

Recent Trends in Semantic SOA

By DMITRI ILKAEV

1.0 Introduction to Semantic Interoperability

The *Semantic Web* is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, but also in a form that can be understood, interpreted and used by software agents, thus permitting them to find, share and integrate information more easily [1]. At its core, the semantic web comprises a philosophy, a set of design principles, collaborative working groups, and a variety of enabling technologies. Some elements of the semantic web are expressed as prospective future possibilities that have yet to be implemented or realized. Other elements of the semantic web are expressed in formal specifications. Some of these include Resource Description Framework (RDF), a variety of data interchange formats (e.g RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL).

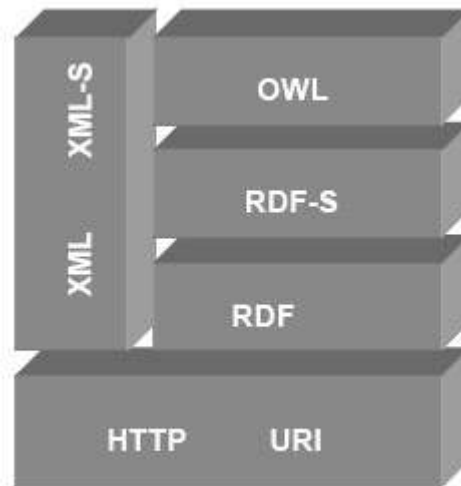


Figure 1. Semantic Web Enabling Standards

All of which are intended to formally describe concepts, terms, and relationships within a given knowledge domain.

Semantic interoperability indicates the meaning of data can be comprehended unambiguously by both humans and computer programs, and that information can be processed in a meaningful way. *Semantic integration* is the means to achieve semantic interoperability and can be considered as a subset of information integration, which includes data access, aggregation, correlation, and transformation.

In a Service-Oriented Architecture (SOA), semantic interoperability ensures that service consumers and providers exchange data in a consistent, flexible way that fulfills non-functional requirements (NFRs) such as performance and scalability, regardless of the diverse information involved. For example, a service requestor from a billing application

needs the customer balance, which is called "BALANCE". Meanwhile, a service provider from an accounting application supplies customer balance, which is called "REMAINDER". Achieving semantic interoperability is to map "BALANCE" in the billing application to "REMAINDER" in the accounting application.

Semantic interoperability is an important architectural quality in a SOA because it enables service consumers and providers to exchange information that make sense, and which then can be acted upon. It is the foundation of a SOA. Without semantics, data is just strings of binary without any meaning. Without semantic interoperability, service consumers and providers could misinterpret and corrupt data, and ultimately bring undesirable effects to a SOA and the business.

In a broader sense, most information integration deals with semantic interoperability. The problem is that people take semantic interoperability for granted and seldom make conscious and informed architectural decisions on it because the semantic interpretation, mapping and transformation are so ingrained with home-grown applications, Enterprise Application Integration (EAI) and Enterprise Information Integration (EII). Therefore, it is commonly overlooked in the development of SOA.

Patterns of semantic interoperability

There are many patterns for achieving semantic interoperability in a SOA. They can be roughly classified by the following [2]:

Pattern One: Point-to-point semantic integration

In this pattern, each data source has its own proprietary semantic meaning, and semantic translation is performed in a point-to-point manner. For example, when two data sources, A and B, need to be integrated, group A and group B print out their own ER diagrams, walk through the meaning of data elements, then perform direct mapping from data source A to B. Using the previous example, "BALANCE" column in a billing application is directly mapped to "REMAINDER" column in an accounting application. If data definition in one data source is changed, the impact to other systems is multiplied and often unpredictable. It does not matter how advanced the technology is that one picks, this semantic integration pattern is messy and a maintenance nightmare when data sources grow. Moreover, it does not easily lead to IT asset reuse. Many ESB and EII projects still perform point-to-point semantic integration in SOA. However, point-to-point integration is not necessarily a bad thing. It can be used selectively to ensure high performance and create a "fast path".

Pattern two: Hub-and-spoke semantic integration

Each system has its own proprietary semantic meaning, but is mapped to a logical data model which can be instantiated as a physical federated model or a canonical message model. Semantic interoperability is achieved within an enterprise via a hub-and-spoke topology, which reduces the redundancy and maintenance cost of point-to-point integration. Well-architected ESBs frequently use this pattern to map messages to a canonical message model and achieve semantic interoperability.

Pattern three: Master data management (MDM) pattern

MDM emerges as a pattern of semantic interoperability responding to data silos produced by departmental solutions. Today, many versions of truth exist in a typical enterprise information management system. A MDM system connects heterogeneous information sources and produces a single version of truth on key information such as customers or products for Online Transaction Processing (OLTP) and Operational Data Store (ODS) systems. The key information could be either a data instance, such as a particular customer, or metadata, such as specifications of products. A MDM system liberates data from individual business applications, package vendors and is based on open standards. As a result, data is truly treated and reused as a corporate asset. It is often built separately from existing systems to reduce the drastic impact to businesses, but legacy systems might eventually migrate to MDM systems overtime. MDM stands up as a distinct pattern from the previous two because MDM holds the single version of truth and effectively integrates various information systems from both logical and physical aspects. With MDM systems, companies gain many proven benefits, such as improved customer relationships, reduced time to introduce new products to market, data integrated with legacy systems and enabling asset reuse.

Pattern four: Industry information model

In order to encourage interoperability within an industry, vertical industry standardization groups develop industry-specific information models, which often include XML messages and message schema, also known as Domain Information Models (DIMs), although some groups produce relational data models as well. DIMs are typically XML-based and used to exchange messages in a business-to-business (B2B) environment. The members of industry standard groups agree to follow the specifications, and they are often required to certify for compliance. For instance, the Association of Retail Technology Standards (ARTS) is used for the retail industry, and the Agency Company Organization for Research and Development (ACORD) for the insurance industry. DIMs prompt a greater level of semantic interoperability, encourage asset reuse and level the playing field so members can spend less time, cost and energy to solve semantic interoperability issues. Some organizations even adopt the industry standard models as their internal enterprise logical models and canonical message models.

Standards organizations tend to look at information from horizontal and cross-industry perspectives. For example, the Open Applications Group (OAGi) is an open standards group building process-based XML standards for both B2B and Application-to-Application (A2A) integration, and it focuses on improving the state of application integration. Another example is RosettaNet, which helps companies from multiple industries meet the demands and challenges of today's global supply chain. Included are the RosettaNet Business Dictionaries and the RosettaNet Technical Dictionaries.

Pattern five: The Semantic Web

The Semantic Web cuts across the boundaries of applications, enterprises and industries. The Semantic Web links and relates elements of the data model to a common ontology. It uses the Resource Description Framework and the Web Ontology Language to allow data to be shared and reused on the Web.

2.0 Reference Model for Semantic Services Oriented Architecture [3]

Service Oriented Architectures represent a huge step towards providing simpler, more dynamic and cheaper integration solutions. By having services to encapsulate discrete piece of functionality that can be later discovered and consumed is without a doubt the critical step in moving from a patchwork of legacy products, monolithic off-shelf applications and proprietary integration to a strongly decoupled, robust yet flexible software architecture.

But SOA cannot (and it is not meant to) solve all the heterogeneity problems inherent to enterprise integration tasks. Such heterogeneity problems could be induced for example by the services (the key players of SOA) themselves or by the environment the enterprises are acting in. In the former case it is natural that services deployed by different providers to have specific requirements regarding data formats or communication protocols. Furthermore, the invocation of such services can be part of complex business process that most probably differs from one vendor to another. Regarding the second case, the World Wide Web proves to be a more and more appealing (and suited) environment for businesses and Web Services could successfully fulfill the roll of services in SOA. Such a context allows for ad-hoc cooperation and on-the-fly business provision, but also pushes the interoperability problems to extreme, beyond the boundaries of the enterprises to be integrated.

Semantics is coming to offer the tools to enable scalable, efficient and cost-effective solutions to these problems. Using ontologies and semantics, services can be unambiguously described, from data, functionality and behavior point of view. These semantics descriptions can be used in operations like discovery, selection, composition or aggregation of services to provide meaningful functionality through a truly Service Oriented Architecture. SOA can provide domain independent solutions for these operations that highly rely on semantics, formal specifications and reasoning.

It is important to note that providing semantics descriptions for SOA is not a trivial task and involves significant efforts: additionally to the actual implementation and technical interfaces an extra layer, the semantic layer, has to be added by each of the service providers. Only after this extra step is accomplished the real benefits of semantics can be exploited: old hard-coded, domain-specific solutions for service operations can be replaced by general, semantic-based ones and used in inter-enterprise (or even intra-enterprise) business scenarios.

As such, bringing semantics to SOA means adding semantics to the services part of SOA and then augmenting SOA with semantic-aware mechanisms to support the elementary operations on services (e.g. discovery, selection, composition, invocation etc). The basis of the Semantic SOA Reference Model is illustrated in Figure , see more details in [3].

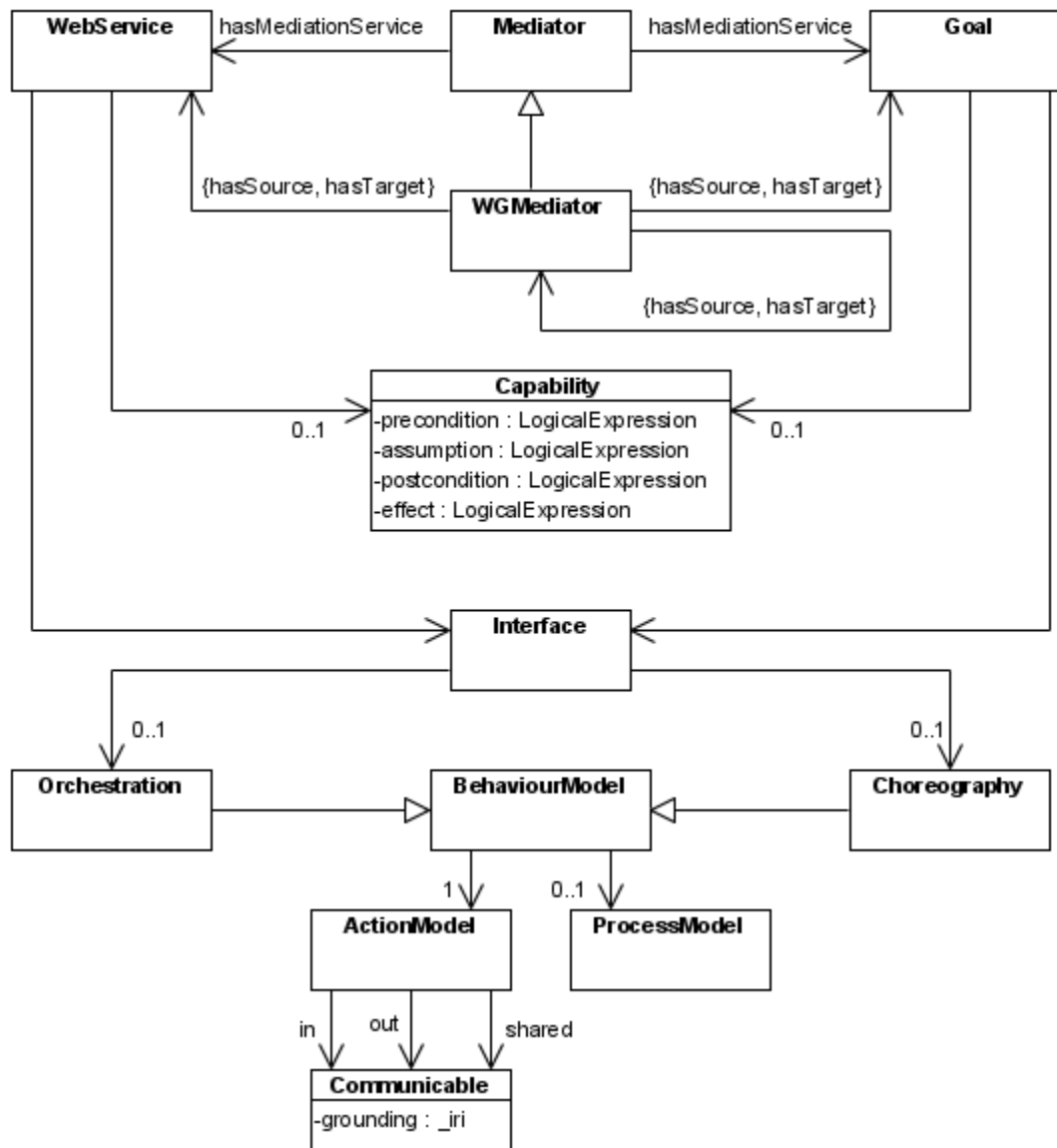


Figure 2. Reference Model as UML Class Diagram

We do not plan to provide a comprehensive analysis of the model, it's done in OASIS documentation, we should note that some basic understanding of the SOA RM (Reference Model) [4] will be helpful for the model review. Below we present the short description of the new notations that the reader may not be familiar with:

Web Service

This is comparable with the notion of service in the SOA-RM, but necessarily describes a service which is capable of *programmatic* invocation and access.

Goal

A major difference between SOA-RM and the Semantic SOA Reference Model it that we explicitly model the intentions of the client to each service. This is represented ontologically as a Goal.

Mediator

The Semantic SOA Reference Model will adopt the principle that any connection made between two elements in the model should be represented by a *Mediator*. These allow the specification of several types of *mediation*, which is to say bridging the gap between heterogeneous descriptions and/or expectations.

WG-Mediator

An important class of Mediator used in the Semantic SOA Reference Architecture is named WG-Mediator since it describes the mediation between goals and web services (and *vice versa*).

Capability

A capability is used both in the description of Goals and Web Services to semantically describe both the requirements for, and the results of, successful interaction from a global point of view. This is specified using logical expressions in the ontology language and consists in four parts, as follow: **Precondition, Assumption, Postcondition, Effect.**

Behavioral model

The behavior model deals with “*knowledge of the actions invoked against the service and the process or temporal aspects of interacting with the service*”. It consists of two distinct aspects: the *action model* and the *process model*. The first one deals with the characterization of actions that can be invoked against the service, while the second deals with the temporal relationships of actions and events associated when interacting with a service. The action model is explicitly modeled in semantic terms in the Semantic SOA Reference Model, which are grounded.

The process model is captured by the interface of a service described in the Semantic SOA Reference Model, in particular in the choreography. It describes in details the order in which interactions can be engaged in the form of a stateful conversation.

Semantic SOA Reference Model as extension to SOA-RM

Additional to the classical Web services of WSDL description and SOAP-based invocations, the Semantic SOA Reference Model provides an extra layer: the semantic description of Web services. In this section we intend to show that exactly these semantic descriptions layered on top of a proper service execution environment, enable what we are calling the new generation of *semantics-enabled* SOA architecture, in compliance with the OASIS SOA-RM standard specifications.

SOA-RM identifies four main aspects regarding the service that have to be considered in SOA:

- *Enable access to one or more capabilities.* In WSMO and SEE services are seen as well as computational entities that enable a requester to gain access and to consume certain functionality. WSMO prescribes that a service should have only one capability – a stronger condition than the one imposed by the SOA-RM.

- *Access through a prescribed interface.* The Semantic SOA Reference Model makes a clear distinction between interface and the capability on one hand, and between the interface and the implementation of the service on the other hand. A service interface must contain all the necessary information one needs to access/invoke the service.
- *Opaque to the service consumer* except from the information and behavioral models in the interface and the information required to assess if a service suits the requester needs. In the Semantic SOA Reference Model the above mentioned two-sided separations fully match this requirement. The separation between the implementation and the interface assures that none of the private business logics or the internal computational/technical details are revealed. By the separation between the interface and the capability, the Semantic SOA Reference Model makes sure that it is clearly stated what information needs to be used when finding the suited service (i.e. during the discovery service) and what information is needed after discovery when the service needs to be invoked. In WSMO the interfaces and the capability are semantically described (this description might include also the grounding to the actual implementation of the service, e.g. the WSDL web service).
- *Consequences of invoking a service* should be either response information to the invocation or a change to the shared state of the defined interface. The Semantic SOA Reference Model goes one step further and as part of the capability, it formally describes the conditions to hold on the outputs after the invocation of the service (i.e. conditions) as well as the changes on the outside world (i.e. effects). The Semantic SOA Reference Architecture defines the functionality (during the discovery phase) to check if the post-conditions and the effects match the requester needs.

It is important to note that because the Semantic SOA Reference Architecture defines operations on services described using the Semantic SOA Reference Model these requirements are wholly fulfilled. Furthermore, by use of semantics, explicit information about service's interface and capability is given; the semantic descriptions offer a significant advantage on the WSDL description (or on the service descriptions in an UDDI repository) which are in general less expressive and in most of the cases rather ambiguous.

3.0 Semantic Web Service Frameworks

Three main approaches have been driving the development of Semantic Web Service Frameworks [5]: IRS-II, OWL-S and WSMF. IRS-II (Internet Reasoning Service) is a knowledge-based approach to SWS, which evolved from research on reusable knowledge components. OWL-S is an agent-oriented approach to SWS, providing fundamentally an ontology for describing Web service capabilities. WSMF (Web Service modeling framework) is a business-oriented approach to SWS, focusing on a set of e-commerce requirements for Web Services including trust and security. We will be focusing on the last 2 approaches as those which are getting most of the recognition and momentum in the industry.

OWL-S

OWL-S (previously DAML-S) [6] consists of a set of ontologies designed for describing and reasoning over service descriptions. OWL-S approach originated from an AI background and has previously been used to describe agent functionality within several Multi-Agent Systems as well as with a variety of planners to solve higher level goals.

OWL-S combines the expressivity of description logics (in this case OWL) and the pragmatism found in the emerging Web Services Standards, to describe services that can be expressed semantically, and yet grounded within a well defined data typing formalism. It consists of three main upper ontologies: the Profile, Process Model and Grounding.

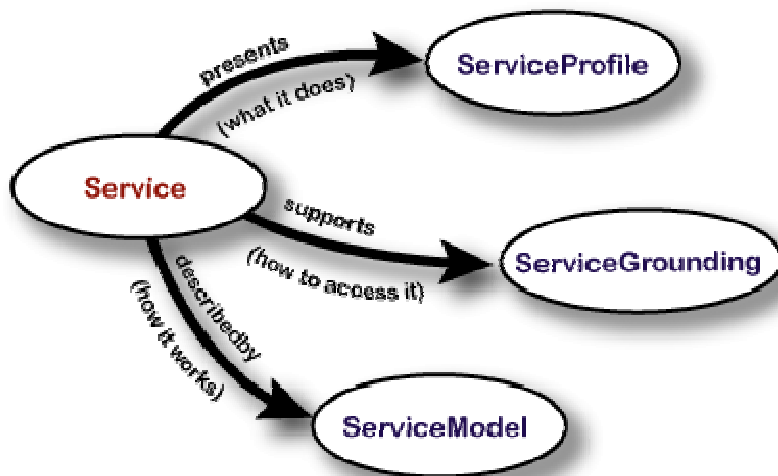


Figure 3. Top Level of the Service Ontology

The Profile is used to describe services for the purposes of discovery; service descriptions (and queries) are constructed from a description of functional properties (i.e. inputs, outputs, preconditions, and effects - IOPEs), and non-functional properties (human oriented properties such as service name, etc, and parameters for defining additional meta data about the service itself, such as concept type or quality of service). In addition, the profile class can be subclassed and specialized, thus supporting the creation of profile taxonomies which subsequently describe different classes of services.

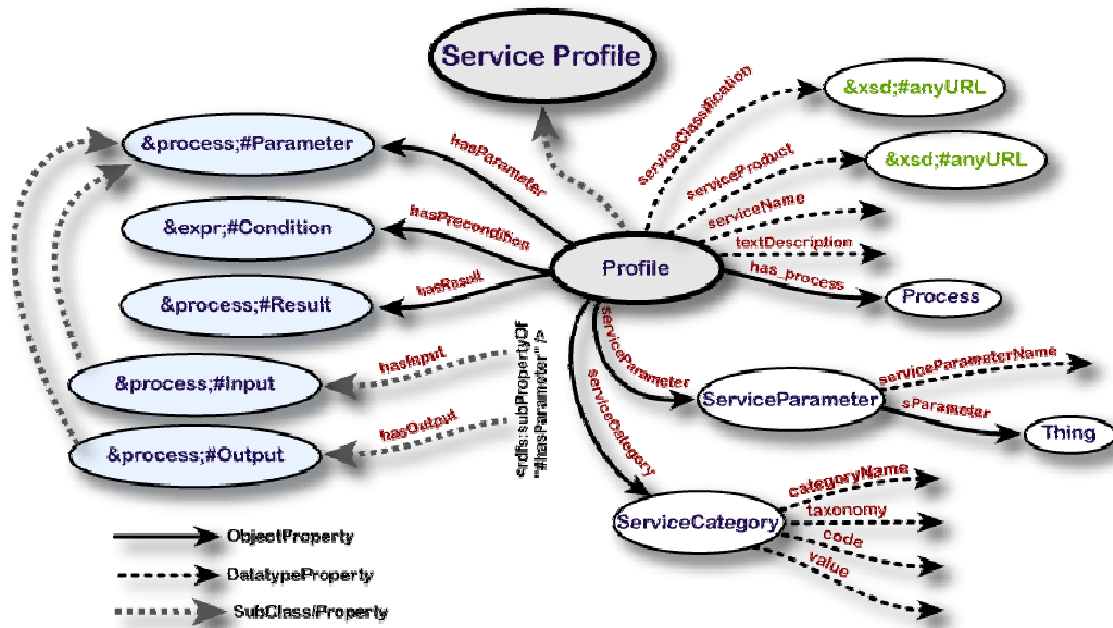


Figure 4. Selected classes and properties of the Profile

OWL-S process models describe the composition or orchestration of one or more services in terms of their constituent processes. This is used both for reasoning about possible compositions (such as validating a possible composition, determining if a model is executable given a specific context, etc) and controlling the enactment/ invocation of a service.

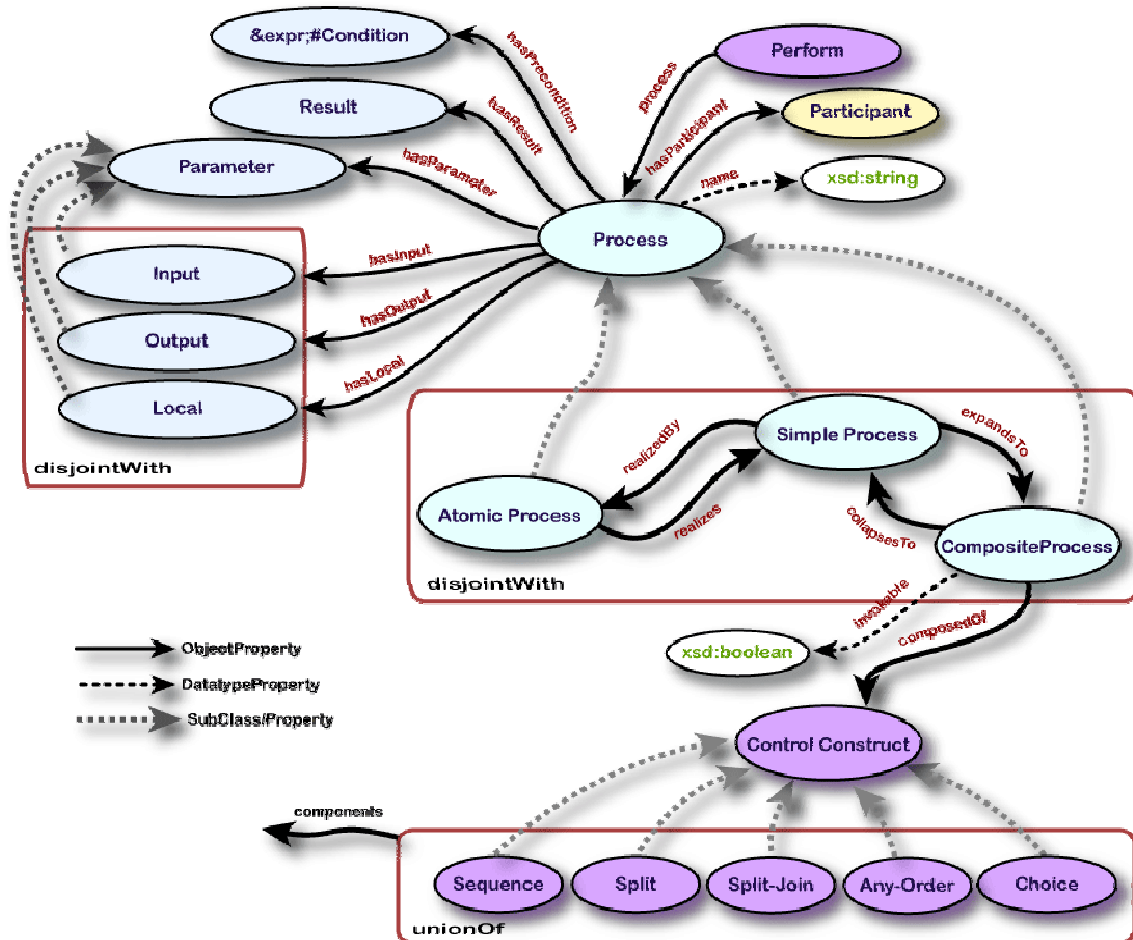


Figure 5. Top Level of the Process Ontology

Three process classes have been defined: the composite, simple and atomic process. The atomic process is a single, black-box process description with exposed IOPEs. Inputs and outputs relate to data channels, where data flows between processes. Preconditions specify facts of the world that must be asserted in order for an agent to execute a service. Effects characterize facts that become asserted given a successful execution of the service, such as the physical side-effects that the execution the service has on the physical world. Simple processes provide a means of describing service or process abstractions – such elements have no specific binding to a physical service, and thus have to be realized by an atomic process (e.g. through service discovery and dynamic binding at run-time), or expanded into a composite process. Composite processes are hierarchically defined workflows, consisting of atomic, simple and other composite processes. These process workflows are constructed using a number of different composition constructs, including: Sequence, Unordered, Choice, If-then-else, Iterate, Repeat-until, Repeat-while, Split, and Split + join.

The profile and process models provide semantic frameworks whereby services can be discovered and invoked, based upon conceptual descriptions defined within Semantic Web (i.e. OWL) ontologies. The grounding provides a pragmatic binding between this concept space and the physical data/machine/port space, thus facilitating service execution. The process model is mapped to a WSDL description of the service,

through a thin grounding. Each atomic process is mapped to a WSDL operation, and the OWL-S properties used to represent inputs and outputs are grounded in terms of XML data types.

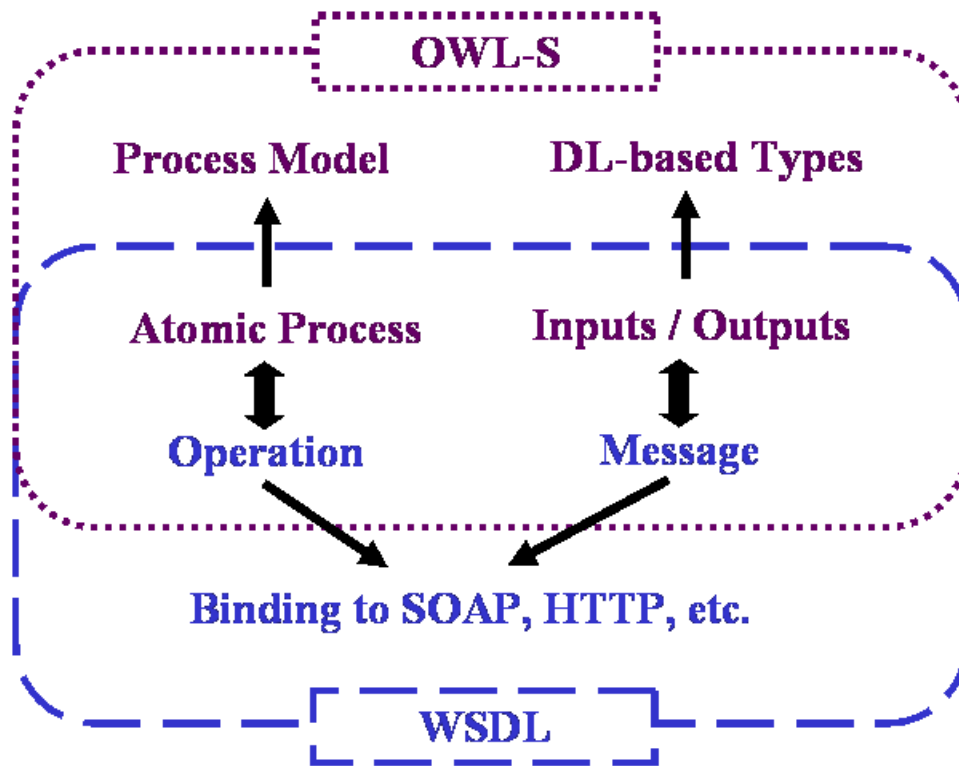


Figure 6. Mapping between OWL-S and WSDL

Additional properties pertaining to the binding of the service are also provided (i.e. the IP address of the machine hosting the service, and the ports used to expose the service).

WSMF

The Web Service Modeling Framework (WSMF) provides a model for describing the various aspects related to Web services. Its main goal is to fully enable ecommerce by applying Semantic Web technology to Web services. WSMF is the product of research on modeling of reusable knowledge components. WSMF is centered on two complementary principles: a strong de-coupling of the various components that realize an e-commerce application; and a strong mediation service enabling Web services to communicate in a scalable manner. Mediation is applied at several levels: mediation of data structures; mediation of business logics; mediation of message exchange protocols; and mediation of dynamic service invocation. WSMF consists of four main elements: ontologies that provide the terminology used by other elements; goal repositories that define the

problems that should be solved by Web services; Web services descriptions that define various aspects of a Web service; and mediators which bypass interoperability problems. WSMF implementation has been assigned to two main projects: Semantic Web enabled Web Services (SWWS) [7] and WSMO (Web Service Modeling Ontology) [8]. SWWS will provide a description framework, a discovery framework and a mediation platform for Web Services, according to a conceptual architecture. WSMO will refine WSMF and develop a formal service ontology and language for SWS. WSMO service ontology includes definitions for goals, mediators and web services. A web service consists of a capability and an interface. The underlying representation language for WSMO is F-logic. The rationale for the choice of F-logic is that it is a full first order logic language that provides second order syntax while staying in the first order logic semantics, and has a minimal model semantics. The main characterizing feature of the WSMO architecture is that the goal, web service and ontology components are linked by four types of mediators as follows:

- OO mediators link ontologies to ontologies,
- WW mediators link web services to web services,
- WG mediators link web services to goals, and finally,
- GG mediators link goals to goals.

Since within WSMO all interoperability aspects are concentrated in mediators the provision of different classes of mediators based on the types of components connected facilitates a clean separation of the different mediation functionalities required when creating WSMO based applications.

WSMX (Web Service Modeling eXecution environment) [9] is the reference implementation of WSMO (Web Service Modeling Ontology). It is an execution environment for business application integration where enhanced web services are integrated for various business applications. The aim is to increase business processes automation in a very flexible manner while providing scalable integration solutions. WSMX internal language is WSML (Web Service Modeling Language). WSMX interprets service requester's goal to:

- discover matching services
- select (if desired) the service that best fits
- provide data mediation (if required)
- make the service invocation

WSMX usage scenario [10] is presented at the Figure 7 below and the overall WSMX bases components and system architecture is shown on the next picture.

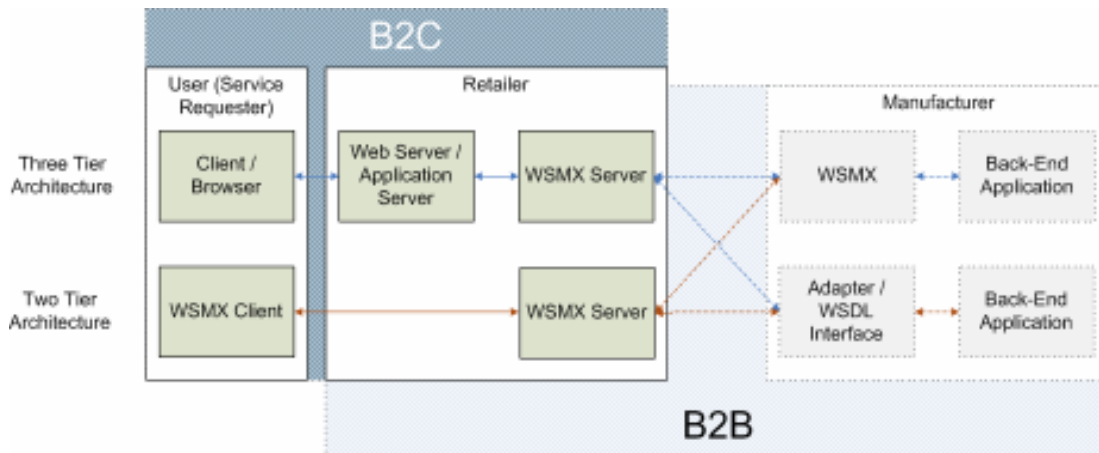


Figure 7. WSMX Usage Example

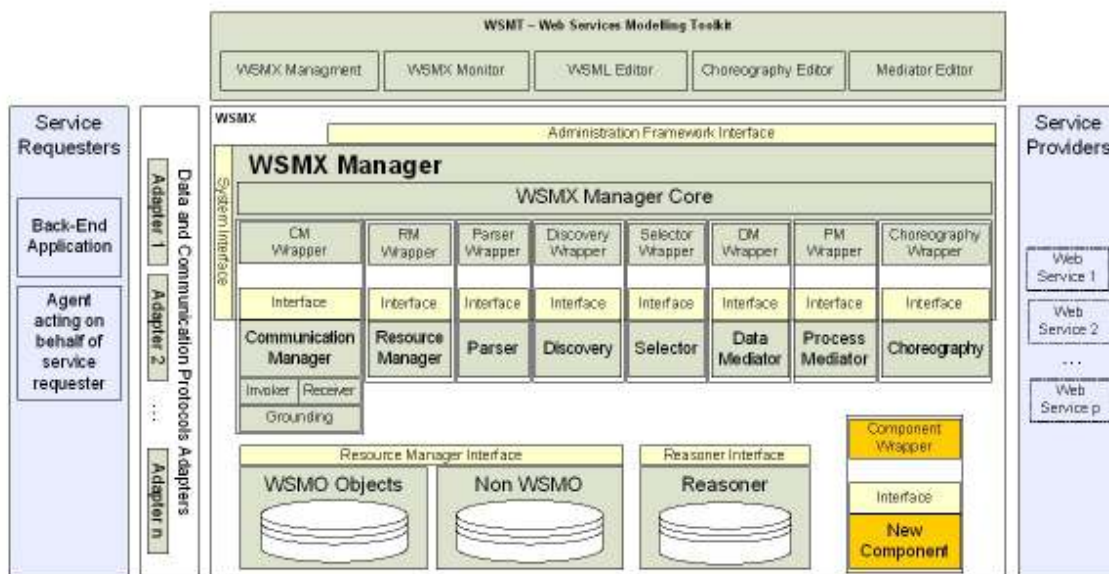


Figure 8. WSMX Components and System Architecture

The Table 1 [5] and Table 2 [10] show comparison between OWL-S and WSMO/WSMF approaches and see [5] for more details on this comparison.

Table 1. Components of different SWS approaches

	IRS-II	OWL-S	WSMF
SWS Activities	Publishing Selection Task Achievement	Composition Discovery Invocation	Discovery
Architecture	Server Publisher Client	Daml-s Virtual Machine Matchmaker	Service Registry Profile Crawler
Service Ontology	Task/PSM Ontology	OWL-S	WSMO
Application tools	IRS Browser and Editor; Publisher; Java API	WSDL2DAML- S	Query interface

Table 2. Comparison between OWL-S and WSMO

	OWL-S	WSMO	<i>current Web Service technologies</i>
Discovery <i>detection of suitable WS</i>	Profile	Goals and Web Services (capability)	<i>UDDI API</i>
Consumption & Interaction <i>How to consume & aggregate</i>	Process Model	Service Interfaces (Choreography + Orchestration)	<i>BPEL4WS / WS-CDL</i>
Invocation <i>How to invoke</i>	Grounding+ WSDL/SOAP	Grounding (WSDL / SOAP, ontology-based)	<i>WSDL / SOAP</i>
Mediation <i>Heterogeneity handling</i>	-	Mediators	-

The main contribution of the OWL-S approach is its service ontology, which builds on the Semantic Web stack of standards. OWL-S models capabilities required for Web services to the extent of grounding, which maps to WSDL descriptions. Additionally, the DAML (DARPA Agent Markup Language) consortium has put a lot of effort in representing the interactions among Web Services through the process model of the OWL-S service ontology. Since the OWL-S service ontology is public and does not

prescribe a framework implementation it has been used as the starting point of individual efforts towards SWS. Nevertheless, the DAML consortium has implemented some components of an architecture based on the DAML inference engine. The invocation activity of OWL-S involves a decomposition of the process model and discovery activity relies on the extension of UDDI registry.

The WSMF approach, although delivering a conceptual framework, invested considerable effort in bringing business requirements into account when proposing a conceptual architecture. Some of the outcomes are still in the form of more detailed specifications. In particular, a service registry has been proposed for which a high level query language is defined according to the service ontology. WSMO distinguished characteristic is the inclusion of mediators in the ontology specification. In common with IRS-II, the WSMF approach builds on the UPML (Unified Problem Solving Method Development Language) framework, taking advantage of the separation of tasks (goals) specifications from the service specifications.

4.0 Conclusion

The state of the art of Semantic Web Services shows that technologies will shape towards accepted enabling standards for Web Services and the Semantic Web. In particular, IRS-II, OWL-S and WSMF promise inter-compatibility in terms of OWL-based service descriptions and WSDL-based grounding [5]. However, an assessment of the delivered results of IRS-II, OWL-S and WSMF approaches show that Semantic Web Services are far from mature. While they represent different development approaches converging to the same objective, they provide different reasoning support, which are based on different logic and ontology frameworks. Furthermore, they emphasize different ontology-based service capabilities and activities according to the orientation of their approaches. While Semantic Web technology positioned well to do the markup and reasoning of Web service capabilities, none of the approaches described provide a complete solution according to the dimensions illustrated, but they show complementary strengths which will continue to grow and merge while Semantic SOA concept and its technical support will continue to mature.

References

1. http://en.wikipedia.org/wiki/Semantic_Web
2. Mei Y. Selva Dan Wolfson Bob Zurek Ed Kahan Achieve semantic interoperability in a SOA
3. OASIS Reference Model for Semantic Services Oriented Architecture docs.oasis-open.org/ex-semantics/sematicsoarm/latest
4. OASIS SOA Reference Model <http://www.oasis-open.org/committees/soa-rm/charter.php>
5. Liliana Cabral, John Domingue, Enrico Motta, Terry Payne and Farshad Hakimpour Approaches to Semantic Web Services: An Overview and Comparisons
6. <http://www.w3.org/Submission/OWL-S/>
7. <http://swww.semanticweb.org/>
8. <http://www.w3.org/Submission/WSMO/>

9. <http://www.wsmx.org/>
10. Michael Stollberg and Armin Haller Semantic Web Services Tutorial 3rd International Conference on Web Services (ICWS 2005) Orlando, Florida, 2005 July 11
11. A.Haller, E. Cimpian, A. Mocan, E. Oren, C. Bussler: [WSMX - A Semantic Service-Oriented Architecture](#), in *Proceedings of the International Conference on Web Service (ICWS 2005)*. Orlando, Florida, 2005.