Semantic Web Services - Concepts and Technology

Michael Stollberg, Cristina Feier, Dumitru Roman, and Dieter Fensel

Digital Enterprise Research Institute
Institut für Informatik, Universität Innsbruck, Austria
{firstname.lastname}@deri.org

Abstract. Although the Internet provides a world wide infrastructure for information provision and communication, the initial web technology stack has substantial draw-backs with regard to automated web content processing. Consequently, the Semantic Web is envisioned as the next evolution step of web technology that shall overcome these deficiencies. Ontologies and Web Services are identified as the key technologies: every resource and every data element interchanged shall be semantically described by ontologies; Web services shall provide access to and usage of computational resources over the Web in order to combine the benefits of the Internet with computational power. The emerging concept of Semantic Web services aims at an integrated technology for amending the Web with semantically enhanced information processing and distributed computing, hence realizing the vision of the Semantic Web. This article explains the idea of Semantic Web services and presents technological solutions developed around the Web Service Modeling Ontology WSMO, a comprehensive framework for Semantic Web Services.

1 Introduction

Invented by Tim Berners-Lee in 1989 as a system for information exchange between researchers, the World Wide Web has established a world wide infrastructure for information provision and communication. The Web is the latest achievement of computer technology with a real impact on society, comparable to the development of affordable personal computers in the 1980ies. Nowadays, there is no computer without Web access, and there is no computer technology without Web support.

The reason for the impressive acceptance and growth of the Web relies on its technical design principles. The foundation is the unambiguous identification of every existing resource via a Uniform Resource Identifier (URI). By combing hypertext technology and internet transfer protocols, the Web allows locating and linking up of arbitrary resources by facile technical means in a decentralized manner [7]. However, the mark-up language for web-content HTML is unstructured which hampers automated web content processing, so that the current Internet is limited to be a world-wide information repository for human consumption only. To overcome these deficiencies, the Semantic Web has been envisioned as the next Web technology evolution step that shall allow automated and seamless information processing and interchange along with computation over the Web. Ontologies and Web services have been identified as the enabling technologies for realizing this vision: the former shall ensure semantic interoperability between Web resources on basis formal terminology definitions, while the latter shall allow usage of computational functionalities over the Internet [6].

Augmenting ontologies and Web services with respective AI-techniques enables more sophisticated Web technology. For turning the Internet into a worldwide infrastructure for distributed computing, respective ontology techniques for semantically enabled information integration or natural language processing allow advanced, semantically enabled information processing over the Web [42]. Moreover, the ultimate aim of Web services is to serve as the basis for dynamic, service-orientated system architectures wherein exactly those services are selected, arranged, and executed that are needed for processing a specific request [17]. For example, consider a langauge technology system that encompasses several components like an automated information extraction tool, some natural language translation facilities, and ontology learning tools. Imagine each component is maintained by different providers and accessible as a Web service. If a user requests a translation from Japanese to English, the system determines the most appropriate Web service for this; for more complex requests like 'extract the information from this Japanese text and let my ontology, which is defined in English learn these', the system shall dynamically determine the usable Web services and execute them in a appropriate order.

While the initial Web service technology stack only provides syntactic means and thus limits Web service usage to manual inspection and integration, the emerging concept of Semantic Web services aims at an integrated technology for realizing the vision of the Semantic Web. Applying ontologies as the underlying data model and on basis of exhaustive semantic description frameworks, intelligent mechanisms are envisioned for automated discovery, composition, contracting, and execution of Web services [35], [21]. This article provides an overview of the concepts and most recent technology developments for Semantic Web services and is structured as follows: Section 2 introduces Semantic Web services, and Section 3 explains their realization within the Web service Modeling Ontology WSMO. Section 4 presents most recent technology developments for Semantic Web services, and Section 5 concludes the article.

2 Semantic Web Services: Aims and Approaches

Ontologies are a state-of-the-art knowledge representation technology developed in the AI-fields of Knowledge Representation and Knowledge Engineering. An ontology is defined as a "formal, explicit specification of a shared conceptualization"[19]. This means that an ontology defines a conceptual model of a domain, representing a commonly accepted consensus among involved parties. An ontology model shall be explicit, i.e. every aspect is modelled precisely without omitting any aspect. This model is formalized into an appropriate ontology language as a machine-readable representation formalism that serves as unambiguous terminology definitions for advanced information processing. Because of these characteristics, ontologies have been identified as a main building block for the Semantic Web. In order to enable semantic interoperability, each Web resource shall be described on basis of ontologies, and every data element exchanged via the Semantic Web is related to an ontology. The World Wide Web Consortium (W3C, website: www.w3c.org) has released a number of technology recommendations for the use of ontology languages on the Semantic Web: the Extensible Markup Language XML for structured web content specification (see: http://w3c.org/XML/), the Resource Description Framework RDF as an initial, low level ontology language [30], and the Web Ontology Language OWL as a comprehensive ontology specification language [34].

However, apart from that sophisticated ontology languages for the Semantic Web are still subject of scientific discussion [15], the usage of ontologies as envisioned for the Semantic Web requires sophisticated means for ontology management and integration. The former refers to facilities for editing, storage, and retrieval, and evolution support for large scale ontologies [45]; the latter is concerned with handling mismatches between heterogeneous ontologies in order to establish semantic interoperability by means of ontology mapping, merging, and alignment [1]. Several research and development efforts are concerned with ontology technologies for the Semantic Web. We observe that methodologies for ontology engineering as well as ontology specifications have reached a preliminary level of maturity [22], while current research and development efforts have identified the requirements for appropriate ontology management technologies and provide prototypical solutions (for example the Ontology Management Working Group OMWG, see website: www.omwg.org).

While ontologies provide the basic means for semantic interoperability and advanced information processing, Web services shall enable computation over the Web. The ultimate goal of Web services is to enable web-based, service-orientated architectures as a novel paradigm of future IT system design [17]. As outlined introductory, Web services shall provide seamless access interfaces to computational facilities accessible on the Web along with means for ad-hoc usage and combination for Enterprise Application Integration (EAI) and E-Commerce as the preliminary application fields [9], [2]. The initial Web service technology stack allows exchange of messages between Web services (SOAP) [25], describing the technical interface for consuming a Web Service (WSDL) [11], and advertising a Web services in registries (UDDI) [13]. However, these technologies only support Web Service usage by manual inspection and integration. In a typical setting, a developer manually searches a Web service that provides the desired functionality in a UDDI repository, then he needs to inspect the WSDL description in order to determine when and how which information need to be interchanged with the Web Service, and then integrate the respective SOAP message handling into the application. Besides, SOAP and WSDL are based on XML as the underlying data model, thus do not support ontologies with respect to semantic interoperability and the vision of the Semantic Web.

Consequently, the emerging concept of Semantic Web services aims at providing more sophisticated Web Service technologies along with support for the Semantic Web. Mentioned first in [35] and [21], Semantic Web services shall utilize ontologies as the underlying data model in order to support semantic interoperability between Web services and its clients and apply semantically enabled mechanisms for automated discovery, composition, conversation, and execution of Web Services. Therefore, exhaustive description frameworks are required that define the semantic annotations of Web services needed for automatically determine their usability.

Chronologically, the first approach for Semantic Web services has been provided by OWL-S [33], the Semantic Web services effort of the DAML-programme (the major US-American Semantic Web research effort, see website: www.daml.org). Using OWL as the description language (s.a.), OWL-S defines an upper ontology for semantically describing Web services that is comprised of three top-level elements: the *Service Profile* holds information for 'service advertisement', containing the name of the service, its provider, a natural language description, and a black-box description of the Service (input and output, and preconditions and effects, short: IOPE); the *Service Model* contains a functional description of a service and its composition out of other services, whereby the service functionality is conceived as a process. The model defines three types of processes (atomic, simple, and composite processes), whereof each construct is described by IOPE as in the Service Profile, with optional conditions over this; and the *Service Grounding* gives details of how to access the service, mapping from an abstract to a concrete specification for service usage. OWL-S provides a default grounding to WSDL (s.a.). Therewith, OWL-S provides a model for semantically describing Web services and serves as a basis for various research and development activities on Semantic Web service technologies [46]. However, OWL-S is criticized for conceptual weaknesses and incompleteness: the meaning of the description elements is not clearly defined and thus used ambiguously, leading to misinterpretations and incompatible service descriptions; furthermore, although OWL-S allows other languages like KIF and SWRL for process descriptions besides OWL, their formal intersection is not defined, hence a coherent formalism for semantically describing Web services is not provided [31].

In contrast, the Web Service Modeling Ontology WSMO [39] defines an overall framework for Semantic Web services consisting of four top level elements: *Ontologies* that provide the semantic terminology definitions used in all other element descriptions as well as for the information interchanged in Web service usage, *Goals* for representing the objective that a client wants to achieve by using Web services, semantic description of *Web services*, and *Mediators* for resolving potentially occurring mismatches between elements that should interoperate. While OWL-S only provides a model for semantically describing Web services, the four WSMO top level elements of WSMO represent the core building blocks for Semantic Web service enabled systems. Hence, the subsequent sections explain Semantic Web services on basis of WSMO in order to provide a comprehensive overview.

3 The Web Service Modeling Ontology WSMO

Taking the Web Service Modeling Framework WSMF as its conceptual basis [21], the WSMO project is an ongoing research and development initiative for defining a capacious framework for Semantic Web services along with a description language (the Web Service Modeling Language WSML [16]) and a reference implementation (the Web Service Execution Environment WSMX [26]).[1]

In order to provide a detailed synopsis of the aims, challenges, and respective solutions for Semantic Web services, the following explains the design principles and element definitions of WSMO in detail. On this basis we explain respective technologies in Section 4.

3.1 Design Principles and Approach

Semantic Web services aim at realizing the vision of the Semantic Web, i.e. turning the Internet from an information repository for human consumption into a world-wide system for distributed Web computing. Therefore, WSMO is based on the following design principles that integrate *Web design principles*, *Semantic Web design principles*, as well as *design principles for distributed, service-oriented computing for the Web.*

Web Compliance: WSMO inherits the concept of IRIs (Internationalized Resource Identifier) for unique identification of resources as the essential design principle of the Web. Moreover, WSMO adopts the concept of Namespaces for denoting consistent information spaces, and supports XML as well as other W3C Web technology recommendations.
Ontology-Based: Ontologies are used as the data model throughout WSMO, meaning that all resource descriptions as well as all data interchanged during service usage are based on ontologies. Following the idea of the Semantic Web, this allows semantically enhanced information processing as well as support for semantic interoperability.
Goal-driven Architecture: User requests are formulated as goals independently of available Web services. Thereby, the underlying epistemology of WSMO differentiates between the desires of clients and available Web services.
Strict Decoupling: Each WSMO resource is specified independently, without regard to possible usage or interactions with other resources. This complies with the open and distributed nature of the Web.
Centrality of Mediation: Mediation addresses the handling of heterogeneities that naturally arise in open environments like the Web. As a complementary design principle to strict decoupling, WSMO recognizes the importance of mediation for the successful deployment of Web services by making mediation a first class component of the framework.

[1] WSMO is a working group of the SDK-Cluster, a joint initiative of European research and development efforts around the Semantic Web and Web services (homepage: www.sdk-cluster.org). All specifications and related information are available at the WSMO homepage: www.wsmo.org.

Description versus Implementation: WSMO differentiates between the *description* and the *implementation* of Web services. The former denotes the unambiguous description of Web services that is needed for automated usage of Web services; the latter is concerned with the internal implementation of the Web Service which is not of interest for Semantic Web service technologies.
Execution Semantics: The formal execution semantics of reference implementations like the Web Service Execution Environment WSMX as well as other WSMO-enabled systems verify the WSMO design and specification.

The design principles are reflected in the four WSMO top level elements shown in Figure 1. Ontologies provide the formal terminology definitions that are used as the data model throughout WSMO, ensuring web compliance by the design of the specification language WSML; Goals are formal specifications of objectives that a client aims to achieve by using Web services, realizing a goal-driven approach that ontologically decouples requesters and providers; WSMO Web Services are formal descriptions needed for automated service handling and usage, whereby the internal implementation of a Web service is not of interest; finally, Mediators are the top level element for handling heterogeneity.

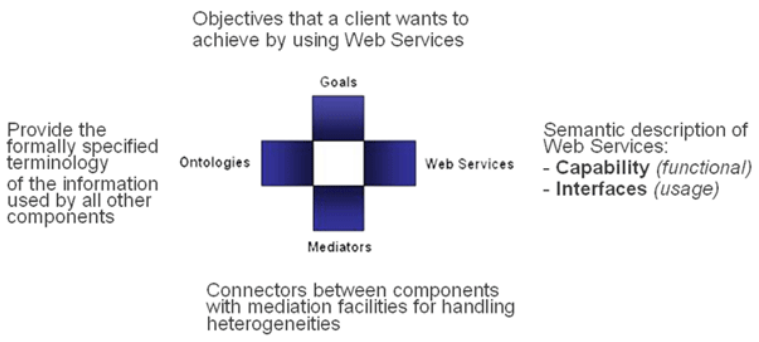
Objectives that a client wants to achieve by using Web Services
Goals
Provide the formally specified terminology of the information used by all other components
Ontologies
Web Services
Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*
Mediators
Connectors between components with mediation facilities for handling heterogeneities

Fig. 1. WSMO Top Level Elements

In terms of the OMG's Meta Object Facility (MOF, a classification framework for system modelling [24]), WSMO provides a comprehensive meta-level ontology for semantically describing the top level elements as the core building blocks of Semantic Web services. Apart from the constitutive description notions for each element, the general structure of WSMO element definition consists four elements: *namespaces* that declare the Web namespaces of used terminology, *importsOntology* and *usesMediator* for denoting the ontologies, respectively mediators used in the element description, and *non-functional Properties* that allow a complete element description used for resource management.[2]

[2] WSMO defines the following non-functional properties (in alphabetical order): Accuracy, Contributor*, Coverage*, Creator*, Date*, Description*, Financial, Format*,

While referring to the WSMO specification for detailed definitions [39], the following explains the WSMO elements with regard to their purpose and constitutive description elements. For exemplifying the modelling of WSMO elements, we consider the following application scenario from the e-tourism domain throughout this section: a user wants to travel from Innsbruck to Venice on a certain date, which is specified as a goal for buying a ticket for the trip. A hypothetical Web service called the "Book Ticket Web Service" is available and capable of solving the goal. This Web service allows searching and buying train tickets for itineraries starting in Austria; the only accepted payment method is credit card that must be a valid visa or mastercard. We refer to [18] for a detailed exemplification of this use case scenario.

3.2 Ontologies

In compliance to the vision of the Semantic Web, WSMO uses ontologies as the underlying data model for Semantic Web services. This means that all resource descriptions and all information interchanged during collaboration execution is based on ontologies, thereby providing the basis for semantically enhanced information processing and ensuring semantic interoperability between Semantic Web services.

In accordance to the AI-theory of ontologies [42], WSMO ontologies consists of the following elements: *Concepts* describe the entities of a domain that are characterized by *Attributes*; *Relations* describe associations between concepts, whereby subsumption and membership relationships define the taxonomic structure of an ontology. An *Instance* is a concrete individual of a concept, and *Axioms* define constraints and complex aspects of the domain in terms of logical expressions. Regarding engineering methodologies developed for the Semantic Web [19], ontology design in WSMO demands and supports *modularization*, i.e. small-sized and concise ontologies, *decoupling*, i.e. distributed and multi-party ontology development, and *ontology mediation* for resolving possibly occurring mismatches between loosely coupled ontologies for a specific usage scenario.

Before discussing the constituting elements of ontologies, the following shows the header of the "Trip Reservation Ontology" that defines the terminology for trip and reservation related information used throughout the running example. This contains the namespace definitions as IRIs (Internationalized Resource Identifier), the IRI of the ontology, and examples for the non-functional properties title, creator, and format. Furthermore, it denotes that two other modular ontologies are imported into the "Trip Reservation Ontology", and that a mediator is used that transforms a person ontology specified in OWL into the required WSML format. The ontology header as well as all other examples are modelled in the Web Service Modeling Language WSML; we refer to [16] for the WSML syntax and semantics definition.

Identifier*, Language*, NetworkRelatedQoS, Owner, Performance, Publisher*, Relation*, Reliability, Rights*, Robustness, Scalability, Security, Source*, Subject*, Title*, Transactional, Trust, Type*, TypeOfMatch, Version; those denoted by (*) are based on the Dublin Core Meta Data Set [47].

```
namespace {_"http://example.org/tripReservationOntology#",
  dc     _"http://purl.org/dc/elements/1.1#",
  loc    _"http://example.org/locationOntology#",
  po     _"http://example.org/purchaseOntology#",
  foaf   _"http://xmlns.com/foaf/0.1/",
  wsml   _"http://www.wsmo.org/wsml/wsml-syntax#",
  prs    _"http://example.org/owlPersonMediator#"
}
ontology  _"http://example.org/tripReservationOntology"
 nonFunctionalProperties
  dc#title hasValue "Trip Reservation Ontology"
  dc#creator hasValue _"http://example.org/foaf#deri"
  dc#format hasValue "text/x-wsml"
 endNonFunctionalProperties

 importsOntology{ _"http://example.org/locationOntology",
                  _"http://example.org/purchaseOntology"}
 usesMediator _"http://example.org/owlPersonMediator"
```

The following provides examples for modeling the constituting elements of ontologies. *Concepts* are defined by their subsumption hierarchy and their attributes, including range specification. The range of the attributes can be a datatype or another concept. The extension of a concept can be defined or restricted by one or more logical expressions embedded in axioms. The corresponding axioms are declared in the `dc#relation` non-functional property of the concept. In the example below, the concept `tripFromAustria` is subsumed by the concept `trip`, and its definition is completed with an axiom, `tripFromAustriaDef`, specifying that for an individual to be an instance of this concept is to have Austria as the value of its `origin` attribute.

```
concept trip
      origin impliesType loc#location
      destination impliesType loc#location
      departure ofType _date
      arrival ofType _date
concept tripFromAustria subConceptOf trip
      nonFunctionalProperties
        dc#relation hasValue tripFromAustriaDef
      endNonFunctionalProperties
axiom tripFromAustriaDef
      definedBy
        forall {?x ,?origin}
          (?x memberOf tripFromAustria
           implies
           ?x[origin hasValue ?origin] and
           ?origin[loc#locatedIn hasValue loc#austria]).
```

Relations describe interdependencies between a set of concepts. A relation declaration comprises the identifier of the relation and (optionally) its arity, its superrelations, the domain of its parameters, and non-functional properties. Like for concepts, axioms can be used for defining or constraining the relation extension. WSMO also supports *Functions* are a special type of relations that have a unary range beside the set of parameters. The example below is a relation with a credit card as its single argument that holds when the credit card is valid. Accompanying this relation is an axiom that compares the credit card expiry date with the current date for establishing its validity.

```
relation validCreditCard(ofType creditCard)
      nonFunctionalProperties
```

```
        dc#relation hasValue ValidCreditCardDef
      endNonFunctionalProperties
axiom ValidCreditCardDef definedBy
      forall {?x, ?y}  (
        validCreditCard(?x)  impliedBy
        ?x[expiryDate hasValue ?y] memberOf creditCard and
        neg (wsml#dateLessThan(?y, wsml#currentDate())).
```

Instances are concrete individuals of concepts or relations, being defined either explicitly by specifying concrete values for attributes or parameters, or by a link to an instance store. The below example shows an instance of the concept `tripFromAustria`.

```
instance tripInnVen memberOf trip
   origin hasValue loc#innsbruck
   destination hasValue loc#venice
   departure hasValue  _date(2005,11,22)
   arrival hasValue  _date(2005,11,22)
```

Axioms denote constraints or complex aspects of the domain, specified as logical expressions. The above definitions present some examples of axioms.

3.3 Web Services

WSMO defines a description model that encompasses those information needed for automatically determining the usability of a Web service. As shown in Figure 2, a WSMO Web service description is comprised of four elements: (1) *non-functional properties*, (2) a *capability* as the functional description of the service; summarized as service interfaces, (3) a *choreography* that describes the interface for service consumption by a client, and (4) an *orchestration* that describes how the functionality of the service is achieved by aggregating other Web services. These notion describe the functionality and behavior of a Web service, while its internal implementation is not of interest.

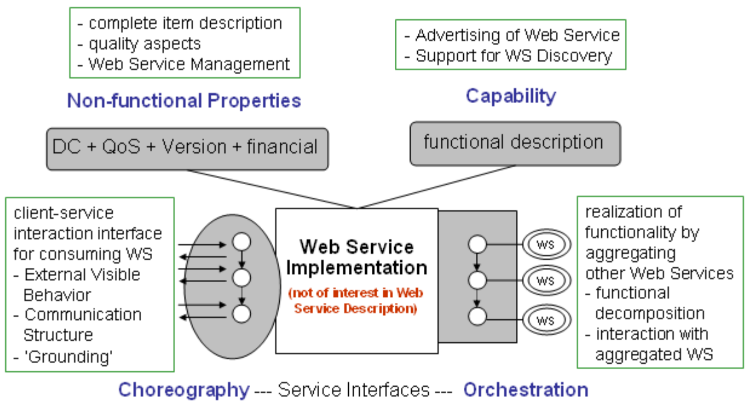
- complete item description
- quality aspects
- Web Service Management
Non-functional Properties
- Advertising of Web Service
- Support for WS Discovery
Capability
DC + QoS + Version + financial
functional description
client-service interaction interface for consuming WS
- External Visible Behavior
- Communication Structure
- 'Grounding'
Web Service Implementation
(not of interest in Web Service Description)
WS
WS
WS
realization of functionality by aggregating other Web Services
- functional decomposition
- interaction with aggregated WS
Choreography --- Service Interfaces --- Orchestration

Fig. 2. WSMO Web Service Description

While the non-functional properties contain an item description, quality of service and financial information, and versioning information of the description used for item management and non-functional selection, the functional service description elements are the Capability and the Service Interfaces. The former describes the functionality of a Web services from a black box perspective for supporting automated functional discovery, meaning to determine whether a Web service can be used to satisfy a user request on basis of its capability. The Service Interfaces describe the interaction behavior of the Web service for consuming, respectively achieving its functionality: a client that wants to utilize the Web service needs to comply with its Choreography Interface; similar, a Web service that realizes it functionality by aggregating other Web services in its Orchestration - which is a main objective of Web service technology - needs to consume these via their respective Choreography Interfaces. This service description model encompasses all information needed for automatically determine usability of a Web service. The following explains the WSMO capability and service interface definitions in more detail.

A capability describes the functionality of a Web service by conditions that need to hold before the service can be executed and by the result that is achieved by service execution. Therefore, WSMO Web Service Capabilities are defined by four notions: *preconditions* for defining conditions on the information space (that is the information used for computation) that have to hold before execution, i.e. on the input required by the service; *assumptions* as conditions on the world (denoting aspects not related to computation) that have to hold before execution; *postconditions* define conditions on the information space after execution, i.e. the output of the service, and *effects* as conditions on the world that hold after service execution. These constitutive elements of capability descriptions are expressed as axioms on ontologies. A set of *shared variables* can be declared which are implicitly all-quantified and whose scope is the whole Web service capability. Informally, the logical interpretation of a Web service capability is that for any values taken by the shared variables, the precondition and the assumption imply the postcondition and the effect.

The below example shows the capability of the "Book Ticket Web Service" in our running scenario. When its execution is successful, the service provides a reservation that includes the reservation holder and a ticket for the desired trip (postcondition) if there is a reservation request for a trip with its starting point in Austria for a certain person (precondition), and if the credit card intended to be used for paying is a valid one, and its type is either visa or mastercard (assumption). As a consequence of the execution of the Web service, the price of the ticket will be deducted from the credit card account (effect).

```
capability BookTicketCapability
    sharedVariables {?creditCard, ?initialBalance, ?trip,
                     ?reservationHolder, ?ticket}
    precondition
       definedBy
          ?reservationRequest[
               reservationItem hasValue ?trip,
               reservationHolder hasValue ?reservationHolder
           ] memberOf tr#reservationRequest and
```

```
        ?trip memberOf tr#tripFromAustria and
        ?creditCard[balance hasValue ?initialBalance
                    ] memberOf po#creditCard.
assumption
   definedBy
     po#validCreditCard(?creditCard)and
     (?creditCard[type hasValue visa] or
      ?creditCard[type hasValue mastercard]).
postcondition
   definedBy
     ?reservation memberOf tr#reservation[
       reservationItem hasValue ?ticket,
       reservationHolder hasValue ?reservationHolder]and
     ?ticket[trip hasValue ?trip] memberOf tr#ticket.
effect
   definedBy
      ticketPrice(?ticket, "euro", ?ticketPrice)and
      ?finalBalance= (?initialBalance - ?ticketPrice)and
      ?creditCard[po#balance hasValue ?finalBalance] .
```

As outlined above, WSMO differentiates two Service Interfaces that are concerned with the interaction behavior of the Web service. The Choreography Interface describes the behavior of the Web Service for consuming its functionality in terms of the information interchange expected, and the Orchestration describes how the Web service interacts with other Web services in order to achieve its functionality.

In contrast to several existing Web service technologies that focus on describing the interaction execution in detail - like WSDL, BPEL4WS, and WS-CDL (see [4] for an extensive discussion) - the aim of WSMO Service Interface descriptions is to provide the foundation for determining automatically whether the interactions between a Web service, its clients, and other aggregated Web services that need to be performed for achieving its functionality. Therefore, WSMO defines a formal model for Service Interface descriptions that supports ontologies as the underlying data model as is based on the Abstract State Machine (short: ASM) framework, a high-level, abstract technique for validating complex systems or programs and provide a highly expressive, flexible, and formally sound means for representing dynamics [8]. According to the ASM principles, WSMO Service Interface descriptions consists of three notions [40]:

– a **Signature** Ω that defines the information space of a service interface on the basis of ontologies. This is defined as the ontological schema of the information interchanged in a service interface by denoting the used concepts, relations, and functions of ontologies. The communicative usage of this ontological schema information is indicated by sub-information spaces for ontologies instances: Ω_{in} denotes the vocabulary of information received by the service interface; Ω_{out} the vocabulary of information that is provided by the service interface; Ω_{shared} denotes the vocabulary of information both received and provided by the service interface; Ω_{static} defines the vocabulary of ontology notions that cannot be changed by the service interface, and $\Omega_{controlled}$ denotes those that can only be changed by the service.
– **States** $\omega(\Omega)$ that denote a status of the information space within the dynamics of a service interface that is defined by the attribute values of the

ontology instances of Ω. A state denotes a stable status within the dynamics of a service interface that is existent as long as attribute values of instances are not changed; changes occur as results of information interchange with interaction partners.

– **Guarded Transitions** T that specify the dynamics of a service interface. The general structure of T is: if *condition*(ω) then *action*, whereby the action denotes the communicative information interchange to be performed for reaching the subsequent state ω'. At a state change from ω to ω', all T are executed whose condition is satisfied.

In principle, this model defines an evolving ontology on the information space that progresses during service usage by the information interchange with interaction partners. Thereby, Ω contains the concepts, relations, functions, and axioms as ontology schema on basis of a domain ontology. A state $\omega(\Omega)$ is stable status of the information space of a service interface, defined by the concrete attribute values of ontology instances; the communicative activities to be performed on instance data are denoted by the sub-information spaces of Ω. The set of guarded transitions T defines state changes with regard to the evolution of the information space. While the general structure of Choreography Interface and Orchestration descriptions is the same, only the actions in Guarded Transitions are modelled differently: while for Choreography Interfaces the action describes the expected communicative interactions in order to reach the next state, defined as updates that define changes in the attribute values of some (or all) instance data of the information space. Actions in Orchestration descriptions consists of so-called operations *operation(WS, update)*, thereby additionally declaring the Web service that the interaction is to be performed with.

The following exemplifies the Choreography Interface definition of the "Book Ticket Web Service" from our running scenario. For the sake of simplicity, we assume the following behavior: at first, the service expects a reservation request for a train trip that starts in Austria. If this is the case, it creates a temporary reservation for that ticket. The next step is to wait for credit card information for payment: if the credit card provided by the client is valid , a reservation is created, otherwise a negative acknowledgement is created. Therefore, the Choreography Interface in this example has three transition rules with respect to the vocabulary definitions. The first one checks whether a reservation request for a trip from Austria instance exists (i.e. it has been already received, since the corresponding concept has the mode *in*). If this is the case, it creates a temporary reservation for that ticket. The second rule creates a reservation for a ticket after reception of a valid credit card for payment by the client, which is sent to the client (denoted by the mode *out* of reservation in the signature). The third rule sends a negative acknowledgement to the client in case of invalid credit card information.

```
choreography BookTicketServiceChoreographyInterface
importsOntology{ _"http://example.org/tripReservationOntology",
                 _"http://example.org/purchaseOntology"}
```

```
vocabulary_in hasValue
 {tr#reservationRequest, tr#tripFromAustria, tr#ticket}
vocabulary_out hasValue
 {tr#reservation, tr#negativeAcknowledgement}
vocabulary_shared hasValue
 {tr#tempoaryReservation, po#creditCard}

guardedTransitions BookTicketChoreographyTransitionRules

 if  (reservationRequestInstance [
          reservationItem hasValue ?trip,
          reservationHolder hasValue ?reservationHolder
      ] memberOf tr#reservationRequest and
     ?trip memberOf tr#tripFromAustria and
     ticketInstance[
         trip hasValue ?trip,
         recordLocatorNumber hasValue ?rln
     ] memberOf tr#ticket
     then
      temporaryReservationInstance[
          reservationItem hasValue ticketInstance,
          reservationHolder hasValue ?reservationHolder
      ] memberOf tr#temporaryReservation

 if (temporaryReservationInstance[
         reservationItem hasValue ticketInstance,
         reservationHolder hasValue ?reservationHolder
     ] memberOf tr#temporaryReservation and
     creditCardInstance memberOf po#creditCard and
     po#validCreditCard(creditCardInstance))
 then
      reservationInstance[
         reservationItem hasValue ticketInstance,
         reservationHolder hasValue ?reservationHolder
      ]memberOf tr#reservation

 if (temporaryReservationInstance [
         reservationItem hasValue ticketInstance,
         reservationHolder hasValue ?reservationHolder
     ] memberOf tr#temporaryReservation and
     creditCardInstance memberOf po#creditCard and
     neg (po#validCreditCard(creditCardInstance)))
 then
      negativeAcknowledgementInstance memberOf tr#negativeAcknowledgement
```

As already said, the "Book Ticket Web Service" accepts credit cards of type visa or mastercard for payment. Assuming that the Web service aggregates other Web services for processing payment, and that these are different for the respective credit card types, it must interact with the correct payment Web service for achieving its functionality during the ticket booking process. Therefore, the "Book Ticket Web Service" defines two sets of transition rules in its orchestration, one for consuming a visa card payment Web service and one for a mastercard payment service. Referring to [40], we omit further examples here.

3.4 Goals

In order to facilitate automated Web service usage and support ontological separation of user desires, service usage requests, and Web service descriptions, Goals in WSMO allow specifying objectives that clients - which can be humans or machines - wish to achieve.

The general structure of WSMO Goal descriptions is similar to Web service descriptions. The client can specify the functionality expected in a *requested capability* that can consist of the same elements as Web service capabilities as discussed above. Also, a Goal can carry information on the expected behavior of an acceptable Web service in so-called *requested interfaces* that can define the excepted communication behavior for consuming a Web service with respect to its Choreography Interfacem as well as restrictions on other Web services aggregated in the orchestration of an acceptable Web service (e.g. only Web services are accepted that utilize a trusted payment facility). It is important to remark that Goal descriptions are defined from the client perspective, thereby decoupled from Web service descriptions.

By intention, WSMO does not prescribe which aspects of goals need to be modelled. For instance in our running example, the user objective is to buy a ticket for traveling from Innsbruck to Venice on a certain date. Therefore, the relevant aspect of a Goal description is the postcondition. As shown is the listing below, this states that as a result of Web service usage, the user wants to receive a reservation for a ticket for the respective trip.

```
goal _"http://example.org/havingAReservationInnsbruckVenice"
  importsOntology {
    _"http://example.org/tripReservationOntology",
    _"http://www.wsmo.org/ontologies/locationOntology"}
  capability
    postcondition
      definedBy
        ?reservation[
           reservationHolder hasValue ?reservationHolder,
           item hasValue ?ticket] memberOf tr#reservation
        and ?ticket[trip hasValue ?trip] memberOf tr#ticket
        and ?trip[
           origin hasValue loc#innsbruck,
           destination hasValue loc#venice,
           departure hasValue 20050715_1800] memberOf tr#trip.
```

The goal resolution process (i.e. how goals are utilized in order to automatically discover, compose, and execute Web Services) is by intention left open to WSMO-enabled applications. Several different possibilities are considered in ongoing applications. Some implementations request complete goals in order to enable completely automated Web service usage (meaning that all aspects of a Goal description should be defined), others utilize Goals with only requested capability definitions along with other means for receiving the input for Web services in the system.

3.5 Mediators

Mediation is concerned with handling heterogeneity, i.e. resolving possibly occurring mismatches between resources that ought to be interoperable. Heterogeneity naturally arises in open and distributed environments, and thus in the application areas of Semantic Web services. Hence, WSMO defines the concept of Mediators as a top level notion.

Mediator-orientated architectures as introduced in [48] specify a mediator as an entity for establishing interoperability of resources that are not compatible a priori by resolving mismatches between them at runtime. The aspired approach for mediation relies on declarative description of resources whereupon mechanisms for resolving mismatches work on a structural, semantic level, in order to allow generic, domain independent mediation facilities as well as reuse of mediators. Concerning the needs for mediation within Semantic Web services, WSMO distinguishes three levels of mediation:

1. **Data Level Mediation -** mediation between heterogeneous data sources; within ontology-based frameworks like WSMO, this is mainly concerned with ontology integration.
2. **Protocol Level Mediation -** mediation between heterogeneous communication protocols; in WSMO, this mainly relates to choreographies of Web services that ought to interact.
3. **Process Level Mediation -** mediation between heterogeneous business processes; this is concerned with mismatch handling on the business logic level of Web services (related to the orchestration of Web services).

With regard to this, WSMO Mediators define a mediation-orientated architecture for Semantic Web services, providing an infrastructure for handling heterogeneities that may arise between WSMO components and implementing the design concept of strong decoupling and strong mediation. A WSMO Mediator serves as a third party component that connects heterogeneous elements and resolves mismatches between them. Figure 3 shows the general structure of WSMO mediators with further explanation below.

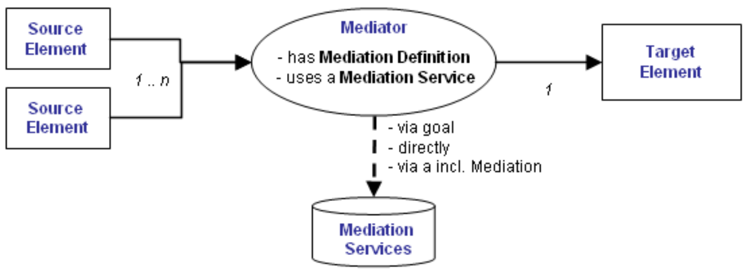

Fig. 3. WSMO Mediator Structure

A Mediator can have several *source elements* that denote the heterogeneous resources that are to be made interoperable, and one *target element* as the resource that applies the mediated source elements. So-called *mediation definitions* define the operations needed to resolve the mismatches between the source components in an appropriate mediation definition language. These operations are executed in a *mediation service* that has processing capability for the mediation definition language. The link to the mediation service used can be defined either directly or indirectly via a goal, including a discovery process for an adequate service to be used, or defined via another mediator that resolves mismatches between the mediator and the mediation service.

Furthermore, WSMO distinguishes four mediator types that connect respective WSMO elements and resolve mismatches between them. The different mediators are named by prefixes, denoting WSMO elements as the respective source and target components. Referring to [41] for further information, the following explains the usage and definition of the WSMO Mediator types.

OO Mediators. OO Mediators are used to establish semantic interoperability between components if this is not given a priori. OO mediation is concerned with ontology integration techniques (i.e. ontology mapping, merging, and alignment), as well as with transformations between languages (transformation between different ontology languages as well as lifting and lowering between structural and ontological languages). Most common usage scenarios for OO Mediators are:

1. for describing a Goal or a Web service, we want to use several modular ontologies O1, O2, O3; therefore, we define an OO Mediator that merges O1, O2, and O3 into one homogeneous ontology that provides the terminology needed to specify the Goal or Web Service.
2. we want to use an ontology defined in OWL in a WSMO Web service description, wherefore we need to transform the OWL ontology into a WSML representation. Therefore, we utilize a "OWL2WSML Mediator" that is specified as follows: the source component is some OWL ontology, and the target component is some WSML ontology. The used mediation service encompasses generic mappings from OWL to WSML (e.g. "owl:Class equals wsmo:Concept"), along with the execution facilities for the transformation mappings. When utilizing this mediator in a concrete usage scenario, we can translate the ontology schema and instance data from the OWL ontology into the WSML representation by execution the mappings.
3. in order to establish interoperability between WSMO components by the other WSMO mediator types, all data level mismatches are handled by (re)using or defining respective OO Mediators.

GG Mediators. GG Mediators specify logical interrelations between Goals in order to allow definition of Goal ontologies. This means that the goal definition process for users is supported by tools on basis of pre-defined Goals, so-called Goal Templates; a user can select a pre-defined Goal and refine or extend it to the concrete objective. For example, there are 2 Goal Templates pre-defined, one "book a flight" and another "book a hotel"; for expressing the objective of booking a trip (consisting of booking a flight and a hotel), these two Goals are combined by a GG Mediator that explicitly states the ontological relationships between the 'source goals' - e.g. that the hotel is located in the destination city of the flight, and the arrival date at the hotel is the same the arrival date of the flight. This supports specifying requirements for Web service composition as well as Web service discovery.

WG Mediators. WG Mediators support establishing the usability of a Web service for resolving a Goal if not given a priori. For example, a Web service provides an online booking facility for train tickets, and a Goal specifies purchase of tickets as a sub-Class of products. A Web service discoverer will not determine this Web service to be usable for resolving the Goal; hence, we define a WG Mediator that resolves terminology mismatches by a respective OO Mediator (stating that a train ticket as specified by the Web service refers to ticket in the Goal), and that the Web service is only usable if the Goal specifies purchase of train tickets, not for other types of tickets. Thus, similar to GG Mediators, WG Mediators explicitly state the ontological relationship between Goals and Web services as auxiliary support for discovery.
WW Mediators. WW Mediators are used to establish interoperability between Web services if not given a priori. With respect to WSMO Service Interfaces, WW Mediators provide mediation facilities on all three mediation levels in order to resolve mismatches that hamper successful interaction of Web services. Apart from using OO Mediators for resolving data level mismatches, the source components are the service interfaces of the Web Services that are ought to interact, and the mediation service encompasses generic mediation patterns for resolving protocol and process level mismatches.

4 Semantic Web Service Technologies

The preceding explorations have introduced the motivation and objectives of Semantic Web services and explained the realization and resource modeling within the Web Service Modeling Ontology WSMO, providing the basis for understanding the requirements for Semantic Web service technologies and position these with respect to thorough next generation Web technologies. This section outlines the overall usage process of Semantic Web services and present recent solutions for selected core technologies.

We can distinguish three phases for the overall usage process of Semantic Web services. At first, service providers as well as potential clients edit the descriptions of their resources and publish them, i.e. making them available on the Web. In the second phase, intelligent mechanisms determine which clients shall utilize which services and resources to achieve their objectives, and the third phase is concerned with execution of Web service usage. Although tool support is required for the editing and publishing phase[3], semantically enabled Web service technologies are mainly allocated in the second and third phase [38].

[3] Several environments for editing, maintaining, and publishing are under development: the Distributed Ontology Management Environment DOME develops an integrated ontology management tool suite for the Semantic Web on basis of WSML (http://dome.sourceforge.net), and the WSMO Studio develops an integrated management environment for Semantic Web services (http://www.wsmostudio.org/); similar tools are under construction for OWL and OWL-S, summarized in the OWL-S Integrated Development Environment (http://projects.semwebcentral.org/projects/owl-s-ide/).

Hence, the main working areas of Semantic Web service technologies for enabling automated Web service usage address the following aspects wherefore we outline most recent approaches below.

– **Discovery and Selection:** how to determine appropriate Web services for solving the goal of a client, and how to select the concrete Web service to be used in case several service can be used?
– **Composition:** if there does not exist a Web service that can completely satisfy a more complex goal, how to determine Web services that can be combined for solving the goal, and how to determine a suitable execution order for these services?
– **Conversation Validation:** given the behavior descriptions of Web services and clients that are ought to interact, how to determine whether the information interchange expected by each party can be achieved successfully?
– **Mediation:** how to resolve mismatches and heterogeneities that hamper Web services and clients to interoperate?
– **Execution Support:** how to manage and control execution of Web services, and how to ensure information interchange with respect to Web scale?

4.1 Discovery

Discovery within Web services is concerned with detecting suitable Web services for achieving a requester's objective, i.e. a goal. As outlined introductory, UDDI supports discovery in conventional Web service technology as follows. The service requester, which in this setting is expected to be a system developer, browses a UDDI repository, retrieving information on available Web services. Then, he manually inspects the usability of a Web service and integrates the respective technical service invocation into the target application. As this technology support appears to be unsatisfactory for automated Web service usage, Semantic Web service discovery aims at automatically determining the usability of a Web service by semantic matchmaking.

Referred to as functional discovery, the general approach is to inspect certain logical relationships between the semantic capability descriptions of requests (i.e. Goals) and Web services. If such a relationship holds, the Web service is considered to be usable for resolving the client's goal. On basis of several preceding works on semantic matchmaking [37], [32], the Web service discovery framework defined for WSMO [28] identifies five matchmaking notions as the core for functional discovery as shown in Figure 4. The important characteristic of these notions is that each one denotes a different logical relationship that has to hold for considering a Web service to be suitable for achieving a given Goal. For instance, the Exact Match holds if and only if for each possible ontology instance that can satisfy the Web service holds that it also satisfies the Goal, and that there exists no possible ontology instance that satisfies only the Goal or the Web service. In contrast, the Intersection Match holds if there exists one possible instance that can satisfy both the Goal and the Web service. Hence, in order to precisely express client objectives with respect to discovery, WSMO

Goals carry an additional non-functional property `typeOfMatch` denoting the matchmaking notion to be applied.

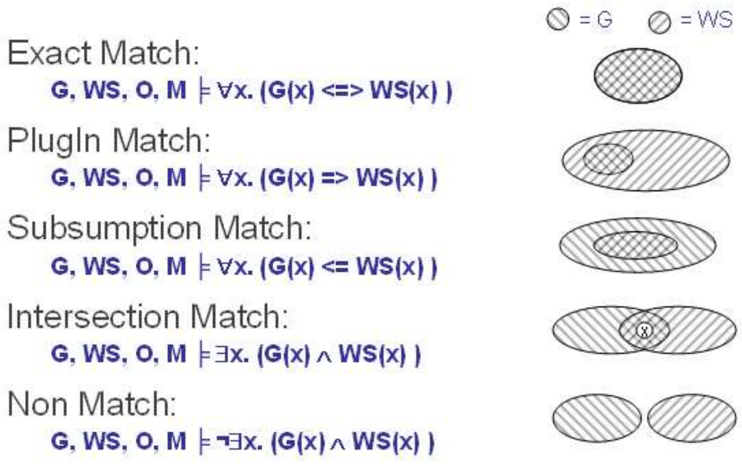
= G
= WS
Exact Match:
G, WS, O, M ⊨ ∀x. (G(x) <=> WS(x))
PlugIn Match:
G, WS, O, M ⊨ ∀x. (G(x) => WS(x))
Subsumption Match:
G, WS, O, M ⊨ ∀x. (G(x) <= WS(x))
Intersection Match:
G, WS, O, M ⊨ ∃x. (G(x) ∧ WS(x))
x
Non Match:
G, WS, O, M ⊨ ¬∃x. (G(x) ∧ WS(x))

Fig. 4. Semantic Discovery Matchmaking Notions

Several prototypical realizations for semantically enabled discovery are existing: [32] and [23] apply Description Logic reasoners, the approach in [29] is based on Frame- and Transaction Logic, and [44] use a FOL theorem prover as the technical platform for semantically enabled discovery. However, all these approaches are restricted to the specific reasoning support provided by the used tools. Hence, it is expected that these approaches will converge towards an integrated discovery framework for Semantic Web services that provides appropriate reasoning facilities.[4]

4.2 Choreography and Orchestration

Another major working area of Semantic Web service technologies is concerned with control and execution of the interactions that need to be performed for consuming Web services.

Therefore, three aspects are differentiated: the *behavior interface* of a Web service or a client that describes the communicative behavior expected or supported for service consumption, so-called *choreographies* that describe the inter-

[4] Description Logic and Logic Programming are the most prominent decidable subsets of First Order Logic; existing reasoners for each subset provide specific reasoning support. For discovery, as well as for several other reasoning tasks on the Semantic, facilities from both fractions are required. Hence, ongoing efforts aim at defining the maximal intersection of Description Logic and Logic Programming languages in order to facilitate integrated reasoning support. See [15] for further discussion.

action protocol between several entities from a global perspective, and *orchestration* that defines how a Web service interacts with other Web services that are used and aggregated to realize the service functionality (see [4] for an exhaustive discussion and analysis of existing technologies).

Classifying most prominent existing related languages and technologies, we can understand the WSDL description of a Web service to be representing its behavior interface by defining the operations (in- and outgoing messages) for consuming its functionality; the Web service Choreography Description Language WS-CDL [27] provides a langauge for describing choreographies as global interaction protocols between several Web services, and BPEL4WS [3] provides a process language along with execution support that can be applied for orchestration. However, these languages relate to the current Web service technology stack, i.e. relying on XML as the data model and missing formal semantics that allow reasoning on interaction descriptions.

As outlined above, the aim of Semantic Web service technologies related to choreography and orchestration is to provide means for determining a priori whether the interactions between a Web service, its clients and aggregated Web services can be performed successfully. As an extension to functional service discovery for detecting usable Web services for resolving a goal as explained above, so-called *choreography discovery* is concerned with determining the existence of a valid interaction protocol between services and clients that are ought to interact on basis of there respective formal interface definitions. This is given if: (1) there exists a valid start state for the interaction, (2) a termination state of the interaction can be reached without any additional input, and (3) if the information to be interchanged between interaction partners in each communicative act are compatible. Choreography discovery provides a technique to automatically determine whether a client can comply with the interaction required for consuming a Web service as defined in its behavior interface, as well as for validating the interactions between the Web service and other services that are aggregated in its orchestration. We refer to [43] for further details and a realization approach for choreography discovery on basis of WSMO Service Interface descriptions as explained in Section 3.3.

Web service composition techniques are related to validating interactions between Services, especially with regard to orchestrations. As outlined above, composition is concerned with determining and aggregating Web services in order to enable resolution of more complex goals. Several approaches have been developed for (semi-)automated Web service composition, ranging from AI-planning techniques that compose services as a sequence of actions, while others compose services for a given goal with respect to behavioral service descriptions (see [5] for an extensive discussion on Web service composition techniques). However, as existing composition techniques are not aligned with Semantic Web service description frameworks like WSMO or OWL-S, the composed Services possibly neglect aspects of interaction validation. Hence, validation of the interaction of a composite Web service and its client as well as with the composed Services needs to be performed after the composition.

4.3 Mediation

While not addressed explicitly in any other approach, WSMO identifies mediation as a central aspect of Semantic Web services. In addition to the ontological definition of Mediators in WSMO (see Section 3.5), the following outlines existing and ongoing work on techniques for the distinct mediation levels identified for Semantic Web services.

Data level mediation refers to the resolution of terminological mismatches that hampers resources from interoperability. Within ontology-based technologies like the Semantic Web and Semantic Web services, this is mainly concerned with ontology mapping, merging, and alignment, summarized as ontology integration techniques [1]. The basis for such techniques is the identification of possibly occurring and resolvable mismatches between heterogeneous ontologies, and an appropriate language for specifying mismatch resolutions along with execution facilities. With respect to previous works on ontology integration, [14] presents an ontology mapping specification language for specifying mismatch resolution between heterogeneous ontologies. This allows defining mappings between ontology concepts, relations, and individuals along with mapping patterns for denoting the ontological relation between elements from different ontologies. The mapping language is language-neutral in order to allow data level mediation between ontologies defined in different ontology languages. An execution environment for this mapping language is under development within the Web Service Execution Environment WSMX, which presents an implementation for WSMO OO Mediators [36].

Process mediation is concerned with resolving behavioral mismatches that hamper entities from interaction with respect to communication protocols and business processes. This relates to choreography, orchestration, and interaction validation as explained above. More precisely: if there does not exist a valid interaction protocol for services and clients that are ought to interact, then a mediator is invoked as a third party component that resolves interaction related mismatches. Similar to the data level, process level mediation relies on a classification of possibly occurring mismatches along with respective mediation patterns and execution facilities for these. We refer to [12] for further elaboration on process mediation along with a prototype implementation.

4.4 Semantic Web Service Execution

Semantic Web services require advanced execution environments. Apart from executing Web services - which means control and monitoring of the information interchange between Web services and their clients while the internal service implementations are executed at their respective physical location - the Semantic Web service usage process needs to be managed and controlled.

The Web Service Execution Environment WSMX develops such an environment as a reference implementation of the WSMO framework: given a WSMO Goal of a requester, WSMX invokes functional components like a service discoverer, mediators, and a service invoker in order to resolve the goal automatically.

The main merits of the WSMX architecture are the event based component architecture along with formal execution semantics that allow dynamic invocation of the system components as needed to process a specific client goal. The ultimate aim of WSMX is to serve as a distributed environment for Web service execution with its functional components to be provided as Semantic Web services themselves [49]. While referring to [26] and the respective specifications of the WSMX architecture and components (see WSMX homepage at: www.wsmx.org), we want to outline a novel approach for communication and information exchange for Semantic Web services developed in WSMX.

Current Web service technologies utilize SOAP as the messaging protocol for exchanging information; apart from not supporting ontologies as the underlying data model, such point-to-point message exchange technologies tightly bind participants to each other in terms of time (HTTP requires participants to communicate synchronously, i.e. to be online at the same time), of reference (messages are addressed to a specific participant, requiring the requester and provider to explicitly know and refer to each other), and of data format (WSDL requires requesters to conform to the data schema defined by the provider). In order to overcome these deficiencies, so-called *Triple-Space Computing* is envisioned [10], [20]: on basis tuple-space computing, wherein communicating parties write information to be exchanged into a common information space wherefrom the respective participants read the data, the interchanged information are ontology instances for supporting semantically enhanced processing. Apart from defeating the limitations on communication support, Triple-Space Computing co-aligns with the Web design paradigm of persistent publishing and reading of information that enables large scale information interchange.

5 Conclusions

This article has introduced the emerging concept of Semantic Web services. Using ontologies as the data model and on basis of exhaustive semantic description frameworks, intelligent techniques shall automatically perform discovery, composition, conversation, and execution of Web services. Supporting seamless and automated information interchange along with distributed computation over the Web, Semantic Web services promise an integrated technology for realizing the vision of the Semantic Web.

In order to give a profound overview and understanding of the objectives, challenges, and solutions for Semantic Web services, we have extensively discussed their realization within the Web Service Modeling Ontology WSMO. We have explained the design principles and motivation for Ontologies, Web services, Goals, and Mediators as the four core elements of Semantic Web services, as well as their definition and modeling in WSMO. On basis of this, we have outlined most recent approaches on semantic discovery, conversation validation, composition, mediation, and execution support as the main areas of research and development within Semantic Web services. Although still under development at the moment, these technologies have the potential to serve as the basis for flexible, dynamic service-oriented architectures on the Web in numerous application areas like Knowledge Management, Enterprise Application Integration, and E-commerce.

Concluding, we remark that Semantic Web services are an ongoing, dynamic research field. Nevertheless, they have the potential of providing a foundational pillar for the next generation of Web technology, as emphasized by the submission of WSMO and OWL-S to the W3C as starting points for upcoming standardization efforts.

Acknowledgements The work presented here is funded by the European Commission under the projects DIP, Knowledge Web, SEKT, ASG, and Esperonto; by Science Foundation Ireland under the DERI-Lion project; and by the Austrian government under the FIT-IT programme. We would like to thank all members of the WSMO, WSML and WSMX working groups for providing input and feedback to this article.

References

1. V. Alexiev, M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen. *Information Integration with Ontologies.* Wiley, West Sussex, UK, 2005.
2. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications.* Data-Centric Systems and Applications. Springer, Berlin, Heidelberg, 2004.
3. T. Andrew et al. (eds.). Business Process Execution Language for Web Services. Specification BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems 1.1, 2003. available at: ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf.
4. A. Barros, M. Dumas, and P. Oaks. Standards for Web Service Choreography and Orchestration: Status and Perspectives. In *Proceedings of the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management at the BPM 2005, Nancy, France*, 2005.
5. D. Berardi. Automatic Service Composition. Models, Techniques and Tools. Phd thesis, University of Rome "La Sapienza", 2005.
6. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), 2001.
7. Tim Berners-Lee. *Weaving the Web.* Harper, San Francisco, USA, 1999.
8. E. Boerger and R.F Staerk. *Abstract State Machines. A Method for High-Level System Design and Analysis.* Springer, 2003.
9. C. Bussler. *B2B Integration: Concepts and Architecture.* Springer, Berlin, Heidelberg, 2003.
10. C. Bussler. A Minimal Triple Space Computing Architecture. In *Proceedings of the 2nd International WSMO Implementation Workshop (WIW 2005), Innsbruck, Austria*, 2005.
11. R. Chinnici, M. Gudgin, J-J. Moreau, J. Schlimmer, and Weerawarana. S. (eds). WSDL. Working draft, W3C, 2004. Available from http://www.w3.org/TR/wsdl20.
12. E. Cimpian and A. Mocan. WSMX Process Mediation Based on Choreographies. In *Proceedings of the 1st International Workshop on Web Service Choreography*

and Orchestration for Business Process Management at the BPM 2005, Nancy, France, 2005.
13. L. Clement, A. Hately, C. von Riegen, and T. (eds) Rogers. UDDI Version 3. Uddi spec technical committee draft, OASIS, 2004. Available from http://uddi.org/pubs/uddi_v3.htm.
14. J. de Bruijn and A. Polleres. Towards an Ontology Mapping Specification Language for the Semantic Web. Technical Report 2004-06-30, DERI, 2004. available at: http://www.deri.at/publications/techpapers/documents/DERI-TR-2004-06-30.pdf.
15. J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In *Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan, 2005.*, 2004.
16. J. de Bruijn (ed.). The Web Service Modeling Language WSML. WSML Deliverable D16.1 final version 0.2, 2005. available from http://www.wsmo.org/TR/d16/d16.1/v0.2/.
17. T. Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, 2004.
18. C. Feier (ed.). WSMO Primer. Deliverable D3.1, final version 0.1, 2005. Available from http://www.wsmo.org/TR/d3/d3.1/.
19. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and E-Commerce*. Springer, Berlin, Heidelberg, 2 edition, 2003.
20. D. Fensel. Triple-Space Computing: Semantic Web Services based on persistent publication of information. In *Proceedings of the IFIP International Conference on Intelligence in Communication Systems, INTELLCOMM 2004, Bangkok, Thailand, November 23-26.*, 2004.
21. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
22. A. Gomez-Perez, O. Corcho, and M. Fernandez-Lopez. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and Semantic Web*. Series of Advanced Information and Knowledge Processing. Springer, Berlin, Heidelberg, 2003.
23. S. Grimm, B. Motik, and C. Preist. Variance in e-business service discovery. In *Proc. of the ISWC 2004 workshop on Semantic Web Services: Preparing to Meet the World of Business Applications, Hiroshima, Japan, Nov. 2004*, 2004.
24. OMG: The Object Management Group. Meta-object facility, version 1.4, 2002.
25. M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, and H.F. Nielsen, editors. *SOAP Version 1.2*. 2003. W3C Recommendation 24 June 2003.
26. A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. In *Proceedings of the International Conference on Web Service (ICWS 2005)*, 2005.
27. N Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon (eds.), editors. *Web Services Choreography Description Language Version 1.0*. 2004. W3C Working Draft 17 December 2004.
28. U. Keller, R. Lara, and A. Polleres (eds.). WSMO Web Service Discovery. Deliverable D5.1, 2004. available at: http://www.wsmo.org/TR/d5/d5.1/.
29. M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A Logical Framework for Web Service Discovery. In *Proc. of the ISWC 2004 workshop on Semantic Web Services: Preparing to Meet the World of Business Applications, Hiroshima, Japan, Nov. 2004*, 2004.

30. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and abstract syntax. Recommendation 10 February 2004, W3C, 2004.
31. R. Lara, D. Roman, A. Polleres, and D. Fensel. A Conceptual Comparison of WSMO and OWL-S. In *Proc. of the European Conf. on Web Services*, 2004.
32. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary.*, 2003.
33. D. Martin, editor. *OWL-S 1.1 Release.* 2004. http://www.daml.org/services/owl-s/1.1/.
34. D. McGuinness and F. v. Harmelen, editors. *OWL Web Ontology Language Overview.* 2004. W3C Recommendation 10 February 2004.
35. S. McIlraith, T. Cao Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16(2):46–53, 2001.
36. A. Mocan (ed.). WSMX Data Mediation. WSMX Working Draft D13.3, 2005. available at: http://www.wsmo.org/TR/d13/d13.3/v0.2/.
37. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference, Springer*, 2002.
38. C. Preist. A Conceptual Architecture for Semantic Web Services. In *Proc. of the Int. Semantic Web Conf. (ISWC 2004)*, 2004.
39. D. Roman, H. Lausen, and U. Keller (eds.). Web Service Modeling Ontology (WSMO). Deliverable D2, final version 1.2, 2005. Available from http://www.wsmo.org/TR/d2/.
40. D. Roman, J. Scicluna, and C. Feier (eds.). Ontology-based Choreography and Orchestration of WSMO Services. Deliverable D14, 2005. available at: http://www.wsmo.org/TR/d14/.
41. F. Scharffe (ed.). WSMO Mediators. Deliverable D29, 2005. available at: http://www.wsmo.org/TR/d29/.
42. S. Staab and R. Studer, editors. *Handbook on Ontologies in Information Systems.* International Handbooks on Information Systems. Springer, 2004.
43. M. Stollberg. Reasoning Tasks and Mediation on Choreography and Orchestration in WSMO. In *Proceedings of the 2nd International WSMO Implementation Workshop (WIW 2005), Innsbruck, Austria*, 2005.
44. M. Stollberg, U. Keller, and D. Fensel. Partner and Service Discovery for Collaboration Establishment on the Semantic Web. In *Proceedings of the Third International Conference on Web Services, Orlando, Florida*, 2005.
45. M. Stollberg, H. Lausen, R. Lara, Y. Ding, H. Sung-Kook, and D. Fensel. Towards Semantic Web Portals. In *Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, NY, USA, May 18, 2004.*, 2004.
46. K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated Discovery, Interaction and Composition of Semantic Web services. *Journal of Web Semantics*, 1(1):27–46, September 2003.
47. S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core Metadata for Resource Discovery. Technical Report RFC 2413, 1998. available at: http://www.faqs.org/rfcs/rfc2413.html.
48. G. Wiederhold. Mediators in the architecture of the future information systems. *Computer*, 25(3):38–49, 1994.
49. M. Zaremba and C. Bussler. Towards Dynamic Execution Semantics in Semantic Web Services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics: Towards Dynamic Business Integration*, 2005.