SOA4All: Enabling Web-scale Service Economies

Reto Krummenacher, Barry Norton, Elena Simperl
Semantic Technology Institute STI
University of Innsbruck
Innsbruck, Austria
firstname.lastname@sti-innsbruck.at

Carlos Pedrinaci
Knowledge Media Institute KMi
The Open University
Milton Keynes, UK
c.pedrinaci@open.ac.uk

***Abstract*—Establishing Web services as resources on the Web opens up highly productive but challenging new possibilities for service economies. In addition, lifting services to the semantic level enables more sophisticated means for automating the service-related management processes and the composition of arbitrary functionality into new services and businesses. In this paper we present the SOA4All approach to a global service delivery platform. By means of semantic technologies, SOA4All facilitates the creation of service infrastructures and increases the interoperability between large numbers of distributed and heterogeneous functionalities on the Web.**

***Keywords*-Global Service Delivery Platform, Semantically-enhanced SOA, Semantic Web services**

I. Introduction

Research and development on service-oriented architecture (SOA) and Web services in general has been particularly productive during the last decade with a wide-range of standards and tools produced and supported by major vendors. However, despite its success, SOA's evolution and uptake seems to be slowing down recently. SOA remains mostly an enterprise-specific solution and its adoption for supporting the creation of distributed systems on the Web has largely fallen behind the initial estimates. SOA is somehow a victim of its own success. The number of services that need to be discovered, integrated, orchestrated, analyzed and maintained, causes considerable problems which are hampering the scaling up of service-oriented architectures to the dimensions of the Web. Furthermore, the technological landscape surrounding Web services is characterized by a heavyweight and ever-growing stack of standards and technologies, be that communication protocols, infrastructures or languages, which hardly contributes to enhancing the adoption of service-oriented solutions in the large.

We argue consequently that SOA needs to be rethought in the light of a number of principles that can be distilled from work in other areas such as the Web, Web 2.0, and Pervasive Computing. In this paper we present an architecture and language stack for a global service delivery platform that fosters the Web-scale adoption of service technologies. The architecture extends SOA with essential principles that made the Web a success, such as its openness, decentralization, and the fact that communication is mostly based on persistent publication rather than messaging. Additionally, we adopt Semantic Web languages as a means to lift services and their descriptions to a level of abstraction that deals with machine-understandable conceptualizations. This increases the level of automation that can be achieved while carrying out common tasks during the life-cycle of services, such as their discovery, composition or invocation. Further ingredients to the service platform come from the application of Web 2.0 principles, notably to emphasize the human as service prosumers (consumers become producers and vice versa in a community of service workers [1]) and RESTful services as a complementary technology to traditional WS-* stack-based Web services. Finally, automated context adaptation capabilities are embedded within the architecture in order to support the use of services in unforeseen contexts, thus increasing the versatility of the services provided while retaining their manageability. As a whole, this delivers a comprehensive framework that enables large enterprizes, SMEs and end-users to get engaged within a Web-scale service-oriented infrastructure.

In the next section we introduce in more details the architecture and principles of the SOA4All service delivery platform. Section III goes further into details about the role of semantics as means towards automation and interoperability. We concentrate in Section III-A on the semantic models and languages for annotating services, and exploit similar techniques in Section III-B for goal-driven service invocation. Section IV offers conclusion from our examination.

II. SOA4All Service Delivery Platform

The global service delivery platform (GSDP) is a Web-scale service-oriented architecture that is under investigation across various recently established European research projects. The GSDP is an open platform with domain independent services that can be used to build domain specific service solutions. SOA4All concentrates on the establishment of an instance of a service delivery platform that is optimized and tailored to the needs of Web services (both traditional WS-* stack-based services, as much as REST APIs), while the ultimate goal is to realize a platform for all types of exposable functionalities, such as mobile services,

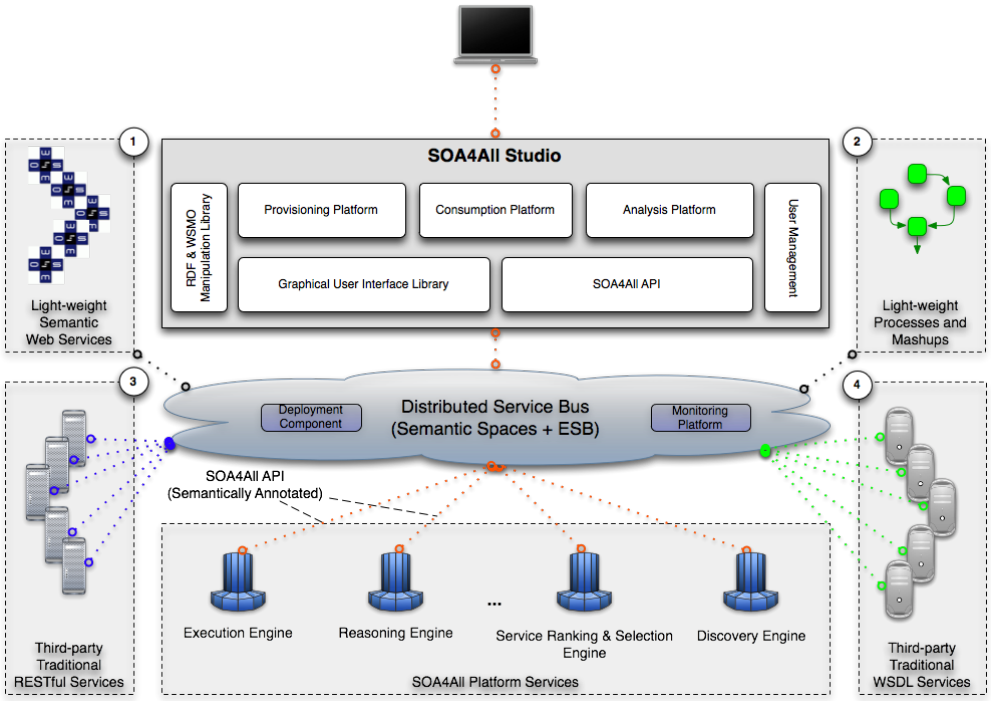

Figure 1. SOA4All overall architecture

sensors and aggregators, or hardware resources. As such the GSDP will provide the core provisioning and consumption functionalities for the emerging XaaS ('Everything as a Service') infrastructures.

In the remainder of this section we present the SOA4All overall architecture. It is built around the following main components: SOA4All Studio, as user front-end, Distributed Service Bus, Platform Services, and Business Services (3rd party Web services and light-weight processes). Figure 1 shows a high-level depiction of these core elements and their relationships.

A. *Service Bus*

The Distributed Service Bus lies at the center of the service delivery platform, as the core communication and integration infrastructure. The DSB augments traditional enterprize service bus technology (ESB, [2]) towards large-scale, open, distributed, and hence Web-scale, computing environments. The extensions to the service bus infrastructure include for this purpose the implementation of distributed service registries, the integration of the bus realization with established Grid-aware middleware for decentralized deployments, and the enhancement of the communication and coordination protocols by means of semantic spaces. From a deployment perspective, the problems to be solved become of similar nature to those raised when programming, deploying, securing, monitoring, adapting a distributed application on a computing grid infrastructure.

The enterprize service bus core is based on PEtALS,[1] an open-source JBI compliant implementation that is hosted by the OW2 Consortium.[2] In terms of Grid-aware middleware, the DSB currently profits from the ProActive Parallel Suite that is also hosted by OW2.[3] The Web-style publish and read infrastructure at last, is provided by so-called semantic spaces that significantly increase the scalability of the bus in terms of communication and coordination of distributed and autonomous services [3].

Semantic spaces are a novel type of communication platform that has recently gained momentum in the middleware community, as a response to the raising challenges of data sharing and service coordination in large-scale, distributed, open and highly dynamic Web environments. Semantic spaces fuse tuple space computing [4], known from parallel processing; blackboard-style problem solving [5], known from artificial intelligence; and semantic technology to a distributed (semantic) data management platform.

In SOA4All, semantic spaces are realized as a virtualization layer on top of distributed but tightly coordinated semantic repositories. The semantic spaces offer a simple but powerful set of operations for publishing RDF statements, querying SPARQL end-points, and coordinating actions via

[1]PEtALS is an open source ESB: http://petals.ow2.org/.

[2]OW2 is a global open-source software community for distributed middleware solutions: http://www.ow2.org/.

[3]ProActive is a middleware for parallel, distributed, multi-core computing: http://proactive.ow2.org/.

pattern-based notification services. More pragmatically spoken, service prosumers can subscribe to particular patterns that they would like to match within the wealth of semantic data (RDF graphs). Semantic patterns are triple patterns or graph patterns, as they are supported by most RDF query languages (cf. for example SPARQL as defacto standard [6]). Upon publication of matching sets of statements, the subscriber is notified and can proceed with its own workflow. In terms of data management, semantic spaces moreover provide means for creating virtual containers in form of so-called subspaces or federations. The latter are temporary read-only views over multiple subspaces, comparable to views in relational databases across different tables. Subspaces and federations are used to create dedicated interaction channels that increase at least local scalability by naturally grouping collaborating services and related data.

The implementation of the semantic space infrastructure is based on well-established P2P technology[4] and is deployed on the same Grid-aware middleware as the service bus nodes. In that way the space nodes and bus nodes are co-existent and coordinated resources in the same grid infrastructure.

From a usage point of view, semantic spaces are exploited to realize many of the large-scale data sharing scenarios for which the volumes of semantic data that has to be processed is likely to exceed the numbers that semantic repositories can currently handle. SOA4All relies on semantic spaces to incorporate various types of repositories for service annotations, goals or process descriptions, as memory infrastructure for shared access to semantically-described monitoring data and user profiles, and most of all as alternative communication channel that enhances the traditionally message-oriented service bus with features for anonymous and asynchronous service coordination based on RDF statements; this aspect will be visited in more details in the part about processes below, in particular in the context of mash-ups.

It is important to note that the complementary communication facilities of the semantic spaces are offered to all services and users as an integrated service of the DSB, without requiring the maintenance of additional access points. In other words, the DSB is conceptualized to provide the core infrastructure services (integration, communication and storage) of SOA4All in an all-in-one solution. This is an important principle for a scalable service delivery platform, as it allows any type of communication efficiently and transparently by means of sharing or exchanging any type of data in between any type and number of distributed parties.

[4]A Chord-ring [7] is used to index the subspaces, while a 3-dimensional CAN overlay [8] offers a natural solution to the storing of RDF statements.

B. SOA4All Studio

The SOA4All Studio delivers a fully Web-based user front-end that enables the creation, provisioning, consumption and analysis of services that are published to SOA4All (Figure 2). It consists of three subcomponents that target the three different service management tasks: provisioning at design time, consumption, and analysis at runtime. Each of them is shortly described in the following:

- The **Provisioning Platform** has two main purposes. First, it provides the tools to annotate services, either WSDL services via WSMO-Lite, or REST APIs via MicroWSMO (cf. Section III for more details). Second, it incorporates a Process Editor that allows users to create, modify, share, and annotate executable process models based on a light-weight process modeling language. SOA4All provides such a language as a considerably simplified subset of the Business Process Modeling Notation (BPMN1.2, [9]) for the abstract parts, and a subset of the Business Process Execution Language (WS-BPEL2.0, [10]) for executable parts. In this way, SOA4All hides much of the service composition complexity whilst providing sufficient notational semantics to understand the interactions between services being composed, and sufficient expressive power for the users to construct useful compositions.
- The **Consumption Platform** is the gateway for users to the service world when they act as consumers. The platform allows them to formalize their desires in several ways, defining and refining goals that can be used to discover and invoke the services that fulfil their needs. The platform stresses personalization by making use of contextual factors to offer a more suitable service consumption to the users, and to adapt the services and the platform based on past use; e.g., by recommending goals for different users in varying situations.
- The **Analysis Platform** obtains information (monitoring events) from the monitoring subsystem of the bus and performs processing in order to extract meaningful information. Monitoring events should come from data collectors that perform basic aggregation from distributed sources in the service delivery platform. Data collectors are installed at the level of the bus (message exchange monitoring), the grid middleware (node monitoring in terms of storage or processing load), and the execution engine (service access and process invocation monitoring).

C. Platform Services

The platform services deliver discovery, ranking and selection, composition and invocation functionality, respectively. These components are exposed via the SOA4All Distributed Service Bus as Web services and hence consumable as any other published service. The distinguishing factor

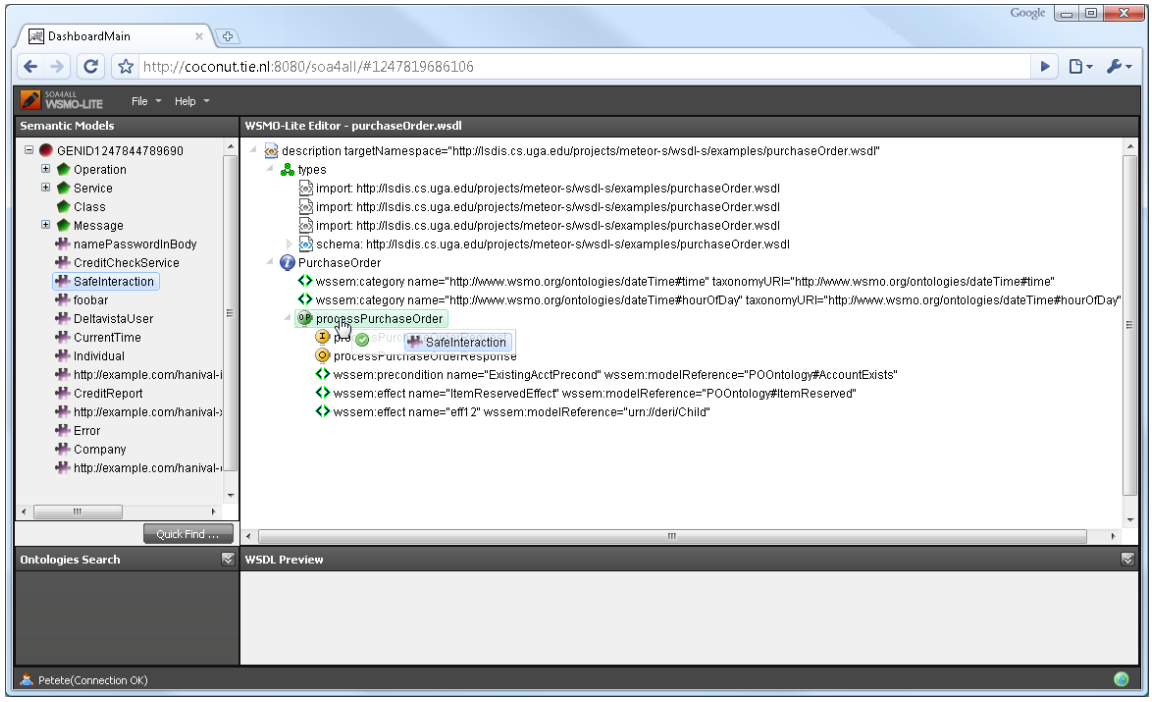

Figure 2. SOA4All Studio: WSMO-Lite Editor (Provisioning Platform)

between platform services and other Web services is the fact that they provide the core functionality required to realize the service platform. A priori, the set of platform services and their roles and interfaces is aligned with recent research in the area of Semantic Execution Environments and Semantic Web services (e.g., the OASIS SEE Technical Committee).[5]

Platform services are mostly used by the SOA4All Studio to offer clients the best possible functionality, while their combined activities (i.e., discovery, selection, composition and invocation) are coordinated via the Distributed Service Bus. The ensemble of DSB, SOA4All Studio and platform services delivers the fully Web-based and Web-enabled service experience of SOA4All: global service delivery at the level of the bus, Web-style service access via studio, and advanced service processing, management and maintenance via platform services.

D. Business Services (Web Services) and Processes

In the corners of Figure 1, semantic service descriptions and processes (composed services) are depicted. They are created and processed by means of the SOA4All infrastructure. First, SOA4All enables available Web services that are exposed either as RESTful services, or as traditional WS-* stack-based services in form of WSDL endpoints [11] (marked with (3) and (4) in Figure 1). Such services are referred to as invocable third-party business services, and are enhanced by SOA4All in terms of automation, composition and invocation via the lifting to the semantic level (cf. Section III). Second, Figure 1(1) depicts the semantic annotations of the business services, so-called Semantic Web services. The semantic descriptions are published in the service registry that is shipped as a platform service, and used for reasoning with service capabilities (functionality), interfaces and non-functional properties, as well as context data. These semantic descriptions are the main enablers of the automation processes related to Semantic Web services. Third, marked as (2), light-weight processes and mash-ups are shown. They are the basis for the realization and execution of composed services. Processes in SOA4All are orderings of Semantic Web services, goal descriptions with associated constraints, and data respectively control flow information. As stated earlier in this paper, a goal is a formal specification of the objective that a user would like to have performed and as such it yields an implicit abstraction over the Web services (REST or WS-* stack-based) that need to be executed. Semantic descriptions of processes are published in form of RDF graphs in the shared semantic space infrastructure and, as such, become a public common and building block for service computing.

In contrast to fully-fledged processes, a mash-up is a data-centric specification over a collection of REST services. A further characteristic of mash-ups is the fact that they are almost entirely executable in semantic spaces. By being data-driven, their composition is enabled by the coordinated access to shared data in semantic spaces. Although being

[5]OASIS SEE TC: http://www.oasis-open.org/committees/semantic-ex.

comparably simple, mash-ups thus provide a very promising approach to Web-style service computing, a pre-requisite for large-scale service economies.

III. Interoperability through Semantics

Semantics, and semantic technologies, is the name given to the general use for computing of Semantic Web technologies, particularly: the consistent and principled use of Uniform Resource Identifiers (URIs) for identification of resources; the underlying use of Resource Description Framework (RDF) triples for knowledge representation; and the use of ontologies to model knowledge in terms of identified resources. The ability to reason over ontology-based representations, and the ability to exchange information in a standardization fashion are built upon in SOA4All with other technologies forming the bottom four layers of the 'Semantic Web layer cake', shown in Figure 3.

In this section we explore the use of semantic descriptions for Web services, an approach called Semantic Web services, and the reasoning-based application of these descriptions to achieve automation in service consumption as goal-driven invocation.

A. Semantic Web Services

In the Semantic Web services approach an ontological description of service functionality, access means, non-functional properties, and often composition, is added to whatever description otherwise exists of these, which usually focusses mainly on access. In the early days of Semantic Web services it was frequently assumed that the so-called WS-* Stack was being extended. In particular it was assumed that services are described in the Web Service Description Language (WSDL) and that they communicate in terms of the SOAP protocol with XML messages. Furthermore Web service description ontologies such as OWL-S and WSMO have used a top-down approach to modeling Web services in which attachment to the actual access means for the underlying service is almost an after-thought through the abstraction of WSDL.

While the potential advantages of reasoning to achieve automation in service tasks has been clearly demonstrated

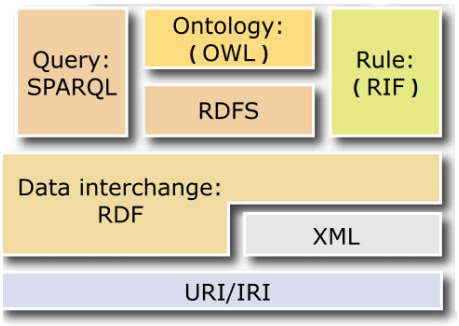

Figure 3. Four layers of the Semantic Web cake

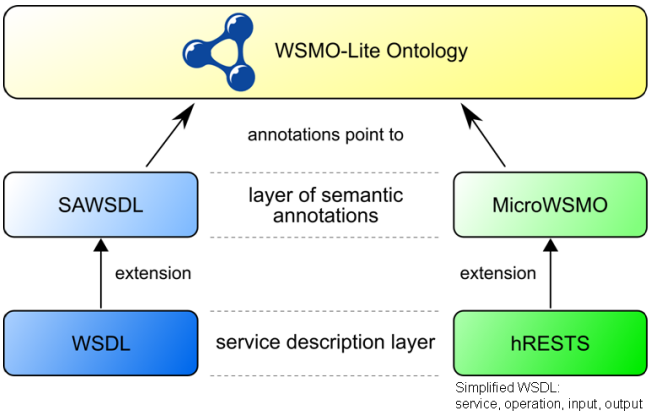

Figure 4. Lightweight service modeling overview

by such models, such a specific and top-down model has been more recently challenged by two developments. First the W3C's standardization approach to the combination of Semantic technologies with Web services has been to challenge the top-down approach and instead provide a recommendation, Semantic Annotations for WSDL (SAWSDL), wherein a bottom up approach of linking WSDL artifacts to semantic descriptions is facilitated. This is seen as a more engineer-friendly approach.

Secondly much of the industry has jettisoned entirely the perceived overheads of the WS-* stack in favor of services which simply communicate using the verbs-and-identifiers view of resources implicit in the Web's HyperText Transfer Protocol (HTTP), over which SOAP is somewhat artificially built. Although reinstating HTTP principles was the original purpose of this so-called RESTful approach, the freedom from need to produce WSDL descriptions and SOAP-compliant messages is often the motivation for its general adoption. At the same time Semantic technology will only reach a wide audience in services if it responds.

The approach of SOA4All to these two challenges is illustrated in Figure 4, respectively as WSMO-Lite and MicroWSMO. In both cases an RDFS-based lightweight service description ontology, the basis of which is shown in Table I, is used as the basis of semantics-based description of services.

In the first case SAWSDL's extensions are used to attach descriptions to these, as shown in Table II where the schema (exemplified by the `ReservationRequest` element), interface and operations (exemplified by `searchForRooms`) of a WSDL-based service description are linked to instances over the lightweight service ontology.

In the second micro-formats are used to attach these descriptions to HTTP documents transferred by RESTful services, as shown in Figure 5. As a first stage hRESTS is used to mark the parts of a document which describe artifacts

Table I
LIGHTWEIGHT SERVICE DESCRIPTION ONTOLOGY USED IN WSMO-LITE AND MICROWSMO

```
Service             a rdfs:Class .
    hasOperation      a rdf:Property .
Operation           a rdfs:Class .
    hasInputMessage   a rdf:Property .
    hasOutputMessage  a rdf:Property .
    hasInputFault     a rdf:Property .
    hasOutputFault    a rdf:Property .
Message             a rdfs:Class .
```

of services and then SAWSDL-style model references are used to attach to semantic descriptions of these in the WSMO-Lite lightweight service modeling ontology.

B. Goal-Driven Invocation

One of the main objectives of a service delivery platform is to make service end-points transparent. In the particular case of SOA4All this implies that services disappear behind the interfaces of the bus, or more generally spoken become integrated elements of the Web as the platform. This emphasizes the fact that the value of the end-point is replaced by the value of the invocable functionality. It is no longer the physical nature of a service that is the determining factor, but rather the deliverable functionality and quality of service. User interaction is no more guided by addressing services, but by specifying objectives through goal descriptions. Goals are hence a means to describe a user's objectives, and as such provide an implicit description of the services that are needed to fulfil the user goal. The service delivery platform then takes care of discovering the necessary services, of potentially composing new processes, and finally of invoking the service(s) on behalf of the user. This procedure is referred to as goal-driven invocation, as the processing of the user request is directly controlled by the information derived from the desired task [12].

All the SOA4All service processing tasks are done at the semantic level (cf. Section III). Analogously to the lightweight semantic description of services, SOA4All thus specifies an ontological model for goals (Table III). Any goal becomes an instance of the class 'Goal' for which a user can indicate the desired input and output information of the service(s) and additional requirements and preferences in terms of functional classifications, non-functional properties, preconditions and effects.

With the exception of the functional classification, the requirements and preferences all point to axiomatic expressions; those are given in SOA4All by logical expression in WSML, or any other language that allows both ontology and rules definition. As we do not intend to enter into WSML reasoning with this paper, we concentrate in the remainder of this section on the use of functional classifications, or as they

Table III
SIMPLE GOAL DESCRIPTION SCHEMA

```
Goal a rdfs:Class .
  hasInput       a rdf:Property.
  hasOutput      a rdf:Property.
  hasPreference  a rdf:Property.
  hasRequirement a rdf:Property.

FunctionalCategorisation a rdfs:Class.
  hasCategory  a rdf:Property.
  hasMatchType a rdf:Property.

MatchType rdf:Alt Strict, Relaxed,
                  Subcategory, Super-category.
```

are termed in the goal ontology, functional categorization. The difference between the WSMO-Lite classification and the goal categorization lies in the possibility to indicate matching types:

- Strict: the service is associated with exactly the same set of categories as the goal (nothing less, nothing more); e.g., the user wants exactly a "travel booking service" or a "reliable service".
- Relaxed: the service is associated with all the categories of the goal and some others (nothing less); e.g., the user desires at least a service that is tagged as "secure".
- Subcategory: the service is associated with a subcategory (simple or composite) of that of the goal; i.e., the service classification is more specific.
- Super-category: the service is associated with a supercategory of that of the goal and the functionality of the service is more general than the one of the goal; e.g., the users wants a "train booking service" but is happy with a general purpose "travel booking service".

Once a goal is given by a user, the required or desired functional classifications can be extracted, and can be used as basis for SPARQL queries on top of the RDF-based service descriptions in the service registry. In a very primitive sense, this provides a first light-weight discovery approach that we term goal filtering. The goal categorization thus helps to select potentially interesting services by means of their classification. The examples in Table IV showcase such a transformation into SPARQL. The first example seeks a service that is described as "travel service", while the latter searches a services that matches the intersection ofthe tags "travel service", "England", and "hotel service".

Efficient service discovery is the most fundamental component for goal-driven invocation, as first of all, user requests must be matched against available services. By means of our light-weight goal descriptions, we provide the basis for multi-level service discovery – from simple SPARQL querying up to full-fledged reasoning with WSML – and as such are able to meet the different needs of users in large-scale SOA4All scenarios.

Table II
WSMO-LITE EXAMPLE

```
<wsdl:description>
  <wsdl:types>
    <xs:schema>
      <xs:element name="ReservationRequest"
                  sawsdl:modelReference="&ex;Reservation"
                  sawsdl:loweringSchemaMapping="&ex;ResMapping.xsparql" .../>
    </xs:schema>
  </wsdl:types>
  <wsdl:interface name="HotelReservations"
                  sawsdl:modelReference="&ex;AccommodationReservationService">
    <wsdl:operation name="searchForRooms"
                    sawsdl:modelReference="&wsdlx;SafeInteraction">
      ...
    </wsdl:operation>
    ...
  </wsdl:interface>
  <wsdl:service name="RomaHotels" interface="HotelReservations"
                sawsdl:modelReference="&ex;RomaHotelReservationPrecondition
                                       &ex;ReservationFee"../>
</wsdl:description>
```

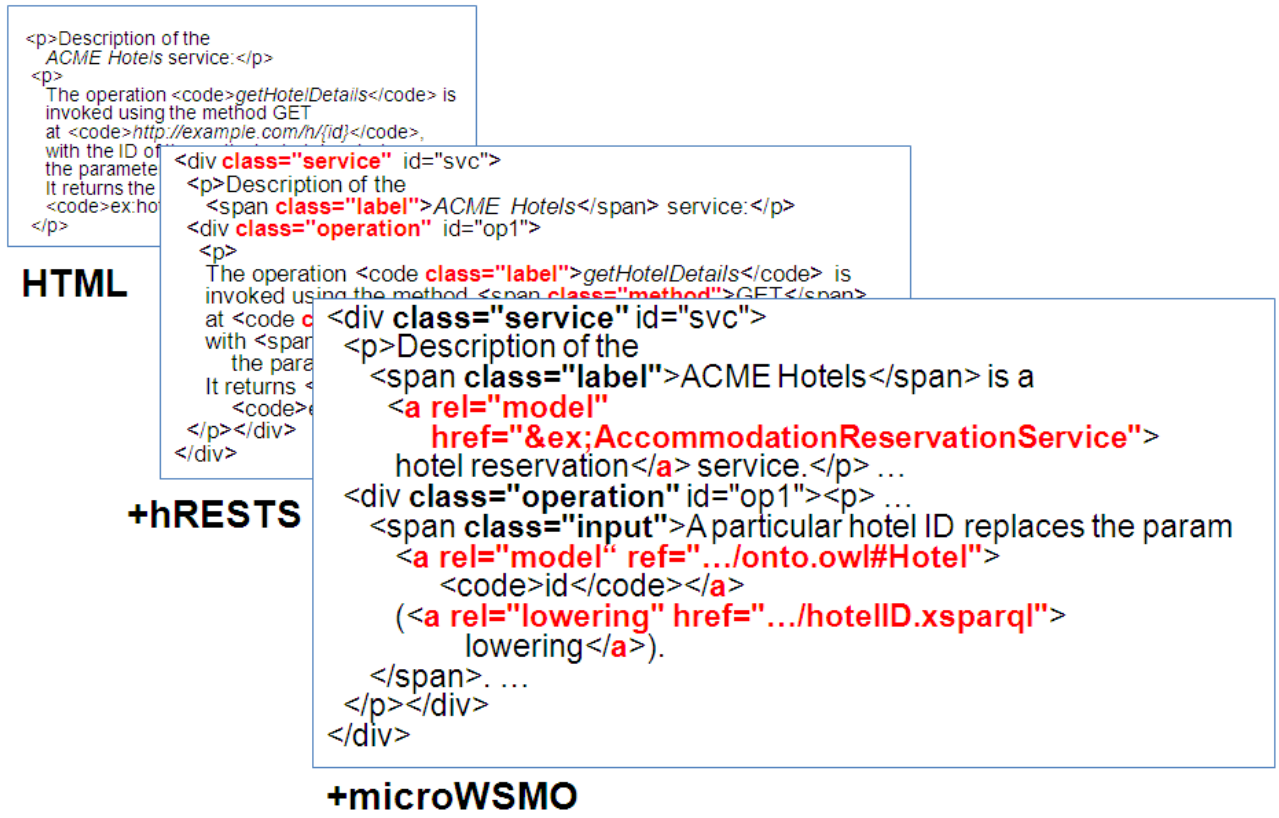

Figure 5. MicroWSMO example

Table IV
GOAL FILTERING WITH SPARQL QUERY

```
@prefix sawsdl: <http://www.w3.org/ns/sawsdl#>.
SELECT DISTINCT ?s WHERE {
?s a wsl:Service;
   sawsdl:modelReference :TravelService. }

-----------------------------------------------

@prefix sawsdl: <http://www.w3.org/ns/sawsdl#>.
SELECT DISTINCT ?s WHERE {
?s a wsl:Service;
   sawsdl:modelReference
              :TravelService,
              :England,
              :HotelService. }
```

IV. CONCLUSION

In this paper, we presented the concepts and architecture of SOA4All. SOA4All produces a first instance of a global service delivery platform, and as such aims at offering billions of services to billions of users in a Web-scale service economy. Through semantic technologies SOA4All helps new business ideas and services to be more easily realized and integrated with others.

In the context of the envisaged global service delivery platform, it becomes evident that automation is required, as otherwise, it would be impossible to handle the wealth of service and process descriptions, the users and their context, and even worse, all the monitoring data that is gathered under the life-time of billions of services. In view of the fact that there is no automation on the Web without semantics, not at last due to the eminent heterogeneity and dynamics of resources, SOA4All has to investigate novel techniques, languages and tools for annotating services, and for processing services at the semantic level. WSMO-Lite, MicroWSMO, WSMO and SAWSDL are some of the recent achievements of the Semantic Web services community that are exploited by SOA4All to do so.

Semantic technologies are a core building block for the realization of a global service delivery platform, however, we also recognize that they only offer the tools for increased automation and interoperability. In order to realize service economies at Web scale and to fully exploit the possibilities and business values of the Service Web, there is more work waiting. In SOA4All we expect to find further benefits in exploring established Web and recent Web2.0 technology to foster increased community involvement, a pre-requisite for reaching the billion services challenge.

ACKNOWLEDGMENT

The work presented in this paper is funded by the EC FP7 IP SOA4All (www.soa4all.eu). The authors would like to thank their colleagues from the consortium for all the active and more passive contributions to this paper.

REFERENCES

[1] D. Tapscott, *The Digital Economy: Promise and Peril In The Age of Networked Intelligence*. McGraw-Hill, 1997.

[2] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman, 2003.

[3] L. Nixon, E. Simperl, R. Krummenacher, and F. Martin-Recuerda, "Tuplespace-based computing for the Semantic Web: A survey of the state of the art," *Knowledge Engineering Review*, vol. 23, no. 1, pp. 181–212, March 2008.

[4] D. Gelernter, "Generative Communication in Linda," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 80–112, January 1985.

[5] R. Engelmore, *Blackboard Systems*, T. Morgan, Ed. Addison-Wesley Publishers, November 1988.

[6] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C Recommendation, January 2008.

[7] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2001, pp. 149–160.

[8] S. Ratsanamy, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," in *ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2001, pp. 161–172.

[9] S. White, "Business Process Modeling Notation," OMG Standard, January 2009.

[10] D. Jordan and J. Evdemon, "Web Services Business Process Execution Language Version 2.0 ," OASIS Standard, April 2007.

[11] R. Chinnici, H. Haas, A. Lewis, J.-J. Moreau, D. Orchard, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts," W3C Recommendation, June 2007.

[12] D. Nau, "Expert Computer Systems," *IEEE Computer*, vol. 16, no. 2, pp. 63–85, February 1983.