology is further clarified below as well as in section 2.2.3.1.

outlined in [21], ontologies are a specific sort of knowledge bases. They usually provide formal definitions for a shared vocabulary, in terms of:

- **Classes** representing *concepts* we wish to reason about in the given domain (documents, people, organizations, etc.);
- **Relations** – (types of) relations holding between (instances of) those classes; e.g. `hasProvider` can be defined as a relation between a service and the organization providing it. **Attributes** are often introduced as binary relations which relate an instance with data values of primitive types (e.g. character strings, numbers, etc.);
- **Instances** – each instance can instantiate one or more classes, be described by several attributes, and linked to other instances by relations (`<service17 type Service>`, `<service17 hasProvider Org23>`, `<service17 label "...">`);
- **Axioms** – logical expressions, which impose additional knowledge or constraints on the interpretation of the classes, relations, and instances; e.g. a formalization of the following rule: “if a service is free of charge then no payment mechanism shall be specified”.

It is widely recommended that KB, containing concrete data⁶ are always encoded with respect to ontologies, which encapsulate a general conceptual model of some domain knowledge, thus allowing easier sharing, reuse, and interoperability.

1.2 The Borderline between Ontologies and Knowledge Bases

Typically, ontologies designed to serve as schema for KBs do not contain instance definitions, but there is no formal restriction in this direction. Drawing the borderline between the ontology (i.e. the conceptual and consensual part of the knowledge) and the rest of the data, represented in the same formal language (i.e. the KB), is not always a trivial task. For instance, there could be an ontology about tourism, which defines the classes `Location` and `Hotel`, as well as the `locatedIn` relation between them and the hotel attribute `category`. The definitions of the classes, relations and attributes should clearly fit into the ontology. The information about a particular hotel is probably not a part of the ontology, as far as it is not a matter of conceptualization and consensus, but is just a description, crafted for some (potentially specific) purpose. Then, suppose that there is a definition of `New York` as an instance of the class `city` – it can be argued that it is either a part of the ontology or just a description of a city. The fact that it is an instance does not necessarily determine that it is not part of the conceptualization.

As a continuation of the discussion on the distinction between ontologies and KB, let us assume that a knowledge engineer is guided by the principle “no instances in ontologies”. Even in this case there are many examples when one and the same concept can be represented as both class and instance, so, this design principle does not help us always to determine what should be part of a schema-ontology, and what – not. As an example, “VW Golf” (as a model) can be an instance of “VW Car”. However, it also makes sense to define a specific vehicle (e.g. `golf-1234`) of this model as an instance of “VW Golf” (as a class). There is no simple way to determine

⁶ Often referred as instance data, instance knowledge, A-Box, etc.

whether “VW Golf” should be defined as class or instance in this case – such modeling decisions are to some extent a function of the intended use of the ontology.

1.3 Analysis of the Different Types of Information

Below we present a few Boolean (or binary) qualia⁷ of the data relevant to the ontology representation and data integration problems⁸:

- **Semantics:** whether the semantics (the meaning) of the data is formally represented, so that a machine can formally interpret it, reason and derive new data⁹? This quale is directly relevant to reasoning and ontology management – reasoning can only be performed on top of “semantic” data. Non-semantic data could be adapted for reasoning by means of mapping it to an ontology, i.e. a semantic schema which defines the meaning of the data externally. There are marginal cases where the specification of a structure bears elements of semantics, e.g. the case of XML schemata. We stay with a relatively narrow definition of what semantics is and consider semantic data only when there is some logical theory defining meaning associated with the representation language used to represent or interpret the data.
- **Structure:** whether the data is formally structured, so that a machine can formally interpret and manage its structure? This distinction is important because the approaches for automated access and management differ considerably between structured and un-structured data.
- **Schema:** here we consider schematic data, which determines the structure and/or the semantics of other data. The schema quale is determined by the (intended) role of the data with respect to other data. This distinction is relevant within the ontology management context, because schemata are important for mediation and evolution, as they determine the consistency and the interpretation of other data. For instance, a change in an ontology can render a dataset, previously compliant with the old version, incompliant with the new one (or vice versa). Further, in many cases, the problem of data integration can be solved at the level of schema integration.

We introduce below an analysis of the different sorts of data, distinguished with respect to the three qualia. The values for the qualia are given in brackets, “_” stands for “any”; nested bullets are used to represent the hierarchy of the sorts:

- **Data**, (`_`, `_`, `_`). Any sort of data.
 - **Datasets**, (`_`, `structured`, `_`). See the definition above, referring to DC.
 - **Knowledge Bases**, (`semantic`, `structured`, `_`). Any sort of a dataset with a well-defined formal semantics. Those are often referred to as instance datasets or instance knowledge. See the definition in the previous sub-section.

⁷ Quale (pl. Qualia), means a primary intrinsic quality, an independent dimension of classification.

⁸ This analysis was first published in

⁹ The newly inferred data is expected to be correct, indisputable from the human perspective and a consequence of the explicit data.

- **Ontologies**, (`semantic, structured, schema`). See the definition in the previous sub-section. Ontologies are used to prescribe both structure and semantics. For instance, an ontology can define the valid attributes for a specific class (like a database schema can do, too) and, in addition, it can specify the semantics of the attributes.
- **Non-semantic schemata**, (`non-semantic, structured, schema`). Such examples are database and XML schemata.
- **Databases**, (`non-semantic, structured, non-schema`). Here databases are used as a generic term for relational databases, XML-encoded data, comma-separated files, and any other structured, non-semantic data that is not intended to serve as a schema, but rather to represent or communicate particular information. Although this is a slightly misleading name, it reflects the fact that relational databases are the most important sort of non-semantic, non-schema data.
- **Mixed datasets**, (`_, structured, schema&non-schema`). Many catalogues and taxonomies can serve as examples. In such datasets one can often find subsumption chains of the sort Location-City-New York, with no formal indication that the first two are classes (schema) and the third is an instance (non-schema).
- **Content**, (`_, non-structured, _`). Any data without a substantial machine-understandable structure. Such examples are free-text documents, pictures, voice or video recordings, etc. In most of these cases, the non-structured data neither bears machine-interpretable semantics nor plays the role of a formal schema.

1.4 Metadata, Annotations, Semi-Structured Information

Metadata is a term of a wide and often controversial or misleading usage. It cannot be “positioned” in the data-qualia framework, presented in the preceding section, because it is a candidate for an information quale, on its own. From its etymology, metadata is “data about data”. Thus, metadata is a role that certain data could play with respect to other data. Such an example could be a particular (structurally) formal specification of the author of a document, provided independently from the content of the document, say, according to a standard like DC, [10]. RDF(S), [5] and [27], has been introduced as a simple knowledge representation (KR) language that is to be used for the assignment of semantic descriptions to information resources on the web. Therefore an RDF description of a web page represents metadata. However, an RDF description of a person, independent from any particular documents (e.g. as a part of an RDF(S)-encoded dataset), is not metadata – this is data about a person, not about other data. Finally, the RDF(S) definition of the class `Person` should typically be part of an ontology, which can be used to structure datasets and metadata, but which is again not a piece of metadata itself.

Semi-structured data is a term used to refer to two different notions. First, in the KM and NLP communities, semi-structured data are usually considered documents that contain free-text fragments, structured in accordance with some schema. Typical sorts of semi-structured documents are forms and tables, which have some strict structure (fields, parts, etc.), whilst the content of the specific parts of the document is

a free-text. Examples are many administrative, insurance, customs, and medical forms. The second usage of the term “semi-structured” is rather different, denoting non-relational data-models. The intuition is that, whilst with databases there is a predefined, strict structure of specific tables, fields, and views, there are other, “semi-structured”, representations with less strict structuring, which are still not unstructured¹⁰. A number of, more or less, graph-based data-model, like RDF and the Associative Data Model, described in [36], match this understanding of semi-structured data. In both cases, there are two levels of structuring. At the logical¹¹ level, there is a very simple model, which can be used as a general carrier or canvas for the representation of the data. On top of it, there could be a “softer” and much more dynamic schema, which supports the interpretation of the data stored in the basic model. If we take the latter meaning of “semi-structured”, RDFS and OWL are semi-structured representations. However, we strongly disagree with the philosophy behind this usage of semi-structured. Languages and models like RDF(S) allow dynamic and flexible structuring, which, in our view, is a higher degree of structuring, instead of a “semi”-one. Thus, further in this section, we will only-use semi-structured as a term for (text) documents with partial structure (i.e. the first meaning).

Fig. 1. presents a simplified map of the positioning of few popular sorts of information with respect to the presence of formal structure and semantic in them.

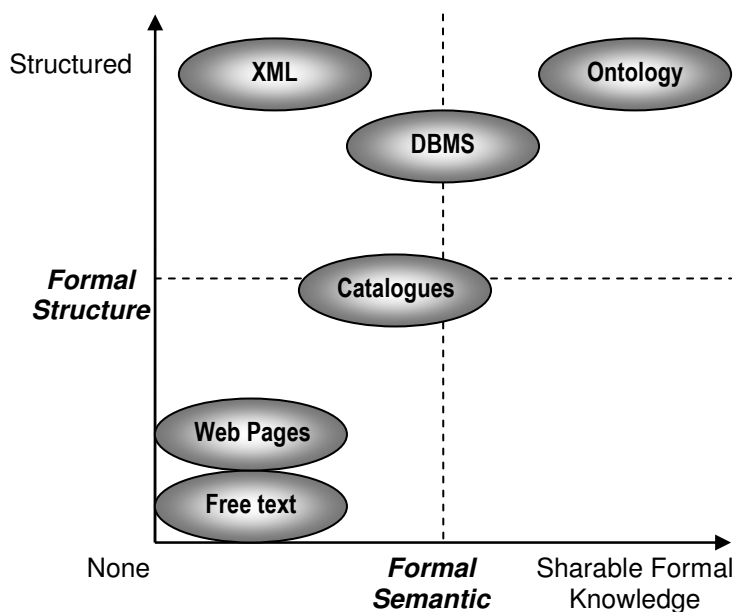


Fig. 1. Structured vs. Semantic Positioning of Different Sorts of Data

¹⁰ See section 3.1.2 of [28] for extended discussion on semi-structured data and its relation to OEM (Object Exchange Model).

¹¹ With regard to the database terminology.

2 Annotations and Reference Models

Annotation is both the process of augmentation of something with links to something else and the result of this process. The later explains why annotation is used as a synonym for metadata, particularly popular in the natural language processing (NLP) community. Here we discuss annotation of documents in general and present a discussion on “semantic annotation” in the context of the Semantic Web.

Annotations on text documents can be distinguished into two groups according to their scope:

- *Document-level annotations*, which refer to the whole document. Such examples are the DC elements (Title, Subject, Creator, etc.);
- *Character-level annotations*, which refer to a fragment of a document, determined by start and end characters. An example might be a comment attached to a particular part of a document. Character-level annotations are usually meant when the term “annotation” is used for text documents without further clarification.

It is worth mentioning that *hyperlinks* can be considered as a specific sort of character-level annotation, when the metadata is, essentially, a reference to another

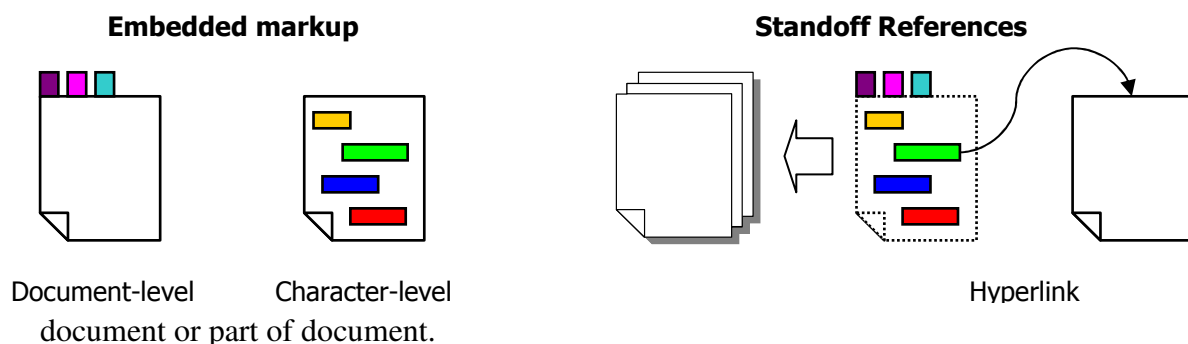


Fig. 2. Types of Annotations

Further, annotations can also be distinguished with respect to the way in which they are attached to the text. The basic choices here are:

- *Embedded mark-up*, when the annotations are incorporated within the document. In this case the metadata is bundled together with the data. Examples are mark-up languages such as HTML, where the annotations are specified through pairs of start and end tags, e.g. `abc<tag>de</tag>gf`. When document-level annotations have to be represented this way, they are sometimes attached in a special section at the start or end of the document – one example is the `<head>` section in the HTML files. An example of character-level embedded annotations is a footnote.
- *Standoff references*, when the annotations are maintained separately from the document to which they refer. In the case of character-level annotations, the reference should also specify the specific part of the document. One approach for this is based on position, e.g. offset and length; another possibility is the

usage of some sort of anchoring and linking mechanism. An example of specification based on standoff annotations is TIPSTER, [18].

The different types of annotation are shown diagrammatically in Fig. 2. In his thesis on architectures for language engineering, [8], Cunningham provides an overview of various annotation models and discusses their advantages and disadvantages in the context of text processing systems and applications. Similar analysis, but in the context of open hypermedia systems (OHS), can be found in [35]. Here we will only briefly mention few of the main characteristics of the embedded and the standoff models:

- Embedded mark-up is not applicable in cases when the author of the metadata has no write-permission to the document;
- Standoff annotations may become inconsistent in the event of change to the document to which they refer;
- Access to embedded annotations requires processing (e.g. parsing) of the documents. Thus, such annotations are not appropriate for applications where random (non-sequential) access to the annotations is important. Conversely, standoff annotations can be maintained and queried efficiently in structured form (e.g. in a database);
- Embedded annotations are simpler to encode, read and manage when the volume of the mark-up is relatively small. However, they are inappropriate when the volume of the mark-up becomes comparable to or bigger than that of the text itself;
- Tagging mechanisms based on embedded annotations have difficulties in handling overlapping (as opposed to nested) annotations;
- Embedded annotations should always be distributed together with the document, which can cause IPR issues, unnecessary redundancy or conflict when multiple sets of annotations are available for one and the same document.

2.1 Semantic Annotations and the Semantic Web

If we abstract the current Web away from the transport, content type, and content formatting aspects, it could be regarded as a set of documents with some limited metadata, attached to them (document-level annotations about title, keywords, etc.), and with hyperlinks between the documents (see the left-hand side part of Fig. 3.).

What does the Semantic Web add to this picture? – Essentially, *semantic* metadata of different kinds, both on the document- and the character-level. Fig. 3. compares links on the current web to those on the semantic web. Typically, the Semantic Web has a greater number of more meaningful annotations, as compared to the current WWW. Many of those annotations represent links to external knowledge, which constitute a new sort of connectivity that is not presented on Fig. 3. , but is extensively discussed in this section (Fig. 4. and Fig. 5.)

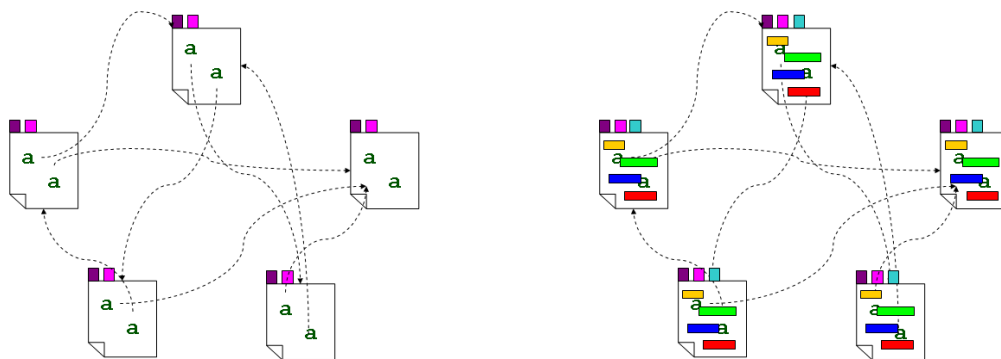


Fig. 3. The Current WWW (left) and the Semantic Web (right)

Suppose, we add a tag <2134> to some portion of a document as follows “... **Abc** <2134>xyz</2134> ...”. Can we call this metadata semantic? – Without further assumptions, the answers are negative. In order to have metadata useful, it should mean something, i.e. the symbols (or expressions or references) that constitute it should allow further interpretation. Interpretation in this context means allowing the assigning of some additional information to the symbols. It is important to realize that interpretation is only possible with respect to something; to some domain, model, context, (possible) world. This is the domain that (the interpretations of) the symbols are “about.” Obviously, annotations in RDF(S), OWL, or some other language refer to a model of the world. Annotations can be expressed in RDF(S), but they are not *about* RDF(S), as depicted on figure Fig. 4.

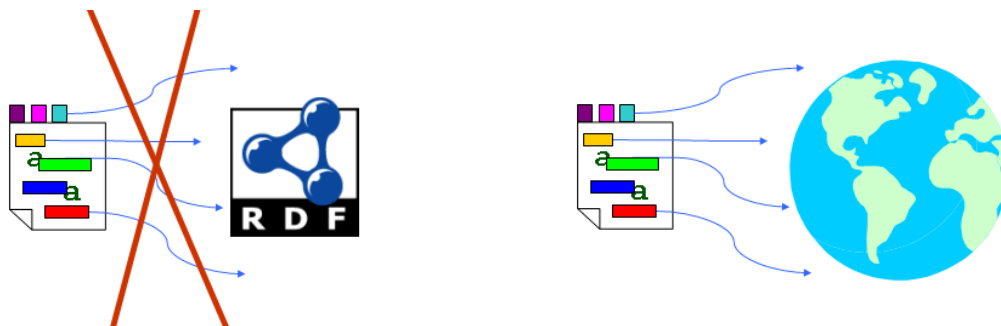


Fig. 4. Metadata about the World, not about RDF

Further, the metadata can hardly refer to (or be interpreted directly with respect to) the world. Such references cannot be formal and unambiguous. What the semantic metadata can be expected to refer to directly is a KB, a formal model of (some aspects of) the world, as depicted in figure Fig. 5. Such a KB specifies some world knowledge which serves as a semantic link from the metadata to the world. Note, that in the Semantic Web context such a KB can be as scattered and heterogeneous as the metadata is. Guha and McCool, [23], consider this KB itself to be the Semantic Web: “the Semantic Web is not a web of documents, but a web of relations between resources, denoting real world objects”. In our view the Semantic Web is the combination of the KB and the semantic annotations referring to it (not just the KB).

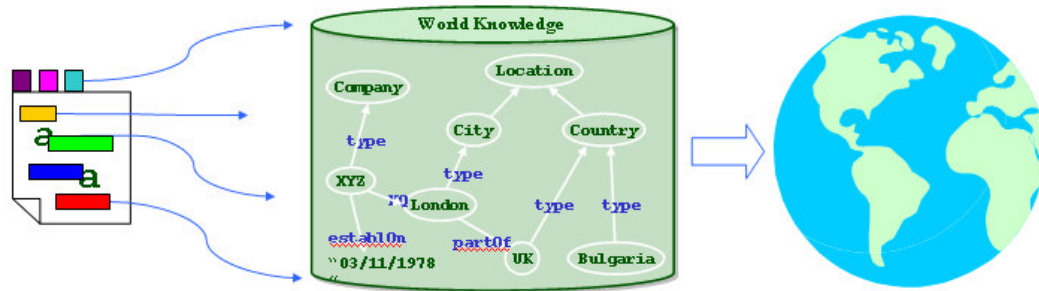


Fig. 5. Metadata Referring to World Knowledge

For automatic processability, the interpretations of metadata should be performed automatically by machines in strict and predictable fashion. This requires a formal definition of how the metadata should be interpreted and, because of this, a formal definition of the context. Assuming that one and the same context can be modelled in different ways, allowing different (and potentially ambiguous) interpretations, what has to be specified is the conceptualization – as defined in [20]: “a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.” This is where ontologies are used to act as logical theories for the “formal specification of a conceptualization” (again in [20], see also section 1.1).

3 Modelling Structured Knowledge

The model for management of heterogeneous information is specified on two layers. In the basis, there is a uniform data-model for management of structured information. It is defined as RDF, extended with support for named graphs and triplesets – groups of “contextualized” triples. A schema for representation of document-related metadata is layered on top of this data-model.

3.1 Modelling Requirements for Structured Knowledge

There are several principle requirements regarding the representation of structured data in the distributed knowledge stores. First of all, they should be able to store and manage any sort of structured information, i.e. all sorts of datasets discussed in section 1.3. This means that it should be at least as powerful as the data models behind the most popular paradigms for management of structured data. Further, the transformation of the data between this model and legacy ones should be straightforward, easy to understand, and to allow for efficient implementation.

Further, this data model shall be aligned with the Semantic Web (SW) standards, for several reasons:

- Most of the contemporary ontology management and reasoning efforts are related to the SW. The most popular ontology language our days is OWL, [15];
- The whole philosophy behind the SW is related to exchange and integration of structured data and metadata in web context. RDF, [27], has been designed to allow for flexible integration of information from diverse data sources with no modelling agreement between each other;
- Serious work on designing framework suitable for structured data interchange has already been performed in W3C’s RDF Access Data Working group (<http://www.w3.org/2001/sw/DataAccess/>).

The above provide sufficient evidence that the data model shall be strongly related to RDF. In this scope, the requirements for a “uniform” structured data-model can be summarized as follows:

- Feasible integration of different structured data sources and more specially relational databases and XML;
- Backward compatibility with the existing RDF specifications and SPARQL query language, [31];
- Easy management of data from several sources within one and the same repository (or computational environment), including:
 - Such are the cases of having data imported from different files, e.g. several ontologies.
 - Efficient processing and storage of meta-data, including such about source and context;
- Grouping statements into manageable collections for the purposes of:
 - Signing and definition of access rights;

- Management of sets of statements which correspond to single construct in a higher level language (e.g. OWL and WSML, [12]);
- Transaction tracking and management.
- Interoperability with the outstanding Semantic Web Services (SWS) formalisms.

3.2 Uniform Structured Data-Model

The uniform structured data model (USDM) that we propose is extension of the RDF data-model. To cope with the requirements, listed in the previous sub-section, we define two extensions: Named Graphs and Triplesets. The RDF model and the extensions are shortly presented below, than we continue with a motivating example, and conclude with comments on reification and links to other models. An extended and more formal presentation of the model is available in section 2 of the ORDI specification, [29].

3.2.1 The RDF Model

RDF, [27], is a metadata representation language, which serves as a basic data-carrier or data-grid for the Semantic Web. It allows resources to be described through relationships to other resources and literals. The resources are identified and referred through URIs (e.g. URLs, [3]). The notion of resource is virtually unrestricted; anything can be considered as a resource and described in RDF: from a web page or a picture published on a web page to concrete entities in the real world, e.g. people, organisations, the number Pi, the musical genre Jazz. Literals (as in XML) are concrete data values e.g. strings, dates, numbers, etc. The main modelling block in RDF is the statement – a triple **<Subject, Predicate, Object>**, where:

- **Subject** is the resource, which is being described;
- **Predicate** is a resource, which determines the type of the relationship;
- **Object** is a resource or a literal, which represents the “value” of the attribute.

A set of RDF triples can be seen as a graph, where resources and literals are nodes and each statement is represented by an arc, labelled with the predicate and directed from the subject to the object. So-called blank nodes can also appear in the graph, representing unique anonymous nodes, used as a sort of auxiliary resources in complex descriptions.

3.2.2 Named Graphs and Datasets

Named Graphs (NG) were introduced in [6], as a mechanism for referring to and describing RDF graphs. The name of each NG is an URI, which can be used to represent the graph; an RDF description for this URI, is considered to represent metadata about the underlying graph. The set of triples which are part of a particular NG shall be explicitly defined, i.e. it may not be derived from the RDF graph. NG can be modelled by quadruples **<Subject, Predicate, Object, NG_Name>** where the first three elements model an RDF statement and the last one represents the name of the NG it belongs to. Let us call this quadruples “contextualized triples” or “contextualized statements”.

One and the same triple can appear in multiple NGs; these appearances should be considered as separate copies of the triple or arcs in the graph. Thus, a set of NG defines RDF multi-graph, where one and the same subject-object pair can be

connected by two arcs, with the same label, which differ only by the NG they belong to. The most typical usage of NG is to model RDF graphs from different data-sources as separate NG.

We adopt from SPARQL, [31], the definition of RDF Dataset as a collection:

$$\{ \mathbf{G}, (\langle \mathbf{U1} \rangle, \mathbf{G1}), (\langle \mathbf{U2} \rangle, \mathbf{G2}), \dots (\langle \mathbf{Un} \rangle, \mathbf{Gn}) \}$$

where \mathbf{G} and each \mathbf{G}_i are RDF graphs, and each $\langle \mathbf{U}_i \rangle$ is a distinct an IRI¹². The pairs $(\langle \mathbf{U}_i \rangle, \mathbf{G}_i)$ represent named graphs. \mathbf{G} is called the default graph – it hosts all the triples which belong to the dataset, but not to any of the named graphs. The notion of default graph is the same, as in Sesame 2.0, [1], despite that graphs are called contexts there and the default one is named “null context”.

3.2.3 Triplesets

Triplesets represent groups (or sets) of contextualized triples. Each triple can be considered as a tag, which can be associated with a specific triple. Triplesets are independent from the NG – triples from different NGs can coexist in one and the same tripleset. In contrast to the NGs, the semantics of the triplesets impose linking of or association triples, instead of copying them. This difference is most obvious with the following two scenarios:

- The removal of a triple from a tripleset, does not reduce the number of the arcs in the RDF (multi-)graph;
- If the arcs in the graph have to be iterated or counted (e.g. in the process of evaluation of a query), the appearance of a (contextualized) triple in several tripleset has no effect – it still counts as a single arc.

An RDF model with NG and triplesets extensions can be modelled via quintuples: $\langle \mathbf{Subject}, \mathbf{Predicate}, \mathbf{Object}, \mathbf{NG_Name}, \mathbf{TripleSet_Name} \rangle$. Formal definition of the tripleset model, including the operations with it, can be found in [29].

3.2.4 Reification

In RDF, reification, [27], allows identifying single triple, so that assertion of facts about it is possible. This mechanism is inefficient, because four additional statements are necessary to reify a single one. As a consequence, several well-known storage implementations like Jena 2, [16], and Oracle Spatial Network Data Model, [2], implement algorithms to optimize the persistence of reified statements and support of contexts by using quadruples internally.

The approach for reification proposed here is usage of singleton triplesets, e.g. whenever statements have to be made about a single statement \mathbf{s} , an auxiliary tripleset \mathbf{ts} is created and \mathbf{s} is associated with it. The name of the tripleset is later on used as an URI of the statement and subject of any statements about \mathbf{s} .

This approach can be easily extended to mimic the standard RDF reification mechanisms, if necessary. For instance, \mathbf{ts} can be declared of type `rdf:Statement` and it can be linked though `rdf:subject`, `rdf:predicate`, and `rdf:object` to the corresponding elements of the triple. A generic implementation of reification in this shape would be as inefficient as any other representation. An optimized schema could be implemented under demand of a specific application.

¹² IRI is the internationalized form of URI, <http://www.ietf.org/rfc/rfc3987.txt>

3.2.5 Tripleset Usage Example

Imagine an RDF repository which holds the customer relationship-related information (CRM) of a company. This data which comes from different sources, represented as named graphs: a CRM database (**ng1**); Yellow Pages-like catalogue (**ng2**); a public companies database (**ng3**), providing data about managers and financial figures.

Fig. 6. depicts a snapshot from the RDF graph, where **pe1** is the contact person for customer **co1**, which has offices in locations **lo1** and **lo2**. As we can see, some of the triples come from more than one data-source; for instance, those defining the position of **pe1** within **co1**. On the other hand, **ng1** and **ng3** provide different variations of the name of the person. Further, **n1** and **n2** “agree” that **co1** has an office in **lo1**, but only **n2** “knows” that it has also office in **lo2**. Note that the corresponding arcs are duplicated, so, if one of them gets deleted, the other remains there to state that this fact is still true according to a different source.

Suppose that access control policy shall be implemented, so, that different people (having accounts associated with different roles) have access to different parts of this database. The data visible to the different roles can be encoded in different triplesets.

- Sales manager (any tripleset): no constraints;
- Account manager (**ts1**): has access to all the CRM information, except the financial figures from **ng3**;
- Sales assistant (**ts2**): has access to the contact information only.

The association of the contextualized triples, with the corresponding triplesets is indicated by means of tags on Fig. 6.

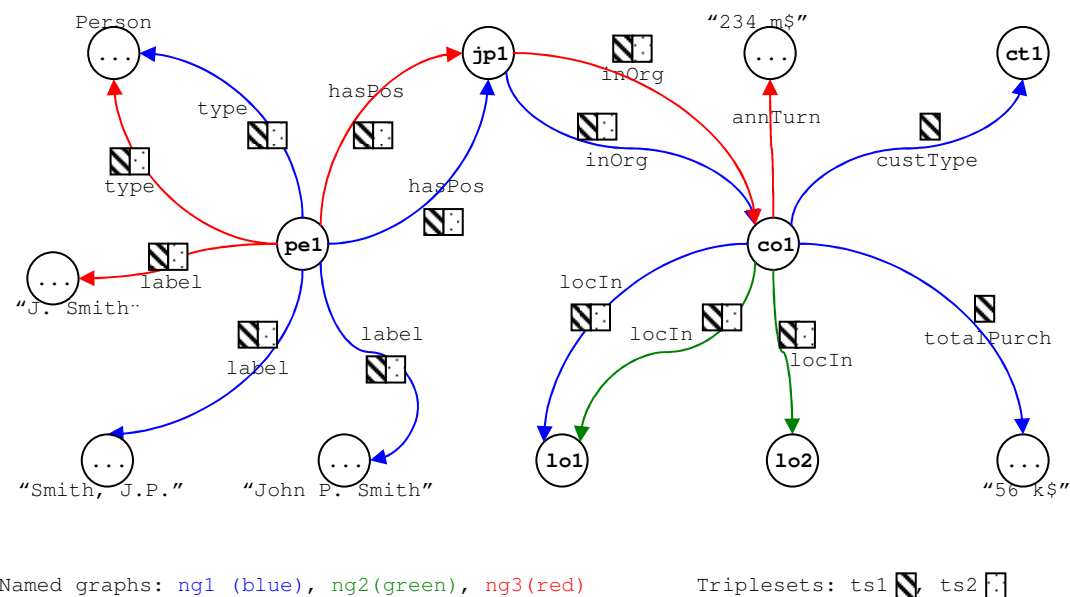


Fig. 6. CRM data from different sources, having different access types

3.3 Interoperability with WSML

Our days, there are two outstanding Semantic Web Services (SWS) initiatives, with their corresponding sets of specifications and tool support: OWL-S, [9], and WSMO, [32]. OWL-S is an ontology, clearly defined on top of OWL, so, the interoperability with the extension of the RDF model we propose here is out of question.

WSML, [13], is a Web Service Modeling Language which provides a formal syntax and semantics for the Web Service Modeling Ontology (WSMO). WSML rerepresents a combination of an ontology and SWS language, so, it provides an alternative to OWL and OWL-S, taken together. The top-level entities which can be modelled in WSML are: ontologies, web services, goals, and mediators. The language has the following dialects: Core, DL, Flight, Rule, Full; those correspond the logical formalism, which define their semantics.

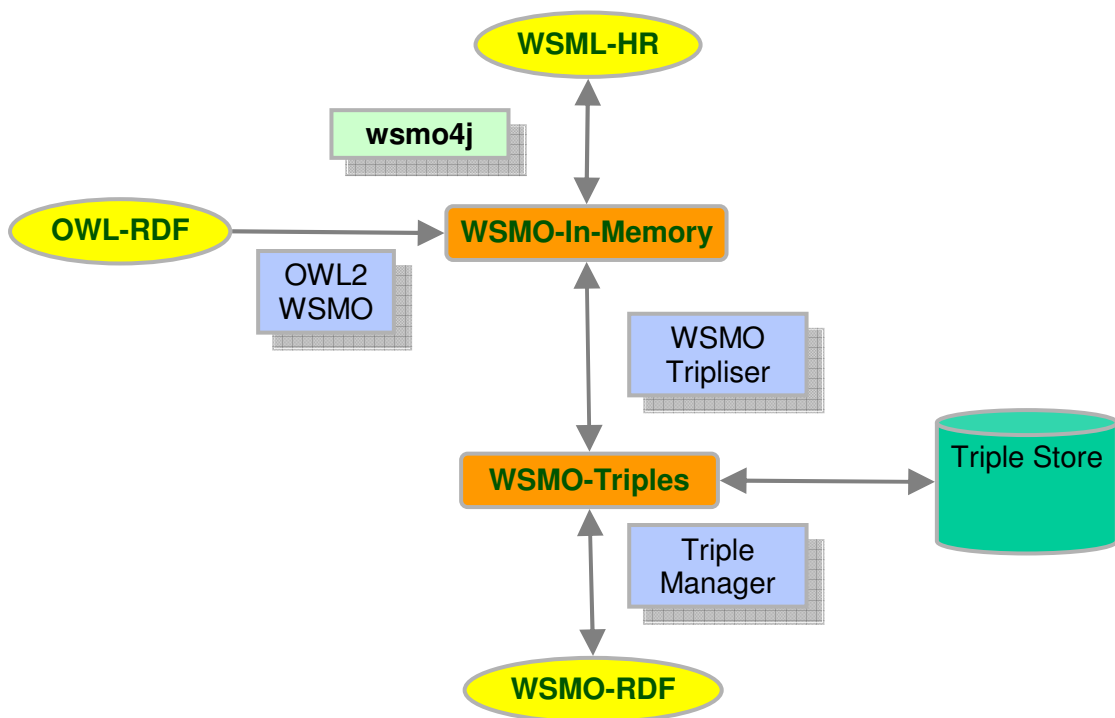


Fig. 7. WSML-to-RDF/OWL Interoperability

While the epistemology of WSML and RDF(S)/OWL differ considerably, the underlying data models are fairly similar. As in RDF, the elementary nodes are: IRIs, data values (compliant with the XML literals), and anonymous identifiers (mappable to RDF's blank nodes). The interoperability between WSML and RDF is defined in the WSMO/RDF mapping, [14]. Despite the several problems discussed there, WSMO/RDF effectively provides a syntax for managing WSML. Those problems are addressed by the model presented here, as discussed in section 3.1 of [29]. Fig. 7. demonstrates the interoperability between OWL/RDF and the different syntaxes (in ellipses) and representations (in rectangles) of WSML, in the way in which it is implemented in *wsmo4j*, [30], and the first generation of ORD, [25]. The partial interoperability between OWL-DL and WSML-Core is defined in section 11 of [13].

4 Modelling Annotations

A schema is proposed here for management of document annotations on top of the Uniform Structured Data-Model, presented in section 3. The schema is focused on semantic annotation with respect to entities, i.e. instances of specific ontology classes. The schema allows for annotation with information about both the class and the instance identifier of an entity. The motivation about this is as follows:

- The annotation with general concepts and/or classes can be modelled as a specific case of the above;
- Named entity (NE, see section 4.1.1) references turn to be rather characteristic for unstructured documents. Evidence for the importance of the named entities is demonstrated by a recent large-scale human interaction study on a personal content information retrieval system of Microsoft, [17].
- Entities are easy to link to instance data within databases and other structured data sources;
- Named-entity annotation is very basic and important scenario for GATE, [33], which is the subject of the first case study.

4.1 Semantic Annotation of Named Entities

Semantic annotation of named entities (SANE) is a specific metadata generation process, aiming to enable new information access methods and to extend some of the existing ones. Moreover, via the use of KBs of external background knowledge, those NE can be related to formal descriptions of themselves and related entities and thus provide more semantics and connectivity to the web or two other structured data.

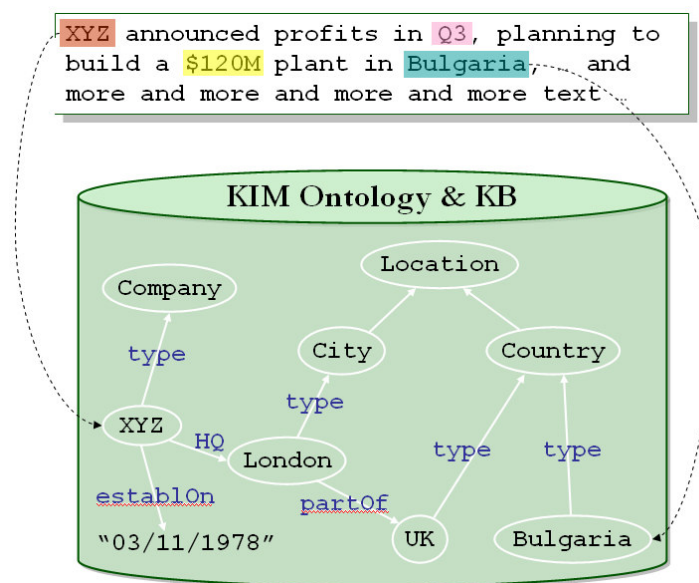


Fig. 8. Semantic Annotation

In a nutshell, SANE is character-level annotation of mentions of entities in the text with references to their semantic descriptions (as presented in Fig. 8.) This sort of metadata provides both class-level and instance-level information about the entities. Such semantic annotations enable many new types of applications: highlighting,

indexing and retrieval, categorization, generation of more advanced metadata, smooth traversal between unstructured text and available relevant structured knowledge. Semantic annotation is applicable for any sort of text – web pages, non-web documents, text fields in databases, etc. Further knowledge acquisition can be performed on the basis of the extraction of more complex dependencies – analysis of relationships between entities, event and situation descriptions, etc.

In other words, SANE solves the following set of tasks: identify and mark references to named entities in textual (parts of) documents and link these references to descriptions of the entities in a KB.

4.1.1 Named Entities

In the Natural Language Processing (NLP) field, and particularly the Information Extraction (IE) tradition, **named entities** (NE) are considered: *people, organizations, locations*, and others, referred to by name, [7]. By a wider interpretation, these also include scalar values (*numbers, dates, amounts of money, addresses*, etc).

NEs should be handled in a different, special way because of their different nature and semantics compared to general words (terms, phrases, etc.) While the former denote particulars (individuals or instances), the latter typically denote universals (concepts, classes, relations, attributes). Even a basic level of formal semantic definition of general word senses involves modeling of lexical semantics and common sense¹³. On the other hand, useful descriptions of named entities can be modelled on the basis of much simpler and more specific “factual” world knowledge.

4.2 Modelling Requirements for Semantic Annotations

The basic knowledge modelling prerequisites for the representation of semantic annotations of named entities are:

- **Ontologies**, defining classes and general concepts and allowing unambiguous references to those;
- **Instance or entity identifiers**, which allow these to be distinguished and linked to their semantic descriptions;
- **Knowledge bases**, providing entity descriptions. Although those are not explicitly necessary for the creation of the annotations, without such the semantic annotations bear no semantics.

Entity descriptions actually make up the non-ontological part of formal knowledge in the semantic repository. The entity descriptions represent a KB, a body of instance knowledge or data. Such KB can either be available as pre-populated background knowledge and/or be extended through information extraction from the documents.

As with other sorts of annotations, major question is whether those should be embedded or stand-off. There are a number of arguments, giving evidence that semantic annotations are best decoupled from the content they refer to. One key reason for this is the ambition to allow for dynamic, user-specific, semantic annotations – conversely, embedded annotations become a part of the content and may not change according to the interest of the user or to the context of usage.

¹³ WordNet is the most popular large scale lexical database, providing partial descriptions of the word senses in the English language. It can be considered also as a lexical ontology or a KB. See the discussion around WordNet and its usage for semantic annotation and IR in Section 2 of Chapter 4.

Further, complex embedded annotations would have a negative impact on the volume of the content and could complicate its maintenance – e.g. imagine that a page with three layers of overlapping semantic annotations needs to be updated without compromising their consistency.

Given that semantic annotations should preferably be kept separate from the content to which they refer, the next question is whether or not (or to what extent) the annotations should be integrated with the ontology and the KB. It is the case that such integration seems profitable – it would be easier to keep the annotation in sync with the class and entity descriptions. However, there are at least three important considerations to be made in this regard:

- Both the number and the complexity of the annotations differ from those of the entity descriptions – the annotations are simpler, but more numerous than the entity descriptions. Even considering middle-sized corpora of documents, the number of annotations typically reaches tens of millions. Suppose that 10M annotations are stored in an RDF(S) store together with 1M entity descriptions. Suppose also that on average annotations and entity descriptions are represented with 10 statements each. The difference, regarding the inference approaches and the hardware that is capable of efficient reasoning and access to a 10M-statement semantic repository and to a 110M-statement repository, is considerable.
- Separation of concerns: if the world knowledge (ontology and instance data) and the document-related metadata are kept independent, this would mean that for one and the same document, different extraction, processing, or authoring methods will be able to deliver alternative metadata, referring to one and the same knowledge store.

4.3 Document and Annotation Modelling

The approach undertaken in TAO is that the Heterogeneous Knowledge Stores will be scalable and manageable to a degree in which semantic annotations can be stored and managed in the semantic repository together with the rest of the structured data. This will allow for efficient evaluation of heterogeneous queries, in the cases in which such setup is suitable.

Still, in the cases when the annotations need to be decoupled from the rest of the knowledge (e.g. for IPR reasons), the following architecture will be possible:

- The annotations will be kept in a separate knowledge store, which implementation may employ storage, indexing and inference mechanisms suitable for the volume and nature of this type of data;
- Knowledge store federation (see section 5) can be applied to provide integrated view and allow queries involving all sorts of information.

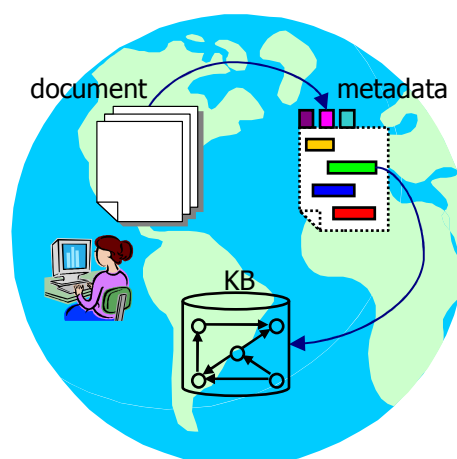


Fig. 9. Distributed Heterogeneous Knowledge

Fig. 9. presents the distributed scenario of decoupled representation and management of the documents, the metadata (annotations), and the formal knowledge (ontologies and instance data). The interoperability with “all-in-one” annotation models is still possible. If, for instance, there is an HTML document with RDF annotation as part of it, which define both a new ontology and provides several instance descriptions, the following strategy is applicable:

- Extract the RDF descriptions and store them into a semantic repository. This is easy as long as RDF is integrated as specific sort of comments into HTML;
- Convert the inline HTML tags into stand-off annotations. GATE already offers such conversion, so, at the technical level this operation is straightforward. Of course, one needs a special purpose pre-processing to identify the semantic annotations and filter out the layout-related HTML metadata (e.g.).

4.4 Annotations Encoding Shema

PROTON, [34], is a light-weight general-purpose upper-level ontology defining about 300 classes and 100 properties in OWL DLP, [19], – a tractable fragment of OWL, less expressive than OWL Lite and suitable for extension towards both Description Logics and Logical Programming. PROTON is split into four modules: System, Top, Upper, and KM (Knowledge Management). The KM module was developed in the course fo the SEKT project, to facilitate semantic metadata extraction and KM applications.

We propose TAO to adopt as a starting point, the schema for encoding of semantic annotation from the KM module of PROTON. Documents are modelled, in accordance with Dublin Core, as a sub-class of **InformationResource**. Each document is a resource, which instantiates this class (or a sub-class of it) and has an URI, so, the association of document-level annotations is straightforward. Character-level annotations are modelled as instances of class **Mention** – each instance of this class represents the mention of an **Entity**, or a class, within an instance of **InformationResource**. The properties, currently defined for the instances of **Mention** are as follows:

- **hasStartOffset** – start offset in the content of the information resource;
- **hasEndOffset** – end offset in the content of the information resource;
- **hasString** – the string of the annotation, if such.
- **occursIn** – relates the mention to the document it appears in;
- **refersInstance** – relates **Mention** with **Entity**.

Additional property **refersClass** needs to be defined, which refers to the class of the entity. It should be an instance of **owl:ObjectProperty**, with domain **Mention** and range **rdfs:Class**. This property shall however be used with special care, only in scenarios when it is necessary. In a typical situation an entity belongs to several classes, either due to multiple explicit type statements or due to class subsumption or other inference. The usage of **refersClass** could be a source for a whole range of representation and inference inefficiencies. Having said this, an updated schema, which contains definition of **refersClass**, will be made available to the consortium by month twelve of the project. It can also host extensions towards annotation of multimedia documents, should such appear necessary.

5 Knowledge Store Architecture

This section comments on architecture of heterogeneous knowledge stores, supporting the unified structured data-model, defined in section 3.2, as well as several other functional requirements as listed in the first sub-section below. The knowledge stores will be developed on top of the second generation of the ORDI framework. An extended description of ORDI's architecture, including diagrams of the most important application programming interfaces (API), can be found in [29].

5.1 Architectural Requirements

There are several major requirements towards the architecture of the heterogeneous knowledge stores:

- They should allow for different (alternative) implementations of the underlying storage and inference modules;
- Tools, designed to use the knowledge stores, should be independent from the implementation of the specific store;
- It should be possible to augment the behaviour of the knowledge stores while retaining the same interfaces.
- Wrapping and integration of different structured data sources should be possible, so, that the data from this sources can be integrated through the knowledge store interfaces, in accordance with the unified structured data model;
- Data federation should be supported, so, that information from several independent knowledge stores (or data sources wrapped as such) could be accessed in integrated fashion, maintaining the abstraction that the data reside in a single store;
- Scalability through distribution – it should be possible that the data is spread across several machines in order to improve the scalability of the stores. In this case, the separate machines may not be able to expose access to fully-functional knowledge stores. This aspect shall be further clarified in deliverable D4.3.

Several non-functional requirements such as security, quality-of-service control, and manageability are not commented here.

5.2 Architecture Overview

The knowledge store specification defines middleware architecture. In its core is a set of interfaces, which determine the interoperability patterns of a repository, allowing management (storage, modification, retrieval) of RDF with the above mentioned extensions. The central interface there, named **store**, is designed to allow for multiple implementations of stores and extensions. Wrappers for data management systems (e.g. RDBMS) shall be developed as implementations of **store**. Higher-levels of modelling – document annotations, concrete ontology languages (e.g. WSML) – shall be implemented as extensions of the **store** interfaces

The architecture is defined as an extension of the Sesame 2.0, [1], architecture. Sesame is one of the most mature, popular, and efficient, RDF management

frameworks. This approach allows for re-use of mature RDF-management infrastructure and better interoperability with it. The architecture of ORDI framework is shown on Fig. 10.

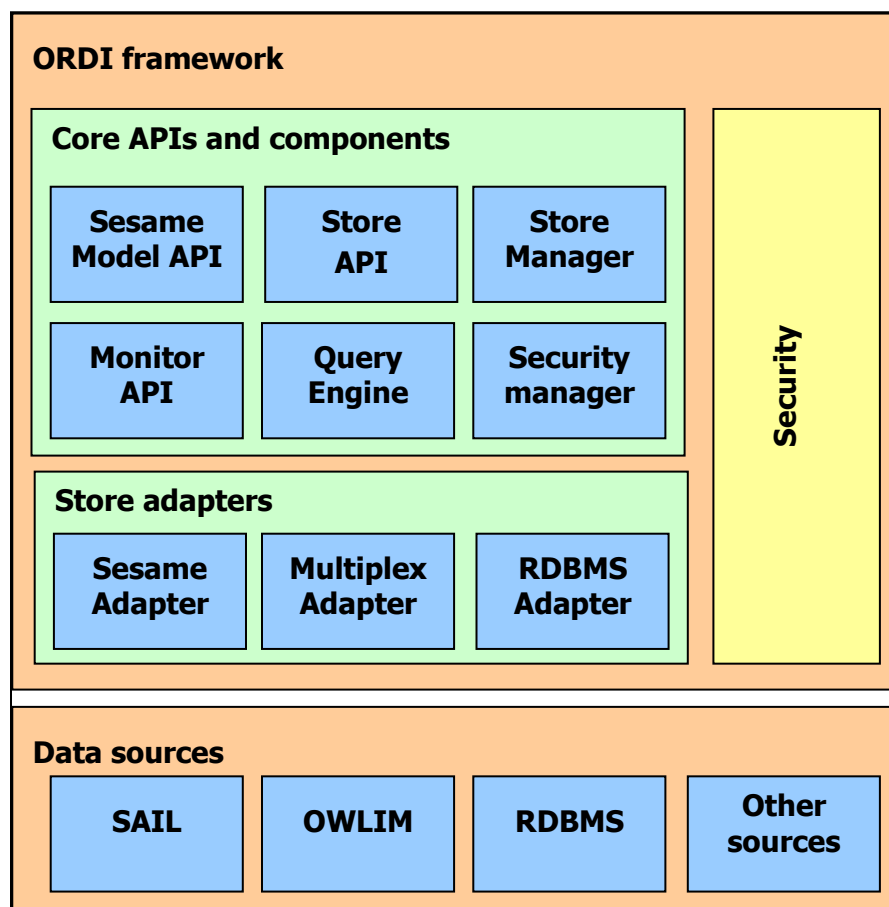


Fig. 10. Architecture of ORDI framework

ORDI framework is composed by core APIs and components and store adapters. The core components and APIs define the main functionality of the framework by introducing set of Java interfaces and implementations to interact with them:

- **Sesame Model** is a lightweight API. It is responsible to describe generic concepts like RDF statement, resource, literal, blank node and URI; more information about it is available at [1]. ORDI framework reutilizes the generic packages `org.openrdf.model` part of Sesame in order to ensure compatibility with the high performance RIO parsers implementations.
- **Store API** describes how the different framework components to interact with the functionality offered by the contained stores. A design principle is to allow no components to interact with implementation outside of its component scope.
- **Store Manager** is some sort of data store registry. It provides basic operation to manipulate the available configured data stores, based on the functionality they implement.

D4.1 / Model and Specification for Distributed Knowledge Stores

- **Monitor API** is still undefined. The requirements towards it are to provide sophisticated mechanism to trace internal system functionality and to optimize the operation over the framework.
- **Query Engine** process SPARQL, [31], based queries and creates an executable conjunction of patterns to be matched against any arbitrary data store. A mechanism for support of multiple different query languages will be added to the framework at a later phase.
- **Security Manager** is component responsible to define the security across the different system parts. Its functionality is still undefined, because of lack of significant requirements

The ORDI store adapters realize the defined interface by the Store API. They could be divided to two main groups concerning the added functionality to the framework:

- Adapters for data-sources. They ensure a mapping between an interface part of the Store API and another one supported by third-party persistent store. Such adapters are the Sesame adapter to map the **TripleSet** data model to SAIL, or RDBMS adapter to transform legacy database information data set to ontology, based on specified schema. The mapping depending on the implementation could be complete or partial; and supported in one or the both direction.
- Adapters to provide new functionality. They work over adapter to provide data to introduce new functionality. Example for functional adapter is the Multiplex Adapter that implements the **MultiTripleSetModel** to union the information located in several tripleset stores.

6 Conclusion

Specification was provided for Heterogeneous Knowledge store, which allows for efficient management of unstructured content (e.g. documents), structured data (e.g. relational databases), ontologies, and semantic annotations, augmenting the content with links to machine-interpretable metadata.

Conceptual groundwork took place in the first couple of sections. We started with discussion on the ontology-related terminology, then provided analysis on the different sorts of data and used it to propose consistent interpretation on the different terms on its basis. Analysis of annotation representation models was presented, to finalize the introduction needed for the specification.

The model for representation of heterogeneous information was specified on two layers. In the basis, there is a uniform data-model for management of structured information. It is defined as RDF, extended with support for named graphs and triplesets – groups of “contextualized” triples. A schema for representation of semantic annotations for documents is layered on top of this data-model.

The knowledge store specification represents middleware architecture. In its core is a set of interfaces, which determine the interoperability patterns of a repository, allowing management (storage, modification, retrieval) of RDF with the above mentioned extensions. The central interface there, named **store**, is designed to allow for multiple implementations of stores and extensions. Wrappers for data management systems (e.g. RDBMS) shall be developed as implementations of **store**. Higher-levels of modelling – document annotations, concrete ontology languages (e.g. WSML) – shall be implemented as extensions of the **store** interfaces.

The knowledge store architecture is specified on the basis of the second generation of the ORDI framework. Its specification, [29], is complementary to this deliverable, as it provides further details on two important aspects: the uniform data-model for representation of structured data and the store architecture. The specification of ORDI will be updated in the course of its development. The short term development plans can be summarized as follows:

- **ORDI v0.4.1** (Nov 2006): a more stable and mature release including SPARQL support; the WSML/RDF mapping will be synchronized with a next version of its specification, [14]; Developer’s Guide to be developed.
- **ORDI SG API** (Dec 2006): the API for the second generation of ORDI will be developed in compliance with Sesame 2.0;
- **First version of ORDI SG, v.0.5** (Jan 2007): the API will be implemented on top of OWLIM, to deliver the first functional prototype.
- **wsmo4rdf v0.1** (Feb 2007): an implementation of the WSMO/RDF mapping as an extension of ORDI SG v.0.5; it will cast ORDI v.0.4;
- **wsmo4rdf v0.2** (Mar 2007): the second version of the wsmo4rdf extension will implement scalable WSML-Core reasoning;
- **Semantic Annotation Store specification** (Mar 2001): specification for ORDI extension supporting storage and management of semantic annotations;
- **Semantic Annotation Store v0.1** (Apr 2001): first implementation.

D4.1 / Model and Specification for Distributed Knowledge Stores

The most notable subjects of middle- and long-term development are: adapter for integration of RDBMS and multi-store implementation allowing for data consolidation.

The specification provided here will serve as a basis for the implementation of Heterogeneous Knowledge Store, in deliverable D4.2 and its extension with support for distributed repositories in D4.3. It is also relevant to the overall architecture and integration specification (D5.2), as one of TAO's key components is specified here. The developments in WP2 and WP3 are related to the knowledge stores, as long as they should store the extracted ontologies and the metadata on the legacy content there.

7 Bibliography and references

- [1] Aduna B.V. 2006. *User Guide for Sesame 2.0*. DRAFT Updated for Sesame release 2.0-alpha-2. <http://www.openrdf.org/doc/sesame2/users/>
- [2] Alexander, N; Ravada, S. *RDF Object Type and Reification in Oracle*. http://download-uk.oracle.com/otndocs/tech/semantic_web/pdf/rdf_reification.pdf
- [3] Berners-Lee, T; Fielding, R; Irvine, U.C; Masinter, L. 1998. *Uniform Resource Identifiers (URI): Generic Syntax*. Network Working Group, Request for Comments: 2396. <http://www.ietf.org/rfc/rfc2396.txt>
- [4] Borst, P.; Akkermans, H.; Top, J. 1997. Engineering Ontologies. *International Journal of Human-Computer Studies*, (46)365-406, 1997.
- [5] Brickley, D; Guha, R.V, eds. 2000. *Resource Description Framework (RDF) Schemas, W3C*, <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>
- [6] Carroll, J. J; Bizer, B; Hayes, P; Stickler, P. 2005. *Named Graphs, Provenance and Trust*. WWW2005, <http://www2005.org/cdrom/docs/p613.pdf>
- [7] Chinchor, N., Robinson, P. 1998. *MUC-7 Named Entity Task Definition* (version 3.5). In Proc. of the MUC-7.
- [8] Cunningham, H. 1999. *Information Extraction: a User Guide* (revised version). Department of Computer Science, University of Sheffield, May, 1999.
- [9] DAML. *OWL-S 1.1 Release*. <http://www.daml.org/services/owl-s/1.1/>
- [10] DCMI Usage Board. 2005. *DCMI Metadata Terms*. <http://dublincore.org/documents/2005/06/13/dcmi-terms/>
- [11] DCMI Usage Board. 2003. *DCMI Type Vocabulary*. <http://dublincore.org/documents/2003/11/19/dcmi-type-vocabulary/>
- [12] de Bruijn, J; Kopecky, J; Krummenacher, R. 2005. *WSML - a Language Framework for Semantic Web Services*. <http://www.w3.org/2004/12/rules-ws/paper/44/>
- [13] de Bruijn, J; Lausen, H; Krummenacher, R; Polleres, A; Predoiu, L; Kifer, M; Fensel, D. 2005. *D16.1v0.21 The Web Service Modeling Language WSML*. WSML Final Draft 5 October 2005, DERI. <http://www.wsmo.org/TR/d16/d16.1/v0.21/>
- [14] de Bruijn, J. (ed.); Kopecky, J; Krummenacher, R. 2006. *WSML/RDF*. Deliverable d32v0.1. WSML Working Draft 15 February 2006. <http://www.wsmo.org/TR/d32/v0.1/20060215/>
- [15] Dean, M; [Schreiber](#), G. – editors; Bechhofer, S; van Harmelen, F; Hendler, J; Horrocks, I; McGuinness, D. L; Patel-Schneider, P. F.; Stein, L. A. (2004). *OWL Web Ontology Language Reference*. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/owl-ref/>
- [16] Dollin, C; McBride, B. *Jena2 reification API proposal*. <http://jena.sourceforge.net/reify-api.html>
- [17] Dumais, S., Cutrell E., Cadiz J., Jancke G., Sarin R. and Robbins D. *Stuff I've Seen: A system for personal information retrieval and re-use*. In proc. of SIGIR'03, 2003, Toronto, ACM Press.
- [18] Grishman, R. *TIPSTER Architecture Design Document Version 2.3*. Technical report, DARPA, 1997. http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/
- [19] Grosz, B; Horrocks, I; Volz, R; Decker, St. (2003). *Description Logic Programs: Combining Logic Programs with Description Logic*. In Proc. of WWW2003, Budapest, May 2003.
- [20] Gruber, T. R. 1992. *A translation approach to portable ontologies*. *Knowledge Acquisition*, 5(2):199-220, 1993. http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html

D4.1 / Model and Specification for Distributed Knowledge Stores

- [21] Guarino, N.; Giaretta, P. 1995. *Ontologies and Knowledge Bases: Towards a Terminological Clarification*. In N. Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. IOS Press, Amsterdam: pp. 25-32.
- [22] Guarino, N. 1998. *Formal Ontology in Information Systems*. In N. Guarino (ed.) *Formal Ontology in Information Systems*. Proceedings of FOIS'98, Trento, Italy, June 6-8, 1998. IOS Press, Amsterdam, pp. 3-15.
- [23] Guha, R.; McCool, R. 2003. *Tap: A semantic web platform*. *Computer Networks*, 42:557 – 577, 2003.
- [24] Kiryakov, A; Ognyanov, D; Kirov, V. 2004. *D2.2: An Ontology Representation and Data Integration (ORDI) Framework*. DIP project deliverable. <http://dip.semanticweb.org>.
- [25] Kiryakov, A; Ognyanov, D; 2006. *An Ontology Representation and Data Integration (ORDI) Framework Implementation*. DIP project deliverable D2.3. ver. 0.4, Jun 2006. <http://www.ontotext.com/ordi/v0.4/FactSheet.html>
- [26] Kiryakov, A.; Popov, B.; Ognyanov, D.; Manov, D.; Kirilov, A.; Goranov, M. 2004a. *Semantic Annotation, Indexing, and Retrieval*. Elsevier's Journal of Web Semantics, Vol. 1, ISWC2003 special issue (2), 2004. <http://www.websemanticsjournal.org/>
- [27] Klyne, G; Carroll, J. J. 2004. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C recommendation 10 Feb, 2004. <http://www.w3.org/TR/rdf-concepts/>
- [28] Martin-Recuerda, F; Harth, A; Decker, S; Zhdanova, A; Ding, Y; Stollberg, M. 2004. *Deliverable D2.1 "Report on requirements analysis and state-of-the-art" within WP2 "Ontology Management" of the DIP project*. <https://bscw.dip.deri.ie/bscw/bscw.cgi/0/3012>
- [29] Momtchev, V.; Kiryakov, A. (2006). *Second Generation Ontology Representation and Data Integration (ORDI) Framework: Specification*. Ontotext white paper. Oct. 2006. http://www.ontotext.com/ordi/ordi_sg_spec.pdf
- [30] Ontotext Lab. 2006. *wsmo4j: an API and a reference implementation for building Semantic Web Services applications compliant with the Web Service Modeling Ontology*. <http://wsmo4j.sourceforge.net/>
- [31] Prud'hommeaux, E; Seaborne, A. 2006. *SPARQL Query Language for RDF*. W3C Working Draft 4 October 2006. <http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/>
- [32] Roman, D; Lausen, H; Keller, U. (eds.) 2005. *Web Service Modeling Ontology (WSMO)*, WSMO deliverable D2v1.2. WSMO Final Draft, 13 April 2005. <http://www.wsmo.org/TR/d2/v1.2/>
- [33] Sheffield University, Natural Language Processing Group. 2006. *GATE, A General Architecture for Text Engineering*. <http://gate.ac.uk/>
- [34] Terziev, I.; Kiryakov, A.; Manov, D. 2005. *D1.8.1. Base upper-level ontology (BULO) Guidance*, report EU-IST Integrated Project (IP) IST-2003-506826 SEKT). Ontotext Lab. July, 2005. http://proton.semanticweb.org/D1_8_1.pdf
- [35] van Ossenbruggen, J., Hardman L., Rutledge L., *Hypermedia and the Semantic Web: A Research Agenda*. *Journal of Digital information*, volume 3 issue 1, May 2002.
- [36] Williams, S. 2002. *The Associative Model of Data*. Second Edition, Lazy Software Ltd. ISBN: 1-903453-01-1. <http://www.lazysoft.com>