Using Semantic Web Spaces to Realize Ontology Repositories

Elena Paslaru Bontas, Lyndon J. B. Nixon, Robert Tolksdorf
paslaru, nixon, tolk@inf.fu-berlin.de
Freie Universität Berlin
Institut für Informatik
AG Netzbasierte Informationssysteme
Takustr. 9, D-14195 Berlin Germany

1 Introduction

As the Semantic Web grows, increasing numbers of private and public sector communities will be developing ontologies which represent their domain(s) of interest. As ontologies are also intended to act as *shared* domain conceptualizations [Gr95] it is expected that ontology developers may also wish to make their ontologies available to other users, possibly under some license. This will bring the benefit of ontology developers being able to re-use existing models or align them to local ones, thus reducing implementation costs, improving the quality of ontological sources, which are, by re-use, subject of continuous revisions and refinements, and increasing the interoperability among ontology-based applications. Besides, access to available ontologies across the Web is a fundamental requirement for the dissemination of Semantic Web Services, which are envisioned to automatically use these sources in order to describe their capabilities and for interprocess communication.

In this paper we consider the current state of the art of the field "Ontology Discovery" and note the outstanding requirements that arise in a real world scenario making use of ontologies across the Web. [1]In order to meet these requirements, we first examine existing ontology repositories on the Web and the technologies underlying these repositories and find them insufficient. We propose the use of a tuplespace-based system as a middleware platform for administrating ontologies and their (semantic) descriptions which can act as a fully-fledged ontology repository. This middleware platform is Semantic Web Spaces [TNL$^+$04, TNPB$^+$05, TPBN05, PBNT05]. We define how the space would store ontologies and how their descriptions would also be modeled as OWL-based metadata. Finally, we illustrate how a Semantic Web Space would function as Ontology Repository and conclude with our observations upon the proposed approach and future work.

[1]By "Ontology Discovery" we mean the activities required for finding ontologies fulfilling user-defined requirements on the Web or in particular Web-accessible repositories.

2 Ontology discovery today

At present the question of how existing ontologies can be found is not trivial. On one hand, the ontology developer could attempt a Web-wide search using a standard search engine such as Google,[2] or choose a Semantic Web-specific search tool such as Swoogle.[3] On the other hand, he or she could try to decide which organizations best represent the domain that is to be modeled and attempt to seek if they have made any ontologies/classifications public. If we consider the case of a supermarket seeking to model its product range, the domain would be 'food and drink'. The latter approach could lead to contacting the Food Marketing Institute, while the former would try to find relevant ontologies using search criteria like 'food and drink filetype:owl' (which in Google finds ontologies on Russia, Geography and TerroristActs!).

Alternatively, ontologies could be grouped into repositories. As long as the developer knew how to access the repository he or she could query there for relevant existing ontologies. The DAML Ontology Library is one of the most representative examples, offering a simple Web-based interface to the source ontologies, in which the ontology user can access them according to different properties (e.g. URI, Keyword, Submitting Organization) or express queries in terms of ontology classes and properties.[4] The keyword view helps us find ontologies for Beer and Wines (there are Food keywords but they are CYC ontologies which are not aimed at product classification). The query is more detailed, as it matches on substrings of 'food' or 'drink', but it is difficult to see from the result list which ontology best classifies the topic. Other examples of ontology collections are the Protegé repository,[5], SchemaWeb[6] (including Web Service access) and (in planning) SemWebCentral.[7]

Ontology repositories appear to be a useful means to provide an access point for developers to locate ontologies. However, the present state of the art does not resolve a number of issues. For example, the ontology description is uploaded into a separate store and is disconnected from changes in the original ontology. The developer needs to remember to update the ontology entry in the repository in case the ontology evolves. The repositories do not appear to support any aspect of ontology versioning. The means of locating ontologies is quite haphazard, and relies on the same type of keyword matching that occurs in non-semantic search engines such as Google. Queries can not draw on the semantics of ontologies themselves in order to be able to find e.g. generalizations, specializations or equivalences of search terms and concepts. Finally, the repositories link to the complete ontologies from their descriptions, meaning that access is on an 'all or nothing' basis, not taking into account the various needs of individual users.

As an additional motivation, let us consider a real world scenario. In large enterprizes, ontologies can be used to model the business domain and classify information according

[2] http://www.google.com
[3] http://swoogle.umbc.edu/
[4] www.daml.org/ontologies/
[5] http://protege.stanford.edu/plugins/owl/owl-library/index.html
[6] www.schemaweb.info/
[7] www.semwebcentral.org/

to this model, thus enabling new solutions for knowledge management problems in these enterprizes. Different departments need access to the enterprize knowledge model for use in different applications and scenarios. As a result, it is likely that they do not each need the exact full set of classes and properties that the model contains and that they eventually maintain their own view upon the relevant ontological entities. Indeed, by restricting to the maximum subset of the domain that is used, computational complexity is avoided and the sub-model can be efficiently evaluated and maintained by its users. On the other hand, each department may need to model additional concepts outside of the core domain, and need to find models for these domains additionally. An ontology repository is made available on the company intra-net, storing both the current versions of the ontologies in use and metadata describing them. As well as the requirements noted previously - the synchronization between the ontologies and their metadata, semantic searching based on semantic-based descriptions and the selection of fragments of ontologies - we add to this:

- The repository needs to demonstrate availability and reliability - ontologies must be available when they are needed
- Ontologies are stored as ontologies and not just links to a file, and they are dynamically associated with their descriptive metadata within the system
- Search and browse functionality is available to developers and is coordinated with the updates being made by ontology providers
- Access to ontologies and their metadata is managed, e.g. that a general user can not make changes to the description of someone else's ontology
- Evolution of ontologies is tracked and versioning managed, so that a developer can e.g. resolve incompatibilities arising from ontology changes

In order to provide a new paradigm for ontology repositories which can better meet the requirements of both ontology providers and ontology developers, we propose in this paper the use of our Semantic Web Spaces in building an ontology repository infrastructure. The tuplespace-based approach provides a logically shared memory which gives a common view upon both ontologies and their metadata to both providers and developers, with a simple yet powerful co-ordination mechanism to support the evolution of the stored ontologies and semantic-based matchings to offer rich query functionality to developers. We believe Semantic Web Spaces also offer better resolution of performance and scalability issues through tuplespace partitioning and distribution. Finally, we demonstrate how the system is extendible to add support for access policies, trust issues, versioning, labeling and other useful requirements for a real world ontology repository.

3 Semantic Web Spaces

Semantic Web Spaces apply the tuplespace paradigm to the Semantic Web to design a middleware for the Semantic Web that inherits the well-known benefits of tuplespace-based coordination such as its mechanisms for asynchronous and spatially and temporally

decoupled communication [RCD01]. For this purpose we extend the traditional Linda-based tuplespace infrastructure [GC92] with the possibility of representing and processing semantically-rich information such as RDF(S) and OWL ontologies [TNL+04, TNPB+05, TPBN05, PBNT05].

3.1 Overview

Semantically enriched tuplespaces require **new types of tuples** with defined semantics. Semantic Web Spaces focus on tuples containing information represented in standard representation languages of the Semantic Web. Moreover, the transfer of the original Linda to the world of Semantic Web requires further work on the conceptional level of the Linda co-ordination model. Examples include **extensions of the Linda primitives** and the definition of new suited **types of tuplespaces**, such nested and overlapping spaces, which correspond to the principles of the Semantic Web.

While common Linda systems deal with *data* represented as tuples, Semantic Web Spaces are intended to manage *information* with formally defined semantics represented in triple form (i.e. *(subject, predicate, object)*) [HM04]. In the latter case tuples have a truth assignment which makes them different from being merely interpreted as data. Changing the scope of the tuplespaces from data to information affects the semantics of the Linda operations involved. Hence the classical semantics of *in* and *out* must be altered for tuplespaces of truth-assigned content.

Any client/agent accessing the Semantic Web Spaces should work within a given *context* that defines a partial view upon the interpreted information. According to this, we consider that there are two views on the tuplespace in Semantic Web Spaces. *The information view* interprets the data from the space according to the semantics of the information it encodes. In this view, Semantic Web Spaces define additional primitives with their own semantics. The *data view* of Semantic Web Spaces preserves the operations *out*, *in* and *rd* as they are defined in Linda. We also add extended matching relations that work on RDF typing and are able to take into account defined RDF(S) semantics, for example to match a subrelation in a tuple for a relation in a template.

Figure 1 shows the structure of Semantic Web Spaces. The traditional Linda primitives operate upon the data view, encompassing simple datatype tuples, XMLSpaces tuples (containing XML documents[CTZ02, TLN04]) and Semantic Web tuples such as RDF triples. The latter also have an information view, where additional primitives are defined to operate upon the data according to the semantics of the information that it contains and hence embedding application-relevant reasoning features (RDFS, OWL Lite, OWL DL etc.).

As in common Linda-like systems Semantic Web Spaces may use different persistent storage paradigms, including for example replication and distribution of the RDF data on several physical systems[BKvH02, HG03].

Semantic Web Spaces is based on the different levels of information published on the Semantic Web (Figure 2). We foresee several “flavors”:

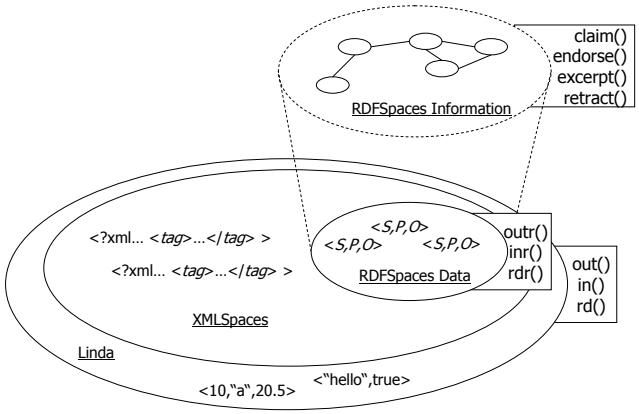

Figure 1: Different views on different spaces in Semantic Web Spaces

- *RDFSpaces* enable the storage and retrieval of RDF triples in a tuplespace with a set of extended matching relations and co-ordination primitives that operate on the additional semantic information provided.
- *OWLSpaces* permit the use of OWL concepts in the tuples as an extension of the RDF(S) concepts already used in the proposed RDFSpace. Due to the syntactical compatibility between OWL and RDF(S) every OWL construct can be represented as a uniquely identified set of RDF statements. The space can interpret the knowledge according to the reasoning capabilities implemented in the application layer/matching functions. Different levels of expressiveness (OWL Lite, OWL DL etc. [PSHH04]) can be achieved by embedding corresponding reasoner modules in the application layer.
- *RuleSpaces* make use of a Semantic Web rule language[HPSB$^+$04] to explicitly represent knowledge in the form of rules. These rules have two purposes in Semantic Web Spaces. They could be set as reactive to particular operations on the tuplespace, and could effect changes in the tuplespace additional to those carried out by agents. Alternatively, agents might need expressivity of knowledge beyond that possible with OWL to be able to draw further inferences that produce new knowledge.
- *ReasonSpaces* include a logical reasoner for determining and publicizing proofs in the tuplespace. This could be based on e.g. PML [dSMF04] to provide the proof to an agent that an operation result is correct.

- *TrustSpaces* include agent policies as tuples and execute matchmaking agents to determine if two agents (source and target) can trust one another before permitting an operation.

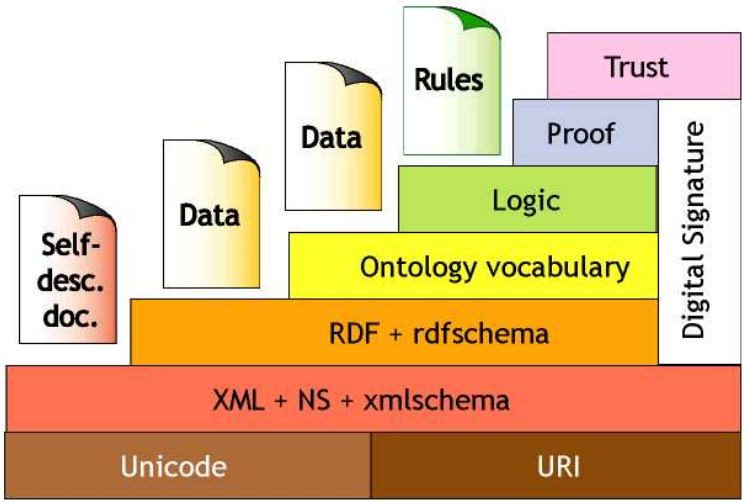

Figure 2: The Semantic Web layered architecture as proposed by Tim Berners Lee [BL00]

As mentioned above different reasoning engines can be linked to the information space depending on its content (e.g. OWL, SWRL) and on the client requirements.

The tuplespace itself is internally represented in terms of a Semantic Web ontology, which contains both content-related metadata and management data such as the types of supported templates, access and trust policies. In this way the architecture of the tuplespace can be adapted to new requirements of the application environment and can support new types of primitives depending on the content of the space.

In the following we take a look at the ways RDF(S) and OWL data is represented and at the semantic matching capabilities within Semantic Web Spaces, as these issues play an important role in understanding the benefits Semantic Web Spaces provide as an infrastructure for Web ontology repositories. In a complementary technical report, we presented the Conceptual Model for the Semantic Web Space in more detail [PBNT05].

3.2 Representing RDF(S) and OWL in Semantic Web Spaces

The data level of Semantic Web Spaces (see Figure 1) contains RDF statements - the core model of Semantic Web knowledge representation. These statements are represented in the tuplespace as tuples with the structure *(subject, predicate, object,id)*. Each tuple also

carries an unique identifier, which is drawn from the RDF ID of the Tuple instance in the tuplespace ontology. In conforming with the RDF data model, all RDF tuples are typed *(?rdfs:Resource, ?rdf:Property, ?rdfs:Resource, ?rdf:ID)*[8].

By re-using the object of one statement as the subject of another, the tuplespace can represent a semantic graph structure of the contained knowledge. In order to support rich typing, tuples in Semantic Web Spaces are stored as an ordered list of typed field values. These values of these types are URIs identifying RDF classes constrained in an ontology. In other words, each field value in the tuplespace is associated to a RDF type. Additionally special attention is paid to blank nodes, RDF containers and reified statements [TPBN05, PBNT05].

Semantic typing also includes namespace support so that types from different vocabularies may be differentiated, also when they share the same name. Each tuple in the Semantic Web Space is addressable through an unique identifier in order to group several semantically interrelated RDF statements. Since the structure and properties of a tuplespace are described using the tuplespace ontology, every tuple of the space is defined by means of this ontology and can be referenced using the corresponding URI.

Ontologies represented in RDF Schema or OWL define classes, properties, restrictions of the domain and range of properties and hierarchies of classes and properties. Due to the syntactic compatibility between RDF and RDF(S) and OWL, representing information in the latter is reduced to representing RDF in the information space. The semantics behind the corresponding representation paradigms is resolved in the information view of the space (Figure 1) by means of pre-defined semantic matching templates complimented by language-specific extern reasoners.

3.3 Semantic Matching

Matching through templates is the fundamental interaction paradigm of tuplespaces. New matching procedures are required to support the RDF-based paradigm of the Semantic Web Space which includes matching not only on values as in the classic Linda-based setting but also on types. As well as resource matching (equivalence of simple datatypes but also of complex datatypes (arrays, lists) and of URIs), matching procedures must also take into account ontological information and different levels of precision matching made available to allow requesters to choose the bounds in which a template may be determined as valid for a tuple.

For example, a query on which wine is located in which region using the template *(?wine:Wine, wine:locatedIn, ?wine:Region)* means match on tuples whose subject (of type Wine) has the predicate *wine:locatedIn* and the object of type Region. [9] Given a tuple *(X, wine:locatedIn,*

[8] In particular the object of a RDF triple – typed as rdfs:Resource – may be a new resource or a value, a literal. The fourth field should not be understood as an ID for the RDF statement, which would be reification, but an ID for the tuple in the space

[9] The forth field of the RDF tuples, the identifier is not relevant to the matching operations and is thus left out from our matching examples.

Z) where Z is of type USRegion, X is of type WhiteWine and the RDF Schema information that USRegion is a subclass of Region, while WhiteWine is a kind of Wine we expect this tuple to match. However the matching will only succeed if the matcher is aware of the RDF Schema information, and the matching procedure selected allows matching on subclasses. The matching functionality for clients in RDFSpaces - the first version of Semantic Web Spaces for representing RDF(S) data - support simple templates for retrieving matching tuples and subspaces, which could then be queried through a suitably expressive RDF query language locally. Special matching rules could be applied to handle cases such as Blank Nodes, reified statements and RDF containers and collections.

OWL primitives and their semantical interpretations are handled by embedding available DL reasoners in the same way as RDFS semantics (i.e. subsumption) is supported in RDFSpaces.

4 Ontologies and Metadata

A fully-fledged ontology repository is inconceivable without a rich and flexible metadata model aggregating additional information about the managed ontologies. The metadata model is used as a framework for search and classification services in the repository - on one hand the repository user (the developer) specifies the properties of the ontologies he or she is looking for in terms of attributes of this model; on the other hand the repository classifies the ontologies according to these dimensions in order to support adequate views aiding the user in navigating the repository's content. According to the requirements in Section 2 the metadata model should satisfy the following conditions:

- It should incorporate significant general-purpose and application-related facts about the described ontologies
- It should be represented semantically in order to enable content-based search on the model's dimensions and automatical classification of new ontological sources in the repository
- It should be extensible to different groups of users and application settings

As mentioned in Section 2 most of the today ontology repositories do not provide support for ontology descriptions, but are merely collections of ontological sources which have to be 'read through' by humans in their attempt to discover potentially relevant ontologies. Others such as DAML Ontology Library and Swoogle classify resources according to an implicit, flat metadata model with very limited search functionality. This situation is justified by the fact that currently the metadata issue w.r.t. ontological sources has been poorly explored in the Semantic Web community, though its importance is well-recognized. Recent approaches such as [Kn04] try to cope with this problem, offering a prototypical model of an XML-based metadata scheme for Web ontologies. However the proposed model lacks formal semantics and can not be optimally used to search or automatically classify ontological resources within the repository.

In order to describe ontologies within the tuplespace-based repository we use the information model introduced in [PB05]. The information model was derived from a comprehensive survey of recent Ontology Engineering literature and from empirical findings from several case studies in the same field. It is intended as a means to offer a formal semantic description of Web ontologies which can be used to improve the results of various human-driven or automatic stages of the engineering process such as ontology discovery, evaluation or matching [PB05]. The model is represented in OWL and contains - according to Stamper's semiotic framework [St91] - three categories of features: syntactical, semantic and pragmatic:[10]

- Syntactical features, originally related to the external form of the metadata-described item, offer quantitative and qualitative information about the ontology and its underlying (graph) topology. Examples of syntactical features include the number of concepts and properties for each class, the depth of an inheritance tree, the number of incoming properties, the number of concept instances, the average path length, the number of connected components. Since ontologies are published in an open network like the Semantic Web, it is also important to consider the links a particular ontology has to other networked information sources [Ne03]. Finally, there is qualitative, representation language-dependent information like the representation language itself, the number of syntax constructs used and syntactical correctness (validity).

- Semantical features are related to the formal semantics of the representation language and the meaning of the ontology content: i). consistency (as measured by a reasoner), ii). correctness (i.e. whether the asserted information is true), iii). readability (i.e. the non-ambiguous interpretation of the meaning of the concept names w.r.t. a lexicon, the usage of human-readable concept names), iv). level of formality (e.g. highly informal, semi-informal, semi-formal, rigorously formal[UG96]), v). type of model (upper-level, domain ontology, thesaurus etc.[Gu98, WW02]), vi). ontology domain (i.e. the modelled domain e.g. medicine), vii). representation paradigm (i.e. the class of representation languages w.r.t. expressivity such as a specific Description Logic), and viii). natural language (the natural language used to denominate ontological primitives e.g. English).

- Pragmatic features refer to information about the usage history of the ontology, for example when, by whom and to which purpose it was developed, whether multiple versions are available or about the engineering process the ontology originally resulted from. The latter topic is relevant for ontology engineers intending to (partially) re-use the ontology within an information system: the original engineering methodology, tools used during the development process and the input information sources. As input information sources one can mention extern ontologies which are, partially or in a modified form, included to the current ontology. Another example is a domain-relevant document corpus used by ontology learning programs.

[10]The rationales behind and the method applied to generate the information model are beyond the scope of this paper and are described in [PB05].

The OWL-based information model can be used to describe ontologies and fragments of ontologies or additional sources related to them, which are contained in the repository (see Figure 3). In contrast to common static metadata approaches, the information model introduced here is meant to offer a description of the complete usage and development history of a particular ontology (or ontology fragments down to concepts and properties). While the former are updated with every change in the ontology they describe, the latter model is of contextual nature, which implies that it monitors the whole range of usage and developmental contexts associated with a specific ontological source. For example the blue ontology in Figure 3 is annotated with more than one metadata objects, depending on the application using the ontology or the user of the metadata. ContextObject1 and ContextObject13 in the same figure are instances of the same class in the metadata ontology, describing the usage of the blue ontology in two application settings, denoted by abc and def. ContextObject2 aggregates the same descriptive information for an ontology user, which is not interested in too many technical details. With every change in an ontology, be that its usage or its content, a new context object is created monitoring the new setting in which the changes are logged for further tasks such as the evaluation of the ontology w.r.t. specific application requirements. The model foresees descriptions for both sub-ontologies (which are quite similar to the metadata used for complete ontologies anyway) and for single ontological primitives. In Figure 3 a new ontology is created by merging the blue, the yellow and the black ontologies on the left side. In this case, the concepts in the new ontology keep the former annotations from the original source ontologies, while the target ontology is described by a new context object. Ontological entities are described by the pragmatical features listed above and by information related to the representation paradigm they are part of. Appendix .1 contains an excerpt of an imaginary instantiation of the described information model. We illustrate the usage of the information model for search and classification purposes for ontology repositories in the next section.

Some of the mentioned features e.g. syntactic and semantic ones can be generated automatically (either by direct computations or using heuristics) in order to ensure the consistency between ontologies and the metadata describing them and thus to ease the annotation process. Remaining features, in particular most of the ones in the pragmatic category are to be provided by humans, due to the lack of pre-defined syntactical constructs in RDF(S) and OWL which could be use unambiguously to represent such information.

5 A tuplespace-based ontology repository

Given Semantic Web Spaces and an ontology for the description of other ontologies (see Section 4), it is possible to realize tuplespace-based ontology repositories. Figure 4 shows the high level architecture of such a repository. In particular we emphasize the following important aspects w.r.t. this architecture:

- The tuple space provides a common view upon both the ontologies and the ontology metadata
- The co-ordination model provides synchronized access to those ontologies and their

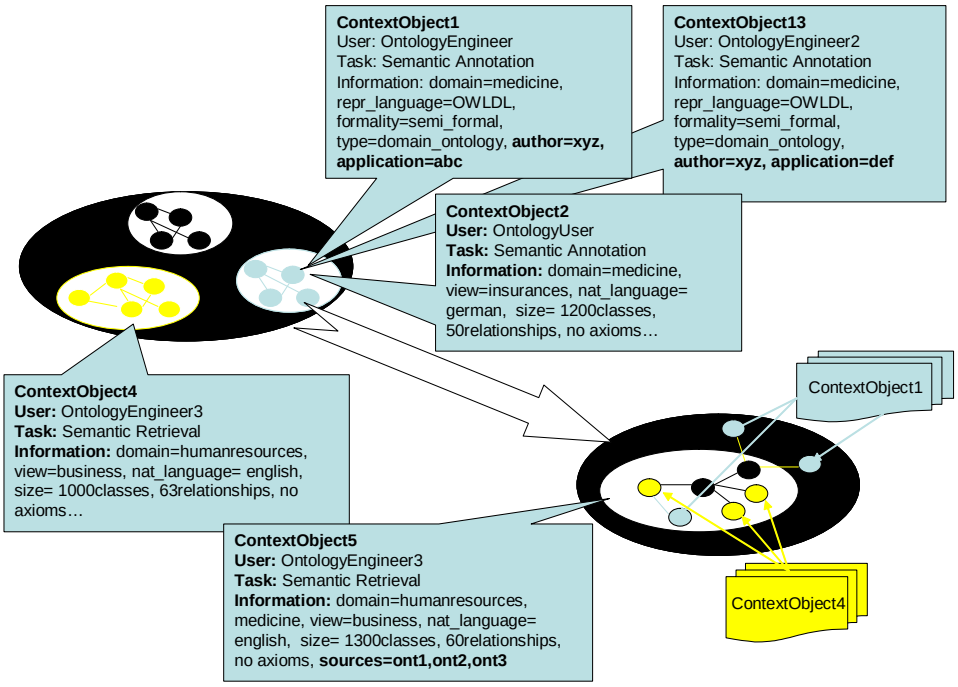

Figure 3: Ontology Metadata

metadata

- The tuple space ontology and administrative services based upon it can provide useful additional functionalities such as versioning or rights management
- The clients can interact with the tuple space using any available GUI on any platform over any network
- The ontology stores can be distributed, based on different platforms and over different networks
- The interaction with the repository is based on semantic reasoning over the stored ontological knowledge

We illustrate the value of the tuplespace-based ontology repository through a feasible usage scenario. We consider different tasks that are foreseeable within this scenario and how these tasks are facilitated by the approach introduced in this report.

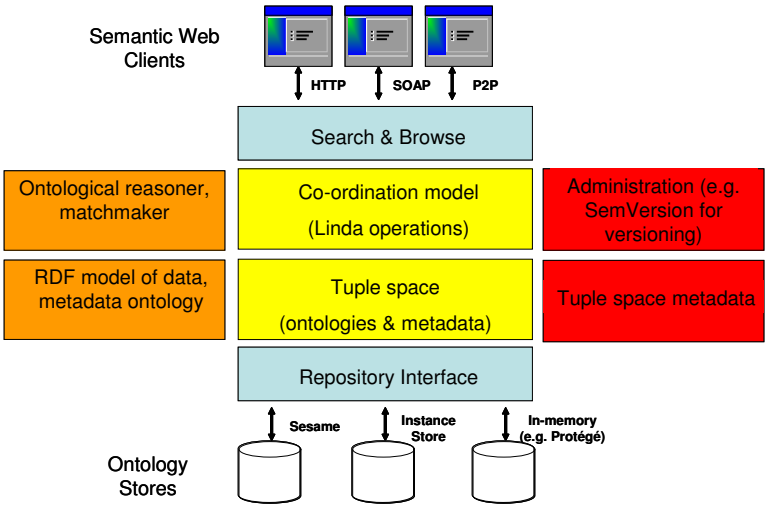

Figure 4: High-level architecture of a tuplespace-enabled ontology repository

5.1 Usage scenario

In 2 we already introduced an enterprize use case, in which ontologies are used to model business domains and views upon these domains in different departments and for different application purposes. Here, a large enterprize, whose departments are distributed geographically and who each independently have their own knowledge requirements, wants to support its knowledge developers in finding and using the ontologies they need. It does this by making available a repository of ontologies that have been provided by previous and existing knowledge management projects within the enterprize or have been accepted by some authority as being external ontologies which are trustworthy and expressive enough for use within the enterprize. The idea is that when a department requires to develop an ontology for some task, it can access other ontologies which are relevant to its domain and particular requirements, in order to re-use it or adapt it to their own needs. This saves development time and promotes interoperability where different departments in the enterprize may share, at least, some common subset of concepts between different ontologies. In the same time the ontologies administrated by the repository system can be accessed and refined by the authorized parties in a coordinated manner, in order to guarantee their consistency in a distributed environment.

We see the scenario as having two types of actor: [11]

- **Providers** have developed ontologies for some task and now make them available on the platform together with some metadata,

[11] Though the same user may be either or both we make this distinction as it is a distinction in how the system is used.

- **Developers** are seeking ontologies for use in a specific project. They are able to express their requirements and are willing to consider any ontology which partly meets that requirement.

5.2 Functionality

For this scenario, we consider the sort of functionalities that an user of the ontology repository would wish to have, and how these are performed using Semantic Web Spaces as the ontology repository platform. Firstly, in the case of providers we have:

- Introducing ontologies to the space (publishing ontologies)
- The automatic and semi-automatic generation of the ontology metadata
- Keeping metadata entries actual, permitting updates generated by the system and allowing other users to add information
- Protecting ontologies and metadata from abuse
- Collaboratively updating ontologies in the space

Then, in the case of developers, we can consider:

- Search by expressing ontology requirements in terms of the metadata vocabulary
- Subscribing to ontology changes
- Additional tests to assess ontology suitability

Finally, we consider a collaborative scenario, in which a cycle of ontology development and provision is occurring, requiring the co-ordination aspects of the Linda model.

5.3 Examples

In this section we demonstrate the implementation of the aforementioned ontology repository functionality in terms of the Semantic Web Spaces middleware. To understand the exact semantics of the operations given here, we refer the reader to [PBNT05].

5.3.1 Introducing ontologies to the space

For the space to act as an ontology repository it is clear that initially ontologies must be introduced into the space. The simplest means to achieve this is to define an instance of the class Ontology and provide an URI which points to the actual ontology [12].

[12]For brevity, the ontology description metadata is given without any namespace prefix, local IDs are given as strings beginning with a hash and the XMLSchema URI is also abbreviated

```
claim("#hrontology1" [Ontology],hasURI,
      "http://.../wz2003.rdf#" [XMLSchema#anyURI])
```

In order to support the correct interpretation of the referred ontology it may also be necessary to provide the Representation Paradigm. However, it may be within the capabilities of the repository to determine the representation from the file (e.g. in a XML encoding, by examining the root element and used namespaces, or by means of specific tools detecting the expressivity level of the representation language in use). Generally, ontologies will be expressed in RDFS or OWL. The file is retrieved from the given URI (we could expect that if the URI returns an error, the claim could be rejected) and stored in the tuplespace (generating RDF tuples in a new context which as a whole represent the RDF graph of the newly submitted ontology). Note that the use of contexts is important here to ensure that RDF statements can be referenced to their ”owner”and hence to its metadata. A simple rule of thumb could be to use the ontology URI (e.g. in this case ”hrontology1”) as the unique-identifier extension to a standard context URI string.

5.3.2 Generation of the ontology metadata

Once the ontology is stored into the tuplespace it is necessary to also have metadata relating to that ontology. To provide the means to maintain the connection between the ontology and metadata both are found within the same context. Some metadata can be generated automatically or using particular heuristics by the repository on the basis of the triples generated when the ontology was introduced to the space. Other metadata must be produced manually by the ontology provider. A possible approach here would be to build this metadata provision into the ontology development tool (by extending a development tool to allow the author to store such metadata within the ontology project files).

Some important metadata that would require manual provision includes the specification of the OntologyAuthor, OntologyApplication and OntologyTask.

```
claim( ("#hrontology1" [Ontology],createdBy,
    "Malgorzata Mochol" [OnthologyAuthor]),
       ("#hrontology1" [Ontology],usedInApplication,
       "#InformationRetrievalSystem" [OntologyApplication]),
       ("#hrontology1" [Ontology],usedFor,
       "#MatchingTask" [OntologyTask]) )
```

Some important metadata that could be generated automatically (eventually by using some heuristics) includes the OntologyDomain (referencing topics in the Open Directory ontology), the NaturalLanguage, SupportedOntologyPrimitives and the UsedSources (see Section 4 and [PB05] for a detailed description of the concepts). In the example, we declare an OntologyPrimitive called RecursiveTypeRelation which represents the non-restriction on the use of the type relation existing in RDFS which is not permitted in Classical Description Logics.

```
claim( ("#hrontology1" [Ontology],hasNaturalLanguage,
```

```
 "de" [XMLSchema#string]),
("#hrontology1" [Ontology],describesDomain,
 "dmoz:HumanResources" [Topic],
("#hrontology1" [Ontology],supportsOntologyPrimitive,
 "#RecursiveTypeRelation" [OntologicalPrimitive]),
("#hrontology1" [Ontology],usesSource,
 "#naicshomepage" [WebDocument]) )
```

5.3.3 Evolution of the ontology metadata

It is recognized that just as the submitted ontology will evolve over time the metadata describing the ontology will need to be updated to reflect the evolution and to provide up-to-date information about the ways the ontology has been used across the tuplespace user community. As for the initial generation of the metadata, some aspects of the metadata can change automatically as the ontology changes, while other aspects need to be manually changed (by a retraction and new claim in the tuplespace).

In the ontology repository, changes are made on the ontology through claims and retractions (additions and removals of RDF tuples in the space). Rules defined in the repository can link changes in the ontology to changes in the metadata of the ontology, in order to ensure that both are synchronized. This is required at the entity level, since every concept of an ontology can be theoretically annotated with metadata information. At the ontology level, a new instance in the metadata ontology is created, recording the changes in terms of actualized metadata. The tuplespace system is responsible for retrieving, aggregating and presenting this information to the ontology developer looking for application-relevant ontologies.

Other aspects of the metadata can change independently of the ontology itself. One important aspect that may change is the version of the ontology. While ontology versioning could be supported by a standalone component, such as SemVersion, new Ontology instances can be added to the space which are versions of existing Ontologies. These ontologies inherit the characteristics of the ontology they are a version of, while new claims supercede any inherited properties. For example, hrontology2 is a new version of the hrontology1 ontology, in which the labels of concepts have been translated into English:

```
claim( ("#hrontology2" [Ontology],isVersionOf,
        "#hrontology1" [Ontology]),
       ("#hrontology2" [Ontology],hasNaturalLanguage,
        "en" [XMLSchema#string]) )
```

5.3.4 Trusting information in the space

Strategies are employed to ensure that the information in the space is not discredited by permitting claims that are false or misleading and which lead to ontology users receiving wrong conclusions about the relevance or suitability of certain ontologies.

The evolution of usage of the system may be able to lead to trusting ontologies from certain authors or organizations, or which have been proven in quality or certified in previous application settings [PB05].

One means of determining trust is through associating trust policies with the tuple space and with the agents which are making claims in the space [PBNT05]. However, here we consider another means in which trust is determined from within the ontology repository. A simple example rule could be that an ontology which is used as a source by a significant number of other ontologies must be an ontology that can be trusted. An agent could check this for a given ontology using the following operation, and then counting how many matches exist in the returned context by destructively reading until the context is empty:

```
excerpt(? [Ontology],usesSource,
    "#dublincore" [Ontology])
```

5.3.5 Search by expressing ontology requirements

The ontology developer would use the ontology repository to discover relevant ontologies by expressing a specific user/application context and matching this to the ontology metadata stored in the repository. The context could express information about the target ontology, about the application (e.g. type of system, industrial sector) and the role the ontology would play in this setting [PB05].

Using the Semantic Web Space, requirements can be expressed as a set of operations through which the developer can drill down through a larger set of candidate ontologies to a smaller number which best match all of his or her requirements.

```
excerpt(? [Ontology],describesDomain,
"dmoz:Human_Resources" [OntologyDomain]) -> C,
retract(? [Ontology],describesDomain,
"dmoz:Human_Resources" [OntologyDomain]) @ C,
excerpt(x [Ontology],usedInApplication, *) -> C1
```

This sample initial query creates within a context C a set of statements about Ontologies which describe the domain of Human Resources. For each of these Ontologies the statement identifying the Ontology is retracted from context C and the related statement about the type of application the Ontology is used in is coped into the context C1.

```
excerpt(? [Ontology],usedInApplication,
"#InformationRetrievalSystem" [OntologyApplication]) @ C1 -> C2
```

Following this, the developer may seek from among the Human Resource Ontologies any that are used in Information Retrieval Systems. To do this, the operation is performed in the context C1, which contains only statements about those Ontologies which describe the Human Resources domain. This can be continued, in that the new set of Ontologies which are the subjects of the statements in context C2 may be retracted, new facts about them excerpted into a new context, and then this new context is queried.

5.3.6 Testing ontology suitability

Having acquired some candidate ontology the ontology user also needs to use contextual information to estimate the quality of the candidate for re-use in the particular target application. While we do not attempt to introduce a novel ontology evaluation methodology, by which ontology developers are aided to discover the most suitable candidate sources for particular application settings, existing elaborated evaluation approaches agree that information about the original purpose and the projects/settings already using these ontologies can give hints about their suitability in a new setting [PB05]. We exemplify these issues for the tuplespace-based ontology repository.

The user could for instance prefer to check if a candidate ontology is already used in the role that the user intends (As OntologyRoles, the metadata defines IndexRole, FilterRole, ModelRole and VocabularyRole). To do this, we excerpt the tasks which are associated to that role and check if the ontology is already used for any of those tasks.

```
excerpt(? [OntologyTask],has-associated-role,
"ModelRole" [OntologyRole]) -> C,
retract(? [OntologyTask],has-associated-role,
"ModelRole" [OntologyRole]) @ C,
endorse("#hrontology1" [Ontology],usedFor,
x [OntologyTask])
```

Another aspect of checking suitability is to determine what sort of changes may still be required in the selected ontology. As a final example, we consider checking the natural language used in the ontology, as it may be required that concept labels or documentation is translated into the language of the intended ontology users.

```
endorse("#hrontology1" [Ontology],hasNaturalLanguage,
? [XMLSchema#string])
```

Finding that the ontology is in German, we can check if there are versions of the ontology which are in French:

```
excerpt(? [Ontology],isVersionOf,
"#hrontology1" [Ontology]) -> C,
retract(? [Ontology],isVersionOf,
"#hrontology1" [Ontology]) @ C,
excerpt(x [Ontology],hasNaturalLanguage,
? [XMLSchema#string]) -> C1,
endorse(? [Ontology],hasNaturalLanguage,
"fr" [XMLSchema#string]) @ C1
```

5.4 Discussion on collaborative scenario

Using the operations of Semantic Web Spaces, which are blocking i.e. they stall the executing process until the operation completes which in the case of a retrieval operation means waiting until a match is introduced in the space, introduces a powerful synchronization functionality to the ontology repository. We must also consider what this means for a collaborative scenario in which ontologies are being introduced and updated while other developers are searching the space for suitable ontologies.

Firstly, it seems circumspect to expect that agents employ a time out on their retrieval operations. While a reasonable amount of time can be allowed to ensure that the entire space has been checked, and possibly to allow for the introduction of a matching claim in the meantime by an ontology provider, in many cases (e.g. checking how many ontologies use Dublin Core as a source, when perhaps none are) a non-match is best considered as a fail than holding the ontology developer process for an indefinite period. However, there are cases where this holding is preferable, e.g. a department begins a new project and seeks an ontology for the Human Resources domain. When it performs its blocking retrieval operation, it may prefer to wait in case an ontology provider introduces a matching ontology to the space than immediately consider the operation to have failed and begin building an ontology from scratch. The user will have the final decision in terms of the period of time he or she is prepared to wait.

Secondly, it is recommended that claims made in the ontology repository block the entire context in which the claim is made until all consequences of the (accepted) claim have been handled by the system. The advantage of this is that if the ontology provider makes a change in the ontology or its metadata, and during the course of the system updating the stored information about the knowledge there is a retrieval request made concerning that information, it is non-determinable whether what is returned will be the updated information or not. By applying this block, retrieval is guaranteed always to receive the most up-to-date information from the repository.

6 Conclusions and Future Work

The purpose of this document has been to describe how Semantic Web Spaces could be used as the basis for an Ontology Repository platform, combining the semantically-enhanced co-ordination power of the Linda language, the distributed shared memory space of the Tuple Space model and a knowledge model for describing ontologies to support their selection and re-use. This serves to provide an illustrative use case for our Semantic Web Space as well as to provide an implementable platform for the use of the proposed ontology description scheme.

References

[BKvH02] Broekstra, J., Kampman, A., and van Harmelen, F.: Sesame: A generic architecture for storing and querying RDF and RDF schema. In: *The Semantic Web - ISWC2002*. 2002.

[BL00] Berners-Lee, T. The Semantic Web. Talk at the XML2000: http://www.w3.org/2000/Talks/1206-xml2k-tbl/. December 2000.

[CTZ02] Ciancarini, P., Tolksdorf, R., and Zambonelli, F.: Coordination Middleware for XML-centric Applications. In: *Proceedings of ACM SAC 2002*. S. 335–343. 2002.

[dSMF04] da Silva, P. P., McGuinness, D. L., and Fikes, R. E.: A proof markup language for semantic web services (Technical Report KSL-04-01). Technical report. Knowledge Systems Laboratory, Stanford University. 2004.

[GC92] Gelernter, D. and Carriero, N.: Coordination languages and their significance. *Communications of the ACM*. 35(2):97–107. 1992.

[Gr95] Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.* 43(5-6):907–928. 1995.

[Gu98] Guarino, N.: Formal Ontology and Information Systems. In: *Proc. of the FOIS'98*. 1998.

[HG03] Harris, S. and Gibbins, N.: 3store:Efficient Bulk RDF Storage. In: *Proceedings of the First International Workshop on Practical and Scalable Semantic Systems*. 2003.

[HM04] Hayes, P. and McBride, B.: Rdf semantics. *Available at http://www.w3.org/TR/rdf-mt/*. 2004.

[HPSB+04] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. *Available at http://www.w3.org/Submission/SWRL/*. 2004.

[Kn04] KnowledgeWeb European Project. Identification of standards on metadata for ontologies (Deliverable D1.3.2 KnoweldgeWeb FP6-507482). 2004.

[Ne03] Newman, M.: The structure and function of complex networks. *SIAM Review*. 45(2):167–256. 2003.

[PB05] Paslaru Bontas, E.: Using Context to Improve Ontology Reuse. In: *Proceedings of the Doctoral Consortium at the International Conference on Advanced Information Systems Engineering CAISE05*. 2005.

[PBNT05] Paslaru Bontas, E., Nixon, L., and Tolksdorf, R.: A Conceptual Model for Semantic Web Spaces (Technical Report TR-B-05-11). Technical report. Free University of Berlin. August 2005.

[PSHH04] Patel-Schneider, P. F., Hayes, P., and Horrocks, I.: Owl web ontology language semantics and abstract syntax. *Available at http://www.w3.org/TR/owl-absyn/*. 2004.

[RCD01] Rossi, D., Cabri, G., and Denti, E.: Tuple-based technologies for coordination. In: Omicini, A., Zambonelli, F., Klusch, M., and Tolksdorf, R. (Hrsg.), *Coordination of Internet Agents: Models, Technologies, and Applications*. chapter 4, S. 83–109. Springer Verlag. 2001. ISBN 3540416137.

[St91] Stamper, R.: The Semiotic Framework for Information Systems Research. *Information Systems Research: Contemporary Approaches and Emergent Traditions*. 1991.

[TLN04] Tolksdorf, R., Liebsch, F., and Nguyen, D. M.: XMLSpaces.NET: An Extensible Tuplespace as XML Middleware. In: *Proceedings of the 2nd International Workshop on .NET Technologies, .NET Technologies'2004*. 2004.

[TNL+04] Tolksdorf, R., Nixon, L., Liebsch, F., Duc Minh, N., and Paslaru Bontas, E.: Semantic Web Spaces (Technical Report TR-B-04-11). Technical report. Free University of Berlin. 2004.

[TNPB+05] Tolksdorf, R., Nixon, L., Paslaru Bontas, E., Nguyen, D. M., and Liebsch, F.: Enabling real world Semantic Web applications through a coordination middleware. In: *Proceedings of the 2nd European Semantic Web Conference ESWC2005*. Springer Verlag. 2005.

[TPBN05] Tolksdorf, R., Paslaru Bontas, E., and Nixon, J. B. L.: Towards a tuplespace-based middleware for the Semantic Web. In: *Proceedings of the International Web Intellingence Conference WI-IAT2005 (to be published)*. 2005.

[UG96] Uschold, M. and Grüninger, M.: Ontologies: Principles, methods and applications. *Knowledge Engineering Review*. 11(2). 1996.

[WW02] Wand, Y. and Weber, R.: Information Systems and Conceptual Modelling: A Research Agenda. *Information Systems Research*. 13(4). 2002.

.1 An example for context information for ontologies

```
<!-- one context object describing the hrontology -->

<Contextrdf:ID="contextobject1">
    <hasTarget rdf:resource="#hrontology1/>
    <hasTask rdf:resource="tasks:matching"/>

    <!-- one system the ontology is used in -->
    <usedInApplication>
     <OntologyApplication rdf:ID="wissensnetzte">
       <isContextInformation rdf:resource="#hr1ontologycontext"/>
       <hasIndustrialSector rdf:resource="wz2003:branch_01"/>
       <hasURL rdf:datatype=".../XMLSchema#anyURI">
       http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/
       </hasURL>
      </OntologyApplication>
   </usedInApplication>
</Context>

<!-- additional descriptive information -->

<Ontology rdf:ID="hrontology1">
```

```
    <hasURI rdf:datatype=".../XMLSchema#anyURI">
    http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze
    /interval/wz2003.rdf#
    </hasURI>
    <!-- pragmatic metadata: engineering method, sources... -->
    <isCreatedUsingMethodology>
       <EngineeringMethodology rdf:ID="wissensnetzemethodology">
         <hasDocumentation rdf:datatype=".../XMLSchema#anyURI">
         http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/
         </hasDocumentation>
       </EngineeringMethodology>
    </isCreatedUsingMethodology>
    <usesSource>
      <WebDocument rdf:ID="naicshomepage">
        <hasURI rdf:datatype=".../XMLSchema#anyURI">
        http://www.census.gov/epcd/www/naics.html
        </hasURI>
        <isContextInformation rdf:resource="#hr1ontologycontext"/>
      </WebDocument>
    </usesSource>
    <usedFor rdf:resource="tasks:matching"/>
    <hasConceptualizationMethod>
      <ConceptualizationType rdf:ID="ManualOntologyConceptualization">
        <isContextInformation rdf:resource="#hr1ontologycontext"/>

    <!-- semantic metadata: natural language, repr. language -->
    <hasNaturalLanguage rdf:datatype=".../XMLSchema#string">de
    </hasNaturalLanguage>
    <hasDomainType>
      <DomainType rdf:ID="Domain">
      </DomainType>
    </hasDomainType>
    <hasFormalityLevel>
      <FormalityLevel rdf:ID="SemiFormal">
      ........................................
    <supportsOntologyPrimitive rdf:resource="#concept"/>
    <supportsOntologyPrimitive>
      <subClassOf rdf:ID="subclassof">
        <isSupportedBy rdf:resource="#hrontology1"/>
      </subClassOf>
    </supportsOntologyPrimitive>
    <describesDomain rdf:resource="dmoz:Human_Resources"/>
       ........................................

</Ontology>
```