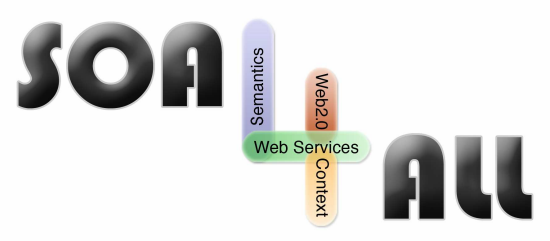


Project Number: **215219**

Project Acronym: **SOA4ALL**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D1.3.1 Semantic Spaces: A Unified Semantic Data Coordination Infrastructure

Activity N:	A1 Fundamental and Integration Activities
Work Package:	1 Service Web Architecture
Due Date:	M6
Submission Date:	30/08/2008
Start Date of Project:	01/03/2008
Duration of Project:	36 Months
Organisation Responsible of Deliverable:	UIBK
Revision:	1.0
Author(s):	Omair Shafiq (UIBK), Tomas Vitvar (UIBK), Yosu Gorroñogoitia (ATOS), Carlos Pedrinaci (OU), Sven Abels (TIE), Imen Filali (INRIA), Matteo Villa (TXT), Marc Richardson (BT) & Fabrice Huet (INRIA) & Rafael Gonzalez Cabero (ATOS).

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	10 June 2008	First draft and table of contents	Omair Shafiq (UIBK), Tomas Vitvar (UIBK)
0.2	13 June 2008	ToC refinement and assigning tasks to partners	Omair Shafiq (UIBK), Tomas Vitvar (UIBK)
0.3	30 June 2008	Semantic Spaces architecture description	Omair Shafiq (UIBK), Tomas Vitvar (UIBK)
0.4	10 July 2008	Added contributions from Open University, ATOS, TIE and INRIA	Yosu Gorroñogoitia (ATOS), Carlos Pedrinaci (OU), Sven Abels (TIE) & Imen Filali (INRIA)
0.5	11 July 2008	Added section for analysis of state-of-the-art	Omair Shafiq (UIBK)
0.6	14 July 2008	Comments of all sections from partners, added functional and non-functional requirements	Omair Shafiq (UIBK)
0.7	18 July 2008	Added inputs from TXT, and some changes in deliverable structure	Omair Shafiq (UIBK), Matteo Villa (TXT)
0.8	20 July 2008	Added input from BT	Marc Richardson (BT)
0.9	21 July 2008	Chapter 1 finalized	Omair Shafiq (UIBK)
1.0	21 July 2008	Updated all UIBK sections	Omair Shafiq (UIBK)
1.1	22 July 2008	Chapter 2 finalized	Omair Shafiq (UIBK)
1.2	23 July 2008	Added input from INRIA, TXT and ATOS	Fabrice Huet (INRIA), Matteo Villa (TXT), Yosu Gorroñogoitia (ATOS)
1.3	24 July 2008	Added input from TIE	Sven Ables (TIE)
1.4	25 July 2008	Analysis of state-of-the-art completed	Omair Shafiq (UIBK)

1.5	26 July 2008	Chapter 3 finalized	Omair Shafiq (UIBK)
1.6	28 July 2008	Added input from INRIA	Fabrice Huet (INRIA)
1.7	29 July 2008	Added Conclusions	Omair Shafiq (UIBK)
1.8	30 July 2008	Chapter 4 finalized	Omair Shafiq (UIBK)
1.9	31 July 2008	Added input from ATOS on Storage of Service monitoring data (in chapter 3)	Rafael Gonzalez Cabero (ATOS)
2.0	31 July 2008	Overall review and updates	Omair Shafiq (UIBK)
2.1	4 August 2008	Updates in section 4.2.4	Fabrice Huet (INRIA)
2.2	26 August 2008	Addressing reviewer's comments	Omair Shafiq (UIBK)

Table of Contents

Glossary of Acronyms

Acronym	Definition
D	Deliverable
EC	European Commission
WP	Work Package
SW	Semantic Web
SOA	Service Oriented Architectures
SOA4All	Service Oriented Architecture for All
WS	Web Services
SWS	Semantic Web Services
WSMO	Web Service Modeling Ontology
WSML	Web Service Modeling Language
WSMX	Web Service Execution Environment
SEE	Semantic Execution Environment
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
UDDI	Universal Description Discovery and Integration
RDF	Resource Description Framework
TSC	Triple Space Computing
TripCom	Triple Space Communication
SS	Semantic Spaces
WSDM	Web Services Distributed Management
ESB	Enterprise Service Bus
KOPE	Knowledge-Oriented Provenance Environment
KOMME	Knowledge-Oriented Monitoring and Management Environment
PSM	Problem-Solving Method
MWWS	Management With Web Services
MOWS	Management Of Web Services

Executive summary

Semantic Spaces have been envisioned as interoperation and coordination for SOA4All services. It defines unified data model for storage and communication of data, service descriptions and ontologies. This deliverable presents the models, concepts required for such universal semantic space infrastructure. The deliverable presents an overview and carries out analysis of state-of-the-art in Semantic Spaces from several different aspects i.e. persistent data storage, communication, information publishing, global accessibility, subscription-notification, information retrieval and reliability. It further draws functional and non-functional requirements for Semantic Spaces and matches them with the challenges that are foreseen for Service Web.

The deliverable further defines the concepts and basic unified data model for Semantic Spaces based on the inherited approaches from state-of-the-art. It then defines infrastructure for Semantic Spaces and its role in SOA4All Distributed Service Bus. It outlines the usage of Semantic Spaces from the perspective of storage and communication of SOA4All services and Service Bus.

1. Introduction

Semantic Spaces have been envisioned as extension of space-based computing systems with semantics. The reason is two fold: first is to allow the communication of semantic technologies over spaces so that the semantic applications do not have to leave the semantic layer in communication, second is to use the semantics for precise search and query of the data published in the space. Triple Space Computing [Fensel, 2004] is one example of Semantic Spaces, which extends tuple-space computing using RDF as the formalism for describing the content of tuples in the space. It makes the fundamental data element as RDF triple in the Semantic Space. In order to store any information, all the data should be represented in RDF. Similarly, communication between different parties will be carried out by publishing and reading RDF triples on the Semantic Space.

Semantic Spaces will be used in SOA4All for two major purposes, i.e. for storage and communication. In case of storage, the data is based on semantic descriptions of services (i.e. WSMO, SAWSDL and WSMO-Lite), contextual information (i.e. to support adapting the provisioning and consumption of services based on contextual information), monitoring data of service execution (i.e. it includes performance and scalability properties) as well as service compositional information (i.e. it includes the business processes based on semantic descriptions of services for service composition). Semantic Spaces in SOA4All will further be used for communication in three different aspects, i.e. for communication of Business or End-point services, communication of platform services in SOA4All Enterprise Service Bus, as well as usage of Web2.0 based collaborative approaches in Semantic Spaces for the communication of SOA4All services.

1.1 Purpose and Scope

The purpose of this document is to explore the usage of Semantic Spaces in SOA4All. Semantic Spaces are envisioned to be a large repository for semantic data that is physically distributed but act as virtually single and globally shared space. This shared space is considered as Web for machines where semantic data is shared among the machines. This document explores the usage of Semantic Spaces to fulfil the needs of SOA4All from the perspective of storage and communication. The result will be a set of options described for the usage of Semantic Spaces in SOA4All for the purpose of storage and communication. From storage perspective, it includes storage of semantic descriptions of services, information regarding service monitoring during execution of a service, user-centric contextual information related to services and information regarding service composition. For each of the storage aspect, the document describes the type and structure of data to be stored, and the possibility to represent it in RDF (if required) to be able to store it on Semantic Space. From communication perspective, it includes semantic-based event-driven and publish-subscribe mechanism for SOA4All Distributed Service Bus, using Semantic Spaces for communication and coordination of SOA4All platform services as well as the business services, building semantic pipes for distributed orchestration on Semantic Spaces, and finally exploring the usage of Web 2.0 based collaborative approaches in Semantic Spaces for SOA4All.

1.2 Structure of the document

This document is structured as follows:

Section **Error! Reference source not found.**2 provides review of state-of-the-art, its analysis followed by requirements analysis on Semantic Spaces (SS), in particular, section **Error! Reference source not found.**2.1 briefly surveys on tuple space computing, section **Error! Reference source not found.**2.2 provides a similar briefing on semantic enabled tuple space

computing technology, while section **Error! Reference source not found.**2.3 provides a critical analysis of previous survey in the context of applicability of SS to SOA4All challenges, and section **Error! Reference source not found.**2.4 collects functional and non-functional requirements driving SS technologies to satisfy SOA4All goals. Section 3 describes how Semantic Spaces can be used for storage and communication purposes in SOA4All, while section 4 defines and describes the Semantic Space as well as its architecture. Finally, Section 5 collects the main conclusions of this document followed by references.

1.3 Methodology

This deliverable aims at defining and describing the Semantic Spaces infrastructure (based on the outcomes of EU FP6 TripCom[1] project) for storage and communication purposes of SOA4All. The methodology in this deliverable is to carry out analysis of state-of-the-art based on storage and communication aspects and derive functional and non-functional requirements for Semantic Spaces in SOA4All. Moreover, in this deliverable, important concepts related to Semantic Spaces have been described and a conceptual architecture for Semantic Spaces and its usage in SOA4All has been prescribed. Furthermore, in this deliverable, storage and communication requirements from different work packages of the project have been collected. From storage perspective, it describes the data that may be required to be stored on space (i.e. semantic descriptions of services, service monitoring information, contextual information and service composition). From communication perspective, it analyses how the communication of platform services of SOA4All Enterprise Service Bus, communication of end-point/standard/business Services, as well as usage of Web 2.0 collaboration techniques for communication in Semantic Spaces. The deliverable outlines the analysis of different possibilities of exploiting Semantic Spaces for storage and communication in SOA4All and its possible benefits over currently available storage and communication solutions.

1.4 Target Audience

The target audience for this document includes researchers as well as practitioners that work in the areas of Service-oriented Computing, Semantic Web, Web Services, as well as applying Semantic Web for semantic description of Web services. Systems analysts and systems architects needing a thorough knowledge of what Semantic Spaces are, and how to use Semantic Spaces for storage and communication purposes in SOA4All, may also benefit from this document. Although no specialized pre-knowledge is required to follow this document, basic knowledge in Service Oriented Architectures, Web Services, Semantic Web, Ontologies, and RDF/S may allow better following the document and for gaining more benefits from it. The work should be of interest to anyone involved with Semantic Web Services and more generally also in Service Oriented Architecture. This document contains description and analysis of state-of-the-art on Semantic Spaces, Requirements analysis for usage of Semantic Spaces in SOA4All, Architecture of Semantic Spaces for SOA4All, as well as detailing the usage of Semantic Spaces in SOA4All in terms of storage and communication.

[1] www.tripcom.org

2. State of the art and Requirements Analysis

This chapter provides a brief review and analysis of related work that has been done in the area of space-based communication as well as space-based communication for Semantic Web. The analysis has been divided into two major sections, i.e. systems that provide space-based communication regardless of semantics, and the systems that support space-based communication with semantic extensions. It includes Tuple-based computing systems, i.e. Linda which is a model to support coordination of among several parallel running processes. Secondly, it is Java spaces, which is a commercial and widely used product providing distributed object exchange and coordination mechanism for Java-objects based on the principles of space-based computing. It further describes TSpaces, which is event-based network communication storage for data exchange for Java applications. Furthermore, it describes the semantically enhanced space-based communication frameworks, i.e. Semantic Web Spaces which is a Linda-based coordination platform for Semantic Web applications. SRRN is another semantic-based distributed repository network that enables a fully distributed network for storage and retrieval of semantic information. It then describes Triple Space Computing (TSC), which is based on the RDF extensions of a space-based communication system (CORSO) for the communication and coordination of Semantic Web applications, as well as Semantic Web Services. TripCom is another semantic enabled communication and coordination framework for Semantic Web based applications as well as Web Services, and provides high performance storage system, distribution mechanisms, distributed querying support as well as grounding of Web Services communication using the TripCom. This chapter further describes usage of P2P solutions for building non-centralized semantic information lookup systems as well as publish-subscribe mechanisms.

2.1 Space based Computing Systems

Space based computing is an innovative and powerful concept for the coordination of autonomous processes. It is based on the notion of a common, abstract space connecting distributed processing entities over a network. Instead of explicitly exchanging messages between individual processes or performing remote procedure calls, processes communicate and coordinate themselves by simply reading and writing distributed data structures in a shared space. This section provides an overview about pioneer efforts made towards Space based Computing systems. It includes Linda, Java Spaces and TSpaces.

2.1.1 Linda

Linda [Gelernter, 1985] is a model of coordination and communication among several parallel processes operating upon objects stored in and retrieved from shared, virtual, associative memory. This model is implemented as a "coordination language" in which several primitives operating on ordered sequence of typed data objects, "tuples," are added to a sequential language, such as C, and a logically global associative memory, called a tuplespace, in which processes store and retrieve tuples. Basic Linda has a simple set of primitives, i.e. out (add a tuple to the tuplespace), in (destructively remove a tuple from the tuplespace), rd (non-destructively remove a tuple from the tuple space), eval (start a new process). Linda has been used for writing parallel applications, designing distributed computing platforms e.g. and for communication in agent-based systems.

Compared to other parallel-processing models, Linda is more orthogonal in that it treats process coordination as a separate activity from computation, and it is more general in that it can subsume various levels of concurrency—uniprocessor, multi-threaded multiprocessor, or networked—under a single model. Its orthogonality allows processes computing in different languages and platforms to interoperate using the same primitives. Its generality allows a multi-threaded Linda system to be distributed across multiple computers, or vice-versa, without change.

Whereas message-passing models require tightly-coupled processes sending messages to each other in some sequence or protocol, Linda processes are decoupled from other processes, communicating only through the tuplespace; a process need have no notion of other processes except for the kinds of tuples consumed or produced (data coupling).

2.1.2 JavaSpaces

JavaSpaces is a tuple space implementation for java objects, providing a distributed object exchange and coordination mechanism.

JavaSpaces is part of the Java Jini[2] technology network architecture for the construction of distributed systems in the form of modular co-operating services.

It is a relatively simple solution for lightweight distributed applications, providing a simple yet powerful API for the manipulation of objects. A JavaSpaces service holds entries, each of which is a typed group of objects. Interactions with an Entry is achieved with three simple operations

1. Write - places a new Entry in the space.
2. Read - returns a copy of an Entry matching a particular template.
3. Take - removes an Entry matching a particular template from the space and returns it to the caller.

JavaSpaces handles the concurrent access, storing and retrieving entries atomically. As with any distributed tuple space there is a lot of extra communication traffic in a JavaSpaces solution, so the only reason to use JavaSpaces is a requirement for computing power or special resources that are not feasible to supply on the server directly.

2.1.3 TSpaces

TSpaces[3] is an event-based network communication shared storage (tuple space) for peer data exchange and persistence. TSpaces allows heterogeneous Java based peers (applications, component, services, devices, etc) to communicate each other by submitting and retrieving data to/from share memory buffers in a very easy and straightforward way.

TSpaces provides some services to support its aforementioned features: for group communication, file transfer, event notification, database persistent and message delivery, among others.

TSpaces offers some advantages upon other traditional even-driven or message driven communication approaches: a) decoupling of data and logic lifetime (applications) so data can outlive its producer or can be created much time before being consumed, b) anonymous mediated communication, which avoid previous mutual awareness between data provider and consumer, c) asynchronous communication, which may be pretty adequate for some long lasting business processing, d) publication-subscription MEP (Message Exchange Pattern) based communication, which allows users to subscribe only to those data categories they are interested on, etc.

While TSpace could be quite suitable to address asynchronous communication among heterogeneous Java based peers participating in long lasting business processes, it may not be so suitable to cover other communication requirements as synchronous MEP, low latency communication process, heterogeneous platform communication (J2EE, .NET, etc), WS* stack

[2] http://www.sun.com/software/jini/

[3] http://www.alphaworks.ibm.com/tech/tspaces

compatibility, etc.

2.2 Semantic enabled Space based Computing

The aim for semantics enabled space based computing systems is to enhance the representation of space model for efficient, effective and precise storage and retrieval of data from space. Moreover, it is also aimed to help for the communication of semantic technologies (i.e. communicate by publishing and reading semantic data), so that they do not have to leave the semantic layer during the communication. This section presents a brief overview of the related efforts for enabling semantics in space based computing. It includes Semantic Tuples (sTuples), Semantic Web Spaces, Semantic Registry and Repository Network (SRRN), Triple Space Computing (TSC) and Triple Space Communication (TripCom).

2.2.1 sTuples

sTuples [Khushraj et. al., 2004], also called as Semantic Tuple Spaces, is based on usage of Semantic Web technologies to represent and retrieve tuples from a Tuple Space. It overcomes the limitations of Tuple Spaces by making use of a web ontology language and a reasoner RACER (Description Logic reasoning engine). Key focus of sTuples is to enhance the Tuple representation and searches by introducing a concept of semantic tuple and extending it to represent data and service descriptions. It further provides user-centric reasoning which is achieved using agents on the space that provide unnoticeable data and services to execute tasks on behalf of the user. It has five major components which are called as Semantic Tuple Manager, Semantic Tuple Matcher, Tuple Recommender Agent, Task Execution Agent and Publish-Subscribe Agent. Semantic Tuple Manager manages the addition, removal, and state changes of the tuples. Semantic Tuple Matcher matches the templates and interfaces with the RACER reasoner to draw additional inferences. sTuples further have some specialized agents that reside on the Semantic Tuple Space to off-load the user and to incorporate user semantics in delivering data and services to the user. Tuple Recommender Agent pushes unwanted data and services into periphery of user's attention and presents only services or data tuples that are of interest for user context. The different aspects of context include location, temporal information, user preferences, environmental factors, proximity of resources, resource availability, schedules etc. Task Execution Agent off-loads certain well-defined tasks that user performs by acting as proxy on behalf of the user. The Publish-Subscribe Agent dynamically delivers data to the subscribed users. A service/agent can publish data or events that are meant to be shared by multiple users by writing a data tuple on the semantic space. The subscription request to this agent is routed the same way as it is done for the other agents.

2.2.2 Semantic Web Spaces

The Semantic Web Spaces[4] [Tolksdorf et. al., 2006] is a Linda-based (cf. section 2.1.1) coordination platform provided by the Institut für Informatik, Fachbereich Mathematik und Informatik at the Freie Universität Berlin (STI Berlin). The platform aims at acting as a middleware fulfilling requirements of real-world usage scenarios of Semantic Web applications. These requirements include the capability to cope with the large scale, distributed, heterogeneous, unreliable and insecure environment of the World Wide Web, if they are to truly represent added value to Web users, as well as issues of persistent storage, efficient reasoning, data mediation, scalability, distribution of data, fault tolerance and security.

[4] http://www.ag-nbi.de/research/semanticwebspaces/index.html

Semantic Web Spaces allows the representation and management of Semantic Web information by extending the classical Linda model with new types of tuples e.g. RDF statements consisting of (subject, predicate, object) and new types of tuplespaces e.g. contexts/scopes expressing the view of a certain agent upon the entire information space.
As the tuplespace now no longer holds only data but also knowledge, the set of Linda operations is extended and refined:

- The classical operations are defined to operate upon the tuples as data
- To permit operations upon agent scope to be added - *addscope, rmscope*
- To add or remove tuples as knowledge to be added - *claim, retract*
- To query tuples as knowledge that is defined as semantic matching relations - *rdiftrue*

The following picture (source: http://userpage.fu-berlin.de/~paslaru/papers/wi2005.pdf) shows the structure of Semantic Web Spaces: it shows both a dataview, encompassing simple datatype tuples, XMLSpaces tuples (containing XML documents) and Semantic Web tuples such as RDF triples, and an information view, where RDF tuples are handled according to the semantics of the information that they contain and reasoning features are embedded (RDFS, OWL-Lite, OWL DL etc.).

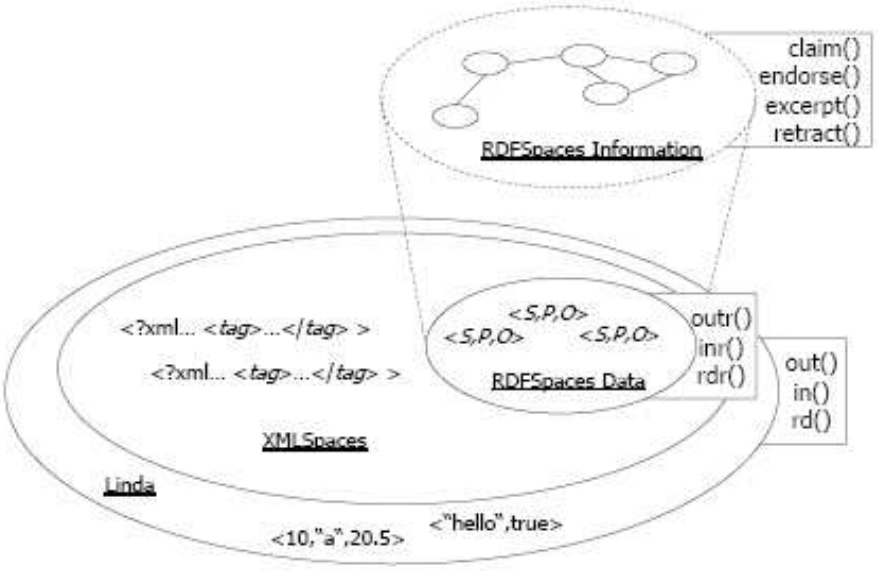


Figure 2.2.2: Semantic Web Spaces

2.2.3 Semantic Registry and Repository Network – SRRN

The SRRN (Semantic Registry and Repository Network) is the core result of the European FP6 SEEMseed project, which was focused on enabling a Single European Electronic Market. It has been further elaborated in the ongoing FP6 SEAMLESS context and is used as a base in the FP6 STASIS project.

The SRRN is a fully distributed network that allows the storage and retrieval of semantic information. The SRRN is a pure P2P network, i.e. no central elements at all. It is accessible via Web Services allowing to query, store and update data. All communication between the different components and between the peers is based on Web Services. The access to the overall SRRN system is also performed via Web Services allowing a tight integration into SOA based

environments.

The SRRN works without any central element. All data is stored and managed by the peers in the P2P network. New peers may join and leave the network spontaneously allowing new partners to add their own servers to the network if required.

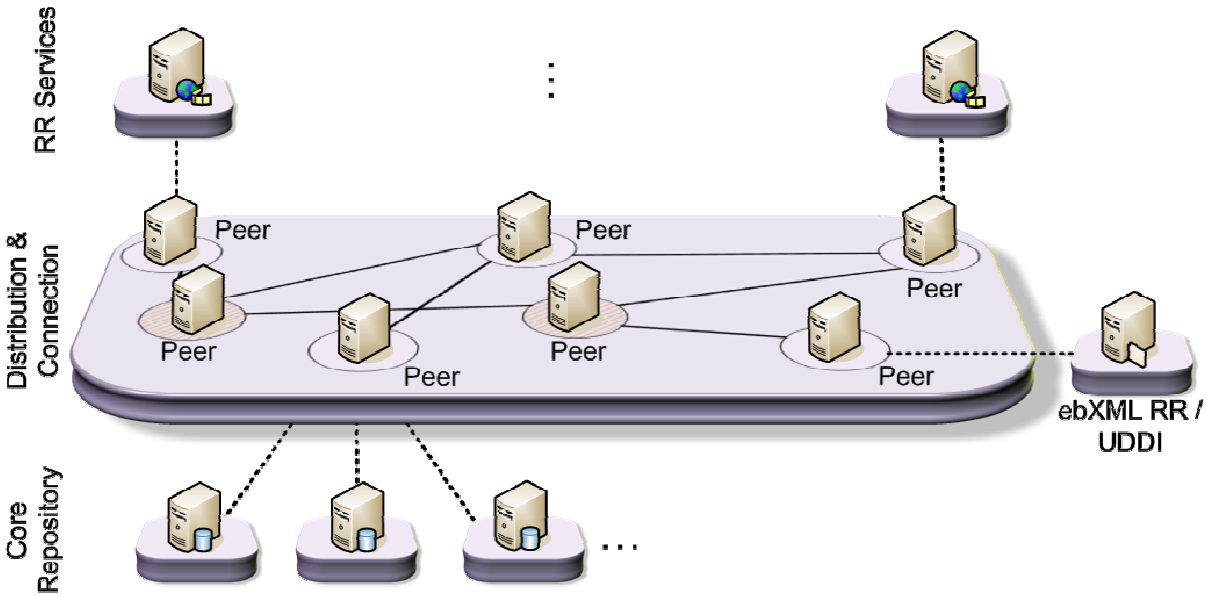


Figure 2.2.3a: SRRN conceptual layers

The SRRN consists of several conceptual layers:

- Repository Layer: This layer stores the data and offers a SparQL query interface. It may be considered the 'database layer' of the SRRN architecture.
- Distribution Layer: This layer cares about distributing data in a P2P-network and it cares about forwarding queries and consolidating results. It provides an asynchronous interface.
- Service Layer: This layer manages registry logic such as, e.g., managing notifications. It offers easy interfaces for application developers to access the SRRN
- Application layer: This is the application used by the user. It's not part of the SRRN itself but it rather represents the applications that are actually using the SRRN.

The following figure shows all layers and their interconnection:

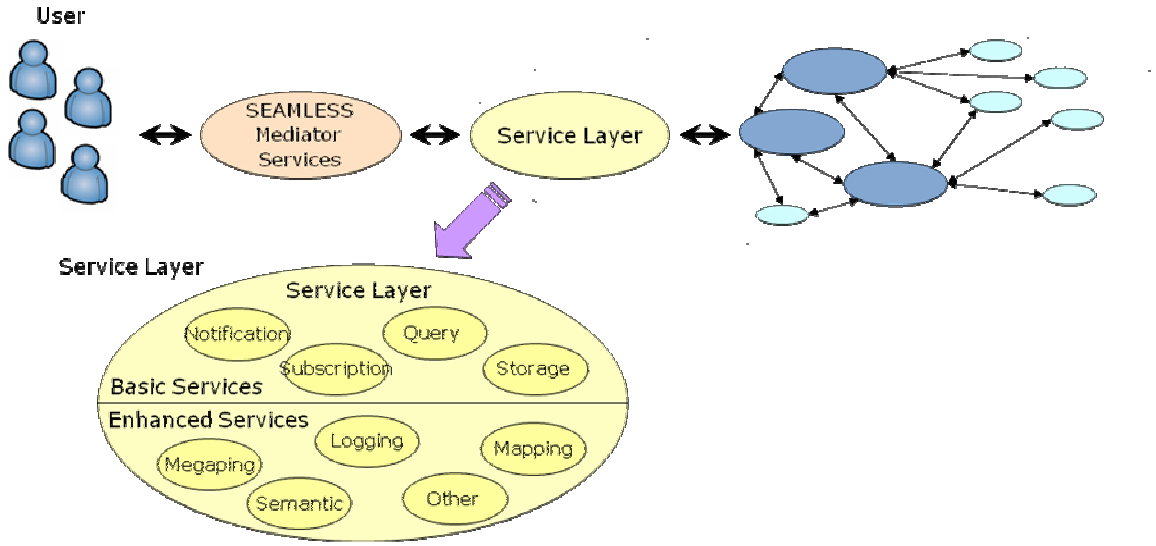


Figure 2.2.3b: SRRN coordination

The SRRN may be used for two different purposes:

- **Semantic Space**: Storing and Managing semantic entities

  The SRRN may be used to directly store and retrieve semantics that are expressed in the OWL or the RDF format. The SRRN provides a comprehensive set of Web Services allowing users to add new OWL/RDF data to the SRRN. This data is stored on one of the peers in the SRRN and it is instantly available in the network. Users may use the SparQL language, which is a W3C recommendation for querying semantic data. The SRRN provides functionality such as load balancing, network monitoring and auditing in order to provide a stable and scalable registry and repository for semantic data

- **Document Storage**: Storing and Managing documents

  In addition to the pure semantic storage, the SRRN may also be used as a repository for documents. In this case, documents may be stored anywhere in the network and may be accessed in an easy way by contacting one the SRRN peers using a Web Service interface. All documents can be annotated with semantic data in either OWL or RDF. Users may use the SparQL query language to access this semantic metadata and the documents that are attached to it.

Depending on the actual scenario, this allows to either store elements as pure semantic data or to store elements in their original data (e.g. BPEL) and to only annotate the existing documents with semantic information. The later alternative is meaningful whenever the original documents are needed at a later stage or whenever a complete conversion into a semantic format is either not worth the effort, impossible or meaningless (e.g. for PDF files). In this case, a mixture is helpful that consists of storing the original data together with some semantic 'chunks' of information.

2.2.4 TSC – Triple Space Computing

Triple Space Computing (TSC)[5] was funded by the Austrian Forschung, Innovation, Technologie -

[5] http://tsc.deri.at

Informationstechnologie (FIT-IT)[6] research programme in the programme line of "semantic systems and services". Triple Space Computing inherits the publication-based communication model from the space-based computing paradigm and extends it with semantics. Instead of sending messages back and forth among participants as in current message-based technologies, It enables the semantic applications communicate with each other by writing and reading RDF triples in the globally shared and virtually single (physically distributed) space. The TSC framework [Fensel et. al., 2007] defines data models, matching algorithms, interaction APIs and security models at the convergence of space-based computing/shared object spaces and the Semantic Web. The interaction API provides operations for writing, reading, removing in blocking and non-blocking manners. In order to allow Web-like communication, the traditional template-based read and take were enhanced with URI-based primitives that allow the extraction of information by identifier. The space infrastructure is implemented through an extension of the CORSO (Coordinated Shared Objects) middleware [CORSO] that provides replication, transaction and coordination of amongst others Java objects (also .NET and C++ classes). Hence, RDF graphs are mapped onto CORSO objects in order to share them amongst participating nodes. The prototype implementation of TSC makes use of the YARS storage framework [Harth et al., 2005] in order to ensure persistency and RDF querying.

The TSC architecture was built on the abstract model called REST (Representational State Transfer), and was motivated by a hybrid architecture called super-peer architecture, which combines traditional client/server and peer-to-peer architectures. In this architecture, there are three kinds of nodes: servers, heavy clients and light clients. In the simplest configuration, a particular Triple Space is realized by a single server, which is accessed by multiple light clients, for example via HTTP, in order to write and read named graphs and to receive notifications about graphs of interest. As the number of light clients increases, the server may become a bottleneck. To overcome this, additional servers can be deployed to provide additional access points to a Triple Space for light clients. As a result, a single Triple Space is be effectively spanned by multiple servers, which uses an inter-server protocol to consistently distribute and collect named graphs to and from other involved servers. Servers can also be deployed to act as caching proxies in order to improve clients-perceived access times. The third kind of nodes is heavy clients, which are not always connected to the system. Like servers, they are capable to store and replicate Triple Spaces and support users and applications to work off-line with their own replicas. While heavy clients can join existing Triple Spaces spanned by servers, they are not forced to do so. The core functionality of TSC servers and heavy clients is realized by a component called Triple Space kernel (TS kernel) [Riemer et. al., 2006]. Heavy clients run in the same address space as the TS kernel, and the TS kernel is accessed by its native interface. Light clients use proxies to access the TS kernel of a server node transparently over the network. As a variation a light client can access a TS kernel via a standardized protocol like HTTP, as already mentioned above. In this case, a server side component, e.g. a servlet, translates the protocol to the native TS kernel interface.

The currently used communication paradigm in Semantic Web Services (SWS) [Fensel & Bussler, 2002] is synchronous, i.e. users communicate with SWS execution environment and SWS communicate with real world Web Services by sending synchronous messages. The problem with synchronous communication is that it requires a quick response as it makes sender halt until the response is received, which is not possible in case of execution process in SWS as it involves heavy processing of semantic descriptions in terms of discovery, selection, composition, mediation, execution. This problem has been overcome by introducing Triple Space Computing as being semantic based asynchronous communication paradigm for communication and coordination of SWS. The Web Services Execution Environment (WSMX) [Harth & Decker, 2005] is our reference implementation for SWS in which the Triple Space Computing middleware is

[6] http://www.fit-it.at

being integrated. Using Triple Space Computing in WSMX enables to support greater modularization, flexibility and decoupling in communication and coordination and to be highly distributed and easily accessible. Multiple TS kernels coordinate with each other to form a virtual space that acts as underline middleware, which is used for communication by reading and writing data. The integration of WSMX and Triple Space Computing was proposed in four major aspects [Shafiq et. al., 2007]: (1) enabling component management in WSMX using Triple Space Computing, (2) allowing external communication grounding in WSMX, (3) providing resource management, and (4) enabling communication and coordination between different inter-connected WSMX systems. In summary, Triple Space Computing acts as a middleware for WSMX, Web Services, different other Semantic Web applications, and users to communicate with each other.

2.2.5 TripCom – Triple Space Communication

Triple Space Communication (TripCom)[7] is an EU FP6 funded STREP project, with an aim to establish the mechanism to publish communication data according to the Web paradigm of "persistent publication and read". In this way, TripCom envisions to bring machine-to-machine communication of Semantic Web applications as well as Web Services according to the principles of Web. TripCom is based on the evolution and integration of several well-known technologies, such as Tuple-Space Computing, Shared Object Space, Persistent Message-based architecture and RDF. Although Triple Space Computing (TSC) and TripCom shares same vision, but TripCom explores it broadly. As a first step, TripCom provides high performance storage implementation for storage of Semantic data (i.e. RDF). It allows integration of various heterogeneous data sources through various adapters, as well as federation of the storage services over the network. A highly optimized and efficient structure for searching and accessing RDF triples has been provided. It allows the integration of relational databases to the Triple Space data model through a RDBMS adapter. It provides support for Triple reasoning based on a high performance storage inference engine that integrates efficient reasoning and performs forward chaining of entailment rules on top of RDF extended named graphs. It allows YARS (Yet Another RDF Store) [Harth & Decker, 2005] to be used for storage purposes. Furthermore, the federation mechanism allows data storage and access in multiple RDF data repositories local to each Triple Space kernel [Sapkota et. al., 2008]. TripCom further refines the model of Semantic Tuple in terms of knowledge representation languages whose semantics is relevant for processing the tuple content in relation with matches. It defines a Triple Space ontology to capture the metadata about the triples and their content [Krummenacher et. al., 2007]. It further provides support for distributed query processing and inconsistent reasoning for processing queries to access data published in Triple Space [Nixon et. al., 2008]. TripCom further explores the Semantic Web Services communication over Triple Space. It includes defining TripCom groundings for Web Service registry. From description language perspective, it describes representing WSDL in RDF. From communication protocol perspective, it describes the TripCom bindings in WSDL, representing SOAP in RDF and describes Web Service invocation in Triple Space. It further explores the grounding issues in Semantic Web Services and details the grounding requirements, specification and implementation guidelines in Semantic Web Service Execution Environment (WSMX) [Shafiq et. al., 2008]. TripCom also provides a comprehensive security models to govern access to Triple Space by defining access control rules and trust [Ghioni et. al., 2007]. The Triple Space architecture defines kernel that provides a physical implementation consisting of all necessary components implementing the Triple Space API. The key components of Triple Space kernel includes API, Storage adapter, Security manager, Mediation manager, Query Processor, Metadata manager, Transaction manager, Distribution manager, as well as components for Web Services discovery and invocation over Triple Space [Murth et. al., 2007].

[7] www.tripcom.org

2.2.6 Survey of distribution techniques

Peer-to-Peer (P2P) overlays networks are large-scale distributed systems build on top of the physical network. Decentralization is one of the major concepts of P2P systems. They offer services such as distributed storage, information sharing as well as control information. Peers form a decentralized, self-organizing fault-tolerant overlay increasing the robustness of the system compared to centralized solutions. Thus, adopting a P2P decentralized architecture for storage and retrieving semantic data based on RDF triples or RDF graphs for example seems to be a natural solution to build scalable systems. We will focus in this section on the basic architectures of P2P networks as well as some proposed P2P solutions for managing and routing of semantic data.

2.2.6.1 P2P Architectures

P2P overlays systems can be classified into two basic types: unstructured and structured overlays. In unstructured P2P networks such as Gnutella [Chawathe et. al., 2003] there is no specific control neither over the network topology nor the data location. When a peer joins the networks, it chooses an arbitrary set of nodes as neighbours. The information retrieval in such network represents an important issue. The main differences among search techniques are the query forwarding process as well as the propagation rules over peers. As shown in [Tsoumakos et. Al, 2003], searching mechanisms in unstructured P2P networks can be categorized as either blind or informed.

In the blind technique, a peer regarding the data or service location holds no information. To find the needed data/service, it is necessary to send a request to a sufficient number of peers rapidly. For instance, Flooding and random walks are two basic blind search mechanisms. In Flooding, a message is forwarded by a peer to all its neighbours. In the (k-) random walks approach, at each step, the (k-) neighbour to which a message will be forwarded is randomly selected.

In informed scheme, each node maintains information about the service location. This makes the search more efficient in terms of number of messages. APS (Adaptive Probabilistic Search) [Tsoumakos et. Al, 2003b] is an informed search mechanism based on k- random walks. The forwarding decision is probabilistic and uses feedback from previous queries to redirect new one.

Unstructured P2P networks seem to support complex queries such as range queries but are not suitable for looking for rare items. Structured P2P overlays such as Chord [Stoica et. al., 2001], CAN [Ratnasamy et. al., 2001] and Pastry [Rowstron et. al., 2001] impose to structure the underlying network of nodes along a predefined topology. These systems are mainly based on Distributed Hash Table (DHTs) where peers are identified by node Ids. It assigns keys to data items and builds a structured graph that maps each key to the node storing the corresponding data. This kind of P2P networks is appropriate for simple queries, i.e. when the hash value of the need information is known. However, performing multi- attributes and complex queries search is difficult compared to unstructured overlays. A comparative study in [Yang et. al., 2006] for a complex query such as search of a full text shows that the average search traffic in the structured overlays is comparable to the traffic generated in the unstructured overlays. As seen, P2P architectures offer useful property for storing data in a distributed and fault-tolerant way. However, depending on the data, some architecture can exhibit better properties than some other. That is why some work has been specifically dedicated to the storing of RDF data and the implementation of distributed RDF repository.

2.2.6.2 P2P solutions for RDF data management and routing

Researches on P2P networks have focused not only on the network architecture but also on the

semantic of the stored data, moving from simple keywords based storage to more sophisticated RDF (Resource Description Framework)-based data storage format. The RDF based data items are described as triples <subject, predicate, object> where the subject identifies the resource to describe, the predicate presents the specific property in the statement and the object is the property value of the predicate. The RDF data model is flexible, i.e., it is simple to express any data in such format. Hence, a need has arisen to efficiently support storing and querying of RDF data. Some existing RDF storage systems such as RDFStore [Rdfstore] and RDFDB [Rdfdb] stores data in centralized repositories. These systems, adopting centralized approaches, have a set of limitations such as fault-tolerance problem since they present a single point of failure.

Many P2P solutions have been proposed to ensure a distributed storage for RDF data. Some of them are built on top of super-peer-based infrastructure. In this model, a set of nodes can act as super-peers and other nodes can be connected to one super-peer. Super-peers, in turn, are interconnected with other super-peers. A super-peer is responsible for message routing in the super-peer backbone as well as between peers belonging to it. Edutella [Nejdl, Siberski et. al., 2003] [Nejdl, Wolpers et. al., 2004] is an example of a RDF-based P2P infrastructure using super-peers. They organized onto a HyperCube overlay and maintain meta-data for a set of peers. Edutella provides a set of services such as query service, replication service, mapping and mediation services. This can facilitate the searching process and make it quite efficient. Others RDF based storage systems are built on structured overlays. In [Gu et. al., 2007], the authors propose a Dynamic Semantic Space (DSS) based on semantics of RDF data maintained by each peer. Peers can be grouped into clusters. Clusters having the same semantics are organized into semantic cluster. While there is no constraint on the overlay topology within the semantic cluster, semantic clusters form a ring structure such as in Chord [Stoica et. al., 2001]. To join the network, a peer has to do a mapping of its RDF data to one or more semantic clusters and join the most suitable one. RDFPeers [Cai et. al., 2004] is a structured P2P RDF store system. The peers are self-organize into a multi-attribute addressable network (MAAN), which is an extension of Chord. The key ideas is to index the RDF triple three times by applying a hash function on the subject, the object and the predicate of the RDF data and so store the triple in three different peers which have closest matching ID as the corresponding indexes.

2.3 Analysis of state-of-the-art

In this section, we present some factors that have been identified for the comparison and analysis of the state-of-the-art. In the state-of-the-art review, we have tried to cover most of widely recognized related work in the area of space-based computing and semantic enabled space based computing systems. There are different factors that have been identified to carry out the comparison. This comparison and analysis will help in partly formulating the requirements specifications for Semantic Spaces.

Factors for analysis/comparison of state-of-the-art:

1. Semantic Support: It concerns with usage of semantics in representation of the data model and then using it to retrieve the information.

2. Query processing: It concerns with support of complex and enhanced query processing in order to retrieve data published in the space.

3. Knowledge representation: It concerns with the formalism used to represent and organize

data published in the space.

4. Data access support: It concerns with the user and management level API support that allows users to access the space as well as publish data in it.

5. Distribution: It concerns with the physical distribution of the spaces over network, as well as the level of distribution, i.e. P2P, client-server, ring, mesh, star, bus, structured or unstructured.

6. Security: It concerns with the support for authentication and authorization provided to the data published on the space.

Attributes	Features					
	Semantic Support	Query Processing	Knowledge Representation	Data Access Support	Distribut-ion	Security
Linda	-	-	-	+	+	-
Java-Spaces	-	-	-	+	+	-
TSpaces	-	-	-	+	+	-
sTuples	+	+	+	+	+	-
Semantic Web Spaces	+	+	+	+	-	-
SRRN	+	+	+	+	+	-
TSC	+	+	+	+	+	+/-
TripCom	+	+	+	+	+	+

Table 2.3: Analysis and comparison of state-of-the-art

Legend:

+ stands for “supported”

+/- stands for “supported to limited extent”

- stands for not “supported”

The table above presents an overview of comparison of the state-of-the-art in Space-based Computing systems and efforts made for enabling semantics in Space-based Computing systems.

Linda is a pioneer effort towards realizing the concept of space based coordination. It does not provide any explicit support for using semantics to represent data in the space, as well as any query processing capabilities in its core primitives. However, it provides support for data access based on simple and basic access methods. It allows distribution across multiple processes and computers connected over the network. Security issues have not been addressed by Linda.

JavaSpaces allows storage of data as java objects, therefore no explicit semantic support to represent data in the space. No explicit query processing support except a basic template matching mechanism for retrieval of data. Knowledge representation can be supported if the ontologies and other semantic data is stored as java objects. It allows distribution by providing a distributed object exchange mechanism. Security issues are dealt based on standard java security package. TSpaces do not offer explicit semantic and knowledge representation support. It allows distribution by allowing heterogeneous java peers to communicate with each other.

sTuples allow semantic support using OWL-DL language for representing data in the space. The query processing based on RACER (a Description Logic reasoner) is used. It allows the addition, removal, and state changes of the tuples as well as access based on user context as data access mechanism. No security requirements have been found dealt in sTuples so far.

Semantic Web Spaces allows explicit representation of data as RDF. For information view, RDF triples are handled according to the information that they contain (i.e. RDFS, OWL-Lite, OWL DL etc.). Data access is based on user scope. Distribution capabilities are inherited from Linda. No security issues have been dealt explicitly.

SRRN allows semantic support based on RDF. Query processing capabilities are based on SPQRQL querying. It allows distribution based on P2P network. It allows knowledge representation to support OWL as data to be stored in semantic space. Security issues have not been explicitly addressed.

TSC allows semantic support by allowing data to be represented as RDF triples. It allows query processing based on template based matching. Data access API is provided to allow storage and retrieval of data. Distribution is allowed based on CORSO (Coordinated Shared Objects) system. Security is addressed only up to theoretical level.

TripCom allows semantic support by allowing data to be represented as RDF triples. It allows query processing based on SPARQL. Distribution is based on P2P overlay networks and it supports distributed query processing and optimization. It further provides support for Web Services to use TripCom for communication. Security is also addressed to a comprehensive level to have appropriate authentication and authorization methods for the data stored and access in the Triple Space.

2.4 Requirement Analysis

This section introduces a couple of requirements for the Semantic Spaces to be built for SOA4All. The requirements have been mainly derived from the vision of Triple Space Computing [Fensel, 2004] that how the persistent publication is required for the communication, enabling machine-to-machine communication compliant with the principles of World Wide Web, as well as the vision of ServiceWeb [Domingue et. al., 2008] for enabling SOA revolution to Web-scale. Moreover, the requirements are further motivated from the idea of semantic extensions to the space-based communication systems, in order to allow better and precise access to the data published on the space. Last but not least, the requirements have also been derived from the analysis of the state-of-the-art and related work, in the above section.

These requirements include functional and non-functional requirements. The functional requirements of the Semantic Spaces are based on the direct features that have been proposed, and consist of persistent storage of data, communication based on persistent storage of data,

publishing of information globally, subscription-notification for users publishing and accessing data, information retrieval from space, as well as the degree of decoupling achieved in the communication. Non-functional requirements of the Semantic Spaces are based on the requirements that specify the criteria that can be used to judge operation of the Semantic Spaces. It includes reliability of data stored in space, versioning of data stored in pace, trust and security of data.

Each of the requirements will be discussed below:

R1. Persistent Data Storage: Information stored in the space should be stored permanently until and unless is explicitly removed by the owner or user (if given rights to do so). The data stored should be safe enough, i.e. not to be changed by any other system process.

R2. Communication: The space should allow complex machine-to-machine communication of applications, i.e. one-to-one communication, one-to-many, many-to-one and many-to-many communication.

R3. Information Publishing: The space should allow users to publish data, which should be visible and accessible for all other users available on the space.

R4. Globally Accessible: The space should be accessed anytime and anywhere over the internet, i.e. global accessible.

R5. Subscription and Notification: The space should allow users to subscribe to particular data or event, i.e. in case of any change/update the users should be notified accordingly.

R6. Information Retrieval (search/querying): The space should allow users with enough querying support to be able to access, search and explore the data published on space by users.

R7. Decoupling in Communication: The space should help in increasing the degree of decoupling among communicating parties

R8. Semantic annotation to data: Data published in the space should be semantically annotated so that it can be retrieved precisely and efficiently.

R9. Reliability: It refers to the reliability of the services and features provided by the Semantic Space infrastructure. In case of any failure that might occur, the system should take alternative path (if possible).

D1.1.1 (Design Principles of a Service Web) [Di Nitto et. al., 2008] of SOA4All describes some challenges that are concerned with SOA4All Service Web architecture. The challenges include Heterogeneity (C1), Worldwide access mechanism (C2), Semantic provisioning of services (C3), Decentralized dynamicity and adaptability (C4), Matching requests and services (C5), enabling

n:m asynchronous interactions (C6), enabling service prosumers (C7), supporting both human and machine based computation (C8). The requirements derived for Semantic Spaces are expected to contribute to face that challenges identified for SOA4All. The table below provides an overview that the requirements collected for Semantic Spaces in SOA4All contribute to challenges identified for SOA4All.

	C1	C2	C3	C4	C5	C6	C7	C8
R1								
R2						●	●	
R3		●						
R4	●	●						
R5				●				
R6					●			
R7		●		●				
R8			●					●
R9								

Table 2.4: Relationship between Requirements and the challenges for SOA4All

Heterogeneity refers to different systems based on different operating systems and implementation languages. The requirement of Global Accessibility will contribute to face the challenge of heterogeneity so that different heterogeneous peers could be able to communicate with each other using Semantic Spaces. Wold-wide access mechanism will be fulfilled by global accessibility, i.e. data on the semantic space and semantic space itself is accessible over the Web, information is published on spaces is also ensured to become globally accessible. Moreover, enabling greater degree of decoupling in communication will help in bringing World-wide accessibility. Semantic provisioning of services will be supported by usage of semantics for representation of data in the semantic space. Decentralized dynamicity and adaptability will be supported by distributed subscription and notification mechanism as well as decoupling in communication and help distribution become as flexible as possible. Semantic Spaces will also help in supporting complex communication over the Web where communication is occurred based on publish and read of data from globally available, shared space, and thus support n:m asynchronous communication. Since the communication through Semantic Space is based on publishing and reading semantic data, therefore, it will support the machine based communication and computation. Semantic Spaces will further investigate the usage of Web 2.0 based collaborative techniques for communication of services, which will contribute to enable service prosumers communicate actively in Service Web.

3. Using Semantic Spaces in SOA4All

This chapter collects all possible proposals for using Semantic Spaces in SOA4All that will further be evaluated and taken into consideration in the next stages of the project. There are two major purposes that have been identified for using Semantic Spaces in SOA4All, i.e. for storage and communication. In case of storage, the data is based on semantic descriptions of services (i.e. WSMO, SAWSDL and WSMO-Lite), contextual information (i.e. to support adapting the provisioning and consumption of services based on contextual information), monitoring data of service execution (i.e. it includes performance and scalability properties) as well as service compositional information (i.e. it includes the business processes based on semantic descriptions of services for service composition). Semantic Spaces in SOA4All will further be used for communication in three different aspects, i.e. for communication of Business or End-point services, communication of platform services in SOA4All Enterprise Service Bus, as well as usage of Web2.0 based collaborative approaches in Semantic Spaces for the communication of SOA4All services. The figure below shows the overall usage scenario for SOA4All services. The black points with sub-section numbers mark each of the storage and communication aspect of the usage of Semantic Spaces for SOA4All. Each of the aspects has been described in the sub-subsections below:

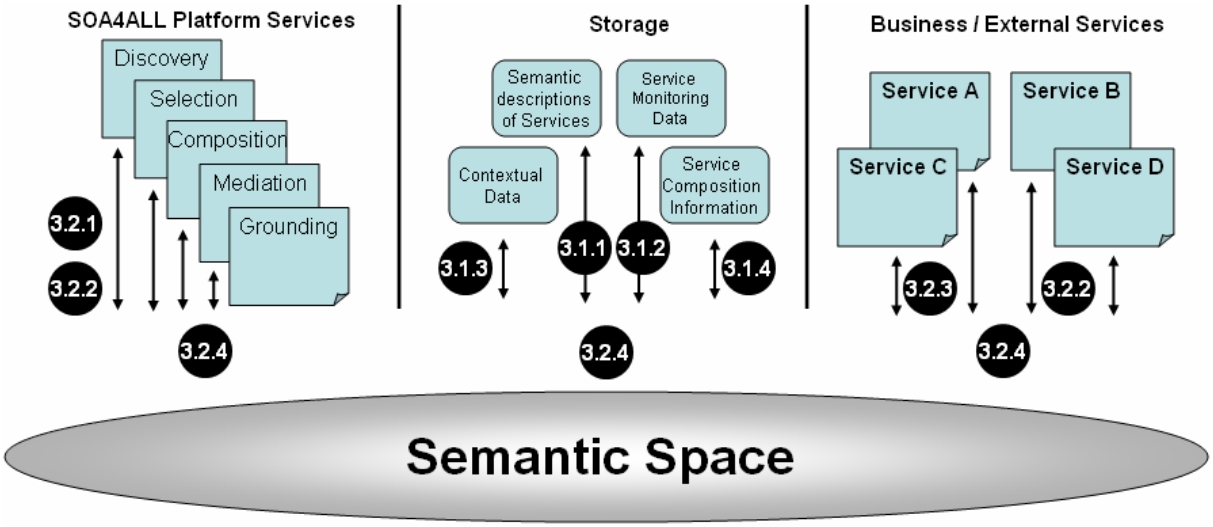


Figure 3: Overall Semantic Spaces usage scenario for SOA4All services

3.1 Semantic Spaces as Repository

This section is concerned with the storage aspect of the Semantic Spaces. Semantic Spaces are envisioned to be based on a large set of repositories physically distributed but connected to each other forming a virtually single shared storage space. We have analyzed the storage requirements in SOA4All and propose the usage of Semantic Spaces as a suitable storage mechanism. The data includes semantic description of services, service monitoring information, contextual information related to services and users, and information about service composition. The subsections below discuss each type of data to be stored. It further discusses the possibility to represent the data in RDF to be able to store on Semantic Space, and argues on whether the RDF transformation is required or not, as it makes sense to transform only that kind of data to RDF for which reasoning may be required.

3.1.1 Enabling storage and access of WSMO, SAWSDL and other Semantic descriptions (ref. WP1 and WP3)

Semantic descriptions of Web Services are one of the important things that will be developed in the SOA4All to increase the interoperability of service providers and service requestors, as well as used for automated reasoning purposes to allow decoupling between service providers and service requestors. The semantic descriptions are motivated from Web Service Modeling Ontology (WSMO) [Feier et. al., 2005], which is an ontology for describing various aspects of Semantic Web Services, i.e. Goals, Mediators, Ontologies and Web Service descriptions. WSMO descriptions are written using a Web Service Modeling Language (WSML) [de Bruijn & Heymans, 2007] which provides a formal syntax and semantics for WSMO.

Semantic Annotations for WSDL and XML Schema (SAWSDL) [Kopecky et. al., 2007] defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0[8], WSDL 1.1[9] and XML Schema extensibility frameworks.

WSMO-Lite [Vitvar et. al., 2007] service ontology is the evolutionary step after SAWSDL that fills the SAWSDL annotations with concrete semantic service descriptions. With the ultimate goal to support realistic real-world challenges in intelligent service integration, WSMO-Lite addresses the issues as follows:

- Identify the types and a simple vocabulary for semantic descriptions of services (a service ontology) as well as languages used to define these descriptions
- Define an annotation mechanism for WSDL using this service ontology;
- Provide the bridge between WSDL, SAWSDL and (existing) domain-specific ontologies such as classification schemas, domain ontology models, etc.

There are two aspects of groundings of semantic descriptions to RDF (Resource Description Framework) that can be seen, in order to allow the grounding of WSML descriptions to currently available Semantic Web standards (i.e. RDF), and to be able to store the descriptions in Semantic Spaces (with RDF as a fundamental storage element): one is to allow RDF representation of WSML [de Bruijn, 2008], and second is to define WSMO grounding in SAWSDL [Kopecký et. al., 2007].

3.1.1.1 RDF representation of WSML

The RDF representation also helps WSML to be an integral part of the Semantic Web as it is assumed that the Semantic Web will most likely represent data in this format. WSML data can be translated to RDF in a straightforward way, making WSML an ontology and the WSML data an instance of that ontology. The following listing shows a simple WSML document and how it could be translated into RDF based on the RDF schema vocabulary provided by WSML working group [de Bruijn, 2008]:

```
// example WSML ontology and Web service
namespace {ns _"http://example.com/"}
ontology ns:Transportation
```

[8] http://www.w3.org/TR/wsdl20

[9] http://www.w3.org/TR/wsdl

```
   concept ns:Vehicle

webService ns:TicketService
   capability ns:TicketServiceCapability


// RDF triples (N3 representation) for the above information
@prefix ns: <http://example.com/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix wsml: <http://www.wsmo.org/wsml/wsml-syntax#>
ns:Transportation
   rdf:type wsml:ontology;
   wsml:hasConcept ns:Vehicle .

ns:Vehicle
   rdf:type wsml:concept .

ns:TicketService
   rdf:type wsml:webService;
   wsml:useCapability ns:TicketServiceCapability
```

Table 3.1.1.1a: Example of WSML Ontology and Web Services and its RDF representation

Such direct translation is a useful solution for most parts of WSML data, but it is suboptimal for WSML ontologies and their instances, because the direct translation of WSML to RDF does not indicate that WSML concepts are similar to RDF classes. In fact, the semantics of WSML concepts and RDF classes differ slightly (for example, rdfs:Resource is a class that contains all resources, i.e. everything, but there is no such concept in WSML; and rdfs:Class contains itself, which cannot be modeled in WSML).

The RDF representation comes with an RDF Schema ontology which describes its classes and properties. Both the RDF representation and the RDF syntax come with a formal mapping from the surface syntax of WSML to the RDF representation and syntax, respectively. It is an alternative syntax for WSML, just like the surface syntax and XML syntax presented in the WSML specification [de Bruijn et. al., 2005]. The advantage of using this RDF syntax over the surface syntax and XML syntax is that it can be stored in Semantic Space.

WSML is the formal language that is used to describe different aspects of Semantic Web Services based on WSMO conceptual model. In the RDF schema proposed to represent WSML in RDF, first of all top level entities are to be represented in RDF. Following RDF schema represents ontology, web service description, goal and mediator in RDF along with their properties, i.e. importsOntology, varients and usesMediator. Once the top level entities have been modeled in RDF Schema, then RDF Schema for each of the entity can be further defined. Bellow we present the classes that represent the top level entities.

```
<rdfs:Class rdf:ID="ontology"
    rdfs:label="ontology">
   <rdfs:comment>
      An ontology can be seen as a part-whole hierarchy. An ontology (whole) has parts concepts, relations, instances, axioms and relation instances. A concept (whole) has parts attribute definitions.
   </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="webService"
    rdfs:label="webService">
   <rdfs:comment>
      A web service can be seen as a part-whole hierarchy. A web service
```

```
(whole) has parts (at most one) capability and (possibly multiple) interfaces.
    </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="goal"
     rdfs:label="goal">
    <rdfs:comment>
       A goal can be seen as a part-whole hierarchy. A goal (whole) has parts (at
most one) capability and (possibly multiple) interfaces.
    </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="mediator"
     rdfs:label="mediator">
    <rdfs:comment>
       A mediator may have a source and multiple targets and may use a service
for its implementation.
    </rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="importsOntology"
     rdfs:label="importsOntology">
    <rdfs:comment>
       Any WSML entity (goal, webService, ontology, mediator, interface,
capability, choreography) may import ontologies in order to reuse vocabularies.
    </rdfs:comment>
    <rdfs:range rdf:resource="#ontology"/>
</rdf:Property>

<rdf:Property rdf:ID="variant"
     rdfs:label="variant">
    <rdfs:comment>
       A WSML entity (goal, webService, mediator, ontology) may have a WSML
variant associated with it.
       Defined values:
       http://www.wsmo.org/wsml/wsml-syntax/wsml-full
       http://www.wsmo.org/wsml/wsml-syntax/wsml-rule
       http://www.wsmo.org/wsml/wsml-syntax/wsml-flight
       http://www.wsmo.org/wsml/wsml-syntax/wsml-dl
       http://www.wsmo.org/wsml/wsml-syntax/wsml-core
       In case the same ontology, goal, etc. has different variants associated with
it, the highest variant is chosen.
    </rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="usesMediator"
     rdfs:label="usesMediator">
    <rdfs:comment>Any WSML entity may use a mediator.</rdfs:comment>
    <rdfs:range rdf:resource="#mediator"/>
</rdf:Property>
```

Table 3.1.1.1b: The RDF-S ontology for WSML/RDF [de Bruijn, 2008]

3.1.1.2 WSMO Grounding in SAWSDL:

WSMO Grounding in SAWSDL will allow making WSMO descriptions more accessible to SAWSDL based tools. The purpose of SAWSDL is to link WSDL and semantic descriptions, therefore it can be used for grounding WSMO descriptions in WSDL.

It concerns with the ability to connect accessible concepts from the Web service choreography description with the appropriate WSDL messages so that we know how to transfer instances of the accessible concepts. Then a client system follows the choreography of a Web service, the choreography dictates when certain data can be sent or received, and the grounding specifies

how exactly the it can be sent or received. In grounding to WSDL, sending means that the client will form an input message of a Web service operation, and receiving means the client will receive an output message from an operation. To ground WSMO choreography, model references on the element declarations are placed, that are inputs or outputs of WSDL operations. In particular, an element that is an input message to a WSDL operation should contain a model reference to an "in" or "shared" concept in WSMO choreography, and an output message element should have a model reference to an "out" or "shared" concept.

Sample WSMO-Lite descriptions in RDF:

```
// namespaces and prefixes
@prefix rdfs: <http://www.w3.org/2000/01/rdf□□schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22□□rdf□□syntax□□ns#> .
@prefix wl: <http://www.wsmo.org/ns/wsmo□□lite#> .
@prefix ex: <http://example.org/onto#> .
@prefix xs: <http://www.w3.org/2001/XMLSchema#> .
@prefix wsml: <http://www.wsmo.org/wsml/wsml□□syntax#> .

// ontology example
<> rdf:type wl:Ontology.
ex:Customer rdf:type rdfs:Class .
ex:hasService rdf:type rdf:Property ;
rdfs:domain ex:Customer ;
rdfs:range ex:Service .
ex:Service rdf:type rdfs:Class .
ex:hasConnection rdf:type rdf:Property ;
rdfs:domain ex:Customer ;
rdfs:range ex:NetworkConnection .
ex:NetworkConnection rdf:type rdfs:Class .
ex:providesBandwidth rdf:type rdf:Property ;
rdfs:domain ex:NetworkConnection ;
rdfs:range xs:integer .
ex:VideoOnDemandService rdfs:subClassOf ex:Service .

// capability description example
ex:VideoOnDemandSubscriptionPrecondition rdf:type wl:Condition ;
rdf:value ”
?customer[ex#hasConnection hasValue ?connection]
memberOf ex#Customer and
?connection[ex#providesBandwidth hasValue ?y]
memberOf ex#NetworkConnection and
?y > 1000
”^^wsml:AxiomLiteral .

ex:VideoOnDemandSubscriptionEffect rdf:type wl:Effect ;
rdf:value ”
?customer[ex#hasService hasValue ?service]
”^^wsml:AxiomLiteral .

// definition of the axiom for WSML language
wsml:AxiomLiteral rdf:type rdfs:Datatype .

// nonfunctional property example
ex:PriceSpecification rdfs:subClassOf wl:NonFunctionalParameter .
ex:VideoOnDemandPrice rdf:type ex:PriceSpecification ;
ex:pricePerChange ”30”^^ex:euroAmount ;
ex:installationPrice ”49”^^ex:euroAmount .

// classification example
ex:SubscriptionService rdf:type wl:ClassificationRoot .
```

```
ex:VideoSubscriptionService rdfs:subClassOf ex:SubscriptionService .
ex:NewsSubscriptionService rdfs:subClassOf ex:SubscriptionService .
```

Table 3.1.1.2 : Sample WSMO-Lite domain-specific service ontology

3.1.2 Storage of service monitoring data (ref. WP2)

This section describes the kind of data and information that will be used in WP2, which is devoted to service monitoring, and more specifically makes a draft description of the information that will be stored in the Semantic Space represented in RDF.

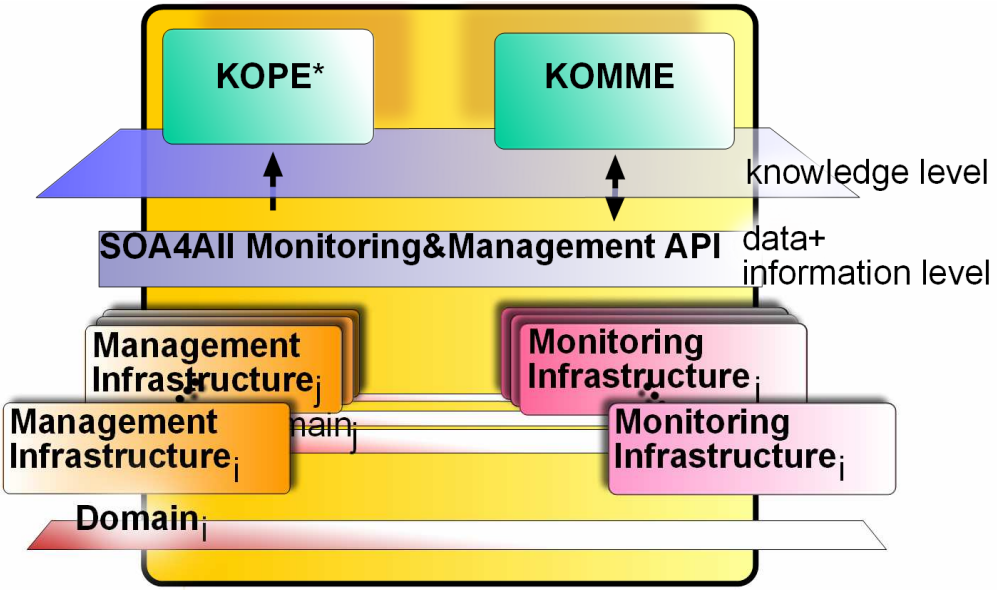


Figure 3.1.2: General overview of the WP2 architecture.

As the figure above depicts, we consider a set of domains that we want to manage, monitor; and where we wish to analyse the information generated by it running processes. That implies that we should handle low level inter/intra domain monitoring; domains that can imply different environments and technologies (in the context of WP2 we will use in our proof of concept PEtALS ESB, ProActive, and TIE Kinetix). We should also carry out management on these domains, providing thus mechanisms for the description and the enforcement of management policies on each of these domains.

All these information handled or generated in the lower layers of the architecture (i.e. in the Management and Monitoring Infrastructures of each of the domains shown in Figure) are accessed in an unified using the Web Services Distributed Management (WSDM) specification; both the specification for the Management Of Web Services (MOWS) and the specification for the Management With Web Services [MUWS Part 1] [MUWS Part 2]. This API (depicted as SOA4All Monitoring & Management API in figure above) provides a single point of access to higher and more abstract layers, composed by KOPE and KOMME.

KOMME will provide knowledge-based techniques in order to detect and diagnose process deviations based on monitoring information and possibly informed by context data. KOPE's goal is to produce domain-oriented interpretations of process executions to increase user understanding of services executions, which can be potentially very large and complex. In order to produce such interpretations, it uses semantic overlays i.e. templates based on reusable Problem-Solving Methods (PSMs), with a high abstraction level, against which KOPE classifies the low-level information documented in logs during process.

Therefore, we will work at the knowledge level both in the description of monitoring and management information and in the interpretation of the logs of services executions. At this level, we will be able to interpret all the results in a more intelligent manner. This knowledge level information is precisely the one that we will store an retrieve from the SOA4All Semantic Spaces. Knowledge level representations can be easily translated in to different representational languages, and more concretely to RDF, which will is the choice for Semantic Spaces.

With this purpose, we are creating an ontology that represents the knowledge-level model in RDFS (see a subpart of this ontology in Table 3.1.2a).

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

  <rdf:Description rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#PrimitiveMethod">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#Method"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#hasDestination">
    <rdfs:range
rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#KnowledgeComponent"/>
    <rdfs:domain rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#Adapter"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#Method">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#FunctionalDescriptor"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#hasOutputRole">
    <rdfs:range rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#Role"/>
    <rdfs:domain rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#Competence"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#KnowledgeFlow">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  </rdf:Description>

  <rdf:Description
rdf:about="http://www.soa4all.org/WP2/PSMOntology.rdfs#KnowledgeComponent">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  </rdf:Description>

</rdf:RDF>
```

Table 3.1.2a: PSM Ontology implemented in RDFS

Once a knowledge level description has been created, it is translated automatically into RDF, instantiating the previously mentioned ontology. Table 3.1.2b contains an example of a generated instance that can be stored in the Semantic Space.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:psm="http://www.soa4all.org/WP2/PSMOntology.rdfs#" >
…
```

```
<rdf:Description rdf:about="http://www.soa4all.org/#Task_Catalogation">
   <rdf:type rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#Task"/>
   <psm:hasInputRole rdf:resource=
"http://www.soa4all.org/WP2#Task_CatalogationInitialObservation_0"/>
   <psm:hasOutputRole rdf:resource=
"http://www.ontogrid.net/#Task_CatalogationInformationRepresentation_0"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.ontogrid.net/#Method_HistoricalCompare ">
   <rdf:type rdf:resource="http://www.soa4all.org/WP2/PSMOntology.rdfs#PrimitiveMethod"/>
   <psm:hasInputRole          rdf:resource="http://www.ontogrid.net/#Method_HistoricalCompare
Observation_0"/>
   <psm:hasInputRole          rdf:resource="http://www.ontogrid.net/#Method_HistoricalCompare
Expectation_0"/>
   <psm:hasOutputRole         rdf:resource="http://www.ontogrid.net/#Method_HistoricalCompare
Observation_0"/>
   <psm:hasOutputRole         rdf:resource="http://www.ontogrid.net/#Method_HistoricalCompare
Normality_0"/>
  </rdf:Description>
…
</rdf:RDF>
```

Table 3.1.2b: Extract from an automatically generated instance of a PSM.

3.1.3 Storage of contextual information (ref. WP4)

Important efforts within SOA4All will be devoted to supporting adapting the provisioning and consumption of services based on contextual information. The main rationale behind this is the fact that a Web scale solution needs to accommodate a huge diversity of individuals and organizations as well as highly dynamic environments. Our understanding of context is very much in line with perhaps the most widely agreed definition which is presented in [Dey, 2001]:

> "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*"

In SOA4All, we do however take a slightly wider approach to context that is not limited to interactions between users and machines but also between machines. After all, one of the main advantages service-oriented technologies provide is the capacity to compose existing services in order to provide ones that are more advanced. Somewhat implicit in this definition of context, is the fact that the relevance of contextual information is dependent on the actual application or service being provided. For instance, the location of the consumer might be irrelevant for some services whereas it might be key in other situations where for example the appropriate legal regulations need to be applied. In fact he underlying essence is that what can be considered as relevant contextual information depends on the task being performed. In a scenario where billions of services are provided and consumed, the diversity of the tasks that will be supported will presumably be outstanding and there needs to be appropriate means for supporting adapting services to contextual factors rather than providing specific services for each and every particular context one may encounter.

In the light of this broad understanding of context, our approach to context adaptation is based on a set of conceptualisations providing the means for modelling contextual information of any kind, and general-purpose machinery able to manage contextual data, use it for recognising concrete contexts within particular situations, and apply this contextual knowledge for adapting services. In order to do so, this general-purpose machinery [Pedrinaci et. al., 2008] referred to as the Contextual Service Adaptation Framework, will provide three core services to the overall SOA4All platform, namely Context Management, Context Recognition and Service Adaptation.

Out of these three services, which are more detailed in [Pedrinaci et. al., 2008], Context Management is of particular relevance here. This service will support the storage, querying and reasoning about contextual information distributed over the Web. Indeed, given our previous definition of context, relevant contextual information will come from a variety of sources such as user information, services information, their previous interactions, and additional domain specific contextual information such as legal regulations, etc. User information will be gathered by service providers based on user provided information or their interactions and it will be enriched with statistical information based on the emergence of communities. Services information will, in a quite similar way be based on existing descriptions and annotations of services, and it will be enriched by monitoring data as gathered by the SOA4All infrastructure. Additionally, the relationships between services and users (i.e., previous executions, user opinion, etc) will enhance the overall body of knowledge, which combined with domain-specific information will provide the level of adaptation pursued.

This broad range of information requires a general purpose machinery able to represent highly diverse and distributed information in an efficient and machine processable way. A context ontology stack will be devised in order to support capturing contextual information so that we can apply advanced knowledge-based techniques (e.g,, classification, parametric design). Furthermore, from an architectural perspective we will investigate the use of Semantic Spaces as the underlying infrastructure for supporting the storage, querying and reasoning about distributed contextual information efficiently over the Web. Indeed Semantic Spaces seem to provide an appropriate abstraction for accessing distributed contextual information in a way that applications are abstracted from the underlying complexities related to storing and reasoning about large amounts of distributed semantic information. We believe that the capability for developing event-based solutions will play a very important role in this respect allowing applications to be automatically notified about relevant contextual changes, such as the incorrect behavior of a service, in order to support the adaptation of the execution accordingly.

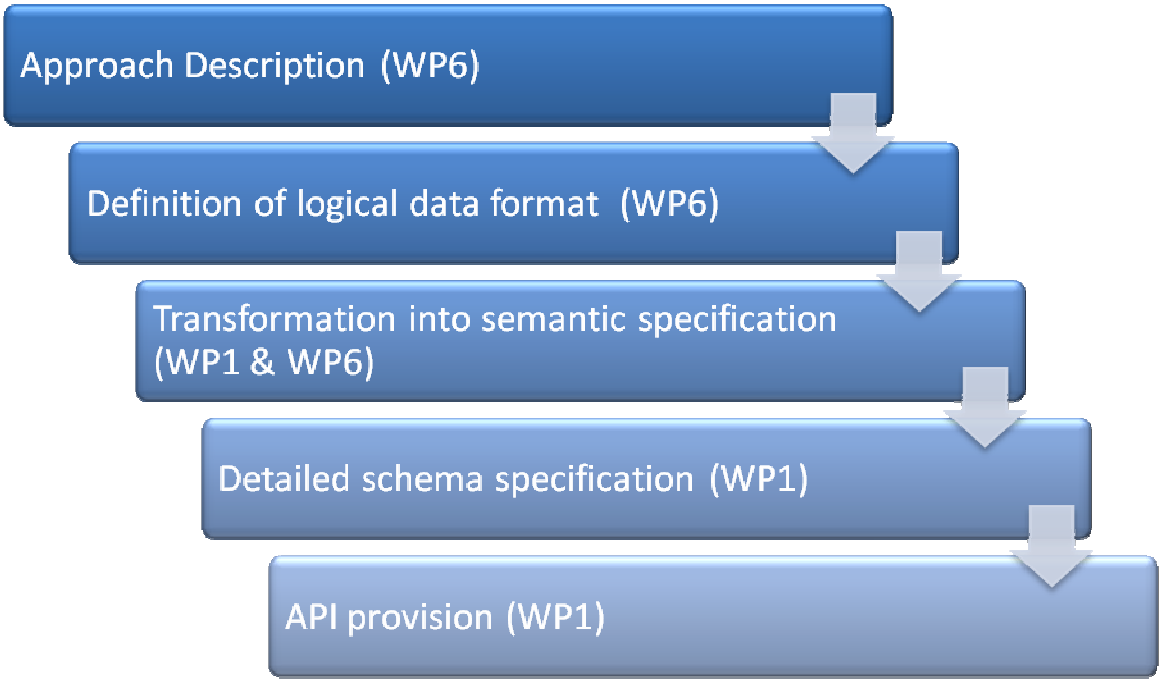
3.1.4 Storage of Composition information (ref. WP6)

Today, Web Services are becoming the de facto standard distributed enterprise computing technology. They are widely used for enabling collaborative business processes between business different partners or even within an organization. Most research on business process management (BPM) mainly addresses issues of highly formalised and heavyweight process descriptions: processes are usually specified once, instantiated very often, highly repetitive and are characterised by a certain degree of temporal stability. In contrast, there is a great need for enabling non-technical users to describe their to-be processes in a lightweight environment. This would allow the untrained user to compose choreographies and mediators to serve their own needs. WP6 is focusing on this aspect. It describes an approach for realizing lightweight business process management and composition. However, although the composition and the end user view might be lightweight, the possibility of storing and retrieving this information in a semantic way (i.e. within a semantic space) has many advantages. The following list shows some examples for those:

- SOA4All will be able to semantically query process descriptions based on different criteria
- Semantic information may be linked to business processes or to parts of business processes. This allows SOA4All to connect this process to other elements of the project
- Processes can be set into relationship with other processes using semantic concepts such as isEquivalent or isUnionOf.
- Within a distributed semantic space, Processes might be stored in a distributed way but they may still be received in a collaborative environment. For example, each company might decide to keep their business processes on their own server only making it available

to a specific group of the semantic space.

In order to enable the usage of the semantic space for service composition, a close collaboration between WP1 and WP6 is necessary. The following figure describes this process and the next steps:



In SOA4All, it is important to provide an easy to use way for users in order to make SOA4All available to non experts. In order to still provide semantic capabilities, the complexity needs to be hidden from the users. SOA4All therefore aims in providing a simple and easy to use interface that allows users to construct business processes and service composition features in a graphical way as described in WP6. Once, the user has performed the graphical description, the data will be generated based on his model. This data will be based on a language that may be interpreted by computers in an automatic way. The precise language will be selected during the technology selection phase of SOA4All.

In order to be able to store the description on the Semantic Space, the business processes need to be described in a semantic language such as RDF or OWL. There are several approaches for realizing this such as OWL-S composition templates. In principle, the description of Web Service compositions needs to be broken down into triples that are describing the process and its elements. For example the triple `(a, invokes, b)` would describe a relationship between two parts of a process.

As the next main step, formats will need to be defined that represent those expressions in RDF. This may result in reusing an existing format, to create a new one or to extend the capabilities of an existing format. As of today, there are several RDF based formats that have been specified in the last years and that might be either reused partly or completely. Popular examples in this domain are WSMO, which is also covered in depth in SOA4All can be used. WSMO uses RDF to represent composed services. Other possibilities include the usage of ebXML Business Process Specification in RDF form (a concrete example may be found here: http://www.w3.org/2001/04/wsws-proceedings/uche/bpsrdf.html) or OWL-S via OWL-S

composition templates. The following code provides an example for the last case[10]:

```
<process:CompositeProcess rdf:ID="BravoAir_Process">
   <rdfs:label>This is the top level process for BravoAir</rdfs:label>
   <process:composedOf>
          <process:Sequence>
          <process:components rdf:parseType="Collection">
            <process:AtomicProcess rdf:about="#GetDesiredFlightDetails"/>
            <process:AtomicProcess rdf:about="#SelectAvailableFlight"/>
            <process:CompositeProcess rdf:about="#BookFlight"/>
          </process:components>
          </process:Sequence>
   </process:composedOf>
</process:CompositeProcess>
```

Table 3.1.4: Sample code of ebXML Business Process in RDF

3.2 Semantic Spaces for Communication

This section explores several possibilities of communication in SOA4All using Semantic Spaces. It includes semantic-based event-driven and publish-subscribe mechanism for SOA4All Distributed Service Bus. P2P overlay networks have been used as baseline for publish-subscribe mechanism to enable event-driven communication and notification mechanism for communication over Semantic Spaces. Space based communication has also been explored for communication of platform and business services in SOA4All. Moreover, usage of Semantic Web pipes for distributed orchestration in Semantic Spaces has also been proposed for SOA4All. Finally yet importantly, Web 2.0 based collaborative approaches in Semantic Spaces for communication of SOA4All services has also been explored. Each of the aspect has been explored in the sub-sections below:

3.2.1 Semantic based Event-driven and publish-subscribe mechanism for SOA4All Distributed Service Bus

Publish-Subscribe systems are used to deliver events from publishers (producers) to subscribers (consumers) distributed across wide area networks. Moreover, the publishers are unaware of the existence of the consumers. The publish-subscribe infrastructure is responsible for matching the events to subscriptions. They are classified into two categories: subject (topic)-based subscribe systems and content-based systems. In topic based systems, each event is assigned to one or a set of topics. A consumer can subscribe to a set of topics and be notified of events associated to these topics. Content based publish subscribe systems allow more complex subscriptions by enabling restrictions on the content of the events. A subscriber can thus specify multiple predicates as a subscription, and receive notifications only from publishers whose events content satisfy all the subscription's predicates.

3.2.1.1 Examples of publish subscribe systems over structured P2P networks

In [Triantafillou et. al., 2004] authors propose a content-based Publish-Subscribe system over the Chord overlay network. In this model an event is characterized by a set of attributes. Each attribute is, in turn, characterized by a type, a name and a value. Note that an attribute can have a

[10] Taken from http://cs.ioc.ee/~tarmo/tsem05/maigre0902-slides.pdf

single value as well as range values. The subscription schema has also a set of attributes such as the name and value as well as a set of constraints on these attributes. Each subscription is characterized by an Identifier which includes the ID of the node that receives the subscription, the subscription itself and the number of attributes to which constraints are specified. The storage of the subscription process starts by applying a hash function on attributes values for the subscription. The return value determines the node ID on the overlay network where the subscription ID will be stored. Since this system supports range attributes values, authors propose an algorithm to store this type of subscription ID. They limit the number of values which can be taken by an attribute, using a notion of precision. A subscription is then registered for every possible value of the attribute. If a description schema has multiple attributes, another procedure is adopted to select node on which the subscription ID will be stored. In this scenario, the same subscription ID can be stored on more than one node in the Chord overlay network. To perform the matching between the subscriptions and the events, attributes are processed separately. If a matching is found for all subscription attributes, the node that generated the subscription is notified by this matching event.

In [Anceaume et, al., 2006] a content-based dynamic publish subscribe system called DPS. In this approach, the overlay network is structured as a virtual forest of logical trees. The basic process behind DPS overlay is that a subscriber joins only the tree corresponding to one attribute of the subscription's attributes randomly selected. During the publication procedure and event has to be published in each tree that matches each of the event's attribute. According to the similarities in their interests, nodes can build their own group based on a common predicate. In addition to that, a ``group predecessor'' relation is defined between groups based on the inclusion relation of predicates. The authors propose a set of algorithms to ensure the group location and the communication between groups and inside the same group. In generic based approach, any node can be a contact node in order to look for the appropriate group in the tree. The search of the group position can be performed in all directions in the tree, whereas in rooted based approach only the root node is chosen as a first contact node so the rooted based approach allows only a downstream propagation of the event.

Regarding the communication within DPS system, two schemes of communication have been proposed. In leader based communication, each group has a leader node and the communication between groups is performed through these leader nodes. To manage the leader group failure, a number of co-leaders is associated to the group's leader. Leaders and co-leaders maintain a complete view about nodes involving in the group as well as others nodes in predecessor and successor group. In this communication model, the leader of the predecessor group is responsible of the invocation of the subscription and the creation of a new group. In the leader-based publication scheme, the leader is responsible to propagate the received events to all group subscribers. The subscribers have to check if the received events match or not their subscriptions. In the epidemic communication model, each node has a partial view about nodes in the same group, in the predecessor and the successor group. A node has to select k neighbors to forward the subscriptions and publications based on the principle of gossiping approaches

Finally, another important feature well studied in this work is that the system can manage node failure not only in the same group but also between groups in the same tree. This can be done by the exchange of update view message between nodes. In the leader-based communication approach, if a leader fails, a co-leader becomes a leader and the new leader identity information will be propagated in all groups.

Authors evaluate the four possible combinations of the system. Based on the obtained results authors show that the leader based approach is more suitable for a small number of nodes and stable network whereas the epidemic communication scheme is more scalable and offer a load balancing on the system even if it generates an important overhead. While the root based approach is more suitable for events publication, the generic based approach is more efficient for subscriptions.

3.2.1.2 Examples of publish subscribe over unstructured P2P systems

SUB-TO-SUB [Voulgaris et. al., 2006] is a content based publish subscribe system build on top of unstructured peer-to-peer networks. It can leverage unstructured P2P networks for features such as self-organization. It supports both exact and range subscriptions. Subscriptions are formed of a set predicates over a set of attributes. In this approach predicates are expressed over a multidimensional naming space. The idea behind this overlay construction mechanism is that subscribers can be clustered based on similarities in their subscriptions. The clustering process is epidemic-based where each peer periodically discovers peers to form a cluster. Specifically, if a subscriber attributes range overlaps with another's then they will be clustered together, so that events are directly routed and efficiently disseminate to the appropriate cluster.
Others solutions can be found, as e.g. in [Chand and Felber, 2005], where each node has to register certain interests (subscriptions) specifying the types of messages that it will be receive. Nodes have similar interests are organized in balanced sub-trees topology with a maximum degree forming semantic communities. Based on this topology, Authors propose an efficient routing mechanism aiming to spread messages within a community that share similar interests and stop forwarding it one it reaches community's boundary. This can reduce the probability that a peer receives a message that does not match its interests (false positives).

3.2.1.3 Publish – Subscribe in Semantic Spaces

Semantic Spaces can be seen as a persistent store for RDF data. It should be possible to put and retrieve RDF triple in that space, which in turn, will be stored on a peer-to-peer network. Each peer will have the responsibility for storing a subset of all available triples. This raises two issues. The first one is the architecture (structured vs. unstructured) and the mapping of the triples on existing peers. We will address this point in section 4.2.4. The second one deals with searching the needed information in the space. This can be done synchronously, i.e. a search returns only the results currently stored in the space. This can be implemented over a DHT if the search is precise for better performances. Another possibility is to have a publish-subscribe mechanism, which will return the information when it becomes available in the space. The main advantage is that it offers an asynchronous solution and can also be used to implement synchronous search.

3.2.2 Space based coordination for SOA4All Platform Services in Distributed Service Bus as well as the Business Services

Currently available Web Services are based on RPC or Document-style invocation where a service requestor sends an invocation request to the service provider's service and waits until it gets the response back. This direct invocation is although simple, but is not scalable in complex application interaction scenarios. Moreover, it requires coupling between the service requestor and service provider, i.e. service requestor has to know which exactly is the service that has to be invoked, and then waiting for the service to send the response/result back. It further effects the compliance of service communication with the principles of Web. Communication in World Wide Web (WWW) is based on persistent publication of information over the Web (i.e. web pages) and then requestors access the information over the Web through Uniform Resource Locator (URL). Similarly, the communication of service in a Web scenario has to be compliant with the principles of Web, i.e. based on persistent publication of information.

In SOA4All, we are exploring on how to enable the service communication using Semantic Spaces, where service requestors requiring a service could just publish the information on space, and any possible/available service upon reception of notification responds. This coordination will be focused only for the platform services of SOA4All, i.e. discovery, selection, composition, mediation etc.

There are following issues that have been identified:

3.2.2.1 Semantic Spaces bindings for WSDL

Web Service Description Language (WSDL) defines the communication protocol bindings: Our goal is to define the bindings for Semantic Space based communication, in WSDL. It will enable Web Services clients to communicate (i.e. invoke) the Web Services through Semantic Space. It concerns with finding a way to have Semantic Space bindings in the service description, secondly it requires binding the communication protocol.

There are following updates that have been identified, to be done in WSDL:

- transport mechanism provides schema information about the transport protocol that is being used
- encoding style mentions the encoding information and namespace information
- address location provides information about exact location that is used for directly accessing the Web Service

The transport information has to be changed that should reflect URI for schema information for RDF representation of SOAP that has been described in 3.2.2.2. Secondly, encoding information for each of the message (in or out or both) has to be provided with the updated information of new RDF encoding of message. Lastly, the address location information has to be updated to provide URI to access the destination client and the ID of sub-space where the message has to be published.

Based on the information above, the proposed updates in the information are to be implemented as follows:

- the transport property has to reflect the underline Semantic Space transport protocol
- Semantic Space based RDF encoding for the SOAP message has to be mentioned to invoke the Web Service
- Address information should contain the client URI and sub-space ID of the triplespace where the target Web Service is subscribed

3.2.2.2 RDF representation of SOAP

SOAP messaging framework defines an XML-based message format and a processing model for Web Service interactions. We will propose the description of a SOAP binding for Semantic Space. It requires RDF representation of SOAP, as RDF is the fundamental data element in the Semantic Spaces. The SOAP messaging framework defines the mechanism of SOAP bindings for enabling transmission of SOAP messages between SOAP processing nodes over a network, using potentially different network transport protocols. For this purpose, they define (i) a serialization of the SOAP infoset in such a way that it can be transmitted by a sender over the chosen network transport protocol (e.g. HTTP, JMS) and reconstructed by the receiver (or the next hop/node in case of multi-hop interactions). Furthermore, they (ii) describe how the services of the underlying transport protocol (i.e. its interface) are used to transmit the chosen serialization of the SOAP infoset between SOAP processing nodes and describe potential failure scenarios that can be anticipated within the binding. Since multiple SOAP bindings for different network transport protocols can be employed along the message path from sender to receiver, the mechanism of SOAP bindings essentially enables multi-protocol message exchange. We propose how to enable the transmission of SOAP messages over Semantic Space. The mapping is based on two major aspects, i.e. mapping SOAP envelope to RDF, and mapping content in the SOAP body part to RDF.

We will propose how to enable the transmission of SOAP messages over Semantic Spaces. The mapping of SOAP to RDF is based on two major aspects, i.e. mapping SOAP envelope to RDF, and mapping content in the SOAP body part to RDF. For SOAP envelop, we use a schema defined in [Shafiq et. al., 2008]. For SOAP body part, we are considering an approach to enable an application specific RDF representation of the SOAP body part. In the first approach, called client-side RDF encoding, the message payload is RDF-encoded by Semantic Space clients (i.e. the applications interacting through Semantic Space). In this case, the application sending a message provides the message payload in form of an RDF graph and the application consuming the message is able to interpret the RDF-encoded message payload. Consequently no mapping to or from RDF has to be performed by Semantic Space. This way of message body encoding can be used e.g. for enabling interaction between Semantic Web Services, using client-side RDF encoding of WSMO instances as defined in the WSML RDF mapping[11].

3.2.2.3 Mapping execution semantics (communication workflow) in Semantic Spaces

In this section, we investigate that how the communication flow can be mapped with Semantic Spaces when a semantic query has to be executed in SOA4All in order to search, select, compose or invoke a Web Service. Communication of internal or platform services of SOA4All Bus will be mapped with the Semantic Spaces, so that whenever there is a request from client, the service execution workflow is executed in the Semantic Spaces, calling any available discovery services (if required), any available composition services (if required), any available invocation services (if required) etc. It will allow the communication within the SOA4All platform by publishing and reading data from the Semantic Spaces, thus enabling the SOA4All platform to become a truly distributed, dynamic and decoupled platform for semantic service execution.

Web Service Execution Environment (WSMX) [Vitvar et. al., 2007] is the reference implementation for Semantic Web Services. If it is used as basis in SOA4All distributed service bus for getting service discovery, selection, composition etc, component management of WSMX will be adapted to use Semantic Spaces. Component management in WSMX has been carried out by its manager that manages the over all execution of the system based on execution semantics supplied to it [Haselwanter et. al., 2005]. In this way there has been made a clear separation between business and management logic in WSMX. Each component in WSMX has a wrapper to handle the communication issues. The WSMX manager and individual components wrappers are needed to be interfaced with Semantic Space in order to enable the WSMX manager to manage the components over Semantic Space. The communication between manager and wrappers of the components will be carried out by publishing and subscribing the data as a set of RDF graphs over space. The wrappers of components that handle communication will be interfaced with Semantic Space middleware. The WSMX manager has been designed in such a way that it could distinguish between the data flows related with the business logic (execution of components based on the requirements of a concrete operational semantic) and the data flows related with the management logic (monitoring the components, load-balancing, instantiation of threads, etc). The Semantic Space is accessed by its client that can be seen as Semantic Space client proxy. The integration will be carried out by embedding these client proxies in wrappers of each of the WSMX component, in order to enable the individual components communicate over Semantic Space.

3.2.2.4 Communication of SOA4All Communication Bus with Users and Business Services

The application services or enterprise application integration middlewares are the typical examples of SOA4All clients. In this scenario, a client can be either a service provider or a service requester involving in the communication in SOA4All. In this scenario, users communication with

[11] http://www.wsmo.org/TR/d32/v0.1

SOA4All Service Bus either to find the services they are interested in or to register description of services they are offering will be provided with a Semantic Space based communication. The interfaces for sending and receiving external messages from clients will be provided a grounding support to alternatively communicate over Semantic Space.

We will also investigate the communication of Platform Services in SOA4All distributed Communication Bus with Business Services (i.e. end-point Services). In order to cover all possible Web Services available over the internet, we will consider the communication of SOA4All communication Bus with standard end-point Web Services (i.e. that are based on W3C standards, using typical WSDL and SOAP), as well as the services enabled with Semantic Space based communication. The communication with standard end-point Web Services will require a mediator mechanism to mediate the communication from Space based communication to direct messaging based communication.

3.2.3 Semantic Web pipes for distributed orchestration on the Semantic Spaces

Service-oriented approaches use services in order to support the development of complex applications out of reusable and distributed existing ones. Services are autonomous and platform-independent software components, which are described, published and discovered in novel ways giving birth to brand-new solutions in relatively fast and cheap development processes. A wide range of technologies have been produced so far that aim to support service-oriented development from the beginning to the end, namely SOAP, WSDL, the so-called WS-* standards, BPEL4WS or the Enterprise Service Bus [Hohpe & Woolf, 2003].

These technologies have proven their benefits and they are widely applied within the industry. However, they also present important drawbacks, which have slowed down or even impeded the initially foreseen adoption on a Web-scale as a means for completely automating services over the Web [Papazoglou et. al., 2007]. Among the limitations identified we can cite for instance the need for semantically-enhanced discovery techniques, the difficulties for integrating heterogeneous data, and the pressing need for achieving self-configuring, self-adapting, self-healing, self-optimizing and self-protecting solutions.

The development of self-* solutions presents interesting challenges since the need for autonomic capabilities contrasts with the “black-box” and passive nature of services. Services are thought as encapsulating functionality that is invoked on demand, directly by a client or by some workflow/orchestration engine. From the perspective of a single service, autonomic capabilities can be achieved in a more or less complex manner by incorporating and directly applying monitoring or in general contextual information. This requires indeed architectural support and most often one would opt for event-based solutions whereby the relevant information is propagated automatically as opposed to actively pulling information. From the perspective of complex orchestrations of services, reaching this level of autonomic behaviour becomes even more complex and proper mechanisms for obtaining contextual information, processing it and taking decisions need to be put in place at a larger scale. Not surprisingly autonomic systems often couple Artificial Intelligence techniques with the appropriate infrastructural support for monitoring and managing components. Other relevant work for instance comes from research in areas like the Semantic Web, Semantic Web Services and Agents that promote a vision where intelligent software components are acting on behalf of the user in a somewhat autonomous way. The promise is to enhance the user experience, to provide functionalities that are more advanced in a more automated way.

There is therefore a growing need and interest in achieving smart and proactive systems but doing so requires adapting our techniques and technologies. A vision like the one pursued in SOA4All requires a certain level of adaptability to contextual changes as we previously showed. Furthermore, the use of semantic technologies paves the way for developing proactive services, which are not solely acting on demand but may instead act on behalf of humans for achieving

certain goals. One could for instance envisage the provisioning of services that monitor auctions on behalf of users in order to bid at the appropriate moments, services that monitor software and hardware and take corrective or even preventive measures to, say, ensure a certain Quality of Service, etc.

Research in Artificial Intelligence focussed on applying the Blackboard Problem-Solving Model and derived architectures for supporting intelligent monitoring and adaptive systems [Engelmore & Morgan, 1988] [Pedrinaci, 2005]. This model promotes a completely decoupled and distributed approach to problem-solving whereby a set of experts are monitoring a blackboard that captures the state of affairs in order to see how they can contribute. Interaction takes place solely through the blackboard in an incremental and opportunistic way. Overall the Blackboard Model was found to be particularly useful for solving complex tasks and obtaining proactive, or as they typically refer to in the literature, opportunistic solutions. The interested reader is referred to [Engelmore & Morgan, 1988] [Pedrinaci, 2005] [Nii, 1986] [Corkill, 2003] for further details.

Semantic Spaces, appears to be an appropriate infrastructural solution for implementing the Blackboard Model of problem-solving and is therefore quite well-suited for supporting the development of intelligent proactive services over the Web. Spaces can simply support the distribution of the blackboard over the Web and the capacity for an event-based notification of changes on the data it holds can provide the appropriate means for ensuring the interested services receive relevant information, as it is available. We will therefore investigate the application of Semantic Spaces for supporting the development of solutions that will complement both traditional passive services and the supporting infrastructures. By doing so we will complement the overall SOA4All infrastructure in two main ways. On the one hand we will support the provisioning of proactive services able to act autonomously on behalf of the user, and on the other hand, we will enhance the architecture with the capacity for including smart services for achieving adaptive infrastructural solutions.

3.2.4 Web 2.0 based collaborative approaches in Semantic Spaces for SOA4All Services

SOA4All will provide an integrated framework upon which to build the Internet of Services, leveraging on four pillars: Web, SOA, Semantic Web, Web 2.0. This integrated framework instantiate several SOA4All processes that are exploited by final users to build up the Web Service Marketplace. Most of those processes are instantiated by final users by accessing to Web 2.0 lightweight final-users-oriented applications (aka Web Rich Applications, WRA), which exchange data and metadata, not only among them, but also with other SOA4All backoffice components, through a common persistence repository and communication space. Provided that those frontend WRA exploits Web 2.0 techniques, it seems quite reasonable to exploit also Web 2.0 techniques for communication between SOA4All frontend WRA and the common persistence and communication layer as Semantic Space (SS).

SOA4All WRA tools are used mostly to support the provisioning and consumption of services within the Web Service Marketplace. Those WPA tools will cooperate each other based on semantic information exchange through the SS. Those WRA tools are:

- Provisioning, Deployment tools: provisioning tools will provide semantic descriptions of services, composite services specifications (together with composition tools for further service enactment), deployment descriptors, etc. that will be stored within the Semantic Space for further usage (for instance, in discovery for further consumption).
- Consumption and composition tools will exploit service descriptions (capability descriptions) and composition specifications stored within the Semantic Space through discovery, ranking and selection processes after posting goals semantic specifications.

- Monitoring and provenance tools will provide and consume monitoring and provenance data, also stored/retrieved to/from the Semantic Space, for subsequent post-mortem or “on the fly” execution analysis.
- Service life-cycle management will manage services life cycle information that will be also stored/retrieved to/from the Semantic Space
- Context information (user’s context, location context, execution context, etc) will be also stored/retrieved to/from the Semantic Space, which is quite relevant for service consumption

Since those SOA4All WRA tools provide nice lightweight Web features and they communicate each other through the SS, SOA4All should provide a Web 2.0 communication API library, the enables SOA4All WRA to exchange common data and metadata through the SS by accessing the methods exposed by that Web 2.0 library. That SS Web 2.0 library should provide features to: a) exchange data and metadata through the common Semantic Space (that is, write and read operations supporting different data/metadata granularity), b) advance SS querying and retrieval (allowing retrieval of single RDF triples or complex constructs), c) exchange of messages with SOA4All infrastructure services and external services, d) a library of WRA SS visual components, e) synchronous and asynchronous message exchange pattern (MEP) support as well as other event oriented MEP such as broadcasting, publication/subscription.

SS Web 2.0 library should also permit SOA4All WRA to refresh concrete content without requiring to retrieve large data content, to build up graphical data viewers (lists, tables, trees, etc), to execute services within SOA4All WRA enabling service enactment, discovery and selection, invocation, etc.

SS WRA API library should permit user-machine and machine-machine communication. Semantic data contained within the SS is not intended for human consumption, but for machine usage. That implies that the most relevant usage of SS WRA API is to support machine-machine communication, but without forgetting that final users are humans accessing to the SOA4All WRA to provide and consume services, therefore this API should also provides a rich widget library to make service responses accessible to humans. This can be achieved as aforementioned with a rich library of widgets for data representation (lists, tables, trees, etc).

We can also consider to build a WRA frontend for SS, exploiting Web 2.0 techniques, quite suitable to manage distribute SS content by SS administrators, which take advantage of WRA feature aforementioned and other common to Web Applications, such us a) accessible everywhere through ubiquitous Web browsers, b)not installations required, c) latest version of SS frontend always available for SS administrators, d) lack of configuration and installation problems, d) user configuration always available regardless the browser used for accessing the SS frontend, etc.

SS Web 2.0 library can be implemented using Web 2.0 techniques such as: a) AJAX and XML/JSON, which are quite suitable for the purposes aforementioned of service communication, data provisioning and retrieval, b) MASHUPS, PIPES to enable service communication and data pipeline among chained services execution, c) JSF, and other rich web widgets libraries supporting complex data representation and other widgets (buttons, labels, etc) quite used in standalone rich client platforms (.NET, SWING, SWT, JFACES, etc), d) RSS, ATOM, and other syndication technologies, to support the common SS accessing approach based on publication/subscription, to populate SOA4All WRA with distribute SS content, according with final user profile and preferences, e) REST Services, since they provide a more natural approach for Web Services (WS) provisioning and consumption in the Web context, since they manage WS as web resources and exploit HTTP transport layer without requiring additional protocols and data exchange formats (SOAP), f) Social Networks/Folksonomies techniques, for collaboratively population of SS with services descriptions, rating of services, service reputation, repudiation, etc.

3.2.4.1 Service communication over shared knowledge spaces naturally calls upon the integration of Web 2.0 motivated collaborative approaches.

In many business situations the semantic description of a service is not sufficient for enabling an effective n:m communication, since no reference meta-model can be developed. This can typically happen with services exchanging a business content (i.e. a business message), and/or when considering organisational and linguistic-cultural differences among business partners. In these cases, a Federated solution is required, based on negotiation and on-the-fly mediation: in absence of a complete well-defined meta-model, negotiation and collaboration, techniques are necessary to try to understand each other. In business terms, this interactivity and negotiation capability is very important for achieving agreements on details, whenever accompanied by some more formal and well defined overall framework like that a unified approach can provide.

As an example, with can consider an e-business supply-chain scenario, where the various actors are using the Semantic Spaces for asynchronous communication, exchanging business document (BOD) based on UBL standard (http://www.oasis-open.org/committees/ubl): although the overall semantic definition of UBL-based documents is universally defined (unified approach), when exchanged in a multi-cultural n:m pattern, the content of some fields may depend upon specific conditions. For example, the “quantity” field may expressed in “Kg” or in “Lbs”, or “Delivery date” can change according to distance.

Web2.0 technologies are the most natural and suitable ones in order to enrich the Semantic Space with a Negotiation Environment, able to customize SOA4All exchanged messages according to negotiation rules that can be agreed upon in a collaborative way by users. Negotiation rules can then be easily stored in some state-of-the-art Rules Repository and even used by some other state-of-the-art rules engine. In this way, the Semantic Space, when used for communication, will be able provide a message with content defined according to the business rules.

4. Semantic Spaces – Conceptual Model

Semantic Spaces have been envisioned as an extension of space-based computing systems with semantics. Triple Space Computing [Fensel, 2004] is one example of Semantic Spaces, which extends tuple-space computing using RDF as the formalism for describing the content of tuples in the space. It makes the fundamental data element as RDF triple in the Semantic Space. In order to store any information, all the data should be represented in RDF as being the W3C standard for publishing data on Semantic Web. Similarly, communication between different parties will be done by publishing and read RDF triples on the Semantic Space. In this way, Semantic Spaces acts as Web for machines, where different services can communicate with each other by publish and read of semantic data, i.e. also making the communication compliant with the principles of the World Wide Web.

This section provides an overview of Semantic Space data model and introduces key concepts about Semantic Spaces based on the work done in Austrian funded FIT-IT TSC[12] (Triple Space Computing) and EU FP6 TripCom[13] (Triple Space Communication) projects. It further specifies the Semantic Space infrastructure and outlines the basic building blocks that are required. Distribution mechanism based on unstructured peer-to-peer (P2P) networks for Semantic Spaces has also been described. Furthermore, communication and storage of SOA4All services based on Semantic Spaces infrastructure has been explained.

4.1 Semantic Spaces Data Model

This section presents the key concepts and the basic data model of Semantic Spaces. These concepts and basic data model has been described based on the results from EU FP6 STREP TripCom (Triple Space Communication) project [Murth et. al., 2007].

4.1.1 Key Concepts

Space: A space is a set of semantic data that is identified by some identifier, i.e. the name of the Space.

Sub-space: A sub-space is a subset of data in a Space that is identified by a specific context. A sub-space complies with the definition of a Space and can have sub-spaces itself.

Semantic Space Kernel: A Semantic Space kernel is a single physical implementation consisting of all necessary components for implementing the Space API. A kernel is deployed on a physical infrastructure and is addressable using a specific physical network address like a single physical machine or a single physical cluster of machines.

Semantic Space Node: A node is one particular instance of Semantic Space Kernel, which is accessible over the Web with a URI and exposes the user API to access the semantic space.

Clients: These entities use Semantic Spaces either for communication or for storage. There are

[12] http://tsc.deri.at

[13] http://www.tripcom.org

three different types of clients that have been foreseen in this scenario, i.e.

- Platform Services of SOA4All
- Business/end-point Services
- SOA4All Components requiring Storage of data in Space

Data Store: All the data and information is required to be physically stored over the space. The data store is used to persistently store the space.

4.1.2 Model

This section describes the basic data model for Semantic Space in order to store the data. RDF (Resource Description Framework) will be used as fundamental data element in the Semantic Space. Therefore, the Semantic Space model will be composed of following elements:

- Identifier: It is the unique identifier that is used to identify an RDF triple
- Subject: It is referred as RDF resource that can have a URI
- Predicate: It is referred as Property
- Object: It is referred as Property value which can be a concrete value or a another RDF resource

The data model of Semantic Space helps in structuring the data in the space. There are different possibilities, i.e. it can be Single, Flat, Nested or Tree like. There are different advantages and disadvantages for each approach. Single data model helps in simplified and straightforward implementation and access through single access point, but on the other hand, it has a single point of failure and is not scalable, and hence not suitable for our requirements in SOA4All. Flat data model allows distribution of data but it provides no structuring and relationship between different sub-spaces. Nested data model provides high flexibility in structuring spaces but it causes difficulties in updating the data afterwards. Tree like data model allows layers and limited nesting but on the other hand, also limits the interlinking of data in space.

The structure of semantic tuples specified in TripCom states the conceptual structure as follows [Simperl et. al., 2007]:

```
<s, p, o, tupleId>
<tupleId, isContainedIn, spaceId>
<tupleId, hasPart, graphId>
```

The tripleset data model is optimized to maintain efficiently contextual information on a single triple level [Sapkota et. al., 2007]. The former three triples could be transformed in a single tripleset statement:

```
<s, p, o, spaceId, {graphId, ...}>
```

The advantage of the optimized approach is that there is significant decrease in the number of required triples. Hence the ontology will scale much better; there is explicit relation between data and meta-data, so the computational complexity to maintain consistent data model is much lower.

4.2 Semantic Spaces Infrastructure for SOA4All

This section provides details about the infrastructure of Semantic Spaces for SOA4All. It describes the structure of kernel based on what Semantic Spaces are composed of. It further identifies the key components to fulfil the needs of Semantic Spaces functionality. It includes persistent storage, query processing, access API, distribution and information management, transaction management and Web Services communication management. Required operations in the API have also been described, followed by an initial proposal for managing distribution in Semantic Spaces.

4.2.1 Semantic Spaces Infrastructure

Semantic Spaces are composed of several inter-connected kernels forming a virtually single view of the data in the space that is physically distributed. The figure below shows the first view of the Semantic Space Kernel. Each Semantic Space Kernel is provided with one or more data stores in which the data is persistently stored. The Semantic Space Kernel is composed of four major parts, i.e. user access, publish-subscribe mechanism, inter-kernel coordination, and data access layer. The purpose of Data access layer is to provide a uniform access to the data stores, i.e. there can be RDF data stores or RDBMS stores can be used. The inter-kernel communication layer will be responsible for distribution in the Semantic Space (i.e. using P2P mechanism). The publish-subscribe mechanism has also to be provided in the Semantic Space so that different clients communicating over Semantic Space could subscribe to some space, and receive notification accordingly. There will be user and management APIs that will be provided by the Semantic Space Kernel. User API will allow Semantic Space Clients to publish and retrieve information from the Semantic Space. The Management API will allow the clients to create, update and delete the logical sub-spaces in the Semantic Space, as well as manage the trust and security information, as may be required.

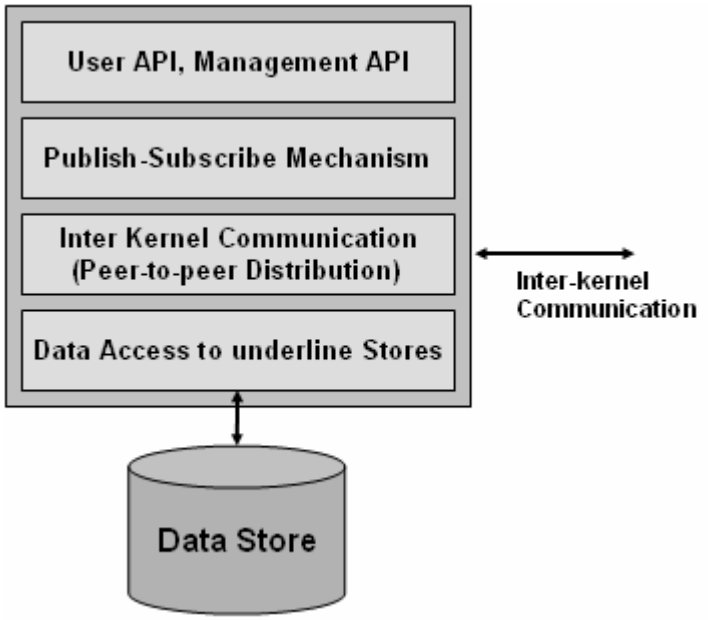


Figure 4.2.1: Semantic Space Kernel

4.2.2 Key components in Semantic Space Infrastructure

This section identifies key components in Semantic Space Kernel as a key to the Semantic Space

Infrastructure that directly contributes to the functionality of Semantic Spaces.

Data Storage: It is supposed to provide a flexible and configurable data storage facility that should abstract the data store level details and provide a uniform access to the data in underlying data stores. There can be one or more data stores underline, where it should provide data federation support to have a single view of data at a Semantic Space Kernel. It should also provide support for supporting integration of semantic with legacy one, i.e. Relational Database Management Systems (RDBMS). Furthermore, it should also provide basic and efficient reasoning facilities for efficient and effective retrieval of data from the stores.

User and Management Access (API): It is supposed to provide an API to enable the Semantic Space clients to publish and access information in the Semantic Space. Similarly it should also provide API for management of the data published in space, i.e. creating, updating or deleting sub-spaces, defining relationship between different sub-spaces, and managing publish-subscribe mechanism.

Query Processing: Query processing facility is required to process the requests made to access, manipulate or publish triples. It should take care for reasoning and optimizing the query before its actual execution, as well as query distribution across different nodes in Semantic Space.

Distribution Management: It is supposed to take care for incoming data access and manipulation requests and route these requests to relevant kernel. There will be the cases that data is stored on one particular kernel of Semantic Spaces, and its access request is made by a client from another kernel. In this case, mechanisms for Distributed Management are required in order to enable the query and data routing across different kernels available in the Semantic Space.

Transaction Management: It refers to handle the transactions that are offered by the access and management API. It is supposed to ensure ACID properties (i.e. Atomicity, Consistency, Isolation and Durability) of the data stored, being accessed and changed in the space.

Information Management: It refers to provide support to manage the structure of data published in Semantic Space and support the operation of the Semantic Space kernel when some data is to be accessed or published in the space. It includes describing information about relationship between data and space, relationship between different spaces.

Web Service Communication Management: Semantic Spaces will be used for communication of Platform as well as Business services in SOA4All platform. This refers to provide support to bridge the two communication paradigms together, i.e. the Web Services communication and Semantic Space based communication. It will enable in adapting Web Services communication using Semantic Spaces, by providing necessary bindings to Web Services description and communication protocol.

4.2.3 Semantic Space API

This section describes the coordination support that has to be supported by Semantic Spaces in

order to allow its clients to publish, retrieve and manage data in the space. The API should support following operations:

- **Publish triples:** this operation should allow users to publish data in the space and retrieve a URI for later access of the data.
- **Read triples:** it should allow users to retrieve data from the space that is published already. It can be based on providing an exact URI of the space or RDF graph, or providing a template (i.e. based on SPARQL).
- **Subscribe:** it should allow user to subscribe to some particular event or data published, retrieved or changed in Semantic Space, and issue notification in case of the change in the state.
- **Unsubscribe:** it should allow users to unsubscribe to particular events or data to which it was previously subscribed to, and stop receiving notifications.
- **Create transactions:** while accessing or manipulating data in the Semantic Space, it should allow users to create transactions to ensure atomicity, consistency, isolation and durability in the operations carried out on Space.
- **Execute transactions:** while accessing or manipulating data in the Semantic Space, it should allow users to execute transactions to ensure atomicity, consistency, isolation and durability in the operations carried out on Space.
- **Delete transactions:** while accessing or manipulating data in the Semantic Space, it should allow users to stop transactions that were created before, for the operations carried out on Space.
- **Rollback transactions:** it should allow to roll back the transaction to undo the manipulations made in the data during an operation.
- **Creation of sub-spaces:** in order to manage the data in the Semantic Space, users could be able to organize data in logical sub-spaces. This operation should allow creating new sub-spaces as required.
- **Deletion of sub-spaces:** it should allow users to delete all the sub-spaces that were created before.

4.2.4 Distribution in Semantic Spaces

The Semantic Spaces will be structured as a peer-to-peer network, more precisely some overlays built on top of a peer-to-peer architecture. Using an overlay allows us to separate low-level constraints like networking and firewall issues, from the high level distribution of data. So the spaces will be organized at two levels. At the low-level, a peer (physical machine or process), will have neighbors based on latency, network proximity… At the overlay level, a peer will have other neighbors, chosen using different criteria such as proximity of the stored data. The low-level architecture will be unstructured as it offers a lot of flexibility. We plan to use structured overlays because they offer good performance.
Our current work focuses on CAN [Ratnasamy et. al., 2001] because of its multi-dimensional nature. The general idea is that all possible values to be stored form a n-dimensional space. As an example, managing information about CPU frequency and memory size would require a 2-dimension space, one for the speed and the other for the memory. As shown in Figure 4.2.4a, each peer is responsible for a part of the global space. To search for some information, one has to first contact a random peer. If it doesn't hold the need information, then it will route the message in the space, contacting only neighbors which are closer to the information.

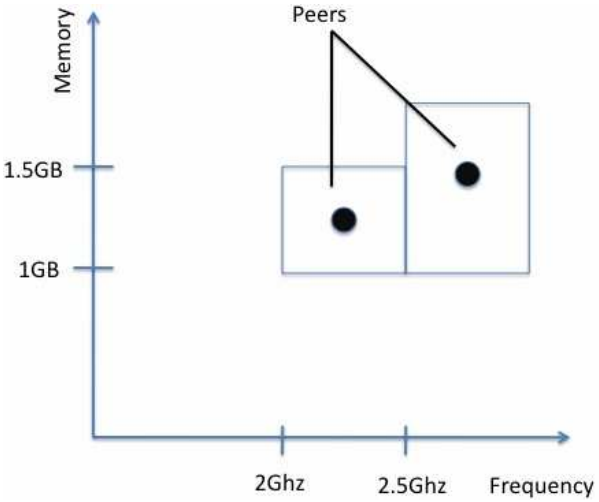


Figure 4.2.4a: CAN 2-dimension space example

A possibility to store RDF data would be to use the predicate as a dimension and the property as the value. This allows efficient searching since it boils down to the traversal of a multi-dimensional space, which has been shown to be efficient. Basically, a peer will be given a region of the space and will store all triples falling in that region. To avoid a fragmentation of the space (each peer responsible for only a triple), clustering will be used. When a peer manages a region, all peers joining the network with data falling close to this region will form an unstructured network with the first one. This mechanism will ensure good performance (as it is dependent on the number of regions) and provide fault tolerance. If the manager leaves the network, then any other peer in his cluster can take over.
As seen previously, semantic spaces are made of several interconnected kernels. They will provide the low level architecture. Each node will host one or many kernels, each managing a part of the CAN space. This use of virtual kernels allows a physical kernel to hold data which could be widely separated in the CAN space, providing a high flexibility in the architecture.

The figure below gives an idea that how different Semantic Space Kernels (as described in the section above) will be interconnected with each other (forming a P2P network) to form a virtually single, physically distributed, shared Semantic Space.

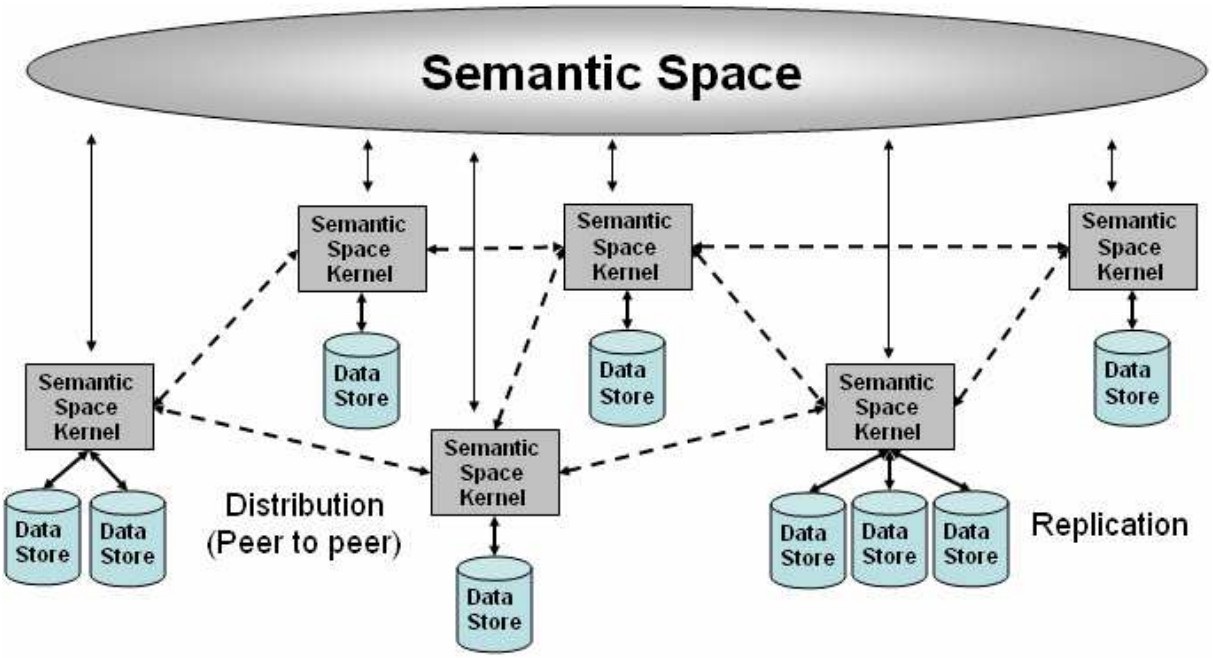


Figure 4.2.4b: Semantic Space Infrastructure based on unstructured P2P overlay network

4.2.5 Semantic Spaces Infrastructure for SOA4All Distributed Service Bus

This section describes the overall scenario that how Semantic Spaces infrastructure will be used within SOA4All for the purpose of communication and storage. Semantic Spaces will be composed of several kernels deployed and different machines, located at different geographical location and interconnected with each other over the internet forming an unstructured P2P network. Usage of unstructured P2P network will give the flexibility for any Semantic Space kernel to be able to connect to any other available kernel, and to become part of Semantic Space network. With each of the kernels, there will be one or more data stores connected to it, in order to fulfil the storage requirements for any particular kernel. All kernels will be interconnected to each other and providing an access API to the clients. The interconnection of kernels will allow giving an illusion to the users that it is a virtually single and globally shared semantic space. Users will interact with Semantic Space by accessing the API of any of the Semantic Space kernel. Storage of data by the clients will be taken care by the kernels. The client will not have to take care of data storage issues. In this case, Semantic Space will become an underlying distributed storage middleware solution, more or less like a semantic data grid. All the components and services in SOA4All requiring to store data on Semantic Space will be able to do it by accessing any kernel over the Semantic Space network. Data that may include semantic description of services, contextual information related to services and its clients, service monitoring data and data related to service composition will be stored on semantic space regardless of the location where it has been stored. The user will only have to have the URI (the unique identifier) of the relevant space or data, to access the data afterwards. Moreover, the data can also be retrieved by query processing capabilities of Semantic Space and by providing a template for search.

The figure below provides an idea that how the Semantic Spaces Infrastructure (described in previous section) will be used in the SOA4All scenario. There will be different kinds of clients using Semantic Spaces, i.e. The SOA4All platform Services, end-point/business Services, and the components in SOA4All bus requiring to store some information.

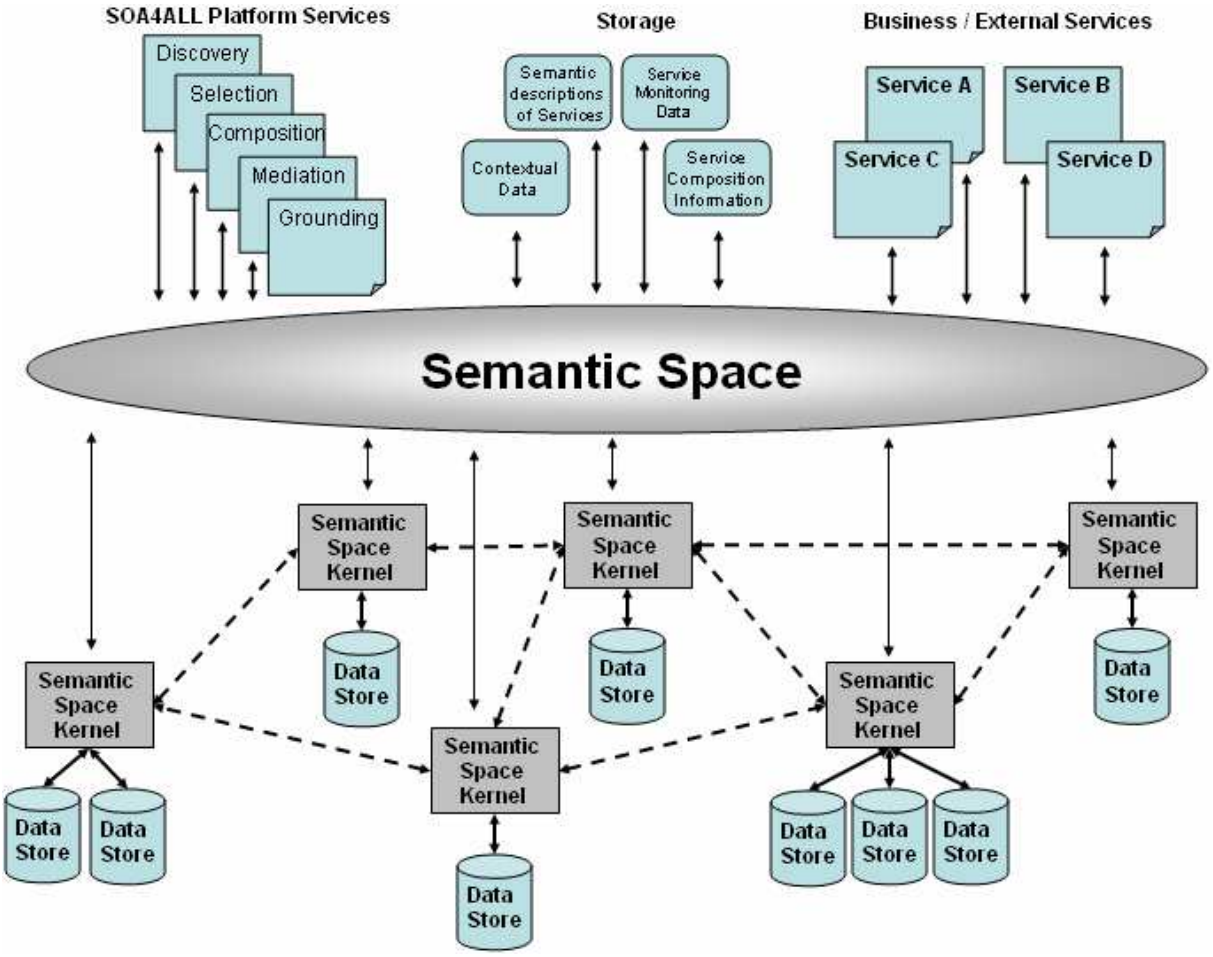


Figure 4.2.5: Semantic Spaces Infrastructure for SOA4All

Semantic Spaces will also be used as underline communication infrastructure for SOA4All services. Semantic Spaces infrastructure allows maximum decoupling between the communicating parties, such that, the sender does not have to know about the receiver in advance. There will be different users on Semantic Space that will be subscribed to particular sub-space or template, and will be notified in case of any change in the state of the Semantic Space. Any user, for example, requiring service discovery publishes a Goal (semantic query describing the type and functionality of service required) in the Semantic Space. All the services that are related to provide service discovery can be notified and provide discovery services to the sender (or service requestor). It can also be possible that during a communication phase, the sender is connected to a different Semantic Space kernel, and receiver is connected to another Semantic Space kernel, both the kernels and interconnected to each other via unstructured P2P network.

Semantic Spaces middleware will hide the complexity related to storage and communication issues and will act as underline middleware for SOA4All. It will provide storage and communication services where components of SOA4All and its clients will be able to store and retrieve data regardless of the any details that where exactly the data has to be stored and organized. Semantic Spaces will take care of storage, distribution and replication of the data, and provide API to access the data accordingly. Having semantic representing the data stored in Semantic will help in precise and efficient retrieval of data from the globally shared Semantic Space. Similarly, SOA4All service bus and its client will be able to use Semantic Spaces for communication with maximum amount of decoupling. The senders will not have to know the receiver in advance. The semantic descriptions used in the Semantic Spaces will help in finding

precise and accurate intended receivers of the data or message published in the shared space, and thus making the SOA4All, a global and distributed Bus for services from all over the World-Wide-Web.

5. Conclusions

Semantic Spaces have been envisioned as semantic extensions to Space-based Computing systems for communication and coordination. This deliverable introduces the use of Semantic Spaces within SOA4All. It carries out a detailed survey and analysis of state-of-the-art in Space-based Computing systems and related approaches for semantic extensions to it. Requirements analysis has also been carried out which are further analyzed to see their potential to contribute to the solutions to be developed to face the challenges of Service Web. The deliverable has further described that how Semantic Spaces can be used in SOA4All for several storage and communication purposes. As a first step, Semantic Spaces will act as distributed, semantic storage for SOA4All. Semantic Spaces will be used to store semantic descriptions of Web Services, Monitoring data for semantic service execution, contextual information related to services and users, and storage of compositional information. Secondly, Semantic Spaces will be used as communication and coordination paradigm for SOA4All. It includes building Semantic-based event-driven and publish-subscribe mechanism for SOA4All Distributed Service Bus, Semantic Space based coordination for SOA4All platform services as well as business (external) services. Furthermore, usage of Semantic Web pipes for distributed orchestration for services in Semantic Spaces has also been investigated. Last but not least, Web 2.0 based collaborative techniques have also been investigated for the communication in SOA4All using Semantic Spaces. Based on this, the deliverable further describes the Semantic Spaces for SOA4All based on the latest research and development results from EU FP6 Strep TripCom (Triple Space Communication) project. The description includes Semantic Spaces data model. It further draws architecture of Semantic Space node and identifies the functional components necessary for features of Semantic Spaces. These key features are persistent storage of data, user and management APIs, query processing, distribution management, information management as well as Web Services communication management support. It also describes the methods that are required to be supported by the API of Semantic Space, followed by an initial proposal for distribution of Semantic Spaces.

6. References

[Chawathe et. al., 2003] Y. Chawathe and S. Ratnasamy and Lee Breslau and Nick Lanham and Scott Shenker, Making gnutella-like P2P systems scalable, SIGCOMM, 407-418 , 2003.

[Gu et. al., 2007] T. Gu and H. Keng Pung and D. Zhang, Information retrieval in schema-based P2P systems using one-dimensional semantic space, Computer Networks, 51,16 , 2007.

[Nejdl et. al., 2003] W. Nejdl and W. Siberski and M. Sintek, Design Issues and Challenges for RDF and Schema-Based Peer-to-Peer Systems, SIGMOD Record, 32(3): 41-46, 2003.

[Triantafillou & Aekaterinidis, 2004] P. Triantafillou and I. Aekaterinidis, Content-Based Publish-Subscribe Over Structured P2P networks, DEBS,, 2004.

[Anceaume et. al., 2006] E. Anceaume and M. Gradinariu and A. Kumar Datta and G. Simon and A. Virgillito, A Semantic Overlay for Self- Peer-to-Peer Publish/Subscribe, ICDCS, 22, 2006.

[Voulgaris et. al., 2006] S. Voulgaris and E. Riviere and A. M. Kermarrec and M. van Steen, Sub-2-Sub: Self-Organizing Content-Based Publish Subscribe for Dynamic Large Scale Collaborative Networks, IPTPS, 2006.

[Chand & Felber, 2005] R. Chand and P. Felber, Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks, Euro-Par 2005 Parallel Processing, Lisbon, Portugal, 2005.

[CORSO] CORSO Tutorial, http://www.complang.tuwien.ac.at/eva/Download/downloadIndex.html

[Fensel & Bussler, 2002] D. Fensel & C. Bussler: The Web Service Modeling Framework (WSMF). Electronic Commerce Research and Applications 1(2). 2002.

[Fensel, 2004] D. Fensel: Triple-space computing: Semantic Web Services based on persistent publication of information: In proceedings of the IFIP International Conference on Intelligence in Communication Systems, INTELLCOMM 2004, Bangkok, Thailand, November 23-26, 2004.

[Fensel et. al., 2007] D. Fensel, R. Krummenacher, O. Shafiq, E. Kuehn, J. Riemer, Y. Ding, and B. Draxler, "TSC - Triple Space Computing", special issue on ICT research in Austria, journal of electronics & information technology (e&i Elektrotechnik & Informationstechnik), January/February 2007.

[Ghioni et. al., 2007] A. Ghioni, D. Cerri, F. Corcoglioniti, J. Kopecky, G. Joskowicz, L. Nixon, D. Cerizza, N. P´erez Crespo, "Definition of security and trust support model for the reference architecture", EU FP6-02732 TripCom (Triple Space Communication) project deliverable (D5.2), March 2007.

[Haller et. al., 2005] A. Haller, E. Cimpian, A. Mocan, E. Oren, C. Bussler: WSMX - A Semantic Service-Oriented Architecture, International Conference on Web Services (ICWS 2005), Orlando, Florida, USA.

[Harth & Decker, 2005] A. Harth and S. Decker. Optimized Index Structures for Querying RDF from the Web. In G. Goos, J. Hartmanis, , and J. van Leeuwen, editors, in proceedings of the 3rd Latin American Web Congress. IEEE Press, November 2005.

[Krummenacher et. al., 2007] R. Krummenacher, E. Simperl, and D. Fensel: An Ontology-Driven Approach To Reflective Middleware. 2007 IEEE/WIC/ACM International Conference on Web Intelligence. Silicon Valley, USA, November 2-5, 2007.

[Murth et. al., 2007] M. Murth, G. Joskowicz, E. Kuehn, D. Cerizza, D. Cerri, D. de Francisco, A. Ghioni, R. Krummenacher, D. Martin, L. Nixon, N. Sanchez, B. Sapkota, O. Shafiq, D. Wutke, "Triple Space Reference Architecture", EU FP6-02732 TripCom (Triple Space Communication) project deliverable (D6.2), March 2007.

[Nixon et. al., 2008] L. Nixon, P. Obermeier, O. Shafiq, J. Saarela and H. Munoz, "Semantic matching in distributed spaces", EU FP6-02732 TripCom (Triple Space Communication) project

deliverable (D3.3), March 2008.

[Riemer et. al., 2006] J. Riemer, F. Martin-Recuerda, Y. Ding, M. Murth, B. Sapkota, R. Krummenacher, O. Shafiq, D. Fensel and E. Kuehn: Triple Space Computing: Adding Semantics to Space-based Computing. In proceedings of 1st Asian Semantic Web Conf., Beijing, China, September 3-7, 2006.

[Sapkota et. al., 2008] Brahmananda Sapkota, Vassil Momtchev, Omair Shafiq, "High-Performance Storage Implementation", EU FP6-02732 TripCom (Triple Space Communication) project deliverable (D1.3), March 2008.

[Shafiq et. al., 2007] O. Shafiq, M. Zaremba and D. Fensel: On communication and coordination issues of Semantic Web Services. In proceedings of IEEE International Conference Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA.

[Shafiq et. al., 2008] O. Shafiq, D. Cerizza, J. Kopecky, D. Martin, M. Murth, B. Sapkota, G. Toro del Valle, A. Turati and D. Wutke, "TripCom Grounding for Semantic Web Services", EU FP6-02732 TripCom (Triple Space Communication) project deliverable (D4.2), March 2008.

[Feier et. al., 2005] C. Feier, D. Roman, A. Polleres, J. Domingue, M. Stollberg, and D. Fensel: Towards Intelligent web Services: Web Service Modeling Ontology (WSMO), In proceedings of the International Conference on Intelligent Computing (ICIC) 2005, Hefei, China, August 23-26, 2005.

[de-Bruijn & Heymans, 2007] Jos de Bruijn and Stijn Heymans. A semantic framework for language layering in WSML. In proceedings of the 1st International Conference on Web Reasoning and Rule Systems (RR2007), pages 103–117, Innsbruck, Austria, June 7–8 2007. Springer.

[Kopecký et. al., 2007] Jacek Kopecký, Tomas Vitvar, Carine Bournez, Joel Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," IEEE Internet Computing, vol. 11, no. 6, pp. 60-67, Nov/Dec, 2007.

[Vitvar et. al., 2007] T. Vitvar, J. Kopecky, M. Zaremba, D. Fensel: WSMO-Lite: Lightweight Descriptions of Services on the Web. In proceedings of the 5th IEEE European Conference on Web Services (ECOWS), IEEE Computer Society, November, 2007, Halle, Germany.

[de Bruijn, 2008] J. de Bruijn (editor): "WSML/RDF", WSML Working Draft D32.2, 14 January 2008. Available at http://www.wsmo.org/TR/d32/v0.2

[de Bruijn et. al., 2005] J. de Bruijn (editor): "The Web Service Modeling Language WSML", WSML Final Draft D16.1, 5 October 2005. Available at http://www.wsmo.org/TR/d16/d16.1/v0.21

[Kopecký et. al., 2007] J. Kopecký, M. Moran, T. Vitvar, D. Roman, A. Mocan, "WSMO Grounding", WSMO Working Draft D24.2, 27 April 2007. Available at http://www.wsmo.org/TR/d24/d24.2/v0.1

[Tolksdorf et. al., 2006] R. Tolksdorf, E. Paslaru Bontas and L. Nixon: "A Co-ordination Model for the Semantic Web", ACM Symposium on Applied Computing (SAC 2006), April 2006, Dijon, France.

[Simperl et. al., 2007] E. Simperl, L. Nixon, R. Krummenacher, V. Momtchev, and H. Mu˜noz, “Representing RDF semantics in tuples:, TripCom project Deliverable D2.1, March 2007. Available at: http://tripcom.org/docs/del/D2.1.pdf.

[Sapkota et. al., 2007] B. Sapkota, A. Harth, Z. Zhou, V. Momtchev, O. Shafiq, "High-Performance Storage Implementation", TripCom project deliverable D1.3a, April 2007.

[Gelernter, 1985] D. Gelernter: Generative communication in linda. ACM Transactions on Programming Languages Systems, 7(1):80–112, 1985.

[Khushraj et. al., 2004] D. Khushraj, O. Lassila, T. Finin, "sTuples: Semantic Tuple Spaces,"

mobiquitous, pp. 268-277, First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), 2004.

[Tsoumakos et. al, 2003] D. Tsoumakos and N. Roussopoulos, a comparaion of peer to peer search methods, WebDB, 2003.

[Tsoumakos et. al, 2003b] D. Tsoumakos and N. Roussopoulos, Adaptive Probabilistic Search for Peer-to-Peer Networks, Networks, in 3rd IEEE Intl Conference on P2P Computing, 2003

[Stoica et. al., 2001] Ion Stoica and Robert Morris and David Karger and M. Frans Kaashoekand Hari Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for InternetApplications, ACM SIGCOMM '01 Conference, 2001.

[Ratnasamy et. al., 2001] Sylvia Ratnasamy and Paul Francis and Mark Handley and Richard M. Karp and Scott Shenker, A scalable content-addressable network, SIGCOMM, 2001.

[Rowstron et. al., 2001] A. Rowstron and P. Druschel, Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001.

[Yang et. al., 2006] Yang, Y. Dunlap, R. Rexroad, M. Cooper, B. F , Performance of Full Text Search in Structured and Unstructured Peer-to-Peer Systems, IEEE INFOCOM, 2006.

[Rdfstore] http://rdfstore.sourceforge.net/

[Rdfdb] http://guha.com/rdfdb/

[Nejdl, Siberski et. al., 2003] Wolfgang Nejdl and Wolf Siberski and Michael Sintek, Design Issues and Challengesfor RDF- and Schema-Based Peer-to-Peer Systems, SIGMOD Record, 32(3): 41-46, , 2003.

[Nejdl, Wolpers et. al., 2004] W. Nejdl and M. Wolpers and W. Siberski and C. Schmitz and M. Schlosser and I. Brunkhorst and A. Loser, Super-peer-based routing strategies for RDF based peer-to-peer networks, J. Web Sem., 1(2), 2004.

[Gu et. al., 2007] Tao Gu and Hung Keng Pung and Daqing Zhang, Information retrieval in schema-based P2P systems using one-dimensional semantic space, Computer Networks, 51,16, 2007.

[Cai et. al., 2004] M. Cai and M. Frank, M. Cai and M. Frank. RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network, International World Wide Web Conference (WWW), 2004.

[Triantafillou et. al., 2004] P. Triantafillou and I. Aekaterinidis, Content-Based Publish- Subscribe Over Structured P2P networks, DEBS, 2004.

[Anceaume et. al., 2006] Emmanuelle Anceaume and Maria Gradinariu and Ajoy Kumar Datta and Gwendal Simon and Antonino Virgillito, A Semantic Overlay for Self- Peer-to-Peer Publish/Subscribe, ICDCS, 22, 2006.

[Voulgaris et. al., 2006] Spyros Voulgaris and Etienne Rivi\`ere and Anne-Marie Kermarrec and Maarten van Steen, Sub-2-Sub: Self-Organizing Content-Based Publish Subscribe for Dynamic Large Scale Collaborative Networks, IPTPS, 2006.

[Chand and Felber, 2005] Raphaël Chand and Pascal Felber, Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks, Euro-Par 2005 Parallel Processing, Lisbon, Portugal, 2005.

[Dey, 2001] A. K. Dey, Understanding and Using Context. Personal Ubiquitous Computing, 2001. 5(1): p. 4-7.

[Pedrinaci et. al., 2008] C. Pedrinaci, N. Mehandjev, C. Ruiz Moreno: Contextual Service Adaptation Framework. SOA4All Deliverable 4.1.1. August 2008.

[Di Nitto et. al., 2008] E. Di Nitto, R. González-Cabero, C. Hamerling, J. Kopecký, O. Shafiq, T. Vitvar, L. Xu: Design Principles for a Service Web, Project Deliverable D1.1.1, EU FP7 SOA4All project, August 2008.

[WSDM-MOWS] I. Sedukhin, Web Services Distributed Management:Management of Web Services 1.0, OASIS Committee Draft, December 2004, available at: http://docs.oasis-open.org/wsdm/2004/12/cd-wsdm-mows-1.0.pdf

[MUWS Part 1] W. Vambenepe, Web Services Distributed Management:Management using Web Services (MUWS 1.0) Part 1, OASIS Committee Draft, December 2004, http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf

[MUWS Part 2] W. Vambenepe, Web Services Distributed Management:Management using Web Services (MUWS 1.0) Part 2, OASIS Committee Draft, December 2004, http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part2-1.0.pdf

[Shafiq et. al., 2008] O. Shafiq, D. Cerizza, J. Kopecky, D. Martin, M. Murth, B. Sapkota,G. Toro del Valle, A. Turati, D. Wutke, "TripCom Grounding for Semantic Web Services", EU FP6-02732 TripCom (Triple Space Communication) project deliverable (D4.2), March 2008.

[Haselwanter et. al., 2005] T. Haselwanter, M. Zaremba and M. Zaremba. Enabling Components Management and Dynamic Execution Semantic in WSMX. WSMO Implementation Workshop 2005 (WIW 2005), 6-7 June, Innsbruck, Austria.

[Vitvar et. al., 2007] T. Vitvar, A. Mocan, M. Kerrigan, Michal Zaremba, Maciej Zaremba, M. Moran, E. Cimpian, T. Haselwanter, D. Fensel: Semantically-enabled Service Oriented Architecture: Concepts, Technology and Application, In Journal of Service Oriented Computing and Applications, Springer London, 2007, ISSN:1863-2386.

[Dey, 2001] A.K. Dey, Understanding and Using Context. Personal Ubiquitous Computing, 2001. 5(1): p. 4--7.

[Hohpe & Woolf, 2003] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 2003.

[Papazoglou et. al., 2007] M. P. Papazoglou et al., Service-Oriented Computing: State of the Art and Research Challenges 40(11): p. 38--45, Computer, 2007.

[Engelmore & Morgan, 1988] R. S. Engelmore and A. J. Morgan, Blackboard Systems. The Insight Series in Aritificial Intelligence, ed. R.S. Engelmore and A.J. Morgan. 1988: Addison-Wesley.

[Pedrinaci, 2005] C. Pedrinaci, Knowledge-Based Reasoning over the Web. 2005.

[Nii, 1986] H. P. Nii, Blackboard Application Systems and a Knowledge Engineering Perspective. AI Magazine, 1986. 7(3): p. 82--106.

[Corkill, 2003] D. D. Corkill, Collaborating Software: Blackboard and Multi-Agent Systems \& the Future. In proceedings of the International Lisp Conference. 2003. New York City, NY, USA.

[Domingue et. al., 2008] J. Domingue, D. Fensel, and R. González-Cabero: SOA4All, Enabling the SOA Revolution on a World Wide Scale. In proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008), August 4-7, 2008, Santa Clara, CA, USA.

Acknowledgements

Authors of this deliverable would like to acknowledge Reto Krummenacher from Semantic Technology Institute (STI) Innsbruck, and Brahmananda Sapkota from Digital Enterprise Research Institute (DERI) Galway for their review and comments.