TripCom

Triple Space Communication

FP6 – 027324

Deliverable

D2.2

Specification of the Triple Space Ontology

Reto Krummenacher
Elena Simperl
Vassil Momtchev
Lyndon J.B. Nixon
Omair Shafiq

March 27, 2007

Executive Summary

This deliverable is about the requirements analysis, the specification, implementation and evaluation of the ontology to represent triple spaces and the published data. The deliverable opens by gathering requirements from other related work packages in the TripCom project. The developed ontology is provided in RDFS, OWL-Lite and WSML-Flight. The document serves as an complement to the online ontology catalogue.

Document Information

IST Project Number	FP6 – 027324	Acronym	TripCom
Full Title	Triple Space Communication		
Project URL	http://www.tripcom.org/		
Document URL			
EU Project Officer	Werner Janusch		

Deliverable	Number	2.2	Title	Specification of the Triple Space Ontology
Work Package	Number	2	Title	Triple Space Knowledge Representation

Date of Delivery	Contractual	M12	Actual	31-Mar-07
Status	version 1.0		final ☒	
Nature	prototype ☐ report ☒ dissemination ☐			
Dissemination Level	public ☒ consortium ☐			

Authors (Partner)	Reto Krummenacher (LFUI), Elena Simperl (FUB), Vassil Momtchev (ONTO), Lyndon J.B. Nixon (FUB), Omair Shafiq (LFUI)			
Resp. Author	Reto Krummenacher		E-mail	reto.krummenacher@deri.org
	Partner	LFUI	Phone	+43 (512) 507-6452

Abstract (for dissemination)	To improve the scalability and discovery a clean description of triple spaces and the contained data is required. This deliverable introduces and evaluates the triple space ontology (TS Ontology), which aims at providing formal descriptions for spaces, kernels, and the data.
Keywords	Triple space ontology, Meta-information, Space representation

Version Log			
Issue Date	Rev No.	Author	Change
January 31, 2007	1	Reto Krummenacher	Conceptual Part Final
February 15, 2007	2	Vassil Momtchev	Implementation Part Final
February 23, 2007	3	Reto Krummenacher	First final working draft
March 3, 2007	4	Reto Krummenacher	Final working draft
March 5, 2007	5	Elena Simperl	Final draft to QA
March 12, 2007	6	Reto Krummenacher	Internal release for QC
March 19, 2007	7	Reto Krummenacher	Final release

Project Consortium Information

Acronym	Partner	Contact
Leopold Franzens University Innsbruck http://www.deri.at	LFUI	Prof. Dr. Dieter Fensel Digital Enterprise Research Institute (DERI) Innsbruck, Austria E-mail: dieter.fensel@deri.org
National University of Ireland, Galway http://www.deri.ie	NUIG	Dr. Laurentiu Vasiliu Digital Enterprise Research Institute (DERI) Galway, Ireland Email: laurentiu.vasiliu@deri.org
University of Stuttgart http://www.iaas.uni-stuttgart.de/	USTUTT	Prof.Dr. Frank Leymann Inst. für Architektur von Anwendungssystemen (IAAS) Stuttgart, Germany E-mail: frank.leymann@informatik.uni-stuttgart.de
Vienna university of Technology http://www.complang.tuwien.ac.at/	TUW	Prof.Dr. eva Kühn Institut für Computersprachen Vienna, Austria E-mail: eva@complang.tuwien.ac.at
Free University Berlin http://www.ag-nbi.de/	FUB	Prof. Dr.-Ing. Robert Tolksdorf AG Netzbasierte Informationssysteme Berlin, Germany E-mail : tolk@inf.fu-berlin.de
Ontotext Lab, Sirma Group Corp. http://www.ontotext.com/	ONTO	Atanas Kiryakov, Vassil Momtchev, Ontotext Lab, Sirma Group Corp. Sofia, Bulgaria E-mail: vassil.momtchev@ontotext.com
Profium OY http://www.profium.com/	Profium	Dr. Janne Saarela Profium OY Espoo, Finnland E-mail: janne.saarela@profium.com
CEFRIEL SCRL. http://www.cefriel.it/	CEFRIEL	Davide Cerri CEFRIEL SCRL. Milano, Italy E-mail: cerri@cefriel.it
Telefonica I+D http://www.tid.es/	TID	Noelia Pérez Crespo Telefonica I+D Madrid, España E-mail: npc@tid.es

Table of Contents

List of Abbreviations

DC	Dublin Core
FOAF	Friend Of A Friend
N3	Notation 3
ORDI	Ontology Representation and Data Integration
OWL	Web Ontology Language
OWLIM	OWL In Memory
RDBMS	Relational DBMS
RDF	Resource Description Framework
RDFS	RDF Schema
SAIL	Storage And Inference Layer
SOFA	Simple Ontology Framework API
SOAP	Simple Object Access Protocol
SeRQL	Sesame RDF Query Language
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
TS	Triplespace
TSC	Triplespace Computing
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WP	Work Package
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
XML	Extensible Markup Language

1 Introduction

In order to optimize the access and management of the triple space middleware a concise meta-information vocabulary is required to describe the structure of the space and the contained data. Meta-information is generally applied in all data management and access installation in order to optimize the access procedures and in case of distributed information sources also to improve the distribution and discovery algorithms. Knowledge about the relationship of data and its context allows for more concise and faster processing. Relational databases for example use meta-information about tables, column types and access privileges. Object-oriented databases rely on information about the data structures, while data warehouses and reference information systems provide meta-information about the provenance and quality of information and their sources.

In this deliverable we first analyze the requirements implied by the various work packages that define the components of a triple space kernel, as well as requirements expressed by the application work packages like the project use cases or the (Semantic) Web service integration efforts. Based on these requirements the triple space ontology (TS Ontology) is defined. The use of an ontology-based meta-information infrastructure brings along two major advantages with respect to triple space computing:

- The inference framework of the space middleware allows reasoning not only about the application data, but also about the administrative data, i.e. the meta-information.
- The use of Semantic Web languages for the formalization of the meta-information vocabularies, in particular the application of RDF-technology allows for an integrated platform without additional requirements on the space infrastructure, i.e. the administrative data is processed by the same means as the application data that is published and consumed by space users.

The meta-information infrastructure will provide the classes, properties and relations that enable the description of spaces, data items and interaction patterns in order to improve the scalability of the space installation and the discovery process (latency and quality) for the desired semantic data. The first part of the TS Ontology engineering process concentrates on the latter aspect, the enhanced discovery of data, i.e. the ontology aims at providing means to annotate the application data in order to allow the definition of refined templates to the space (queries that take the context of data into account: e.g. publisher, access logs). At a later point in the project the ontology will have to be extended in order to cope with the requirements implied by the semantic clustering and distribution algorithms (cf. Chapter 4).

Another major target of the meta-information is the modeling of security and trust aspects of the triple space infrastructure such as user roles and access permissions. In accordance with work package 5: Security and Trust it was decided that the definition of the corresponding ontology models is complementary to this work, however not an integrate part of the current effort. The incomplete definition of security and trust related concepts and properties would limit the possibilities of the respective working group and would thus have a negative impact on the resulting ontological model. The security and trust extension resulting from ongoing and future efforts in WP5 will be based on the TS Ontology introduced in this deliverable and is thus fully compatible with the core concepts hereof. First ideas of the security and trust extension is given

in Chapter 4, for more detailed technical details the reader is however referred to deliverable D5.2: Definition of security and trust support models for the reference architecture [8].

After the requirements analysis in Chapter 2, the ontology is defined in Chapter 3. The usage and visibility of the ontology, as well as the features of the ontology are defined. Moreover the core results of the initial evaluation round and the evaluation questionnaire are depicted in this chapter. In Chapter 4 we shortly discuss the above-mentioned extensions to the core ontology: security and trust extension and the semantic clustering and distribution extensions. Finally the implementation is shortly discussed in Chapter 5, while the formalized versions of the TS Ontology are given in the Appendices A (RDFS), B (OWL-Lite) and C (WSML-Flight).

2 Requirements Analysis

This chapter presents and shortly evaluates the requirements expressed by the various working groups of the project. Special emphasis is given to the requisites of the three work package directly related to the space implementation: storage (WP1), triple space knowledge representation (WP2), and triple space interaction (WP3). Moreover, the needs of the security work package and the use cases are also considered in a separate section.

First the requirements of each work package are outlined. These requirements were written by the respective work package members using their own terms. This is for example the reason why the requirements section for WP2 uses the term tuple rather than triple. After the outline of the requirements we give a discussion of the issue raised with respect to the development of the TS Ontology. Finally a set of related competency questions [10, 18] are stated. These questions were used to determine the relevant terms of the meta-information vocabulary and eventually applied to define the TS Ontology (Chapter 3).

2.1 Requirements from Storage (WP1)

The storage layer is not per se imposing any requirements on the triple space. WP1 will provide a stored infrastructure to automate all tasks related to the efficient persistence and retrieval of large quantities of information represented as triples (cf. TripCom D1.2 [16]). The storage layer is considered to be a passive component or some sort of data service. Because it will be used as a service, the triple space needs to maintain some specific information about the existing storage infrastructure and the used underlying data sources, as it might be possible to use multiple storage instances for a set of data sources.

The storage work package expects that knowledge about the type of storage, the performance, readability and connectivity thereof could be of interest to optimization tasks. The following list outlines roughly some possibly interesting terms that could help to efficiently use the storage layer:

- Type of storage
  - type (e.g., RDBMS, File, RDF-Store, Multistore)
  - retrieval methods, i.e. query language support
  - read only (yes — no)
  - transaction safe (yes — no)
- Performance
  - operation latency of insert, delete, retrieve
- Readability
  - uptime number
  - number of executed queries
- Connectivity
  - connected data sources
  - average connection latency

- average network speed
- last network speed

Discussion

The TS Ontology is about modeling spaces and data in order to optimize the management thereof. As stated in the beginning, the storage infrastructure is, with respect to the management layer, a passive entity and hence does not impose any requirements. The outlined terms seem to be highly relevant for the description of the storage infrastructure, but are consequently out of the scope of an ontology that describes the data models.

There are however some aspects touched that might influence the behavior of the space management layer: the retrieval and access methods like transaction support or query language expressivity.

The other characteristics like performance and readability will in turn not be considered and are seen to be descriptive factors of the storage service. This means only that the facts remain out of scope of the TS Ontology and not that they are not relevant for an optimized space implementation. One possible approach could be the definition of a storage service endpoint in form of a WSML service description that is linked to the space ontology.

Competency Questions

- which types of repositories are available at kernel K?
- which query engine support is provided by the storage layer?
- which query language is understood by the local storage access point?

2.2 Requirements from Triple Space Knowledge Representation (WP2)

In the context of WP2 the TS Ontology fundamentally provides a means to represent the tuple and tuplespace models in a controlled and formalized way. For the terms used in this section the reader is referred to the specifications in TripCom D2.1 [21]. The ontology should first of all provide concepts and properties to describe the organization, content and characteristics of the space. With respect to the former we mention the following aspects:

- in which spaces is a tuple visible?
- how is a space structured in terms of sub-spaces and which sub-spaces are contained in a particular space?
- how are related spaces referring to each other?

Regarding the Semantic Web content of the tuples the ontology should also consider the fact that tuples can now be linked to other tuples:

- which tuples are interlinked or connected (so-called nested tuples)?
- what is the semantics of the tuple fields and values?

Both tuples and spaces can be associated to meta-information related to provenance, time stamps or access rights. The latter is however out of the scope of the core tuplespace ontology and will be elaborated in the context of WP5, which is dedicated to security issues.

The ontology should moreover provide an overview and a formal description of the physical organization of a space:

- how is a space physically partitioned, where are the partitions, what kind of data do these partitions contain?
- which replicas are available for a space or one of its partitions, where are these replica, what data do they contain?

Last but not least, the ontology offers implicitly a means to create and address tuples and spaces. In ontologies all terms and instances are named by URIs, which in turn provide the identifiers to directly address tuples and spaces orthogonally to the associative addressing based on the template mechanism commonly used in Linda systems [7].

We expect that the ontology will be used to refine the query processing algorithms provided in WP3, e.g. by restricting queries to particular spaces based on contents or meta-information. A second use case relates to issues of distribution, clustering and replication. In the scope of these upcoming tasks it can provide the information necessary to deduce distribution and clustering algorithms based on reasoning results, and to enrich distributed query answering heuristics.

Discussion

The knowledge representation work package is an important source for requirements for the prospected ontology, as it is obviously, just as WP3, one its main users. The desired features to describe organizational models, content and distribution (respectively replication and partitioning of data) will hence constitute the core of the TS Ontology.

WP2 works based on the assumption that triple spaces are semantically enhanced tuplespaces, hence also the use of the general terms *tuplespace* and *tuples* throughout the requirements analysis. The terms *triple space* and *triple* will be in turn used to refer to the data model in the specific target setting of this project. In consequence within TripCom, and in the context of this deliverable, the terms are often used interchangeably.

Competency Questions

The competency questions listed hereafter are partly based on further considerations of the structure of RDF data. Several RDF triples are often grouped to so-called RDF graphs. Based on statements and requirements of other work packages it is foreseen that it shall be possible to group triples in graphs and hence to manipulate and annotate sets of triples in addition to individual triples. Therefore some of the competency questions are about RDF graphs too.

- which RDF triples belong to the RDF graph G?
- what spaces does a RDF triple $\langle$subject, predicate, object$\rangle$ belong to?

- what spaces does space S include?
- which space is the super-space of space S?

- which kernels are associated to space S?
- which triples are managed at kernel K?
- on which kernels do we find the triple ⟨subject, predicate, object⟩?
- which partitions/replica are available for space S and where are they located?
- which other kernels contain information on a given topic?

2.3 Requirements from Triple Space Interaction (WP3)

The interaction work package is about enhancing the Linda primitives and common template-based matching with semantics in order to have more expressive means of communication (cf. TripCom D1.3 [17]). In particular this work package looks at the integration of Semantic Web query languages with the operations and matching algorithms of traditional Linda systems.

Interesting features that might enhance the service of WP3 are related to the flexibility of the matching/querying procedures applied. The ontology should provide some means to model the use of and tools for underlying query languages and query engines in order to choose the desired matching algorithms. In the context of WSMO it is assumed that the actual algorithm is described by use of a WSML Web service description (just as possible for all other components of the TS-SOA). This descriptions need however to be bound to the supporting space installation, as not all spaces on all nodes will support the same matching technologies and languages. In summary WP3 expects means to:

- describe and annotate the template matching procedure
  - choose the best matching algorithm to be applied (possibly via WSML)
  - describe functionality of applied query engine
- type of query language support for template matching and data formats
- description of means of mapping between query languages and data formats
- identification of the formalism applied for data encoding

The last point in this listing refers to the fact that also languages with higher expressivity like the Horn facts of plain RDF can be serialized in triple form and thus represented as RDF; there is OWL/RDF and WSML/RDF for example. The semantics of information encoded in OWL or WSML can however only be fully understood if respective inference or reasoning support can be provided, hence the desire to reflect the language formalism of sets of triples. The potential heterogeneity of data formats and query engines at triple space nodes may lead to matching procedures which take advantage of available mappings to provide a 'best match' functionality.

Discussion

On the one hand the desired features expressed by WP3 overlap with some of the requirements derived from WP1: nomination of query engine and query language support. Knowledge about the query engine and the supported query language might

influence the type of templates that can be expressed. Such knowledge could be taken into consideration by the space users at API level or internally by the Query Processor [5] to map to the right language. On the other hand this work package requires additional information about language formalisms and access possibilities. This elements are clearly data targeted and hence the work package simply requires a more expressive annotation infrastructure for published data. In that sense these features are expected to be fully integrated into the TS Ontology.

Competency Questions

- which triples model OWL data?
- what language formalism is used to encode the data contained in the graph G?
- which query engine support is provided at this triple space node?
- which query language is understood by the local (storage) endpoint?
- how do I access the chosen query engine and what type of response will I get?

2.4 Requirements from Other TripCom Work Packages

In this section the adjacent work package on security and trust (WP5) is considered, as well as the use case work packages. The Triple Space and Semantic Web services work package (WP4) is considered in this deliverable to be another use case with respect to Semantic Web service, as the infrastructure defined in WP4 is clearly making use of the space from outside of the core installation.

The security and trust support for triple space computing requires means to model users, roles, security credentials on the one hand and trust, in particular distributed trust mechanism on the other. In the course of the ontology engineering process it was however noted that these elements are better taken out of the TS Ontology and integrated in a dependent, but separate security and trust ontology. This extension to the TS Ontology will be defined in WP5 (cf. Section 3.3).

WP4 addresses issues like the discovery, invocation of services, as well as the sharing of Semantic Web service descriptions. The Web Service Execution Environment (WSMX, [14]) as a reference implementation of Semantic Web services is considered in the work package and analyzed to explore the possible benefits the WSMX can get by incorporating triple space computing. Concerning this, the integration of WSMX and triple space computing has been analyzed and proposed in four different aspects, i.e. (1) enabling component management in WSMX using triple space computing, (2) external communication grounding in WSMX using triple space computing, (3) Resource Management in WSMX using triple space computing, and (4) enabling the communication of different inter-connected WSMX and then to build an application scenario to show the viability. Each of the integration aspect has been explored in detail in the work package.

These descriptions of Web services are expected to be provided in form of WSML-files, consist, when serialized into triple form, of many related triples without meaning when considered individually; thus WP4 requires support for communication and coordination by RDF graphs. The retrieval of service descriptions is often based on the unique identifiers of the metadata elements or optimized by help of context information like the publisher of a graph, the regency of publication or trustworthiness of the implementer.

The Enterprise Application Integration use case of WP8a needs to store EDI messages and services in the triple spaces. A service mainly involves a set of EDI messages and must be identified by a unique identifier that refers to all the EDI messages included. The main requirement of WP8a is again the identification by URI of complex objects, i.e. graphs.

The core feature of the eHealth use case is the sharing of patient summaries and records. Such patient records imply serious consideration of privacy issues. On the one hand the privacy concerns are addressed by the technologies defined in WP5, on the other they require precise auditing. WP8b hence requests the installation of access logs per data, which records who and when a data item was read, mortified or written.

Discussion

First of all we repeat that security and trust aspects are not considered in the TS Ontology presented in this deliverable (cf. Section 3.3).

The three use cases (WP4, WP8a and WP8b) consistently require the support for resource identification on set level, i.e. treatment of information in form of graphs with associated identifiers. In this respect the reader is also referred to the concept of named graphs [2]. The user needs the publish, retrieve, remove and update features of the named graphs. A flexible set of ways is required to enable search for RDF graphs based on any possible information the user may have. The clients require subscription mechanisms to get the notification of any updates for any particular information on triple space. The clients also require the mechanisms to distinguish the on-going communication between different clients simultaneously.

Moreover the existence of access and provenance information seems to be highly relevant in order to ensure the integrity and quality of information. The TS Ontology is thus foreseen to provide means to model access log entries with addresser information and timestamping.

Competency Questions

- what triples did author A publish?
- about how many triples published by user A are in the space S?
- which spaces contain triples published by user A?
- which users publish data to space S?
- was the triple ⟨subject, predicate, object⟩ written by user A?
- who is the author of a triple ⟨subject, predicate, object⟩
- who is the author of the triple with id T?
- what triples have been published in space S by user A?
- when was a given triple T published to the space S?
- what is the latest triple written by user A?
- what is the latest triple added in space S?
- which graphs were modified by user A in space S after 5:00 pm GMT?
- when did user $A1$ modify the triple ⟨subject, predicate, object⟩ in space S published by user $A2$?
- who last modified a particular graph G?
- at what time was the graph G modified last?
- which user read the graph G last?

A further set of questions that were posed during the competency question collection phase are related to the collection of statistical data. These question are expected to be answerable with help of reasoning support and statistical information is always changing constantly. Hence, this kind of data however has not been assumed to be determined explicitly as part of the ontology.

- how often has the triple ⟨subject, predicate, object⟩ been accessed in the last two hours?
- how often has space S been accessed in the last two hours?
- which is the most "popular" triple?
- which is the most "popular" space?

3 TS Ontology Specification

Based on requirements analysis from the previous chapter, we identify the set of primitives of the ontology to represent triple spaces, data and other organizational issues with respect to space modeling and data distribution.

In Section 3.1 we first outline the features of the ontology, i.e. discusses the functionality which is provided and which is not. Thereafter we present the TS Ontology itself and explain in detail the different parts, classes and properties, before depicting shortly some results of the initial evaluation and the applied evaluation framework in Section 3.3.

In Section 3.4 we discuss in detail the expected applications of the ontology and the usage and access rules. The TS Ontology provides the vocabulary for the administrative data, which, in contrast to user data, models information that should not be altered by external space users.

3.1 Description of Ontology Features

In this section the features of the TS Ontology are presented and discussed in informal ways. A well-structured, but only semi-formal description will be given in Section 3.2, while the implementations are available online.[1]

Due to the technical focus and the specificity of the domain modelled by the TS Ontology and the novelty of the application scenario in the context of Semantic Web, ontology reuse did not provide a feasible instrument to aid the development process. In order to keep the recognition effect and the interoperability high, a number of renowned vocabularies were used and integrated with the ontology. They provide partial vocabularies, but no larger solutions to the problems of the TS Ontology. Of particular interest due to the target domain were typical vocabularies for the modeling of data, users and their dependencies:

- http://www.foaf-project.org/
  Friend-Of-A-Friend for the annotation of user agents
- http://www.dublincore.org/
  Dublin Core for the annotation of data
- http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/
  Simple part-whole for the part and wholes relationship of data and spaces

Moreover the traditional Semantic Web vocabularies provided by RDF[2], RDFS[3] and XML Schema datatypes[4] were reused where appropriate.

The development process focused however on the collection of competency questions presented in Chapter 2, as the aforementioned vocabularies only provide marginal support to the domain in question. At first the questions delivered a set of key terms which were required to cover the domain. According to [18] the particular role, class, instance, attribute or relationship, did not matter at this point. It is necessary to note

[1] http://www.tripcom.org/ontologies/tsonto-core.php
[2] http://www.w3.org/RDF/
[3] http://www.w3.org/TR/rdf-schema/
[4] http://www.w3.org/TR/xmlschema-2/

that hereafter only the terms that appear in the TS Ontology are considered. The competency questions also raised issues that were removed after the evaluation round, e.g. distribution and security and trust elements (cf. Section 3.3).

The following paragraphs introduce and shortly explain all of the extracted key terms:

> A ***triple space*** is a virtual container for semantic ***data*** and is generally a ***subspace*** of another triple space (unless the space is a subspace of the global root space). A virtual space is envisioned to provide a scoped interaction platform specified for a particular ***topic***. Furthermore the vocabularies which are mainly applied within a given space might be seen to ***define*** the space and hence provide additional information about the content of it. A space can thus be characterized by two different types of annotations: (1) an external resource or topic giving a hint about the target of the space, and (2) a vocabulary or ontology that defines the scope of the space.
>
> The semantic data can be stored in form of ***triples*** with the same syntax and semantics as RDF triples or as RDF ***graphs*** which consist of several triples [12]. A triple can thus be ***a part of*** a graph. The triple format is for many languages besides pure RDF only a serialization and does hence not reflect the semantics of the language ***formalism*** applied. The issue with the language becomes obvious when considering the work of WP4 where Web service descriptions originally written in WSML are published in RDF triple format to a space. A Semantic Web service description written in WSML-DL would result in a graph a several dozens of triples. The graph is tagged with the language identifier of WSML-DL in order to make sure that any potential reader knows how to deserialize and interpret the triples of the graph.
>
> Data is always written by a particular user ***agent*** at a given ***point in time*** into a triple space; this agent is then the ***publisher*** of that piece of information. The information about the publisher and the time of publication constitute the first ***access log entry***. Whenever a data item in a space is accessed: ***written***, ***read***, ***modified*** or eventually ***deleted***, a new access log entry is written to the administrative data.
>
> In reality a user does not access a space, as it only exists virtually, but rather a so-called ***kernel***. A kernel is the implementation of a triple space access point, which could, according to be TripCom D6.2 [5], be installed on several nodes. From a WP2 management point of view we however only care about processes and hence a kernel implementation. The actual node(s) on which the space processes are running are in this situation of no importance. Furthermore a kernel ***manages*** a number of spaces which are known and accessed by the local user. A kernel does not only manages several spaces, but a space might be ***shared*** by more than one kernel when accessed by multiple users. In order to ensure persistency most kernels will locally run a data store, a ***repository*** which is ***accessed*** by use of a ***query engine***. In that sense we see a kernel to be using a query engine which in turn uses a repository to store and request data. In order to retrieve data, a particular ***query language*** must be applied. Such a query language

could be SPARQL, the currently most renown language in the Semantic Web.

So far we introduced the terms that will constitute the core TS Ontology. A particular strength of ontology-driven middleware management is the possibility of reasoning about participating nodes and disperse data. This becomes particularly evident when dealing with distributed data. Above, in the last paragraph, we already shortly mentioned the possibility of running spaces over several kernels. At a later point of the project the knowledge representation work package will look at data distribution algorithms, i.e., replication, caching and semantic clustering.[5] Considering such measures it will be necessary to extend the TS Ontology. A first attempt adds the subsequently introduced terms in form of a first extension to the TS Ontology:[6]

> A piece of data can be ***managed at*** any kernel that shares a given space in which the data item was stored; inversely a kernel ***manages*** this piece of information. Indirectly this also relates different kernels and hence one kernel can ***point*** to another related kernel that contains similar data.
>
> Just as much as data can be distributed over interlinked kernels, it is possible to ***store it in a particular repository*** whenever a kernel has a hand on more than only one local storage unit. Hence, data can exist in a repository independently of a particular kernel, while from a kernel point of view, data is always stored in a repository. The properties of the repository and the corresponding query engine support can influence the decision of where and what to retrieve.

As mentioned previously, and also obvious from the limited dimension of terms, this is only a first outline of possible direction to go. These terms are seen as hooks for later work and presented in order to give a first idea of potential extension in the direction of ontology-driven space management and data distribution and clustering.

Consequently the core TS Ontology defines the following classes:

- Triple $\sqsubseteq$ rdf:Statement, Data
- Space
- Graph $\sqsubseteq$ Data
- Agent
- AccessLogEntry
- Kernel
- Repository
- QueryEngine

Triple subsumes rdf:Statement in order to inherit the semantics and the attributes defined for the latter; that are rdf:subject, rdf:predicate and rdf:object that are needed to address the individual fields of the triple. Figure 3.1 depicts the relationship of Triple, Graph and Spaces as described above. To simplify the outline we neglected the Kernel class.

The prefix-namespace binding for *rdf* is given in the listing below together with all other relevant bindings that will be applied throughout the remainder of this document:

[5] The tasks T2.4 semantic clustering solutions and T3.5 semantic matching in distributed spaces are dedicated to such issues.

[6] http://www.tripcom.org/ontologies/tsonto-distr.php

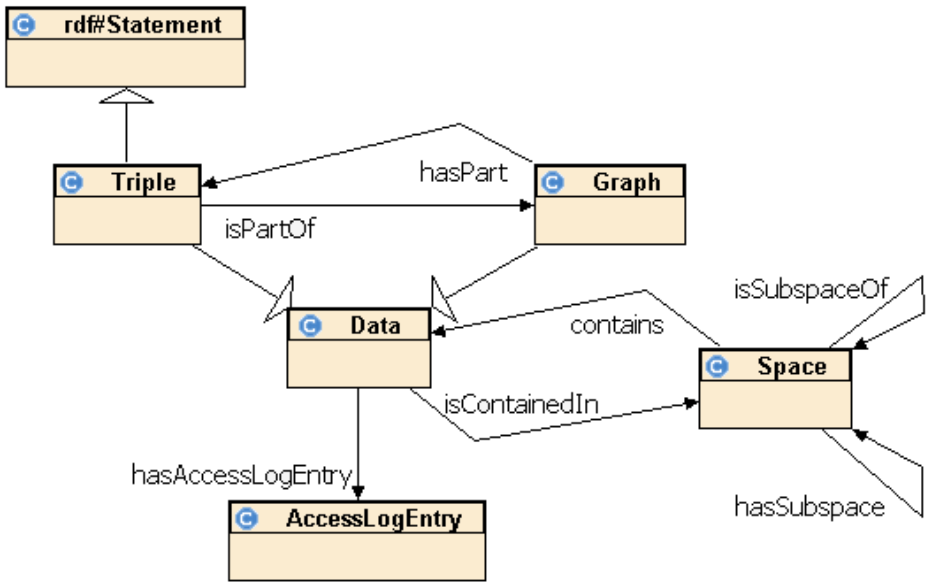

Figure 3.1: Partial TS Ontology: Triples, Graphs, and Spaces

rdf : `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
rdfs : `http://www.w3.org/2000/01/rdf-schema#`
xsd : `http://www.w3.org/2000/10/XMLSchema#`
foaf : `http://xmlns.com/foaf/0.1/`
dc : `http://purl.org/dc/elements/1.1/`
spwh : `http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#`

To conclude the outline of the ontology features we provide a listing of mappings from the key terms introduces above to the actual properties depicted in the next section.

In Table 3.1 we first present the properties taken from existing vocabularies, while the second listing (Table 3.2) provides the relations and attributes newly defined for the TS Ontology. The 'Property' column contains the name of the property followed by an indication of its domain and range: property [domain → range].

The TS Ontology presented within this deliverable aims thus at the modeling of spaces and data which includes means for auditing access records. Moreover the ontology already addresses first possible approaches for the management of data and spaces of more advanced, distributed and clustered implementations. The facets, i.e., cardinalities, transitivity, symmetry or reflexivity of properties were not given so far. These traits are directly depicted in the semi-formal ontology listing of the next section.

3.2 TS Ontology

The following outline of the TS Ontology provides a semi-formal (non language specific) definition of the classes and properties and their facets. The definitions are based on the description of the domain given in Section 3.1. The listing served as the basis for the implementation (formalization) of the ontology in RDFS, OWL-Lite and WSML-Flight (cf. Chapter 5). RDFS provides the ground implementation and most

Key Term	Property
a part of a graph	spwh:isPartOf [Triple → Graph]
inverse of above	spwh:hasPart [Graph → Triple]
language formalism of a data item	dc:format [Data → URI]
link to a topic of a space	rdfs:seeAlso [Space → URI]
link to a defining vocabulary	rdfs:isDefinedBy [Space → URI]
name of a user agent	foaf:name [Agent → xsd:string]
the publisher/owner of a data item, also used to link the agent that accesses a triple	dc:publisher [AccessLogEntry → Agent]
time, the time of publication or access	dc:date [AccessLogEntry → xsd:dateTime]
identifier of a query language	dc:language [QueryEngine → URI]

Table 3.1: Mappings from key terms to properties (1)

obvious approach for spaces tailored at RDF-triple data. However the facets cannot be reflected without OWL support. The WSML-Flight version is provided in order to have a hand on the language constructs of WSMO and moreover to support the basics of an LP-language.

For the formalized ontologies the reader is referred to the appendix of this deliverable or to the online catalogue of the TS Ontology at `http://www.tripcom.org/ontologies/`. Please note that in this section we provide a consolidated version of what was previously referred to as tsonto-core and tsonto-distr.

Triple

$\sqsubseteq$ rdf:Statement, Data

- spwh:isPartOf inverse(spwh:hasPart) $< 0 \ldots * >$ Graph

Graph

$\sqsubseteq$ Data

- spwh:hasPart inverse(spwh:isPartOf) $< 1 \ldots * >$ Triple

Data

- isContainedIn inverse(contains) $< 1 \ldots * >$ Space
- hasAccessLogEntry $< 1 \ldots * >$ AccessLogEntry
- dc:format $< 0 \ldots 1 >$ rdfs:Resource

  Every piece of data is encoded by a particular formalism and should preferably be interpreted by an engine understanding this formalism. Example are

  - rdfs: http://www.w3.org/2000/01/rdf-schema#
  - owl: http://www.w3.org/2002/07/owl#
  - wsml: http://www.wsmo.org/wsml/wsml-syntax#

Key Term	Property
data is contained in the space it is written to	isContainedIn [Data → Space]
whenever a data item is accessed a new log entry is written	hasAccessLogEntry [Data → AccessLogEntry]
from tsonto-distr: data is managed at minimally one kernel	isManagedAtKernel [Data → Kernel]
from tsonto-distr: data is persistently stored in a repository	isStoredInRepository [Data → Repository]
besides the root space, any space is subspace of exactly one other space	isSubspaceOf [Space → Space]
inverse of above	hasSubspace [Space → Space]
inverse of isContainedIn	contains [Space → Data]
any space is shared by all users kernels accessing it	isSharedAtKernel [Space → Kernel]
an access log is associated with a data item: *inverse of hasAccessLogEntry*	refersToData [AccessLogEntry → Data]
type of access to a data item	type [AccessLogEntry → Write, Read, Modify, Delete]
inverse of isSharedAtKernel	sharesSpace [Kernel → Space]
inverse of isManagedAtKernel	manages [Kernel → Data]
a kernel can point to another kernel with related scope	seeAlsoKernel [Kernel → Kernel]
a kernel uses a query engine to access a repository	hasQueryEngine [Kernel → QueryEngine]
a query engine uses a repository to find data to a request	accessesRepository [QueryEngine → Repository]

Table 3.2: Mappings from key terms to properties (2)

- isManagedAtKernel inverse(manages) $< 1 \ldots * >$ Kernel
- isStoredInRepository $< 1 \ldots * >$ Repository

Space

- contains $< 0 \ldots * >$ Data
- hasSubspace inverse(isSubspaceOf) $< 0 \ldots * >$ Space
- isSubspaceOf inverse(hasSubspace) $< 0 \ldots 1 >$ Space
- isSharedAtKernel inverse(sharesSpace) $< 1 \ldots * >$ Kernel
- rdfs:seeAlso $< 0 \ldots * >$ rdfs:Resource

  This is used to indicate a resource that might provide additional information about the subject resource: e.g., a topic of the content of the space.
- rdfs:isDefinedBy $< 0 \ldots * >$ rdfs:Resource

  This is used to indicate a resource defining the subject resource. This property may be used to indicate an RDF vocabulary in which a resource is described. In the case of the TS Ontology, the source would be the ontology or schema from which the data in the space is derived, hence the range is the URI of an ontology.

Agent

- foaf:name $< 0 \ldots 1 >$ xsd:string

AccessLogEntry

- dc:publisher $< 1 >$ Agent
- dc:date $< 1 >$ xsd:dateTime
- refersToData inverse(hasAccessLog) $< 1 \ldots * >$ Data
- type $< 1 >$ read,modify,write,delete

Kernel

- manages inverse(isManagedAtKernel) $< 0 \ldots * >$ Data
- sharesSpace inverse(isSharedAtKernel) $< 0 \ldots * >$ Space
- seeAlsoKernel symmetric, transitive $< 0 \ldots * >$ Kernel
- hasQueryEngine $< 1 \ldots * >$ QueryEngine

Repository

QueryEngine

- dc:language $< 1 >$ rdfs:Resource
- accessesRepository $< 0 \ldots * >$ Repository

3.3 Outline of Evaluation Results

The outline of the ontology model for the meta-information framework of TripCom given above is based on an initial draft of the conceptual model and the evaluation conducted with related experts of the consortium. In this section we shortly depict the feedback given and the main changes resulting thereof.

Before discussing the outcome of the evaluation we shortly introduce the procedure taken. The evaluation was done by a group of related experts from the use case work packages, the security and trust work package and the involved partners in WP1, WP2 and WP3 directly concerned with the applicability of the ontology. The evaluation framework was adopted from [9] and the following questionnaire provided to the reviewers.

The Questionnaire

There were seven questions defined in order to give guidance to the evaluators of the TS Ontology. The seven questions are shown in the section, as they were sent to the evaluators.

Question 1 - Domain of ontology

What types of information should be represented in the TS Ontology and why?

Question 2 - Usage of ontology

Please provide extended feedback to the description of the scope and the usage of the TS Ontology. In particular elaborate on the aspects captured by the ontology which might be of use for space users, and the ones which should exclusively remain space-internal, if any.

Question 3 - Ontology classes

Do the classes defined so far for the TS Ontology contain enough information to answer the competency questions? Do they cover irrelevant information as well? Which aspects could be excluded from the conceptual model and why?

Question 4 - Properties (Relations and Attributes)

Do the properties defined so far for the TS Ontology clearly reflect the relationships between the classes? Are there superficial properties defined? Are any properties missing (in particular to classes that you added in Question 3)?

Question 5 - Consistency

Have you detected any inconsistencies in the ontology? Does it contain any false or contradictory statements?

Question 6 - Extensibility

How would you rate the degree of extensibility of the ontology (1 low to 3 high)? How could the extensibility of the ontology be improved?

This criterion refers to the possibility of adding new definitions to the ontology without altering the existent content. This question is in particular of high importance with respect to fields like security and trust that are currently under-specified and will probably be conceptualized as an extension of the model.

Question 7 - Readability

How would you rate the degree of readability (1 low to 3 high)? How intuitive and self-explaining are the labels denominating the classes and properties included in the ontology? Do you have any suggestions for changes in the names of these ontological primitives?

The Results

In summary there are two major changes required with respect to the first draft of the conceptual model. First, initiated by the security and trust work package all concepts and properties related to the modeling of the access policies vocabulary had to be removed from the ontology. These initial trials would restrict the work of the security and trust working group and imply a certain model. Now, they can independently develop a complete and concise ontological model suited for their needs, and provide it as an extension to the core TS ontology. Second, a common expectation of the ontology is the modeling of information about the content of a space or graph. Several reviewers raised the issue of adding categorizations or topic and ontology bindings to the TS Ontology in order to also describe spaces and hence to cover the whole domain usage. In consequence the RDFS properties rdfs:seeAlso and rdfs:isDefinedBy are reused for the Space concept.

The outcome of the second question do not need further discussion here. The results are shown in Section 3.4.

On the conceptual level one major concern was resolved and incorporated in the final draft of the TS Ontology: the layering and dependencies of Triple, Graph, Data and Spaces was clarified. There were doubts about the initial ideas of having the Graph class as a Container[7] and in consequence the difference between Space and Graph was unclear. With the introduction of the part-relationship between Triple and Graph it was possible to remove the Container class from the ontology. The Triple and Graph classes are now solely used to model data, while the Space class represents the communication medium.

A minor issue was related to the AccessLog class, which was correctly renamed to AccessLogEntry, as each instance only represents one single entry and only all instances together provide the access log of one data unit of class Data.

Question 4 of the evaluation questionnaire concerned the definition of the properties. As feedback several minor comments were provided:

[7]Container was initially installed as super-class to Graph and Space, as a Graph was seen as container for Triples.

- The relationships between Kernel, Repository and QueryEngine were not clear. In consequence it was decided that a kernel uses a query engine with a given language support, while the query engine retrieves the corresponding data from a given repository. In fact, a kernel does not need to have any knowledge about the repository, but rather about the way data can be queried from the persistency framework.

- Some of the reviewers requested more reuse of properties from existing vocabularies like FOAF or DC.

- Some reviewers suggested the use of categorizations or topic and ontology bindings for the description of spaces.

- The logging procedure was also simplified. Initially every data instance was associated with (1) a publishing agent, (2) a timestamp indicating the time of publication and (3) a number of access log entries. However the properties (1) and (2) embody simply the first access log entry of every published data instance and are therefore redundant.

By resolving the above-mentioned issues with respect to the classes, properties and the domain of the ontology all concerns regarding the consistency and extensibility were naturally clarified. On basis of readability there were no significant remarks posted in the evaluation.

Herewith we conclude the short summary of the evaluation process and results. In this last section of this chapter we discuss the usage and visibility of the TS Ontology within the triple space infrastructure. Primarily the ontology is used for the management of the space and thus used internally to the system. However, some elements of the ontology, in particular with respect to the data and spaces are also of use to space users. It is therefore necessary to clearly define who creates the metadata and who and how this information can be accessed. This discussion is subject to the next section.

3.4 Usage and Visibility of TS Ontology

There are two different types of data distinguished in triple space computing. On the one hand there is the application data that is published by the space users. There are basically no restrictions on the potential data published, besides on the format as defined by the interaction primitives of the semantic Linda interface [17]. On the other there is the administrative data used to manage the space infrastructure and the data. On a conceptual level the administrative data is modeled and stored by use of the TS Ontology, i.e. all information encoded by use of the TS Ontology is referred to as administrative data.

However, as there are a priori no restrictions on the application data published in a triple space it is necessary that the system clearly separates the administrative from the application data. Otherwise it would be possible for malicious users to falsify organizational information and manipulate the space infrastructure and the published data. We thus propose in this section measures to keep the two types of data apart.

Users can get in contact with administrative data when either writing data or reading data from the space. For both interaction patterns we define bridges between

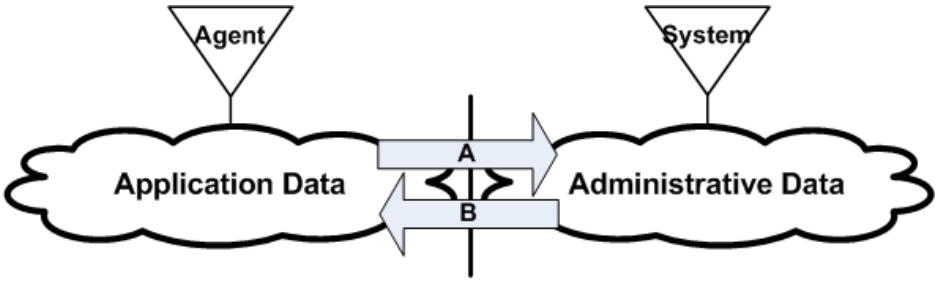

Figure 3.2: Software Agent

the disjoints sets of triples (Figure 3.2). These bridges allow users on the one hand to create administrative data when writing data to the space (Bridge A) and on the other to jointly use administrative meta-information and application data when refining templates for data retrieval (Bridge B). In the following we outline the policies that apply the creation (when writing) and the usage (when reading) of TS Ontology-based administrative data.

Writing : In principle administrative data is created by the space infrastructure itself and independently of any user will. There are however exceptions, as for example when creating a new space - a task clearly triggered by a space administrator and thus an agent external to the space system. In such cases the creation meta-information is initiated by use of dedicated methods that are a part of the Triple Space API: `create(space,superspace)` for example triggers the writing of the triples

```
space rdf:type Space .
space isSubspaceOf superspace .
```

Any other instance data that is based on the TS Ontology and written to the space is considered to be regular application data and will thus not influence the management of the space.

Reading : As outlined earlier in this deliverable one of the targeted uses of the TS Ontology is the refinement of templates, e.g. triples from a particular source, with a given topic or a chosen time of publication. As administrative data is separated from the application data it is necessary to provide special tool not only to write it, but also to make use of it at retrieval. We suggest the usage of appropriate built-in functions that are used within templates to address particular information, e.g. `publishedBy(publisher)` as in Example 2 below that could be used to indicate that only triples published by the given agent should be considered. These built-in functions depend on the expressivity of the matching mechanism defined in WP3.

In the continuation of this section we provide a set of simple examples to outline the chosen approaches. As mentioned previously, on a conceptual level the administrative data is stored in the TS Ontology. The implementation of the space infrastructure requires however a mapping to the tripleset data model of ORDI (of the form `<s p o c {ts_1 ... ts_m}>`) that was introduced in [16, 15] in order to persistently store the instance data in the space memory. The tripleset model is used to provide the link (bridge B in Figure 3.2) between user data and the meta-information without fully

integrating and thus mingle the two disjoint sets. In Table 3.3 the triple data for the subsequent examples is given (for the namespace-prefix bindings the reader is referred to Section 3.1). These triples denote the administrative data for the applications data triple `<a b c>`. To simplify the showcase we assume that only this one single triple is written to the space `S` by use of the `out` operation [17]: `out(<a b c>, S, g)`, where `g` denotes a graph identifier.

`<t rdf:type Triple>`	`<g rdf:type Graph>`
`<t rdf:subject a>`	`<S rdf:type Space>`
`<t rdf:predicate b>`	`<L rdf:type AccessLogEntry>`
`<t rdf:object c>`	`<L dc:publisher P>`
`<t part:isPartOf g>`	`<L dc:date "2007-02-16">`
`<t isContainedIn S>`	`<L ts:type ts:Write>`
`<t hasAccessLogEntry L>`	`<f rdf:value http://www.w3.org/RDF/>`
`<t dc#format f>`	

Table 3.3: Example triples for ontology usage showcase

While these 15 triples seem a lot to annotate in the simplest way one single triple we can largely diminish this number by help of the tripleset data model of ORDI: `<a b c S {g L f}>`. The published triple is implicitly linked to the meta-information through the binding to the triplesets, where `g` still identifies a graph, `L` the access log entry and `f` the formalism of the semantic data, i.e. RDF. Of course also with this approach we have to describe the semantics and the values of the three URIs in the tripleset:

```
<a b c S {g L f}>
<g rdf:type Graph>
<L rdf:type AccessLogEntry>
<L dc:publisher P>
<L dc:date "2007-02-16">
<f rdf:value http://www.w3.org/RDF/>
```

The first data item shows in what way the application data is stored within the space implementation, while the remaining triples indicate the administrative data which remains clearly separated from the user's input. The type of S is implicitly given by the definition of the forth field within TripCom; the context field is always used to indicate the space a triple was published in. Moreover the triple `<S rdf:type Space>` and all additional information about S was already created when some administrator called `create(S)`.

At retrieval time a user may use meta-information to refine the templates. To guarantee the right usage and in order to acknowledge the separation of the administrative data it is necessary make use of built-in functions. The ORDI specification introduced the function `isMemberOfTS()` to be used within SPARQL-query to filter out only the triples that belong to the given tripleset. This member function takes five parameters for the RDF triple, the context field (space identifier) and a tripleset membership URI (Example 1). This approach is advisable in case where the meta-information is fully provided by a single URI like it is e.g. the case with a given graph name. However, whenever the meta-information relies on more specific information that is not included

in the tripleset member URI, e.g. in the case of access log entries, the user is required to specify this additional information within the body of the template (Example 2). We think this approach has a negative impact on the understandability for the user, and we therefore suggest to define appropriate built-in function for all the possible meta-information queries .

Example 1:

In this example the user likes to retrieve all triples of subject `a` if they are part of the graph with identifier g. We provide the query in form of a SPARQL selection query, as this is the formalism understood by the applied ORDI framework:

```
SELECT ?p ?o
FROM S
WHERE { a ?p ?o .
        FILTER isMemberOfTS(a, ?p, ?o, S, g)
      }
```

Example 2:

Using the `isMemberOfTS()` function also for this second example where a user likes to retrieve all triples published by an agent `P` demands a more complex where-clause:

```
SELECT ?s ?p ?o
FROM S
WHERE { ?s ?p ?o .
        L rdf:type AccessLogEntry;
          dc:publisher P .
        FILTER isMemberOfTS(?s, ?p, ?o, S, L)
      }
```

It can be easily observed that the agent that puts together this request is required to have clear insight in the TS Ontology and that such queries could become increasingly complicated. We therefore suggest to provide appropriate built-ins that simplify the template refinement and that suggest the semantics of the filter expression to the user agent. These built-ins should be provided first as part of the ORDI implementation, but also as part of the Triple Space API implementation to enable their usage for templates:

```
SELECT ?s ?p ?o
FROM S
WHERE { ?s ?p ?o .
        FILTER publishedBy(P)
      }
```

The space implementation will internally match this user query to the previously depicted SPARQL-query for resolution. Principally this is a straightforward task, as the semantics of `publishedBy` is known by the space installation. It is expected that the execution of such a mapping is one of the tasks of the Metadata Manager [5].

The reader shall note that these are solely suggestions on how to access and write meta-information in triple space computing while maximizing the visibility for space users and without decreasing the guarantee for secure usage. The precise syntax and expressivity of the triple space templates is defined by WP3 and this section serves only as input to their work.

4 TS Ontology Extensions

As mentioned in Section 3.3 it was decided to exclude security and trust related metadata vocabularies, as well as distribution and replication related terms from the core TS Ontology. These two areas are examples of potential extensions to the core TS Ontology. It is a good practise to keep ontologies small, well modularized and self-contained within a small domain of application [20]. On a larger scale it is nonetheless very important to consider the integration and interoperability of ontologies to model more complex domains with the combination of several such small ontologies. We therefore paid attention to the extensibility and reuse of our core TS Ontology, while future extensions must be well-founded in the core ontology for integrate usage within the triple space infrastructure.

In this chapter we provide a short outline of these future extensions that will be modeled in a integrated way by other tasks of the TripCom project: Section 4.1 is about security and trust, Section 4.2 about distribution-related issues.

4.1 Security and Trust Extension

In the Security and Trust work package (WP5), task T5.2 (due in M18) will include an ontological extension that is used to model access policies based on XACML (eXtensible Access Control Markup Language, [6]), a declarative access control policy language in XML, published by the OASIS standards organization. XACML incorporates a processing model that describes how to interpret the policies. This standard was installed to respond to the need for a common language for expressing security policies. The goal of XACML is to allow enterprises to manage the enforcement of all the elements of its security policy in all the components of its information systems. The main idea is to model high level security policies instead of low level access rights to particular pieces of data.

A complete policy applicable to a particular decision request may be composed of a number of individual rules or sub-policies. For instance, in a personal privacy application, the owner of the personal information may define certain aspects of disclosure policy, whereas the enterprise that is the custodian of the information may define certain other aspects. In order to render an authorization decision, it must be possible to combine the two separate policies to form the single policy applicable to the request (example from [6]). XACML defines how multiple rules and sub-policies are evaluated together.

WP5 intends to ontologize the XACML language or its subset. This policy ontology will be integrated with the TS Ontology presented in this deliverable; it is expected to reuse and extend the TS Ontology at three distinct locations:

- Space: the access policies are specified for a particular space
- Agent: the policies will be related and thus about some particular user agent
- AccessLogEntry: this class of the core TS Ontology will have to be extended, in order to allow the logging of security specific records.

These three access points to the core TS Ontology ensure that the extension will be well integrated with the rest of the metadata vocabulary of the TripCom project.

Even though kept separately for organizational reasons, this approach allows integrated usage and computation.

4.2 Distribution and Replication Extension

As discussed throughout this deliverable the core TS Ontology will be extended by later work in the knowledge representation work package (WP2) to integrate the necessary vocabulary to model the distribution of data and kernels.

First, provisional attempts to showcase possible approaches were already given in this deliverable (Section 3.2). The relations modeled in this respect help to interlink kernels and to bind particular spaces and data to kernels (below from the kernel point of view).

```
Kernel
    manages inverse(isManagedAtKernel) <0...*> Data
    sharesSpace inverse(isSharedAtKernel) <0...*> Space
```

In that way we expect that information can more easily be discovered and retrieved also in a highly distributed infrastructure. However, this is clearly only a first step in addressing the distribution and scalability issues arising when moving away from the one node solution installed with the first prototype of the project.

The distribution and replication extension will have to address issues like semantic clustering (linking of semantically related information) similar to semantic overlay networks in P2P [1, 3, 13], or semantic caching that was first introduced as alternative to page or tuple caching in the late 90's [4, 19]. The idea behind semantic overlay networks is to create pointers or additional links to nodes in a P2P network that are semantically close to one another. In a way this semantic links provide shortcuts and hence shorten the search pass when queries are resolved. Semantic caching on the other hand applies similar ideas for cache replacement policies. Instead of applying traditional strategies like FIFO, LRU (least recently used), LFU (least frequently used), semantic clustering relies on the semantic proximity of queries and data.

These technologies will be further investigated in the upcoming tasks T2.4-T2.7 and the results and ontology extensions presented in the respective deliverables. As the first attempts already reveal, the access points for the integration of the core TS Ontology and this extension will essentially be the kernel concept. The kernel is the physical representant of the space installation and thus embodies the physical characteristics of a triple space.

5 Implementation

The ontology implementation is the last phase of the process of the ontology development. It aims to formalize the specified conceptualization, and bring it to concrete ontology formalization compatible with existing semantic environments and reasoning infrastructures. TS Ontology is implemented in three different ontology languages to reach maximum compatibility with the existing software. The implemented ontologies strictly follow, as far as possible, the semantics prescribed by the conceptualization process.

The RDFS version in Section 5.1 provides the ground implementation and most obvious approach for spaces tailored at RDF-triple data. However the facets cannot be reflected without OWL support, hence the OWL-Lite version described in Section 5.2. The WSML-Flight version is provided in order to have a hand on the language family of WSMO and moreover to have support for a decidable subset of logic programming (Section 5.3) in contrast to description logics that is at the basis of OWL.

5.1 RDFS formalization of TS Ontology

RDF Schema, or RDF's vocabulary description language, is a general-purpose language used to describe resources in the Web and to add limited semantics. The resources are described by properties in terms of the classes of resources to which they apply. The supported expressivity is quite limited and it is mostly suitable to represent taxonomic typing of individuals and resources.

Utilized RDF Schema semantics:

- Class
- rdf:Property
- rdfs:subClassOf
- rdfs:domain
- rdfs:range
- Individual

The RDFS ontology language impose limitations over the correct implementation of the TS Ontology. The properties in RDFS do not support formal ways to define cardinality constrains. Several properties like hasQueryEngine or refersToData are not constrained, which mean in the terms of the RDFS semantics, that arbitrary number of values are allowed. Another limitation imposed by the RDFS vocabulary is the definition of inverse, transitive and symmetric properties semantics. The TS Ontology implementation in RDFS is given in Appendix A.

5.2 OWL-Lite formalization of TS Ontology

OWL stands for Web Ontology Language and introduces several ontology language dialects. OWL-Lite is regarded roughly as extended version of RDF's vocabulary description language to support additional semantics. Its expressivity is enough to apply

cardinality constrains (restricted to 0 or 1) to the properties and specify transitive, inverse and symmetric properties.

Utilized OWL-Lite semantics:

- Class
- rdf:Property
- inverseOf
- TransitiveProperty
- SymmetricProperty
- rdfs:subClassOf
- rdfs:subClassOf applied to class expressions
- rdfs:domain
- rdfs:range
- Individual
- minCardinality (restricted to 0 or 1)
- maxCardinality (restricted to 0 or 1)
- imports
- versionInfo

A design guide-line to implement the ontology in OWL-Lite is to ensure its backward-compatibility with the existing RDFS reasoners. As result all properties are defined as:

```
<owl:ObjectProperty rdf:about="accessesRepository">
  <rdfs:domain rdf:resource="QueryEngine"/>
  <rdfs:range rdf:resource="Repository"/>
</owl:ObjectProperty>
```

Note that owl:ObjectProperty is defined as subclass of rdf:Property, so for RDFS reasoner being able to load OWL ontology located at http://www.w3.org/2002/07/owl it is possible to infer that owl:ObjectProperty is also of type rdf:Property.

Our final recommendation is to use OWL-Lite ontology version even in cases when only strict RDFS reasoners are used. The OWL-Lite ontology is expressive enough to fully map the specified conceptualization to the concrete ontology language. Its implementation is depicted in Appendix B.

5.3 WSML-Flight formalization of TS Ontology

The Web Service Modeling Language (WSML) is full-fledged ontology and rule language used to describe Web services. WSML-Flight layers on top of WSML-Core, which correspond to the intersection of Description Logic and Horn Logic without function symbols and without equality [11]. WSML-Flight offers modeling primitives to enrich WSML-Core with values and integrity constraints.

Utilized WSML-Flight semantics:

- concept
- attribute
- inverseOf
- transitive
- symmetric
- subConceptOf
- cardinality constrains
- impliesType
- instance

The WSML-Flight version of TS Ontology fully formalizes the prescribed semantics by the conceptualization process and is given in Appendix C.

6 Conclusion

This deliverable presents the first results towards an ontology-driven management infrastructure for semantic spaces. The use of ontologies for the management of the distribution, clustering and discovery tasks is seen to be the major asset of triple space computing compared to traditional space-based computing approaches.

Based on the requirements expressed by the other work packages of the TripCom project an ontology model was developed that allows the description of spaces and space hierarchies, the data together with provenance information, the kernel implementations and the relationships amongst these core entities of any space installation.

Security and trust related concepts and properties were neglected on purpose in order not to harm the work of the respectively dedicated work package WP5. The security and trust work package will however develop ontological extensions that are integrated with the core TS Ontology presented in this deliverable. Similarly there will be extensions towards the modeling of the necessary terms and relations to implement discovery, clustering and scalability related algorithms for the distributed setting of future triple space implementations. This clearly shows that the work done for this deliverable is only a begin to a complete vocabulary collection for the management of triple spaces. Follow-up work is expected to be published in the deliverable D5.2 [8] and D2.4.

This deliverable depicted the requirements collection, the ontology features, an outline of the evaluation that was conducted with representatives of the related work packages, as well as additional experts, and an analysis of the visibility of the TS Ontology together with possible approaches to the problem of the separation of application and administrative data. Moreover, we shortly discussed the envisioned extensions to the TS Ontology, as well as the current implementations. The implemented ontologies are available online at `http://www.tripcom.org/ontologies/`.

References

[1] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *3rd Int'l Semantic Web Conf.*, pages 107–121, November 2004.

[2] J.J. Carroll, Ch. Bizer, P. Hayes, and P. Stickler. Named Graphs. *Journal of Web Semantics*, 3(4), 2005.

[3] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, Stanford University, October 2002.

[4] S. Dar, M. Franklin, B. Jónsson, D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. In *22nd Int'l Conf. on Very Large Data Bases*, pages 330–341, September 1996.

[5] M. Murth (ed.). Triple Space Reference Architecture. TripCom Project Deliverable D6.2, March 2007.

[6] T. Moses (ed.). eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, February 2005.

[7] David Gelernter. Generative Communication in Linda. *ACM Transactions on Prog. Languages and Systems*, 7(1):80–112, January 1985.

[8] A. Ghioni, J. Kopecky, and G. Joskowicz. Definition of security and trust support models for the reference architecture. TripCom Project Deliverable D5.2 (forthcoming), September 2007.

[9] A. Gómez-Pérez. Evaluation of Ontologies. *International Journal of Intelligent Systems*, 16(3):391–409, 2001.

[10] M. Gruninger and M.S. Fox. Methodology for the Design and Evaluation of Ontologie. In *Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95)*, 1995.

[11] A. Pollers H. Lausen, J. de Bruijn and D. Fensel. WSML - a Language Framework for Semantic Web Services. In *W3C rules workshop*, April 2005.

[12] P. Hayes and B. McBride. Rdf semantics. W3C Recommendation, February 2004.

[13] A. Loeser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic Overlay Clusters in Super-Peer Networks. In *Int'l Workshop on Databases, Information Systems and Peer-to-Peer Computing*, pages 33–47, September 2003.

[14] A. Mocan, M. Zaremba, M Moran, and E. Cimpian. Filling the Gap – Extending Service Oriented Architectures with Semantics. In *Proc. of the 2nd IEEE Int'l Symposium on Service-Oriented Applications, Integration and Collaboration*, October 2006.

[15] V. Momtchev and A. Kiryakov. ORDI: Ontology Representation and Data Integration Framework. 2nd Generation Specification, Ontotext Lab., October 2006.

[16] V. Momtchev and A. Kiryakov. Specification of the Store Architecture and Interfaces. TripCom Project Deliverable D1.2, September 2006.

[17] L. Nixon, E. Simperl, R. Krummenacher, M. Murth, G. Joskowicz, and E. Kuhn. Specification and Implementation of a Semantic Linda Model. TripCom Project Deliverable D3.1, March 2007.

[18] N.F. Noy and D.L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report SMI-2001-0880, Stanford Knowledge Systems Laboratory, March 2001.

[19] Q. Ren and M.H. Dunham. Semantic Caching and Query Processing. Technical Report TR-98-CSE-04, Southern Methodist University, May 1998.

[20] M.-Ch. Rousset. Small Can Be Beautiful in the Semantic Web. In *3rd Int'l Semantic Web Conf.*, pages 6–16, November 2004.

[21] E. Simperl, L. Nixon, R. Krummenacher, V. Momtchev, and H. Mu noz. Representing RDF semantics in tuples. TripCom Project Deliverable D2.1, March 2007.

A Appendix – TS Ontology in RDFS

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY dc "http://purl.org/dc/elements/1.1/#">
 <!ENTITY p "http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#">
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
 <!ENTITY foaf "http://xmlns.com/foaf/0.1/#">
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://www.tripcom.org/ontologies/tsonto#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#"
    xmlns:p="http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

    <!-- Classes -->
    <rdfs:Class rdf:about="AccessLogEntry"/>
    <rdfs:Class rdf:about="AccessType" rdfs:comment="Class to denote the different types of
        updates defined as instances."/>
    <rdfs:Class rdf:about="Agent" rdfs:comment="The user that performed the changes."/>
    <rdfs:Class rdf:about="Data" rdfs:comment="Class to represent abstract structure as data
        in the Triple Space."/>
    <rdfs:Class rdf:about="Graph">
        <rdfs:comment>Graph is term used in the Triple Space to denote a set of triples .</
            rdfs:comment>
        <rdfs:subClassOf rdf:resource ="#Data"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="Kernel" rdfs:comment="Kernel represents an autonomous node part
        of the space"/>
    <rdfs:Class rdf:about="QueryEngine"/>
    <rdfs:Class rdf:about="Repository" rdfs:comment="Repository represents an RDF Store."/>
    <rdfs:Class rdf:about="Space" rdfs:comment="Space is a Linda term to denote an
        autonomous shared data space."/>
    <rdfs:Class rdf:about="Triple">
        <rdfs:subClassOf rdf:resource ="Data"/>
        <rdfs:subClassOf rdf:resource ="&rdf;Statement"/>
    </rdfs:Class>

    <!-- Properties -->
    <rdf:Property rdf:about="&dc;date">
        <rdfs:domain rdf:resource ="AccessLogEntry"/>
        <rdfs:range rdf:resource ="&xsd;dateTime"/>
    </rdf:Property>
    <rdf:Property rdf:about="&dc;format">
        <rdfs:domain rdf:resource ="Data"/>
        <rdfs:range rdf:resource ="&xsd;string"/>
    </rdf:Property>
    <rdf:Property rdf:about="&dc;language">
        <rdfs:domain rdf:resource ="QueryEngine"/>
        <rdfs:range rdf:resource ="&xsd;string"/>
    </rdf:Property>
    <rdf:Property rdf:about="&foaf;name">
        <rdfs:domain rdf:resource ="Agent"/>
        <rdfs:range rdf:resource ="&xsd;string"/>
```

```
</rdf:Property>
<rdf:Property rdf:about="&dc;publisher">
    <rdfs:domain rdf:resource="AccessLogEntry" />
    <rdfs:range rdf:resource="&xsd;string" />
</rdf:Property>
<rdf:Property rdf:about="accessesRepository">
    <rdfs:domain rdf:resource="QueryEngine" />
    <rdfs:range rdf:resource="Repository" />
</rdf:Property>
<rdf:Property rdf:about="contains">
    <rdfs:domain rdf:resource="Space" />
    <rdfs:range rdf:resource="Data" />
</rdf:Property>
<rdf:Property rdf:about="hasAccessLogEntry">
    <rdfs:domain rdf:resource="Data" />
    <rdfs:range rdf:resource="AccessLogEntry" />
</rdf:Property>
<rdf:Property rdf:about="&p;hasPart">
    <rdfs:domain rdf:resource="Graph" />
    <rdfs:range rdf:resource="Triple" />
</rdf:Property>
<rdf:Property rdf:about="hasQueryEngine">
    <rdfs:domain rdf:resource="Kernel" />
    <rdfs:range rdf:resource="QueryEngine" />
</rdf:Property>
<rdf:Property rdf:about="hasSubspace">
    <rdfs:domain rdf:resource="Space" />
    <rdfs:range rdf:resource="Space" />
</rdf:Property>
<rdf:Property rdf:about="isContainedIn">
    <rdfs:domain rdf:resource="Data" />
    <rdfs:range rdf:resource="Space" />
</rdf:Property>
<rdf:Property rdf:about="&rdfs;isDefinedBy">
    <rdfs:domain rdf:resource="Space" />
</rdf:Property>
<rdf:Property rdf:about="&p;isPartOf">
    <rdfs:domain rdf:resource="Triple" />
    <rdfs:range rdf:resource="Graph" />
</rdf:Property>
<rdf:Property rdf:about="isSubspaceOf">
    <rdfs:domain rdf:resource="Space" />
    <rdfs:range rdf:resource="Space" />
</rdf:Property>
<rdf:Property rdf:about="refersToData">
    <rdfs:domain rdf:resource="AccessLogEntry" />
    <rdfs:range rdf:resource="Data" />
</rdf:Property>
<rdf:Property rdf:about="&rdfs;seeAlso">
    <rdfs:domain rdf:resource="Space" />
</rdf:Property>
<rdf:Property rdf:about="sharesSpace">
    <rdfs:domain rdf:resource="Kernel" />
    <rdfs:range rdf:resource="Space" />
</rdf:Property>
    <rdf:Property rdf:about="isSharedAtKernel">
    <rdfs:domain rdf:resource="Space" />
    <rdfs:range rdf:resource="Kernel" />
```

```
    </rdf:Property>
    <rdf:Property rdf:about="type">
        <rdfs:domain rdf:resource="AccessLogEntry" />
        <rdfs:range rdf:resource="AccessType" />
    </rdf:Property>

    <!-- Instances -->
    <rdf:Description rdf:about="Modify">
        <rdf:type rdf:resource="AccessType" />
    </rdf:Description>
    <rdf:Description rdf:about="Read">
        <rdf:type rdf:resource="AccessType" />
    </rdf:Description>
    <rdf:Description rdf:about="Write">
        <rdf:type rdf:resource="AccessType" />
    </rdf:Description>
    <rdf:Description rdf:about="Delete">
        <rdf:type rdf:resource="AccessType" />
    </rdf:Description>
</rdf:RDF>
```

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY dc "http://purl.org/dc/elements/1.1/#">
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://www.tripcom.org/ontologies/tsonto#"
         xmlns:dc="http://purl.org/dc/elements/1.1/#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

    <!-- Classes -->
    <rdfs:Class rdf:about="#Data" rdfs:comment="Class to represent abstract structure as
        data in the Triple Space; already defined in the core ontology/>
    <rdfs:Class rdf:about="#Kernel" rdfs:comment="Kernel is Linda term to represent an
        autonomous node part of the space; already defined in the core ontology/>

    <!-- Object Properties -->
    <rdf:Property rdf:about="isManagedAtKernel">
        <rdfs:domain rdf:resource="Data" />
        <rdfs:range rdf:resource="Kernel" />
    </rdf:Property>
    <rdf:Property rdf:about="isStoredInRepository">
        <rdfs:domain rdf:resource="Data" />
        <rdfs:range rdf:resource="Repository" />
    </rdf:Property>
    <rdf:Property rdf:about="manages">
        <rdfs:domain rdf:resource="Kernel" />
        <rdfs:range rdf:resource="Space" />
    </rdf:Property>
    <rdf:Property rdf:about="seeAlsoKernel">
        <rdfs:domain rdf:resource="Kernel" />
        <rdfs:range rdf:resource="Kernel" />
    </rdf:Property>
```

```
</rdf:RDF>
```

B Appendix – TS Ontology in OWL-Lite

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY owl "http://www.w3.org/2002/07/owl#">
 <!ENTITY dc "http://purl.org/dc/elements/1.1/#">
 <!ENTITY p "http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#">
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
 <!ENTITY foaf "http://xmlns.com/foaf/0.1/#">
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>

<rdf:RDF xml:base="http://www.tripcom.org/ontologies/tsonto#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:p="http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

    <!-- Ontology Information -->
    <owl:Ontology rdf:about="http://www.tripcom.org/ontologies/tsonto-core.wsml"
        owl:versionInfo="0.1">
        <owl:imports>
            <owl:Ontology rdf:about="&p;" />
        </owl:imports>
    </owl:Ontology>

    <!-- Classes -->
    <owl:Class rdf:about="AccessLogEntry">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
                <owl:onProperty rdf:resource="&dc;publisher" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
                <owl:onProperty rdf:resource="&dc;date" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
                <owl:onProperty rdf:resource="type" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                    owl:minCardinality>
                <owl:onProperty rdf:resource="refersToData" />
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
```

```
<owl:Class rdf:about="AccessType" rdfs:comment="Class to denote the different types of
    updates defined as instances."/>
<owl:Class rdf:about="Agent" rdfs:comment="The user that performed the changes">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:maxCardinality>
            <owl:onProperty rdf:resource="&foaf;name"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="Data">
    <rdfs:comment>Class to represent abstract structure as data in the Triple Space.</
        rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:maxCardinality>
            <owl:onProperty rdf:resource="&dc;format"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
                owl:minCardinality>
            <owl:onProperty rdf:resource="&dc;format"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:minCardinality>
            <owl:onProperty rdf:resource="isContainedIn"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:minCardinality>
            <owl:onProperty rdf:resource="hasAccessLogEntry"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="Graph">
    <rdfs:comment>Graph is term used in the Triple Space to denote a set of triples .</
        rdfs:comment>
    <rdfs:subClassOf rdf:resource="Data"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:minCardinality>
            <owl:onProperty rdf:resource="&p;hasPart"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="Kernel">
    <rdfs:comment>Kernel represent an autonomous node part of the space.</rdfs:comment
        >
```

```
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:minCardinality>
            <owl:onProperty rdf:resource="hasQueryEngine"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#QueryEngine">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            <owl:onProperty rdf:resource="&dc;language"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="Repository" rdfs:comment="Repository represents an RDF Store."/>
<owl:Class rdf:about="Space">
    <rdfs:comment>Space is a Linda term to denote an autonomous shared data space.</
        rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:minCardinality>
            <owl:onProperty rdf:resource="isSharedAtKernel"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:maxCardinality>
            <owl:onProperty rdf:resource="isSubspaceOf"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="Triple">
    <rdfs:subClassOf rdf:resource="Data"/>
    <rdfs:subClassOf rdf:resource="&rdf;Statement"/>
</owl:Class>

<!-- Datatype Properties -->
<owl:DatatypeProperty rdf:about="&dc;date">
    <rdfs:domain rdf:resource="AccessLogEntry"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&dc;format">
    <rdfs:domain rdf:resource="Data"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&dc;language">
    <rdfs:domain rdf:resource="QueryEngine"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&foaf;name">
    <rdfs:domain rdf:resource="Agent"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&dc;publisher">
```

```
    <rdfs:domain  rdf:resource ="AccessLogEntry" />
    <rdfs:range   rdf:resource ="&xsd;string" />
</owl:DatatypeProperty>

<!-- Object Properties -->
<owl:ObjectProperty rdf:about="accessesRepository">
    <rdfs:domain  rdf:resource ="QueryEngine" />
    <rdfs:range   rdf:resource ="Repository" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="contains">
    <rdfs:domain  rdf:resource ="Space" />
    <rdfs:range   rdf:resource ="Data" />
    <owl:inverseOf   rdf:resource ="isContainedIn" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasAccessLogEntry">
    <rdfs:domain  rdf:resource ="Data" />
    <rdfs:range   rdf:resource ="AccessLogEntry" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&p;hasPart">
    <rdfs:domain  rdf:resource ="Graph" />
    <rdfs:range   rdf:resource ="Triple" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasQueryEngine">
    <rdfs:domain  rdf:resource ="Kernel" />
    <rdfs:range   rdf:resource ="QueryEngine" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasSubspace">
    <rdfs:domain  rdf:resource ="Space" />
    <rdfs:range   rdf:resource ="Space" />
    <owl:inverseOf   rdf:resource ="isSubspaceOf" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="isContainedIn">
    <rdfs:domain  rdf:resource ="Data" />
    <rdfs:range   rdf:resource ="Space" />
    <owl:inverseOf   rdf:resource ="contains" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&rdfs;isDefinedBy">
    <rdfs:domain  rdf:resource ="Space" />
    <rdfs:range   rdf:resource ="&owl;Thing" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&p;isPartOf">
    <rdfs:domain  rdf:resource ="Triple" />
    <rdfs:range   rdf:resource ="Graph" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="isSubspaceOf">
    <rdfs:domain  rdf:resource ="Space" />
    <rdfs:range   rdf:resource ="Space" />
    <owl:inverseOf   rdf:resource ="hasSubspace" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="refersToData">
    <rdfs:domain  rdf:resource ="AccessLogEntry" />
    <rdfs:range   rdf:resource ="Data" />
    <owl:inverseOf   rdf:resource ="hasAccessLogEntry" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&rdfs;seeAlso">
    <rdfs:domain  rdf:resource ="Space" />
    <rdfs:range   rdf:resource ="&owl;Thing" />
</owl:ObjectProperty>
```

```
    <owl:ObjectProperty rdf:about="sharesSpace">
        <rdfs:domain  rdf:resource ="Kernel" />
        <rdfs:range   rdf:resource ="Space" />
        <owl:inverseOf  rdf:resource ="isSharedAtKernel" />
    </owl:ObjectProperty>
        <owl:ObjectProperty rdf:about="isSharedAtKernel">
        <rdfs:domain  rdf:resource ="Space" />
        <rdfs:range   rdf:resource ="Kernel" />
        <owl:inverseOf  rdf:resource ="sharesSpace" />
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="type">
        <rdfs:domain  rdf:resource ="AccessLogEntry" />
        <rdfs:range   rdf:resource ="AccessType" />
    </owl:ObjectProperty>

    <!-- Instances -->
    <rdf:Description  rdf:about="Modify">
        <rdf:type  rdf:resource ="AccessType" />
    </rdf:Description>
    <rdf:Description  rdf:about="Read">
        <rdf:type  rdf:resource ="AccessType" />
    </rdf:Description>
    <rdf:Description  rdf:about="Write">
        <rdf:type  rdf:resource ="AccessType" />
    </rdf:Description>
    <rdf:Description  rdf:about="Delete">
        <rdf:type  rdf:resource ="AccessType" />
    </rdf:Description>
</rdf:RDF>
```

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY owl "http://www.w3.org/2002/07/owl#">
 <!ENTITY dc "http://purl.org/dc/elements/1.1/#">
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
 <!ENTITY foaf "http://xmlns.com/foaf/0.1/#">
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>

<rdf:RDF xml:base="http://www.tripcom.org/ontologies/tsonto#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

    <!-- Ontology Information -->
    <owl:Ontology rdf:about="http://www.tripcom.org/ontologies/tsonto-distr.wsml"
        owl:versionInfo="0.1">
        <owl:imports>
            <owl:Ontology rdf:about="&p;" />
            <owl:Ontology rdf:about="http://www.tripcom.org/ontologies/tsonto-core.wsml" />
        </owl:imports>
    </owl:Ontology>

    <!-- Classes -->
```

```
    <owl:Class rdf:about="Data">
        <rdfs:comment>Class to represent abstract structure as data in the Triple Space.</
            rdfs:comment>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                    owl:minCardinality>
                <owl:onProperty rdf:resource="isManagedAtKernel"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                    owl:minCardinality>
                <owl:onProperty rdf:resource="isStoredInRepository"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="Kernel" rdfs:comment="Kernel represents an autonomous node part
        of the space."/>

    <!-- Object Properties -->
    <owl:ObjectProperty rdf:about="isManagedAtKernel">
        <rdfs:domain rdf:resource="Data"/>
        <rdfs:range rdf:resource="Kernel"/>
    </owl:ObjectProperty>
        <owl:ObjectProperty rdf:about="isStoredInRepository">
        <rdfs:domain rdf:resource="Data"/>
        <rdfs:range rdf:resource="Repository"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="manages">
        <rdfs:domain rdf:resource="Kernel"/>
        <rdfs:range rdf:resource="Space"/>
        <owl:inverseOf rdf:resource="isManagedAtKernel"/>
        <rdf:type>
            <rdf:Description rdf:about="&owl;TransitiveProperty"/>
        </rdf:type>
        <rdf:type>
            <rdf:Description rdf:about="&owl;SymmetricProperty"/>
        </rdf:type>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="#seeAlsoKernel">
        <rdfs:domain rdf:resource="Kernel"/>
        <rdfs:range rdf:resource="Kernel"/>
    </owl:ObjectProperty>
</rdf:RDF>
```

C Appendix – TS Ontology in WSML-Flight

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.tripcom.org/ontologies/tsonto#",
    rdf _"http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    rdfs _"http://www.w3.org/2000/01/rdf-schema#",
    xsd _"http://www.w3.org/2001/XMLSchema#",
    foaf _"http://xmlns.com/foaf/0.1/",
    dc _"http://purl.org/dc/elements/1.1/",
    part _"http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#" }

ontology _"http://www.tripcom.org/ontologies/tsonto-core.wsml"

concept Triple subConceptOf {rdf#Statement, Data}
  part#isPartOf impliesType Graph

concept Graph subConceptOf Data
  part#hasPart impliesType (1 *) Triple

concept Data
  isContainedIn inverseOf(contains) impliesType (1 *) Space
  hasAccessLogEntry impliesType (1 *) AccessLogEntry
  dc#format impliesType (0 1) rdfs#Resource
    nfp
      dc#description hasValue "Every piece of data is encoded by a particular
        formalism and should preferably be interpreted by an engine understanding
        this formalism. Example are
          rdfs: http://www.w3.org/2000/01/rdf-schema\#
          owl: http://www.w3.org/2002/07/owl\#
          wsml: http://www.wsmo.org/wsml/wsml-syntax\#"
    endnfp

concept Space
  contains impliesType Data
  hasSubspace inverseOf(isSubspaceOf) impliesType Space
  isSubspaceOf inverseOf(hasSubspace) impliesType (0 1) Space
  isSharedAtKernel inverseOf(sharesSpace) impliesType (1 *) Kernel
  rdfs#seeAlso impliesType rdfs#Resource
    nfp
      dc#description hasValue "This is used to indicate a resource that might provide
        additional information about the subject resource: e.g., a topic of the
        content of the space."
    endnfp
  rdfs#isDefinedBy impliesType rdfs#Resource
    nfp
      dc#description hasValue "This is used to indicate a resource defining the
      subject resource. This property may be used to indicate an RDF vocabulary in
      which a resource is described. In the case of the TS ontology, the source
      would be the ontology or schema from which the data in the space is derived,
      hence the range is the URI of an ontology."
    endnfp

concept Agent
  foaf#name impliesType (0 1) xsd#string
    nfp
      dc#description hasValue "A human readable name for the agent"
```

```
    endnfp

concept AccessLogEntry
  dc#publisher impliesType (1) Agent
  dc#date impliesType (1) xsd#dateTime
  refersToData inverseOf(hasAccessLog) impliesType (1 *) Data
  type impliesType (1) AccessType

instance Read memberOf AccessType
instance Modify memberOf AccessType
instance Write memberOf AccessType
instance Delete memberOf AccessType

concept Kernel
  sharesSpace inverseOf(isSharedAtKernel) impliesType Space
  hasQueryEngine impliesType (1 *) QueryEngine

concept Repository

concept QueryEngine
  dc#language impliesType (1) rdfs#Resource
  accessesRepository impliesType Repository
```

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.tripcom.org/ontologies/tsonto#" }

ontology _"http://www.tripcom.org/ontologies/tsonto-distr.wsml"

importsOntology _"http://www.tripcom.org/ontologies/tsonto-core.wsml"

concept Data
  isStoredInRepository impliesType (1 *) Repository
  isManagedAtKernel inverseOf(manages) impliesType (1 *) Kernel

concept Kernel
  manages inverseOf(isManagedAtKernel) impliesType Data
  seeAlsoKernel symmetric transitive impliesType Kernel
```