



# NESSI Semantic Technologies Working Group

*Research Roadmap 2007-2010*

Version 1.2 10<sup>th</sup> June 2007

## **Editors**

Michal Zaremba, Leopold-Franzens-Universität Innsbruck, DERI Innsbruck, Austria

Tomas Vitvar, National University of Ireland, DERI Galway, Ireland

Dieter Fensel, Leopold-Franzens-Universität Innsbruck, DERI Innsbruck, Austria

John Davies, BT, UK

Marc Richardson, BT, UK

Tomas Pariente, ATOS, Spain

## **Abstract**

This document defines the research roadmap for the Semantic Technologies working group within NESSI European Technology Platform for the period of 2007-2010. This roadmap is focused on goals to be addressed within the Service Oriented Architecture (SOA) research and development with aim to resolve fundamental challenges related to open SOA environment. We expect that in the near future a service-oriented world will consist of an "uncountable" number of services. The computation will involve services searching for services based on functional and non-functional requirements and an interoperating with those that they select. Services will not be able to interact automatically and existing SOA solutions will not scale without significant mechanization of service provisioning process. Hence, machine processable semantics are critical for the next generation of service-oriented computing to reach their full potential. The goal is to define methods, algorithms and tools forming a skeleton of the Semantically-enabled Service Oriented Architecture (SESA) introducing automation to the service provisioning process including service discovery, negotiation, adaptation, composition, invocation, and monitoring as well as service interaction requiring data and process mediation. SOA outside of tightly controlled environment cannot succeed until semantic issues will be addressed and critical tasks within

the service provisioning process will be automated leaving humans to focus on higher level problems.

## Table of Contents

1. Introduction .....	4
2. Governing Principles .....	5
3. SESA Vision.....	6
Stakeholders Layer .....	6
Problem-Solving Layer .....	7
Service Requesters Layer .....	7
Middleware Layer .....	8
Service Providers Layer.....	9
Services in SESA .....	9
4. Scope of Research Roadmap.....	10
5. Research Areas and Goals.....	11
1 – Ontologies.....	11
2 – Applications.....	13
3 – Developer Tools .....	13
4 – Discovery.....	15
5 – Adaptation .....	16
6 – Composition .....	17
7- Choreography .....	17
8 – Mediation.....	18
9 – Grounding.....	19
10 – Fault Handling .....	20
11 – Monitoring.....	20
12 – Formal Languages .....	21
13 – Reasoning .....	22
14 – Storage and Communication .....	23
15 – Execution Management.....	24
16 – Security .....	24
6. Summary.....	24

## 1. Introduction

The most important issue in today's design of software architectures is to satisfy increasing software complexity as well as new IT needs, such as the need to respond quickly to new requirements of businesses, the need to continually reduce the cost of IT or the ability to integrate legacy and new emerging business information systems. In the current IT enterprise settings, introducing a new product or service and integrating multiple services and systems present unpredicted costs, delays and difficulty. Existing IT systems consist of a patchwork of legacy products, monolithic off-shelf applications and proprietary integration. It is even today's reality that in many cases users on the "spinning chairs" manually re-enter data from one system to another within the same organization. The past and existing efforts in Enterprise Application Integration (EAI) don't represent successful and flexible solutions. Several studies showed that the EAI projects are lengthy and the majority of these efforts are late and over budget. It is mainly costs, proprietary solutions and tightly-coupled interfaces that make EAI expensive and inflexible.

Service Oriented Architecture (SOA) solutions are the next evolutionary step in software architectures. SOA is an IT architecture in which functions are defined as independent services with well-defined, invocable interfaces. SOA will enable cost-effective integration as well as bring flexibility to business processes. In line with SOA principles, several standards have been developed and are currently emerging in IT environments. In particular, Web Services technology provides means to publish services in a UDDI registry, describing their interfaces using the Web Service Description Language (WSDL) and exchanging requests and messages over a network using SOAP protocol. The Business Process Execution Language (BPEL) allows composition of services into complex processes as well as their execution. Although Web services technologies around UDDI, SOAP and WSDL have added a new value to the current IT environments in regards to the integration of distributed software components using web standards, they cover mainly characteristics of syntactic interoperability. With respect to a large number of services that will exist in IT environments in the inter and intra enterprise integration settings based on SOA, the problems of service discovery or selection of the best services conforming user's needs, as well as resolving heterogeneity in services capabilities and interfaces will again be a lengthy and costly process. For this reason, machine processable semantics should be used for describing services in order to allow total or partial automation of tasks such as discovery, selection, composition, mediation, invocation and monitoring of services.

In [1], the vision of *serviceware* as the next natural step beyond hardware and software is introduced: *"After four decades of rapid advances in computing, we are embarking on the greatest leap forward in computing that includes revolutionary changes at all levels of computing from the hardware through the middleware and infrastructure to applications and more importantly in intelligence"*. There we refine

this towards a comprehensive framework which integrates two complimentary and revolutionary technical advances, namely Service-Oriented Architectures (SOA) and Semantic Web, into a single computing architecture, that we call Semantically-enabled Service Oriented Architecture (SESA). While SOA is widely acknowledged for its potential to revolutionize the world of computing, that success is dependent on resolving two fundamental challenges that SOA does not address, namely integration, search and mediation. In a service-oriented world, millions of services must be discovered and selected based on requirements, then orchestrated and adapted or integrated. SOA depends on but does not address either search or integration.

The combination of semanticx and Web 2.0 technologies is another important research area and we touch on this in Appendix 1 of this document. It should be noted that the User Service Interactions NWG is also looking at this area.

## 2. Governing Principles

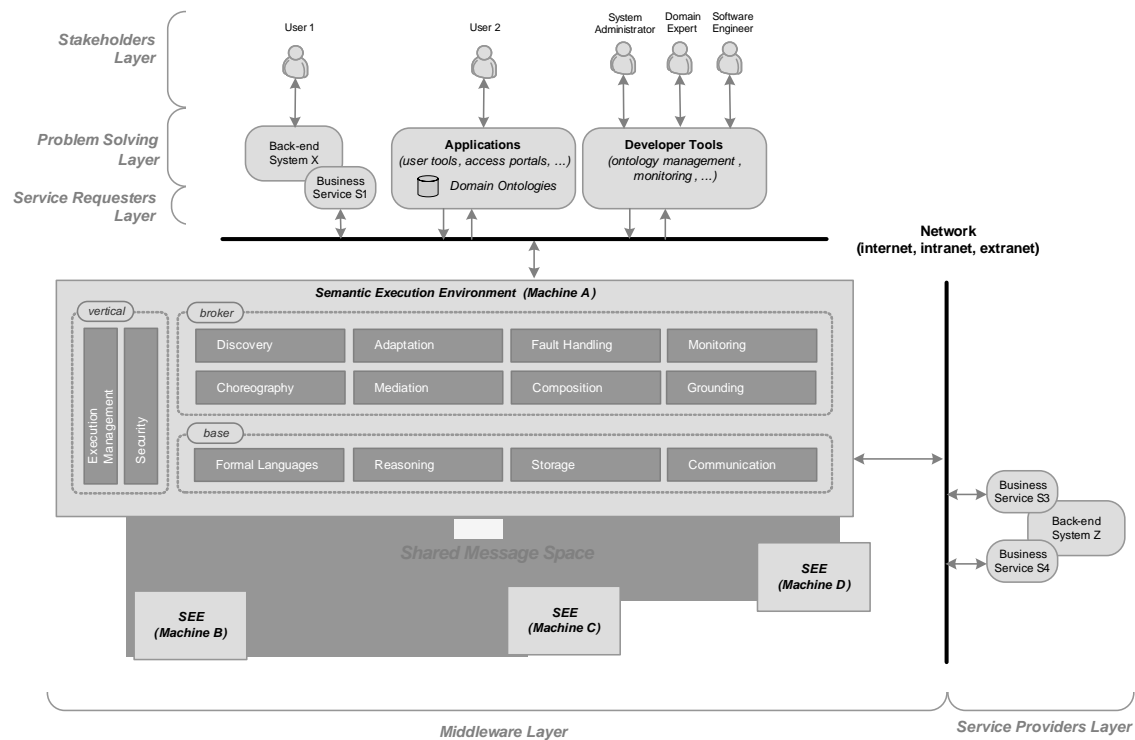
For purposes of the roadmap following principles have been identified which define essential background knowledge governing the architecture research, design and implementation. These principles reflect fundamental aspects for service-oriented and distributed environment which all promote intelligent, user-centric and seamless integration and provisioning of business services. These principles include:

- **Service Oriented Principle** represents a distinct approach for analysis, design, and implementation which further introduces particular principles that govern aspects of communication, architecture, and processing logic. This includes service reusability, loose coupling, abstraction, composability, autonomy, and discoverability.
- **Semantic Principle** allows a rich and formal description of information and behavioral models enabling automation of certain tasks by means of logical reasoning. Combined with service oriented principle, semantics allows to define scalable, semantically rich and formal service models and ontologies allowing to promote total or partial automation of tasks such as service discovery, contracting, negotiation, mediation, composition, invocation, etc.
- **Problem Solving Principle** reflects Problem Solving Methods as one of the fundamental concepts of the artificial intelligence. It underpins the ultimate goal of the architecture which lies in so called *goal-based discovery and invocation of services*. Users (service requester's) describe requests as goals semantically and independently from services while architecture solves those goals by means of logical reasoning over goals' and services' descriptions. Ultimately, users do not need to be aware of processing logic but only care about the result and its desired quality.
- **Distributed Principle** allows aggregating the power of several computing entities to collaboratively run a task in a transparent and coherent way, so that from service requester's perspective they can appear as a single and centralized system. This principle allows executing a process across a number of components/services over the network which in turn can promote scalability and quality of the process.

- **User-Centric Principle** puts the user into the centre of the architecture perspective. With respect to problem solving principle, the user-centric approach promotes one-stop-shopping when user is aware of his/her needs while architecture provides a complete solution for these needs.

### 3. SESA Vision

The global view on the architecture, depicted in Figure 1, comprises of several layers, namely (1) *Stakeholders* forming several groups of users of the architecture, (2) *Problem Solving Layer* building the environment for stakeholders access to the architecture, (3) *Service Requesters* as client systems of the architecture, (4) *Middleware* providing the intelligence for the integration and interoperation of business services, and (5) *Service Providers* exposing the functionality of back-end systems as Business Services. In this section we describe these layers and define service types in the architecture.



**Figure 1: Global View on SESA Architecture**

#### Stakeholders Layer

Stakeholders form the group of various users which use the functionality of the architecture for various purposes. Two basic groups of stakeholders are identified: *users*, and *engineers*. Users form the group of those stakeholders to which the architecture provides end-user functionality through specialized applications. For example, users can perform electronic exchange of information to acquire or provide products or services, to place or receive orders or to perform financial

transactions. In general, the goal is to allow users to interact with business processes on-line while at the same time reduce their physical interactions with back-office operations. On the other hand, the group of engineers forms those stakeholders which perform development and administrative tasks in the architecture. These tasks support the whole SOA lifecycle including service modeling, creation (assembling), deployment (publishing), and management. Different types of engineers could be involved in this process ranging from domain experts (modeling, creation), system administrators (deployment, management) and software engineers.

### **Problem-Solving Layer**

The problem solving layer contains applications and tools which support stakeholders during formulation of problems/requests and generates descriptions of such requests in the form of user goals. Through the problem solving layer, user will be able to solve his/her problems, i.e. formulate a problem, interact with the architecture during processing and get his/her desired results. This layer contains *back-end systems* which directly interface the middleware within business processes, specialized *applications* built for specific purpose in a particular domain, which also provide specific domain ontologies, and *developer tools* providing functionality for development and administrative tasks within the architecture.

*Developer tools* provide a specific functionality for engineers, i.e. domain experts, system administrators and software engineers. The functionality of developer tools cover the whole SOA lifecycle including service modeling, creation (assembling), deployment (publishing), and management. The vision is to have an Integrated Development Environment (IDE) for management of the architecture. It should aid the developers through the development process including *Engineering of Semantic Descriptions (services, goals, and ontologies)*, *Creation of Mediation Mappings*, *Interfacing with Architecture Middleware and External Systems*. By combining this functionality, a developer will be allowed to create and manage ontologies, Web services, goals and mediators, create ontology to ontology mediation mappings and deploy these mappings to the middleware.

*Applications* provide a specialized functionality for architecture end-users. They provide a specialized domain specific ontologies, user interfaces and application functionality through which stakeholders interact with the architecture and its processes. Through specialized applications in particular application settings, the technology and its functionality is validated and evaluated.

### **Service Requesters Layer**

Service requesters act as client systems in client-server settings of the architecture. They are represented by Goals created through problem/request formulation by which they describe requests as well as interfaces through which they wish to perform conversation with potential services. Service requesters are present for all applications and tools from problem solving layer and are bound to specific service semantics specification.

## Middleware Layer

Middleware is the core of the architecture providing the main intelligence for the integration and interoperation of Business Services. For purposes of the SESA, we call this middleware Semantic Execution Environment (SEE). The architecture defines the necessary conceptual functionality that is imposed on the architecture through the underlying principles defined in section 2. Each such functionality could be realized (totally or partially) by a number of so called *middleware services* (see section Services in SESA). This functionality is further distinguished in the following layers: *base layer*, *broker layer*, and *vertical layer*.

### Vertical Layer

The Vertical layer defines the middleware framework functionality that is used across the Broker and Base Layers but which remains invisible to them. This technique is best understood through so called "Hollywood Principle" that basically means "Don't call us, We'll call you". With this respect, framework functionality always consumes functionality of Broker and Base Layers, coordinating and managing overall execution processes in the middleware. For example, Discovery or Data Mediation is not aware of the overall coordination and distributed mechanism of the Execution Management.

- **Execution Management** defines a control of various execution scenarios (called execution semantics) and handles distributed execution of middleware services.
- **Security** defines a secure communication, i.e. authentication, authorization, confidentiality, data encryption, traceability or non-repudiation support applied within execution scenarios in the architecture.

### Broker Layer

The Broker layer defines the functionality which is directly required for a goal based invocation of Semantic Web Services. The Broker Layer includes:

- **Discovery** defines tasks for identifying and locating of business services which can achieve a requester's goal.
- **Choreography** defines formal specifications of interactions and processes between the service providers and client
- **Monitoring** defines a monitoring of the execution of end point services, this monitoring may be used for gathering information on invoked services e.g. QoS related or for identifying faults during execution.
- **Fault Handling** defines a handling of faults occurring within execution of end point Web services.
- **Adaptation** defines an adaptation within particular execution scenario according to users preferences (e.g. service selection, negotiation, contracting).
- **Mediation** defines interoperability at the functional, data and process levels.
- **Composition** defines a composition of services into an executable workflow (business process). It also includes orchestration, which defines the execution of

a composite process (business process) together with a conversation between a service requester and a service provider within that process.

- **Grounding** defines a link between semantic level and a non-semantic level (e.g. WSDL) used for service invocation.

### Base Layer

The Base layer defines functionality that is not directly required in a goal based invocation of business services however they are required by the Broker Layer for successful operation. Base layer includes:

- **Formal Languages** define semantic languages used for semantic description of services, goals and ontologies.
- **Reasoning** defines reasoning functionality over semantic descriptions.
- **Storage and Communication** defines persistence mechanism for various elements (e.g. services, ontologies) as well as inbound and outbound communication of the middleware.

The SEE middleware can operate in a distributed manner when a number of middleware systems connected using a shared message space operate within a network of middleware systems and empowering this way a scalability of integration processes. The SEE consists of several decoupled services allowing independent refinement of these services – each of them can have its own structure without hindering the overall SEE architecture. Following the SOA design principles, the SEE architecture separates concerns of individual middleware services thereby separating service descriptions and their interfaces from the implementation. This adds flexibility and scalability for upgrading or replacing the implementation of middleware services which adhere to required interfaces.

### Service Providers Layer

Service providers represent various back-end systems. Unlike back-end systems in service requesters layer which act as clients in client-server setting of the architecture, the back-end systems in service providers layer act as servers which provide certain functionality for certain purpose exposed as a business service to the architecture. Depending on particular architecture deployment and integration scenarios, the back-end systems could originate from one organization (one service provider) or multiple organizations (more service providers) interconnected over the network (internet, intranet or extranet). The architecture thus can serve various requirements for Business to Business (B2B), Enterprise Application Integration (EAI) or Application to Application (A2A) integration. In all cases, functionality of back-end systems is exposed as semantically described business services.

### Services in SESA

While some of SESA functionality is provided as services, the others remain the entities required to let the overall system to function – they are not services in terms of a Service oriented Architecture. While the Middleware and Service requester's layers build SESA architecture in terms of services (with some exceptions), the

Problem Solving layer adds the set of tools and entities which makes SESA a fully-fledged Semantic Service Oriented Architecture.

The core aspect of SOA is the service. With this respect, we distinguish two types of services in SESA, namely *middleware services* and *business services*.

- **Middleware** services are necessary to enable particular functionality of the architecture – they are the main facilitators for integration, search, and mediation of business services.
- **Business Services** are exposed by service providers, their back-end systems which are external to SESA. Business services are subject of integration and interoperation within the architecture (facilitated by the middleware services) and usually provide a certain value for architecture stakeholders. Such services are in SESA architecture semantically described conforming to a specific semantic service model.

With this respect, the SESA defines **the scope of particular middleware services** in terms of the functionality they should provide. In addition, the SESA defines a **semantic service model for the business services** on which the SESA operates. Particular business services are however subject of modeling in application-oriented scenarios. For this purpose, domain-specific ontologies can be designed as part of SESA application design or evaluation. With respect to middleware services and business services distinction, the SEE middleware is designed as the Service Oriented Architecture on its own. However, it is designed as the facilitator for the integration of semantic business services and as such is not currently considered as *semantically-enabled* but rather as *semantic-enabling*.

#### 4. Scope of Research Roadmap

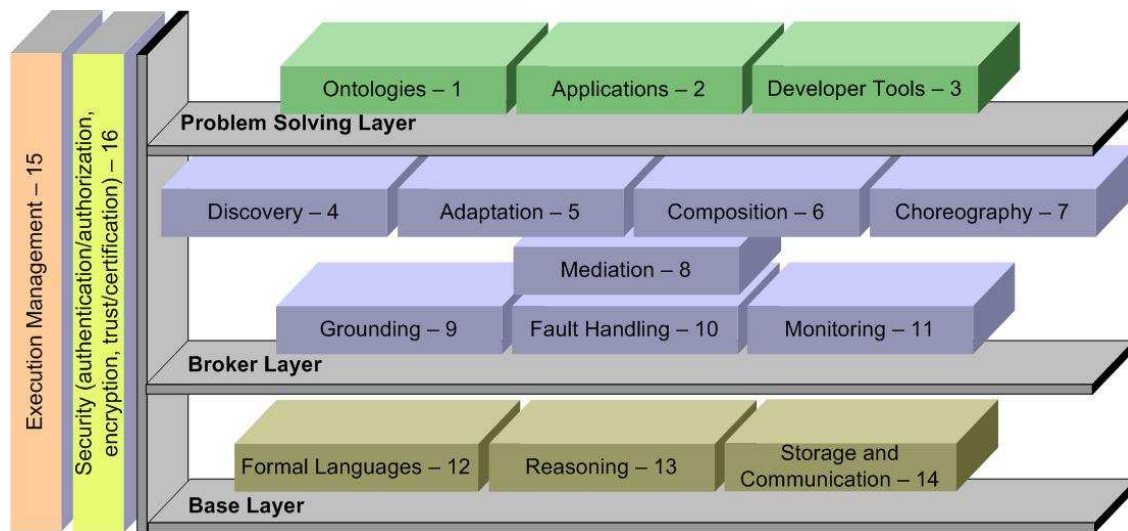
With respect to the architecture vision described in section 3, we define the scope of **research roadmap for the period of 2007-2010**. Research roadmap is defined in a number of research areas, each having defined its goals for the period of the roadmap. Each research goal usually combines major research challenges in Semantic Web Services and SESA together with an implementation effort related to it. Research area and its goals have a corresponding architecture component. Thus, based on the architecture vision and the global view, we identify following research areas as *architectural components* which are further distinguished in layers as depicted in Figure 2.

- The *problem-solving layer* consisting of (1) **Ontologies**, (2) **Applications** (e.g., e-tourism, e-buiness, e-government) and (3) **Developer tools** (GUI tools such as those for engineering ontology/web service descriptions; generic developer tools such as language APIs, parsers/serializes, converters, etc.).
- The *broker layer* which consists of (4) **Discovery**, (5) **Adaptation** (including selection and negotiation), (6) **Composition** (web service composition techniques such as planning), (7) **Choreography**, (8) **Mediation** ((a) Ontology mediation: techniques for combining Ontologies and for overcoming differences between Ontologies; (b) Process mediation: overcoming

differences in message ordering, etc.), (9) **Grounding**, (10) **Fault Handling** (Compensation, etc.), and (11) **Monitoring**.

- The *base layer* that is providing the exchange formalism used by the architecture, i.e., (12) **Formal languages** (static ontology and behavioral, i.e., capability/choreography/orchestration languages, connection between higher-level descriptions), (13) **Reasoning** (techniques for reasoning over formal descriptions; LP, DL, FOL, behavioral languages, etc.) and (14) **Storage and Communication**.
- *Vertical services* such as (15) **Execution management** and (16) **Security** (authentication/authorization, encryption, trust/certification).

Each of these components forms a research area for which further goals are identified in following sections of the report.



**Figure 2: SESA Architecture Components**

## 5. Research Areas and Goals

This section describes in more detail research areas which as a result build the functional components that play a role in the SESA architecture. For each research area, the description of the area and the goals are described.

### 1 – Ontologies

Ontologies are *community contracts about a representation of a domain of discourse*. Representation in here includes (1) formal parts that can be used for machine reasoning, and (2) informal parts like natural language descriptions and multimedia elements that help humans establish, maintain, and renew consensus about the meaning of concepts. Ontologies as formal representations of a domain have been proposed for quite a long time as a cure to interoperability problems and problems of application integration, and the Semantic Web community has made a

lot of progress in developing stable infrastructure and standardized languages for the representation of ontologies. Also, impressive tools and validated methodologies are available. However, a major bottleneck towards business applications of Semantic Web technology and machine reasoning is the lack of industry-strength ontologies that go beyond academic prototypes. The design of such ontologies from scratch in a textbook-style ontology engineering process is in many cases unattractive, for it would require significant effort, and because the resulting ontologies could not build on top of existing community commitment. Also, real-world problems of data and systems interoperability can only be overcome using Semantic Web technology if ontologies exist that represent the very standards currently in use in systems and databases. Such standards, though mostly informal in nature, are likely the most valuable asset on the way to real business ontologies that can help solve real business interoperability problems, since they reflect some degree of community consensus and contain, readily available, a wealth of concept definitions. However, the transformation of such standards into useful ontologies is not as straightforward as it appears

### Goals and Tasks

- **Maturing Semantic Web Foundations**, so that they become compatible with the real world complexity and scale. In particular, the social interaction and economic dimension of ontologies must be defined.
  - *Ontology Engineering* – Methodologies for and prototypes of industry-strength business ontologies, e.g. the gen/tax methodology for deriving ontologies from existing hierarchical standards and taxonomies (UNSPSC, eCl@ss, ...) and eClassOWL, the first serious attempt of building an ontology for e-business applications; and in general advancing the state of the art in e-business data and knowledge engineering, including metrics for content.
  - *Community-driven Ontology Building* – Methods and techniques for evolution of ontologies managed by user community, including semi-automated approaches and OntoWiki – a Wiki-centric ontology building environment.
  - *Economic Aspects of Ontology Building and Usage* – Building ontologies consumes resources, and in an economic setting, these resources are justified and will be spent (by rational economic actors, at least) only if the effort needed to establish and keep alive a consensual representation of a domain of discourse is outweighed by the business gain, either in terms of cost, added value, or strategic dimensions, e.g. process agility. This research branch fuels the use of ontologies in business applications.
- **Building ontologies for core challenges of Information Systems** in order to realize and evaluate the business benefits and to identify the open research challenges.
  - *Application ontologies* – ontologies developed for particular domain such as in e-business, e-government, e-tourism, etc. This includes semantics-supported Business Process Management for mechanization of business process management, ontology-supported electronic

procurement and analysis of the true complexity of business matchmaking, financial reporting, etc.

## 2 – Applications

All the activities gathered around development of SESA framework must be tightly coupled to the development and the implementations of the use cases proving the viability of SESA framework. There are many technologies in the area of semantic Web Services mainly presented in academic workshops and conferences where various use cases are being defined in alienation. In addition, there does not exist any unified methodology which could be used for comparing these technologies and more importantly, there is no way for industry to evaluate the robustness, applicability and added values of these technologies. Therefore, progress in scientific development and in industrial adoption is thereby hindered.

### Goals and Tasks

It is the goal of Applications to analyze processes, infrastructures, and results of existing real-world scenarios, develop a standard set of problems and a public repository for such problems. In addition, the goal is to develop and standardize a community-agreed evaluation methodology.

- **Define the set of standard problems and their levels** - Developing a methodology for evaluating the functionality (versus performance) of semantic service technologies. Based upon our experience with a group working on SWS challenge initiative<sup>1</sup>, Applications aim to get involved into a larger community through the W3C Test-bed incubator group.
- **Support of scalable public collaborative development of new problem scenarios and associated services** – In this development, the aim is to standardize the methodology and the infrastructure.
- **The standard methodology for peer-review of solutions** - The Applications should refrain from recommending technologies or providing solutions to the semantic web service problems. The focus should be on standardizing the evaluation methodology.

## 3 – Developer Tools

An Integrated Development Environment (IDE) is defined as *a type of computer software that assists computer programmers to develop software* and such IDEs like the Eclipse Java Development Toolkit (JDT)<sup>2</sup> or the NetBeans IDE<sup>3</sup> for developing software in the Java programming language have proven that the productivity of the Java developer can be improved by providing all the tools required by the developer side by side and integrated with one another. The breadth of the field of semantics means that one IDE for all the different languages and technologies is

---

<sup>1</sup> <http://www.sws-challenge.org>

<sup>2</sup> <http://www.eclipse.org/jdt/>

<sup>3</sup> <http://www.netbeans.org>

unlikely to happen, however using technologies like the Eclipse<sup>4</sup> platform will allow for the developer to place individual tools or indeed individual IDEs together in order to build the IDE with the tools they need to perform the job at hand. A good example of where such a combination of IDEs would be useful is the Semantic Web Service field, the Eclipse Web Tool Platform (WTP)<sup>5</sup> provides a collection of tools for simplifying the process of building Web service based applications, combining the WTP with an IDE for describing Web services semantically. With semantic to become more centric to modern computer science, many different combinations of tools that at this stage cannot even be contemplated will be required. The flexibility of platforms like Eclipse gives a form of future proofing and puts the design and scale of the resulting IDE into the hands of the user. When describing tools for semantic technologies it is very easy to become focused primarily on ontologies and to forget that there are many different technologies that require tool support. Research topics like Semantic Web Services and Semantic Business Processes are producing many forms of semantic description that must be created by some developer in order to deploy such technologies. Only now are developers of semantic descriptions receiving limited tool support that allow them to focus on the problem at hand and stop grappling with low level problems like syntax, testing and deployment of their descriptions. Within the scope of this roadmap the aim is to provide guidelines for the types of tools that should exist within an IDE for a given semantic technology.

### Goals and Tasks

- **Creation and Maintenance** – tool support must be available for creating the actual descriptions themselves. It is important that users of different skill levels are supported within the IDE, thus editing support at different levels of abstraction should be provided. Some users may be very comfortable with dealing with textual semantic descriptions, while others may require more visual paradigms for creating descriptions. These different levels of abstraction can also benefit the skilled engineer. Considering ontologies, it may be more convenient for the engineer to create an ontology using a textual representation within a text editor and then to use a graph based ontology engineering solution to learn more about the ontology and tweak the model that has been created.
- **Validation** – The most common problem that occurs when creating semantic descriptions is incorrect modeling. It can be very easy for an engineer to make a mistake without any tool support. Validation of semantic descriptions is a non trivial task and validation at both the syntactic and semantic levels can vastly reduce the time a developer spends debugging their descriptions. By syntactic validation we mean checking that the actual syntactic structure of the semantic description is correct and by semantic validation we refer to checking that syntactically correct descriptions are semantically valid.
- **Testing** – Once valid semantic descriptions exist the engineer needs to ensure that they behave in the expected manner in their intended environment prior to deploying them. Having testing integrated into the development environment reduces the overhead of the user performing a lengthy,

---

<sup>4</sup> <http://www.eclipse.org>

<sup>5</sup> <http://www.eclipse.org/webtools/>

iterative, deploy-test scenario. The engineer will more than likely perform a deploy-test scenario anyway, but having an initial cycle within the development environment can significantly reduce the length of this cycle and the time taken to perform it.

- **Deployment** – Ultimately the descriptions created within the development environment must be used in some run-time system. Deploying descriptions can also be a huge overhead on the engineer and having tool support in an IDE can prevent mistakes occurring at this crucial stage of the process.

## 4 – Discovery

Within a service-oriented-architecture the discovery of services is the essential building block for creating and utilizing dynamically created applications. However, current technologies only provide means to describe service interfaces on a syntactical level, providing only limited automation support. Existing solutions for Service Discovery include UDDI, a standard that allows programmatically publishing and retrieving a set of structured information belonging to a Web Service, however it allows to retrieve services by pre-defined categories and keywords, not by their actual semantic properties. There is a lack of means that allow describing functional and non-functional properties of a service on a semantic level. Only with such descriptions a precise discovery is possible.

Given a description of a Service, the problem of discovering a desired service can be seen as information retrieval problem, that uses keywords to express the desire and uses a document index to match them. However for mechanizing the discovery task a more fine grained approach to discovery is required, e.g. to restrict the search space along specific parameters like location, provider, price, etc. It can be seen as a search for semi structured entity. Here approaches developed in the context of the Semantic Web, in particular the use of ontologies are a promising approach.

### Goals and Tasks

- **Service and Domain Ontologies** – In order to provide a semantic discovery service both service requests and offers need to be described on a semantic level. While there exist some proposals for upper level ontologies like WSMO and OWL-S, we need to refine them and provide guidelines for their usage and accompanying domain ontologies.
- **Language and Reasoning integration** – Potentially many different logical formalism can be used to annotate services. Each comes with a specific trade off between expressivity and computational complexity. It has to be investigated for which use cases a particular formalism is suitable. In addition the reasoning engine for a particular formalism needs to be integrated into the discovery context, such that its usage becomes transparent to the user of a discovery engine.
- **Non-Functional Properties** – Research around service discovery has so far paid much attention to the specification of the functional properties of a service, however only little effort has been spent on the investigation of the usage of non-functional properties within the discovery process. Specific ontologies and matchmaking techniques have to be developed in order to allow a semantic retrieval on non-functional properties.
- **Field Deployment and Verification of Existing Discovery Strategies** – While many concrete formalism have been proposed only a few case studies have been performed to validate the appropriateness of a particular approach.

Further real world use cases are required in order to adopt the existing semantic discovery approaches for the practical needs.

## 5 – Adaptation

After discovering a set of potentially useful Web services, a semantic user agent needs to find out the concrete offers available at the Web services and relevant to the user goal, generally by communicating with the Web service or with its provider. This process filters out the discovered Web services that cannot fulfill the goal. This step is required as it is not feasible for a Web service to provide an exhaustive semantic description of all its potential offers. The process of checking whether and under what conditions a service can fulfill a concrete Goal is called negotiation in SESA. The results of negotiation are filtered using the functionality of the Discovery component to consider only the services that have the appropriate functionality and also non-functional properties acceptable by the user. Filtering is followed by building a ranking/order relation based on non-functional properties criteria like price, availability, etc. Once a list of Web services that can fulfill the user's concrete goal is prepared, a SESA must then choose one of the services to invoke. It is important that this selection is tailored to the user's needs, as for example while one user may require high quality, another may prefer low price. This process is called selection. *Negotiation*, *ranking* and *selection* are tasks of the Adaptation working group.

### Goals and Tasks

- **Semantic and Multi-criteria based ranking and selection** – Ranking and selection of services could be done along more than one dimension. For example users might be interested in the cheapest and fastest service providing a certain functionality. A ranking and selection solution which uses semantic descriptions of multiple non-functional properties and ranks the services based on their attached logical expressions non-functional properties representations should be developed. This involves Development of semantic and multi-criteria ranking algorithms and Design and implementation of a semantic and multi-criteria ranking component.
- **Context-based ranking and selection** – Service ranking and selection must consider contextual information in order to provide relevant results. The goal is to develop models and algorithms for context-aware ranking and selection. This involves Development of context ranking algorithms.
- **Social-based ranking and selection** – Service ranking and selection remains an open and controversial problem in terms of both social and technical aspects. From a social point of view an honest and fair mechanism is required. An aspect which might be useful especially for ranking services is “social” aspect of consuming services. Previous customers which have used a service could provide feedback about the service. Furthermore not only users but group of users, communities can be used to “compute” the ranking values of services.
- **Negotiation Algorithms** – negotiation support with use of communication to establish details of Web service offers relevant to user's needs.

## 6 – Composition

Composition involves methods for Web Service composition (WSC), starting from web service descriptions at various levels of abstraction, specifically, the functional level and process level. The WSC area is yet in a very early stage of its development. Several techniques have been proposed, mostly based on AI Planning or on logical deduction, but all of those still have severe shortcomings. The existing techniques either: (I) largely or even completely ignore the constraints given in the background ontology in which the web services are specified; (II) largely or even completely ignore the complex inner behavior and interfaces of web services; (III) have severe scalability problems, solving only toy problems with few services; or suffer from several of these deficiencies.

### Goals and Tasks

- **Development of a scalable tool for WSC with powerful background ontologies and partial matches** – the goal is to overcome the lack of a technique that combines adequate treatment of background ontologies (I) with scalability (III). This will be achieved by building on logics-based search space representations and heuristic functions originating in the area of AI Planning.
- **Development of a scalable tool for WSC with plug-in matches, dealing with Business Policies** – the goal is to overcome the lack of a combination of (I) and (III), with a particular focus on Business Process Management (BPM) scenarios. In those, scalability is particularly urgent, since enterprises deal with thousands of services from which the composed service should be combined. Further, business policies -- rules governing how services can be executed within or between enterprises -- are of paramount importance. Scalability, even in the presence of business policies, will be achieved by exploiting plug-in matches rather than partial matches, and by exploiting the typical forms of ontologies occurring in practice.
- **Integration of techniques for functional-level and process-level WSC** – the goal is to overcome the lack of a combination of (II) and (III), and potentially to build technology that overcomes all the three deficiencies (I) -- (III). The idea is to combine techniques from functional-level and process-level WSC, first by establishing ways for their interplay, later by integrating their underlying core principles. Regarding their interplay, functional-level and process-level WSC essentially provide different trade-offs between accuracy and computational cost; they can be combined to mutually profit from each other's benefits. Regarding the underlying principles, the most effective process-level composition methods today are based on binary decision diagrams; this will be replaced with the logics-based search space representation underlying advanced functional-level composition. This approach allows for more flexibility and can be used to model and take into account also the background ontology, while at the same time reducing computational costs through consequent exploitation of problem structure.

## 7- Choreography

Techniques for service choreography play a key role in creation of new opportunities for collaborations between service requesters and providers, and thus for creation of new services. The Choreography part of SESA defines formal specifications of

interactions and processes between the service providers and clients. Current approaches to service choreography languages have been criticized for being too procedural-oriented. With the move towards service-orientation, where entities are autonomous and need to agree on the collaborations between them, where no central point of control might exist, a more declarative modeling style for interactions is required (i.e. "what" without having to state the "how"). Moreover, reasoning techniques for such a language that would enable a flexible and dynamic integration of service requesters and providers in a collaborative environment are currently missing.

### Goals and Tasks

- **Declarative choreography language.** The goal here is to develop a declarative process language which should allow for formal specifications of interactions and processes between the service providers and clients. Such a declarative language would enable non-IT experts to easily represent service behaviors and interactions, enabling a more flexible way of engaging in new collaborations.
- **Reasoning tasks for choreography.** The goal here is to define reasoning tasks that should be performed using the declarative language. Verification techniques such as contracting or enactments are examples of reasoning tasks. Such techniques will enable an automated, flexible, and dynamic integration of service requesters and providers in a collaborative environment.
- **Tool support for choreography.** The goal here is to implement an engine to support the execution of interactions, as well as to support reasoning in the proposed declarative language.

## 8 – Mediation

Heterogeneity problems and mediators have been intensively investigated in the last decade but still the key solution that would enable the decisive leap towards automation is yet to be found. Semantics is changing the problem specifications and service orientation paradigms offers new ways of designing, deploying and using mediators while in the same time, poses new challenges and sets new requirements. That is, data and processes can be formally and unambiguously described while services and service oriented architectures allow the development of decoupled, transparent and flexible software components, including mediators.

### Goals and Tasks

- **Advanced support for data mediation.** Semi-automatic design-time tools should be developed in order to allow domain experts to identify and capture the heterogeneity problems between different models of overlapping domains. Special attention will be given to user profiles and expertise levels in order to separate the tools for trained domain experts and the tools designated for casual users of ontologies. Furthermore, at this level, alignments between various models will be part of a community validation process where users can add and remove links between the models in order to achieve and maintain agreed upon interlinked models.

- **Advanced support for process mediation.** Heterogeneity appears on the process level as well no matter if these processes are enterprise internal processes or public processes used in describing the visible behavior of particular services. Such heterogeneous processes need to be part of collaborative scenarios that can range from simple peer-to-peer interaction to complex compositions. Semi-automatic tool support allowing the tailoring of such process in order to overcome the heterogeneity problems should be provided. Further more such tools should support annotation of existing process representation standards with semantics based on ontological domain models.
- **Service Mediation by Mediation Services.** Mediator systems able to resolve specific types of heterogeneity problems should be encapsulated and deployed as mediation services. Such services should be developed for well defined mediation scenarios while preserving the generality of their offered functionality.
- **Semantic descriptions for the mediation services and mediation libraries.** Mediation services should be semantically described as any other resources. In this way their functionality can be properly advertised and their intended usage explicitly stated. Furthermore, they can become part of intelligent mechanism for service discovery, composition and invocation. Additionally, such semantically described mediation services will be organized in semantic mediation patterns that can be directly applied on complex heterogeneity scenarios. In addition, such services should be organized in libraries supporting intelligent mediation service retrieval (by providing customizable mediators classifications), patterns construction (based on the semantic descriptions of the mediation service and on the mediation goal to be achieved) and governance mechanisms (by exploring service and patterns dependencies and impact analysis).

## 9 – Grounding

Legacy systems represent valuable assets for most of their owners and usually completely replacing them is not an option. As such, methodologies that will allow the integration of these systems with the new emerging paradigms and technologies have to be developed. For example, XML Schemas and XML data have to be lifted to ontological level in order to allow semantic-aware systems to act on this data. In addition, since all tasks related to discovery, selection, composition, etc. operate in SESA on semantic descriptions the link between semantic service and underlying technology for communication (e.g. HTTP, SOAP, etc.) needs to be defined. The basis for grounding has been established within the W3C Semantic Annotations for WSDL WG (SAWSDL WG) allowing hooking semantic descriptions with WSDL elements.

### Goals and Tasks

- **Semi-automatic tools allowing the creation of transformation between syntax-based and semantic models** - specifying the transformations between Web service XML messages and the semantic data currently requires deep knowledge both of the structure of the XML message and the ontology. Methods for semi-automated creation of grounding should be created.

- **Grounding to other specifications** – grounding definitions to other specifications apart from WSDL should be defined. It should be possible to use e.g. REST services with semantic descriptions, etc.

## 10 – Fault Handling

Fault Handling defines the handling of faults or errors occurring within execution of end point Web services. Fault handling is important in SESA as the effect of a web service failure will reach beyond the scope of the individual service if it is part of a composition. The aim of fault handling is to ensure that the failure of the service has the minimal impact, and the most appropriate action is taken resulting from the fault. This may be simply returning an error message to the user and terminating any further actions. It may involve more complex tasks such as replacement of a failed service with a substitute service, or rolling back previously actions carried out by other services in a composition .Currently The BPEL specification has some provision for fault handling in via the catches and some support for rollback. The WS-Transaction family of specifications from OASIS also has provision for basic fault handling including distinguishing applications faults from communications faults

### Goals and Tasks

- **Increased automation of fault handling**- Semantic descriptions of services in SESA will enable an increased level of automation in fault handling. Current fault handling techniques allow either a very generic level of fault handling (e.g. gracefully exit and report errors) for all services, or require specific actions to be hard coded for each service. The aim will be able to make full use of the semantic descriptions and reasoning to make intelligent decisions about fault handling automatically (e.g. In this context a particular error can be ignored)
- **More complex fault handling tasks** - Tasks such as automatic service substitution is rarely carried out in current SOA based systems, as they is not sufficient semantic information about services to allow an equivalent service to be identified and substituted. The aim will be to increase the use of more complex fault handling tasks in SESA with the aid of semantic descriptions and reasoning.

## 11 – Monitoring

Monitoring is concerned with checking the progress of service execution and advising user/agent of abnormal events. Within SESA monitoring of services is essential as behavior of individual services has an effect on successful completion of a composition. Detecting failure or abnormalities will require some action to be taken (e.g. rollback previous actions, or substitute failed service)

Currently the most popular methods for Web Service monitoring are based around the MUWS (Management Using Web Services) OASIS specification, which defines a flexible, expandable approach to monitor manageable resources. Specifically Web Service monitoring is handled under a sub specification called MOWS (Management of Web Services).

MUWS defines how the ability to manage, or, how the manageability of, an arbitrary resource can be made accessible via Web services. Manageable resources can be accessed with a Web Service endpoint. The manageability consumer exchanges messages with the endpoint in order to subscribe to events, get events, and request. The type of information available for are things such as as: Number of Requests, Number Of Failed Requests, Number Of Successful Requests, Service Time, Max Response Time, Last Response Time.

### Goals and Tasks

- **Monitoring framework based on ontologies** – Currently MUWS defines parameters for monitoring based on a rigid XML schema. A move to a more flexible ontology based definition would give more expressivity and allow monitored to be tailored to specific context.
- **Increased automation of monitoring and link to fault handling** In a more automated SESA based system, the system should be able to proactively monitor the execution of services, identify when a particular abnormality has occurred and take the best action at the time to deal with it. For example if a service failed the system should detect it and replace with another service that performs the same task. Current methods of monitoring will identify problems with web services but there is no automatic next step to fault handing and recovery. This should be included the scope of SESA.

## 12 – Formal Languages

Descriptions in SESA need different formal languages for the specification of different aspects of knowledge and services. These descriptions can be decomposed into four dimensions: Static knowledge (ontologies), Functional description (capabilities), Behavioral description, Non-functional Properties. There are several knowledge representation formalisms used for formal languages including Description Logic and Logic Programming, Datalog subset of F-Logic, Horn subset of F-Logic with negation under the well-founded semantics, Description Logic SHIQ, first order language with nonmonotonic extensions, etc.

### Goals and Tasks

Major objective is to integrate FOL-based and nonmonotonic LP-based languages, the explicitization of context for use with scoped negation, and the development of rules for the, Semantic Web (through the W3C RIF working group). Furthermore, requirements on the functional descriptions of services and as well as a semantics for web service functionality need to be devised. Requirements need to be gathered on the description of a choreography and an orchestration and a semantics needs to be devised. Finally, purpose and usage of non-functional requirements need to be investigated. In particular, the goals will be:

- **Integrating knowledge based on classical first-order logic and nonmonotonic logic programming** – important issues are the representational adequacy of the integration, as well as decidable subsets and a proof theory, so that reasoning becomes possible; scoped default negation; rules for the Semantic

Web – RIF working group; connection between Semantic Web languages RDF, OWL

- **Functional specification of services and a semantics needs to be devised** which can be combined with the language for the description of ontologies, in order to enable the use of ontologies for the description of web service functionality. An important use case for the functional description of services is discovery. Therefore, it is expected that many requirements on the functional description of services will come from the discovery research goal.
- **Advanced Behavioral description** – there exist several formal languages which are suitable for behavioral description. Examples are transaction logic, situation calculus, and action languages. Requirements need to be gathered on the description of choreography and an orchestration and semantics needs to be devised. A key challenge is the combination of this language with ontology languages in order to enable the reuse of ontology vocabulary in the choreography and orchestration descriptions. Finally, this language needs to be connected to the language for capability description in order to prove certain correspondences between the functional and behavioral description of services.
- **Non-functional Properties** – Non-functional properties can at least be divided into two categories: (1) meta-data, e.g., author, description, etc., of the WSML statements in a description and (2) actual non-functional properties, i.e., actual properties of services (e.g. pricing, QoS, transactions). NFPs require a deeper investigation into their purpose and their usage.

### 13 – Reasoning

The SESA necessitate effective reasoning for different tasks such as service discovery, process and data mediation and integration. To enable processing of these tasks in an automated manner, the SESA utilizes machine reasoning over formally represented service specifications.

We are developing Integrated Rule Inference System (IRIS)<sup>6</sup> which is a scalable and extensible reasoner tool for Web Service Modeling Language (WSML)<sup>7</sup>. The system implements different deductive database algorithms and novel optimization techniques.

#### Goals and Tasks

- **Reasoning techniques with large data sets** – In the context of Semantic Web, applications might require vast volumes of data to be processed in a short time. Current reasoning algorithms are developed rather for small, closed, trustworthy, consistent, and static domains. Therefore these algorithms need to be extended and adapted in order to be applied on large and dynamically changing knowledge bases. One challenging approach to achieve a scalable reasoning is to combine existing deductive and database techniques with methods for searching the Web (utilizing semantic annotations). This line of research considers reasoning in distributed environments as well.

---

<sup>6</sup> <http://www.sourceforge.net/projects/iris-reasoner>

<sup>7</sup> <http://www.wsmo.org/wsml/>

- **New techniques for Description Logics reasoning** – Description Logics (DLs) are a family of knowledge representation formalisms characterized by sound, complete and (empirically) tractable reasoning. However applications in areas such as e-Science and the Semantic Web are already stretching the capabilities of existing DL systems. Key issues here are the provision of efficient algorithms that allow (advanced) applications (1) to scale up to knowledge bases of practical relevance and (2) to leverage expressive languages for capturing domain knowledge.
- **Reasoning with integrating frameworks based on classical first-order logic and nonmonotonic logic programming** – Two lines of research will be explored:
  - Reasoning with decidable fragments of such integrating frameworks.
  - Reasoning with undecidable fragments using proof-theoretic techniques.

## 14 – Storage and Communication

**Storage and communication form the underlying mechanisms of the SESA architecture needed for coordination of the execution of middleware services within the platform. The novel communication and coordination paradigm is called Triplespace Computing (TSC). TSC is recently receiving attention in open distributed systems like the World Wide Web and pervasive computing environments. TSC supports the Web's dissemination idea of persistently publish and read. Furthermore, it is based on the convergence of tuplespace technology (originating from Linda) and Semantic Web (service) technology where RDF triples provide the natural link from tuplespaces to triplespaces. Having machine understandable semantics integrated in the middleware makes this approach particularly useful for SEE architectures. TSC can be used for dynamic management of middleware services, middleware services coordination, resourcemanagement and external communication with SESA.**

### Goals and Tasks

- **TSC establishment** – interaction interfaces, specification of security and trust support, ontology-driven space management for the development of self-adaptive and reflective TS Kernels for the integration of scalable semantic clustering and distributed querying.
- **Dynamic management of middleware services** – asynchronous coordination of middleware services through local triplespaces, running middleware processes over triplespaces.
- **Resource management** – a unified storage infrastructure with standardized access policies and interfaces replacing dedicated repositories and hiding the complexity of different types of resources. Thus, interface TS kernels with the resource management, install RDF-based access to resources, and management of distributed resources needs to be developed.
- **External communication with SESA based on TSC** – SOAP-enabled Web service execution over triplespaces, grounding of SOAP messages to triplespace computing, lifting of RDF-based messages to SOAP.

## 15 – Execution Management

Execution management as the kernel of the SESA architecture is responsible for the coordination of middleware services. It realizes the overall operational semantics of the middleware which lets the system achieve the functional semantics of its client-side interface. It orchestrates the functionality of the middleware services into a coherent process in an orderly and consistent fashion. This process is defined by so called execution semantics which defines how particular middleware services need to interact so that SESA can provide particular functionality to its users. The research focuses on the functional as well as the operational combination of the individual services of the middleware.

### Goals and Tasks

- **Definition and Refinement of execution semantics** – definition of various execution semantics for particular execution scenarios
- **Triple Space Computing Integration** – inter-kernel communication and coordination, distributed execution of tasks; The communication between execution management and middleware services through publishing and subscribing the data as sets of triples over triple space. Other than the benefits of asynchronous communication, it will achieve decoupling of individual middleware services.

## 16 – Security

In the Context of SESA security will cover many areas including: authenticating access to services, preventing misuse of services, encryption & data privacy. Security will be an important concern to ensure that services are accessed correctly, by the authorized people and that confidential or sensitive data is securely stored and transmitted. There are currently many Web Services standard relating to security, the most prominent being the recently completed WS-Security 1.1 specification from OASIS

## 6. Summary

This document outlined a comprehensive framework that integrates two complimentary and revolutionary technical advances, Service-Oriented Architectures (SOA) and Semantic Web, into a single computing architecture that we call Semantically Enabled Service oriented Architecture (SESA). While SOA is widely acknowledged for its potential to revolutionize the world of computing, this success is dependent on resolving two fundamental challenges that SOA does not address, namely integration, and search or mediation. In a service-oriented world, millions of services must be discovered and selected based on requirements, then orchestrated and adapted or integrated. SOA depends on but does not address either search or integration. The contribution of NESSI semantic technology is to provide the semantics-based solutions to search and integration that will enable the SOA revolution. The report provides a vision of the future, enabled by SESA, that

places computing and programming at the services layer and places the real goal of computing, problem solving, in the hands of end users. It defines a research roadmap for NESSI Semantic Technology WG for the timeframe 2007 – 2010.

## **Appendix 1 Semantic and Web 2.0 technology – some brief remarks**

The combination between semantics, Web 2.0 and SOA technologies is another promising research field. There is not a common definition for Web 2.0. However, it is commonly accepted that Web 2.0 is about a new generation of applications or services aimed at fostering the collaboration and participation of users in the web based on informal annotations. According to Tim O'Reilly (O'Reilly, 2005), among the main basic principles of Web 2.0 are to take advantage of the web as a platform, the collective intelligence, the software ubiquity and the importance of the data, as well as providing rich user interfaces. These principles can be clearly combined to those of SOA.

There are many Web 2.0 related initiatives that explore the principle of a web as a platform:

- Web as a platform for high-performance applications, such as Amazon Web Services, Amazon Storage Service, Amazon Computing Cloud or HousingMaps. They offer data storage, high-performance computation or housing services.
- Web as a platform for Portal customization based on Mashups, such as GoogleIG, NetVibes, Live.com, or YouOS. These examples show the tendency to personalize the access to services and contents.
- Web as an architecture encouraging people participation. Examples like Writely, NumSum and Google Calendar show how their services gain from the people participation in their services.

On the other relation between Web 2.0 technologies and SOA is not new. The use of AJAX for creating dynamically enhanced web user interfaces for SOA-based applications, or Web services used in Web 2.0 applications are quite common. However, due to the fact that the Web 2.0 is based on informal annotation, the fusion between Web 2.0 and SOA needs clearly the use of semantics to glue. This is a research work only partially undertaken. Aspects such as user modelling (including the definition of a user context and a common vocabulary for characterising the user activity and their social and business preferences), collaborative ontology development, service discovery and composition based in social context, and the use of rich and personalized interfaces for the different SOA users are among the research fields.