Semantics for Service-Oriented Architectures

Michael Stollberg, Martin Hepp, and Dieter Fensel

Semantic Technology Institute (STI),
University of Innsbruck, Austria

Abstract. The concept of Service-Oriented Architectures (SOA) is the latest design paradigm for IT systems. The aim is to use Web services as the basic building blocks, which provide reusable functionalities that are invocable of the Internet. The initial Web service technology stack around WSDL, SOAP, and UDDI allows to technically realize the provision and usage of Web services. However, the support for the detection of the suitable Web services for a specific client application is limited to manual inspection. To better support this for SOA applications with larger numbers of available Web services which can be expected in real-world scenarios, the emerging concept of Semantic Web services (SWS) develops inference-based techniques for the automated discovery, composition, and execution of Web services. This article provides an overview on the SWS approach as well as the latest technology developments.

1 Introduction

The concept of Web services as been invented by a consortium of leading IT vendors in the late 1990ies. Essentially, a Web service is a program that can be invoked over the Internet. It is accessible via an interface that specifies the physical address as well as the messages via which a client can consume the Web service. The actual consumption is realized by the exchange of XML data over the Web. This technology is independent of the actual implementation, and allows to exchange information over the Internet and to as well as to seamlessly reuse functionalities within and in between organizations.

Because of this, Web services have been proclaimed as the core technology for Service-Oriented Architectures (SOA): future IT systems shall be composed of Web services as the basic building blocks instead of proprietary solutions. The aim is to exploit the potential of the World Wide Web (WWW) as an infrastructure for computation, and also to reduce the development and maintenance costs for IT systems. The adaptation of Web services and the SOA paradigm within industry as well as by non-profit software developers has been facilitated by the early standardization of the necessary technologies. Commonly referred to as the initial Web service technology stack, these are (1) the *Web Service Description Language* (WSDL) for specifying the technical information as well as the messages for invoking and consuming a Web service, (2) SOAP as a messaging technology for exchanging XML data over the Web, and (3) the *Universal Description, Discovery and Integration Protocol* (UDDI) which provides a registry technology for Web services.

This allows service providers to offer functionalities as Web services, and also supports the technical usage of Web services by clients. However, the descriptions remain on a syntactic level which limits the Web service usage to manual inspection: the developer of a client application needs to search a suitable Web service within a UDDI repository, then inspect the WSDL description in order to determine how and in which order the necessary messages shall be exchanged, and finally integrate the Web service invocation into the application.

In order to overcome these deficiencies, the emerging concept of Semantic Web services (SWS) develops techniques for better supporting the detection and usage of Web services on the basis of semantic descriptions. The aim is to better support and eventually automate the Web service usage process, and to facilitate the dynamic detection and execution of the necessary Web services for solving a particular client request within SOA systems. For this, inference-based techniques for the automated *discovery* as the detection of suitable candidates out of the available Web services, *composition* as the automated combination of several Web services, and the *automated execution* of Web services are developed. The SWS approach uses ontologies as the underlying data model, which are formally specified knowledge models propagated as the base technology for the Semantic Web, another amendment of the existing Web technologies.

This article provides an overview of the SWS approach as well as the latest technology developments. At first, Section 2 explains the initial Web service technology stack and outlines the vision of Service-Oriented Architectures. Then, Section 3 introduces the concept and the most prominent frameworks for Semantic Web services, and Section 4 presents recent developments on SWS techniques for automating the detection, usability analysis, and execution of Web services. Finally, Section 5 summarizes the article and outlines perspectives for the future development and standardization of semantic SOA technologies.

2 Web Services and SOA

The following explains the concept of Web services and the initial technology stack, and discusses the intended usage of Web services as the base technology for Service-Oriented Architectures (SOA). We also identify the deficiencies of the initial Web service technology stack in order to motivate the need for semantic technologies to enhance the quality of SOA systems.

2.1 Web Services

The concept of Web services has been invented in the late 1990ies by a mostly industry-driven initiative. The aim was to define a new technology that on the one hand makes use of the WWW as an infrastructure for computation, and, on the other hand, allows to effectively tackle the intra- and inter-organizational integration of information and services. For this, three contiguous technologies have been specified which are commonly referred to as the initial *Web service technology stack*: WSDL as the language for describing the interface of a Web

service, SOAP as a messaging protocol for exchanging XML data over the Web, and UDDI as a registry technology for Web services. These have been published by the World Wide Web Consortium W3C (see `www.w3.org`), respectively by OASIS as a most industry driven standardization body (`www.oasis-open.org`). The standardized specifications have been first released in the years 2000 to 2002; the latest versions have been published in 2007.

The following explains the basics of the three central Web service technologies. We refer to the technical specifications as well as to extensive secondary literature for details, e.g. [5,24,44].

Web Service Description Language (WSDL) [11]. As the heart of Web service technology, this is an XML-based language for describing the interface of a Web service. Essentially, a WSDL description specifies the supported operations for invoking and consuming the Web service, its physical location, and it supports bindings to several transport protocols and formats for the actual information exchange between the Web service and the requester.

The WSDL description of a Web service is defined as an XML document that consists of the following elements as illustrated in Figure 1. The *service* element describes the name and the physical location of the Web service, mostly in form of a URI. A Web service can have several physical endpoints. These are called *ports*, for which a *binding* defines the supported transport protocols and formats. While this specifies how to carry out the actual information exchange, the *port type* element specifies the set of operations that are supported by the Web service. An *operation* consists of a set of messages and their direction (i.e. in- or outgoing). A *message* describes the data being communicated between the requester and the provider. The message content is described in terms of XML Schemas; the *type* element allows to specify the complex data types used in the WSDL description.

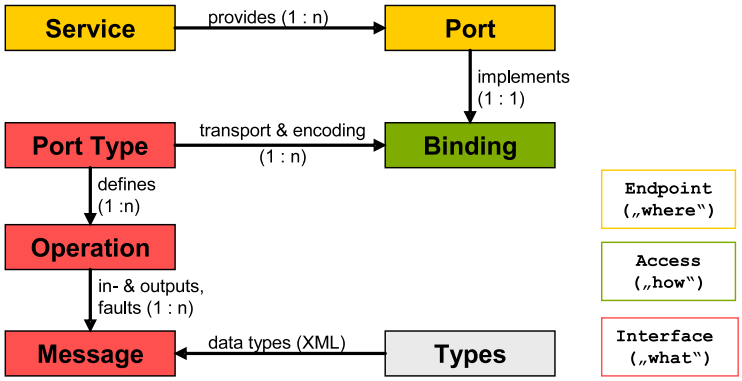

Fig. 1. Ingredients of WSDL

The main merit is that WSDL is independent of the technologies used for the actual implementation of a Web service. Thus, in principle any program can be provided as a Web service by defining a WSDL description. This is supported by existing development environments, e.g. the JAVA2WSDL tool from the Apache AXIS tool kit which allows to automatically generate the WSDL description for a Java program (see `http://ws.apache.org/axis/`).

SOAP [51]. Formerly the abbreviation for *Simple Object Access Protocol*, this is a messaging technology for exchanging XML data over the Web. Although SOAP is not restricted to the context of Web services, it has become the standard communication protocol for consuming Web services by the exchange of messages over the Internet.

As outlined above, every operation in a WSDL description is associated with one or more messages. To consume a Web service, these need to be instantiated with concrete values and then are exchanged between the endpoints via a specific transport protocol. While in the context of Web services SOAP is mostly bound to HTTP in order to allow document exchange over the WWW, it can also be bound to other transport protocols.

A SOAP message is a XML document which consists of a *header* with technical information, and a *body* that carries the actual content in form of XML data. This is wrapped into an envelope, which then can be bound to a transport protocol for conducting the actual information exchange. Listing 1 shows an example for a SOAP message for invoking a Web service for weather forecast. These messages are processed by respective SOAP engines, which denote the heart of execution environments for Web services.

```
<?xml version="1.0" encoding="UTF-8"?>
 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetWeather xmlns="http://www.webservicex.net">
      <CityName>Innsbruck</CityName>
      <CountryName>Austria</CountryName>
    </GetWeather>
  </soap:Body>
</soap:Envelope>
```

Listing 1. An example SOAP message

Universal Description, Discovery and Integration Protocol (UDDI) [18]. This is a registry technology intended to support publishing, management, and discovery of Web services. It defines a generic data model for describing Web services with respect to the providing business entity, the technical access information, a natural language description, and a keyword-based classification scheme. In addition, the detailed specification of Web services can be bundled in so-called technical models. The specification comes along with an API in order to support programmatic access to UDDI registries.

The purpose of a UDDI registry is to allow service providers to publish and advertise their Web services, and also to facilitate the search and inspection of

suitable Web services by clients. Initially, big vendors such as Microsoft, SAP, and IBM maintained the UDDI Business Registry (UBR) as a single repository for publicly available Web services. However, this effort has been abandoned because the used categorization scheme as well as the UDDI support for publishing and searching Web services occurred to be insufficient. Nowadays, most SOA systems employ registry techniques that are specialized for the specific application scenario. Nevertheless, these proprietary registries follow the principles of UDDI – i.e. describing and organizing Web services in a classification scheme to support clients in the detection of the suitable Web services.

Concluding, the initial Web service technology stack is comprised of three cohesive, standardized technologies:

1. WSDL as the standardized description language for Web services
2. SOAP as the communication protocol for executing Web services
3. UDDI as a registry technology for publishing and searching Web services.

In addition, several accessory technology standards have been specified, which are concerned with usage policies, addressing schemes, security, and other aspects that occur to be relevant for real-world applications [71]. A reliable indicator for the thorough adaptation and success of Web services is that essentially all big software vendors committed to this technology.

2.2 Service-Oriented Architectures

The invention of Web services and the standardization of necessary technologies has initiated the concept of Service-Oriented Architectures (SOA) as a new IT system design paradigm [24]. The idea is to use Web services as the basic building blocks of software systems in order to exploit the potential of this new technology. The motivation for this is manifold:

- software fragments from distributed locations that are offered as Web services can be seamlessly integrated, which allows to ease the aggregation of services from different providers [5]
- Web services can help to reduce the development and maintenance costs of IT systems by reuse of existing services and by flexible replacement [44]
- Web services allow to tackle the integration problem, i.e. the exchange of data and services between business partners that use different technologies: if two businesses agree on a common data model and provide their public processes as Web services, then the relevant information can be interchanged while the internal processes remain unchanged [13].

The initial Web service technology stack as explained above provides a suitable basis for realizing the SOA vision, and the standardization has triggered major research and development efforts in industry as well as in academia. Existing SOA technologies range from freely available tools (e.g. the Methods Web service browser) and open source development kits (e.g. AXIS from Apache) to

exhaustive development and management environments from the major software vendors, e.g. the Microsoft's .NET framework, IBM's WebSphere, Oracle's SOA Suite, NetWeaver (SAP), or Crossvision (Software AG). Moreover, the rising interest in Web service and SOA has led to further technology developments such as the integration into business process management (e.g. BPEL4WS, [6]) as well as to service orientation as a new business model [4].

However, the development of sophisticated SOA technologies is an immense challenge. A central challenge is the adequate support for the detection of suitable Web services for a concrete client application. This requires an appropriate description that allows clients to determine whether a Web service is actually suitable for the given problem, and SOA systems should support this in an adequate manner. We discuss the deficiencies of the basic Web service technologies for the usability analysis in more detail, which will reveal the motivation for Semantic Web services that we shall discuss in the next section.

Figure 2 illustrates the procedure of Web service usage by clients on the basis of WSDL, SOAP, and a registry technology like UDDI. The client – which in most cases is the developer of an application wherein Web services shall be used – wants to find a suitable Web service for a certain problem setting. As the first step, the client searches the UDDI registry of the available Web services. When a candidate has been found, its actual usability must be determined. This means that the client needs to figure out in what order which messages with what content and under which transport binding must be exchanged with the Web service in order to consume the desired functionality. The relevant information for this is available in the WSDL description of a Web service. However, the client needs to manually analyze the supported operations as well as the required data in order to determine how to invoke the Web service in a way such that it will solve the given task. This problem remains when using automatically generated client stubs for WSDL descriptions, because the generated code merely reflects the description in a programmatic environment. Once the usability analysis is completed successfully, the Web service can be invoked and consumed over the specified binding (which usually is SOAP as explained above).

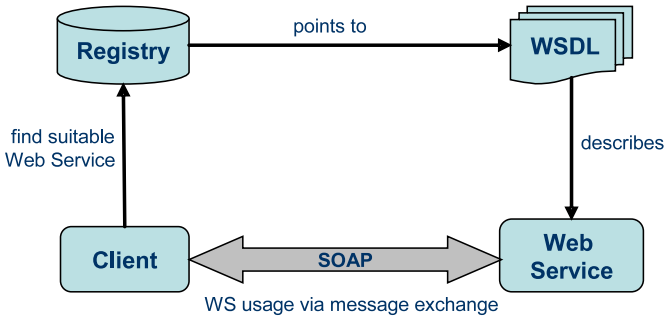

Fig. 2. Web Service Usage Procedure

Obviously, the outlined procedure can not be considered to provide sophisticated support for the detection of suitable Web services, because most of the usability analysis tasks are left to manual analysis by the client. Moreover, several problems may occur during the analysis, e.g. that the classification scheme in the repository is too inexpressive so that the candidate search result is imprecise, or that the data of the client and the Web service are incompatible. Thus, more appropriate technologies are needed for supporting Web service detection and the usability analysis, which is at least as important for realizing the SOA vision as the technical infrastructure for the publication and consumption of Web services. One prominent approach that addresses this problem is the emerging concept of Semantic Web services that we will explain in the following.

3 Semantic Web Services

The aim of Semantic Web services (SWS) is to overcome the deficiencies of the initial Web service technologies, especially for the service detection and usability analysis as discussed above. The approach is to extended Web service descriptions with sufficiently rich annotations and, upon this, provide inference-based techniques for automating the detection and usage of Web services [49,27]. Several research and development efforts work on SWS technologies, and there exists a wealth of work on this. We here provide a concise overview, referring to exhaustive literature for further details (e.g. [15,28,65]).

Essentially, SWS technologies apply reasoning techniques on formalized descriptions in order to better support the usability analysis of Web services and also to handle the integration problem on a semantic level. The primary tasks that can beneficially be supported by SWS technologies are *discovery* as the detection of suitable Web services for a given task, *composition* as the combination of several Web services to solve a more complex task, and *mediation* as the handling of heterogeneities that may occur between the requester and the provider. For this, the SWS approach extends Web service descriptions as follows:

1. Instead of XML, **ontologies are used as the data model** for describing Web services. These provide formalized knowledge models of a domain that allow advanced information processing. Moreover, this pursues the alignment of Web services with the Semantic Web for which ontologies are considered as the base technology (see below).
2. Apart from non-functional aspects such as the owner, usage rights, quality-of-service and financial information, also the **provided functionality** of a Web service is formally described. The primary purpose is to support semantic matchmaking techniques for more precise Web service discovery.
3. The **Web service interface for consumption**, i.e. the WSDL description, is formally described in order to support automated compatibility analysis of the communication behavior supported by the client and the Web service.
4. In addition, the **aggregation of Web services** describes how a complex Web service achieves its functionality by combining several other Web ser-

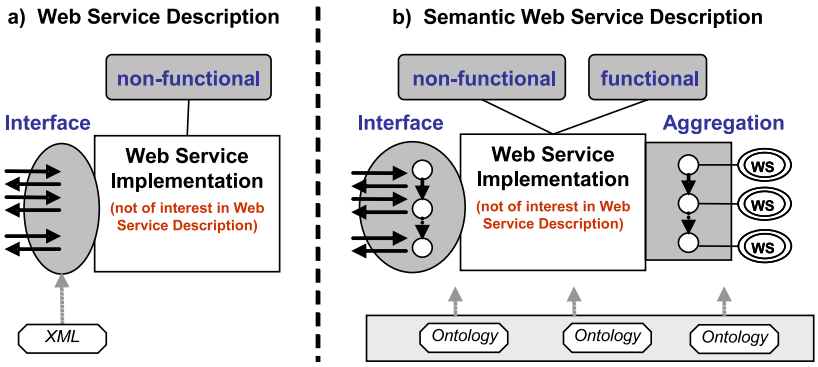

Fig. 3. From Web Services to Semantic Web Services

vices. This aims at automated techniques for analyzing the executability of Web service aggregations in more complex SOA applications.

The following explains the foundations of the SWS approach in more detail. We commence with the Semantic Web and ontologies, and then present the most prominent SWS frameworks that have been developed in the last years. We shall discuss the state-of-the-art in the development of SWS techniques for automating the detection and execution of Web services in the next section.

3.1 Ontologies and the Semantic Web

Ontologies are a modern AI knowledge representation technique. They have been identified as the base technology for the Semantic Web – the grand vision for the further evolution of the WWW [9] – and they are used as the formalized domain knowledge specifications for SWS descriptions. The following explains the definition and the benefits of ontologies as formalized knowledge models, and depicts the status of Semantic Web technology developments.

Adopting the denotation from the philosophical study of being and existence, an ontology is defined as a "formal, explicit specification of a shared conceptualization" [30]. This means that an ontology defines a conceptual model of a domain that ideally represents an agreed consensus among involved parties. The conceptual model is defined in terms of *concepts* that denote the entities in the domain of discourse. These are associated with *attributes* for describing specific properties, and *relations* allow to specify the relations between concepts. The subsumption and membership relations define the taxonomic backbone of the ontology. In addition, further knowledge on the domain can be specified in terms of logical statements referred to as *axioms*. Individuals in the domain are represented as *instances* of a concept. The conceptual model is then represented in a formal, machine-processable language upon which reasoning techniques can be applied for advanced information processing. The major merit is that ontologies provide a technology independent model of the domain of discourse, which

allows to better bridge the gap between the real world and IT systems [26]. Furthermore, ontologies allow to integrate heterogeneous data on the semantic level by defining mappings between ontologies [3].

The Semantic Web envisions that Web resources are described on the basis of ontologies, and then to exploit their potential for advanced and meaning-preserving information processing. Proposed by Tim Berners-Lee – inventor of the WWW and director of the W3C – this is embedded in a larger vision for subsequently augmenting the current WWW with additional languages and technologies that shall be standardized by the W3C. Figure 4 shows the so-called *Semantic Web Layer Cake* that illustrates the overall vision: the bottom layers are the already existing WWW technologies (URI, XML, Namespaces). Upon this, several ontology languages are defined that are the current focus of standardization work. Op top of this, languages for proof and trust on the Web are targeted as future work.[1]

The Semantic Web has received high interest in academia and industry, resulting in a steadily growing, international research community. This has produced a wealth of work that mainly covers (1) formal ontology languages (e.g. [21]) and efficient reasoning techniques (e.g. [53]); (2) ontology management technologies [34], i.e. methodologies and tools for ontology engineering [29], scalable ontology repositories (e.g. [33]), and ontology evolution support (e.g. [23]); (3) ontology-based data integration techniques (e.g. [56]); (4) applications that demonstrate the benefits of Semantic Web technologies [20].

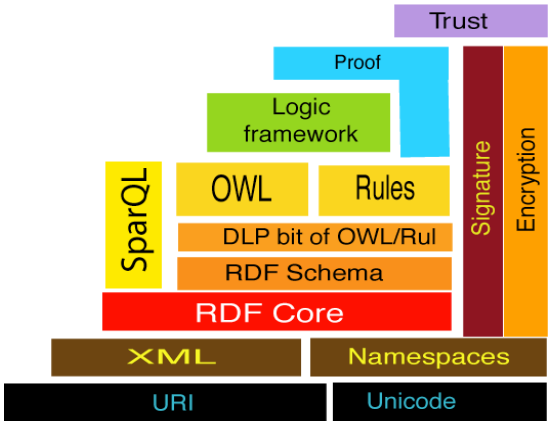

Fig. 4. The Semantic Web Layer Cake (revised version, 2005)

[1] Figure 4 is taken from a keynote talk by Tim Berners-Lee, see `http://www.w3.org/2005/Talks/0511-keynote-tbl/`. At the time of writing, W3C standard recommendations exist for the Resource Description Framework RDF (see `www.w3.org/RDF/`), the Web Ontology Language OWL [48], and the RDF query language SPARQL [43]; standardization work on a rule language is ongoing, e.g. in the RIF working group (see `http://www.w3.org/2005/rules/wg`).

3.2 SWS Frameworks

We now present SWS frameworks that define comprehensive specifications for semantically describing Web services, in general following the approach as outlined above. The aim is to provide an overview of the conceptual frameworks that most of the research on Semantic Web services is based upon. As the most relevant ones, we here depict the approaches that have been submitted to or published by acknowledged standardization bodies.

OWL-S [46]. As the chronologically first approach for SWS, OWL-S defines an upper ontology for semantically annotating Web services. This has been developed in the years 2003 - 2005, driven by a mostly US-based consortium under the DAML programme (see `www.daml.org`).

The OWL-S model defines three elements for describing Web services as shown in Figure 5 (taken from [46]). Every description element is defined on the basis of an domain ontology, and the current standard ontology language OWL is used as the specification language (see above):

1. the *Service Profile* holds information for Web service advertisement, containing the name of the service, its provider, a natural language description, and a formal functional description defined in terms of the in- and outputs, preconditions and effects (short: IOPE)
2. the *Service Model* describes how the Web service works whereby the service is conceived as a process. The description model defines three types of processes (atomic, simple, and composite processes), whereof each construct is described by IOPE along with basic control- and dataflow constructs.
3. the *Service Grounding* gives details of how to access the service, which is realized as a mapping from the abstract descriptions to WSDL.

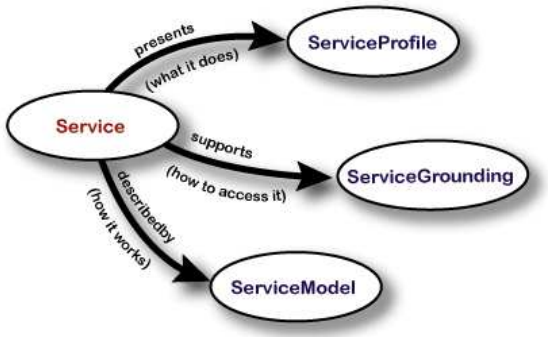

Fig. 5. Overview OWL-S

The intended usage of OWL-S description is as follows. The service profile relate to the information stored in UDDI repositories. While the natural language

descriptions are for human consumption, the formal functional description is used for automated Web service discovery by semantic matchmaking (see Section 4 below). The service model formally describes the external visible behavior of a Web service, i.e. how to invoke and consume the service and what happens when it is executed. This is used to determine whether the communication between a client and the Web service as well as with other aggregated Web services can be carried out successfully. Finally, the service grounding maps the abstract, semantic descriptions to conventional Web service technologies in order to conduct the actual message exchange for execution. Although being criticized especially on the inadequacy of the process description language [40], OWL-S has served as the basis for various SWS research and development activities.

WSMO [41]. The Web Service Modeling Ontology WSMO is developed by a European initiative since 2004 (see `www.wsmo.org`). It takes a broader approach than OWL-S, aiming a comprehensive framework for semantically enabled SOA technologies [12]. For this, it defines four top-level notions: *ontologies* that define formalized domain knowledge, *goals* that describe objectives that clients want to achieve by using Web services, semantic description of *Web services*, and *mediators* for resolving potentially occurring heterogeneities (see Figure 6).

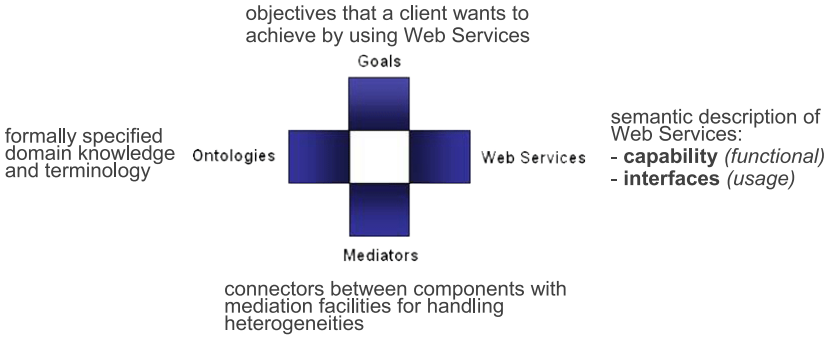

Fig. 6. WSMO Top Level Notions

In contrast to the other frameworks, WSMO does not only cover the semantic annotation of Web services but propagates a goal-based approach for Semantic Web services along with mediation as an integral part. The idea is that a client formulates requests in terms of a goal, which formally describes the objective to be achieved while abstracting from technical details; the system then automatically detects and executes the suitable Web services in order to solve the goal [64]. The notion of goals provides an explicit element for the client side of SOA applications which allows to lift the Web service usage by clients to the level of problems that can be solved. In addition, integrated mediators allow to handle and resolve potentially occurring heterogeneities that can be expected in open

and decentralized environments like the Web and may hamper the successful interaction between clients and Web services [17].

The WSMO framework defines description models for all four elements along with an own specification language. Analogous to Figure 3 above, Web services in WSMO are described by non-functional properties, a capability that specifies the provided functionality in terms of preconditions, assumptions, postconditions, and effects, a choreography interface that describes how a client can invoke and consume a Web service, and an orchestration interface that describes how the Web service interacts with other Web services to achieve its functionality. WSMO provides an own specification language called WSML [22], which is a conceptual language for the WSMO elements along with five variants of logical languages that corresponds to the ontology languages developed for the Semantic Web (*cf.* Figure 4). Several tools are provided for WSMO, including a suite of reasoners for the different variants and an API for the programmatic management of WSMO elements and definitions. Moreover, there are implementations of execution environments for Semantic Web services, namely WSMX as the WSMO reference implementation (see `www.wsmx.org`), and IRS that provides a broker for Semantic Web services [14].

SWSF [7]. The Semantic Web Services Framework (SWSF) has been developed by a joint working group of industrial and academic researchers. Essentially, it provides an extension of OWL-S that aims at replacing the initial, insufficient specification model and language for the *Service Model* with an appropriate formal process language. The major contribution of SWSF is a rich behavioral process model based on the Process Specification Language (PSL) [31]. SWSF provides two axiomizations: (1) FLOWS is based on first-order logic with extensions form situation calculus to model changes of the world; (2) SWSLRules is a logic programming language that serves as both a specification and implementation language and provides support for tasks like discovery, contacting, and policy specification for Semantic Web services.

WSDL-S [1]. The WSDL-S approach has been defined in a joint effort of IBM and the University of Georgia. Instead of defining a comprehensive framework for semantically describing Web services, WSDL-S defines extensions to WSDL in order to semantically annotate the XML data types as well as the messages and operations in a WSDL description. For this, a WSDL document is augmented with additional tags that refer to an external domain ontology. While not fixing the ontology language, WSDL-S proposes three types of annotations:

1. WSDL types (i.e. XML data elements) are referenced to concepts in the domain ontology
2. WSDL operations can be described by preconditions and effects that refer to respective axioms
3. a categorization of Web services can be defined on the basis of the ontology taxonomy.

SAWSDL [25]. While the previously presented approaches have been published as W3C member submissions, *Semantic Annotations for WSDL and XML Schema* (short: SAWSDL) is the only official W3C technology recommendation for Semantic Web services existing at this point in time. It essentially follows the idea of WSDL-S, i.e. the annotation of WSDL documents with additional tags that reference to a domain ontology. SAWSDL consists of two parts as illustrated in Figure 7 (taken from [39]): the mappings of XML schema defines to ontology concepts in order to define the correspondence of SOAP message contents to ontology data, and the semantic annotation of WSDL operations. For the latter, SAWSDL limits the annotation by referring to ontology concepts but does not support the definition of preconditions and effects, which limits the annotations to merely consists of keywords associated with a domain ontology.

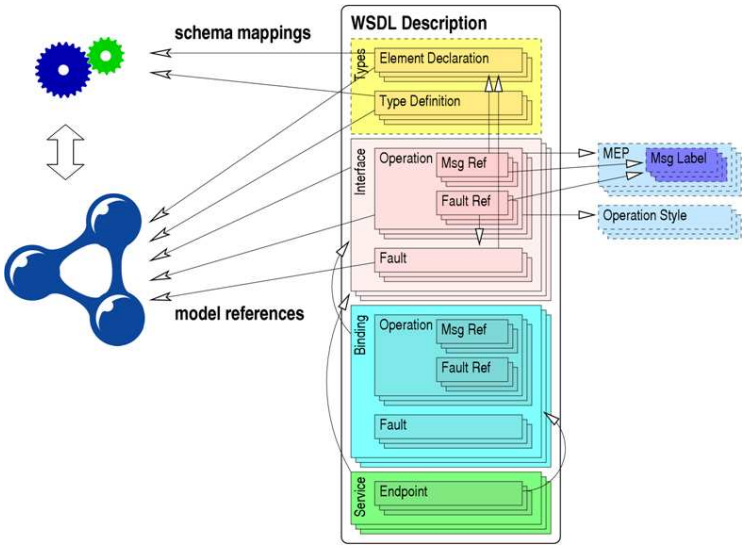

Fig. 7. SAWSDL Overview

A comparison of the frameworks reveals the following commonalities and differences. OWL-S as the first approach defines a description model for Web services that covers all aspects of the SWS approach as described above, i.e. non-functional aspects, a formal functional, and descriptions of Web service interfaces for consumption and aggregation. This uses OWL as the specification language, thus is compliant with the W3C standards for the Semantic Web. SWSF extends and refines this approach with a richer process description model and more expressive specification languages. WSMO is a more exhaustive framework that propagates a goal-driven approach along with integrated mediation facilities. This goes beyond the idea of merely annotating Web services, aiming at an all-embracing semantic SOA technology. WSMO defines an own specifi-

cation language that covers all ontology languages that are considered for the Semantic Web, and provides reasoners for this along with a set of development tools as well as reference implementations. WSDL-S only partially realizes the SWS approach, therewith can be considered as a light-weight framework. However, it follows the W3C tradition of extending existing standards, and it has served as the conceptual basis for SAWSDL as the only approach for the semantic annotation of Web services that is recommended by a standardization body as of today. Recent works take over this approach, e.g. [47].

4 Semantic Techniques for Automating SOA

After explaining the motivation and prominent approaches for Semantic Web services, we now turn towards the techniques for automated support of Web service detection and execution. The following first identifies the central techniques, and then presents the state-of-the-art in research and development.

As outlined above, the ultimate aim of the SWS approach is to automate the complete Web service usage process with inference-based techniques that expose a sophisticated processing quality. Figure 8 illustrates the workflow of SWS environments for this, which has been defined in early works on SWS system architectures [60] and is currently being specified in detail by a OASIS standardization working group [55]. The input is a concrete client request that shall be solved by detecting and executing the suitable Web services; the gray boxes denote the necessary techniques for this. The first processing step is *discovery*, which is concerned with the detection of suitable candidates out of the available Web services. This commonly is realized by semantic matchmaking of formal functional descriptions. Then, the usability of the discovered candidates is in-

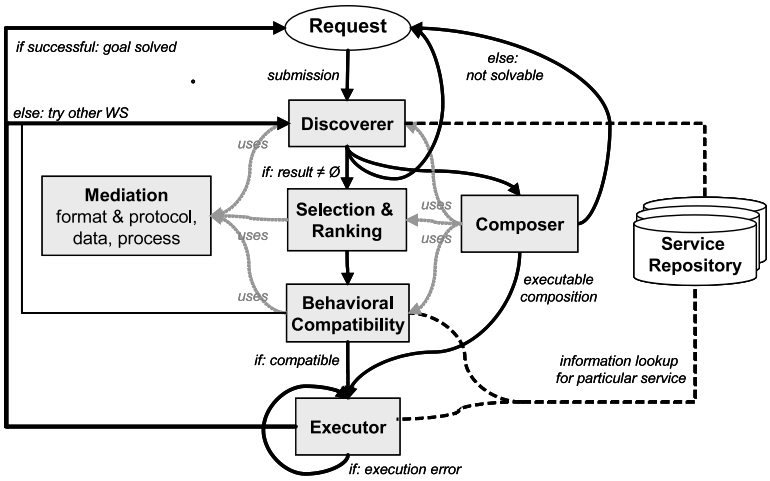

Fig. 8. SWS Techniques for Automated Web Service Usage

spected in more detail. The *selection and ranking* component either selects one of the candidates or determines a priority list for the further processing with respect to quality-of-service criteria as well as other non-functional aspects, and the *behavioral compatibility* component checks whether the communication between the requester and the Web services can be carried out successfully. If this is given, then the *executor* automatically invokes the Web service in order to solve the client request. If a single suitable Web service does not exist, then the *composer* is invoked which tries to construct a combination of several Web services for solving the request; this utilizes the previously mentioned components. In addition, *mediation* facilities can be employed in order to handle potentially occurring mismatches that hamper the successful interaction between the requester and the provider. In the following, we explain each of the techniques in more detail and depict the latest research works on this.

4.1 Discovery

Web service discovery is concerned with the detection of the suitable Web services for a given request out of the available ones. This is a central operation in SOA systems for which significant quality increase can be achieved by SWS techniques: on the basis of more precise Web service descriptions, discovery techniques can be developed that expose a higher precision and recall than the syntactic keyword-based search supported by UDDI.

A wealth of work exists on semantically enabled Web service discovery. Most approaches address this by semantic matchmaking of formally described requested and provided functionalities, i.e. OWL-S service profiles or WSMO capabilities as explained above. This is commonly referred to as functional discovery, which allows to determine whether a Web service can solve the given request with respect to the preconditions and effects a successful execution. Prominent works for this are [59,42,36]. In addition to this, techniques have been developed for handling cases where a match is not given but can be established by relaxing requirements in the request (e.g. [19]), and approaches that integrate other techniques for discovery (e.g. [37,54]).

In principle, semantically enabled Web service discovery techniques can achieve a very high retrieval accuracy. One the basis of sufficiently rich functional descriptions with expedient formal semantics and a exhaustive domain ontology, one can specify semantic matchmaking techniques that allow to very precisely determine whether a Web service can be used for the given client request or not. This occurs to be desirable in comprehensive SWS environments as outlined above; however, such techniques require a considerable effort in the creation and validation of the necessary formal descriptions. Thus, also more light-weight discovery techniques are developed which can merely achieve a lower but whose employment requires less effort on the necessary formal specifications.

With respect to this, we can distinguish six categories of semantic discovery techniques. The following explains them in an ascending order with respect to the achievable retrieval accuracy, and Table 1 illustrates which of the discovery techniques are supported by the SWS frameworks presented in Section 3.

Table 1. Support for Automated Web Service Discovery

	OWL-S	**WSMO**	**WSDL-S**	**SAWSDL**
goal-based		x		
precond. / effect heavy	x	x		
precond. / effect light	x	x	x	
input / output	x	x	x	x
categorization	x	x	x	x
keyword-based	x	x	x	x

1. **keyword-based:** the simplest techniques perform discovery on the basis of keywords, e.g. the flight booking Web service from United Airlines is annotated with the keyword "flight, booking, UA". Usually, the keywords are based on a domain ontology - e.g. by the referencing mechanism defined in WSDL-S and SAWSDL. This is relatively easy to realize, but only a very low retrieval accuracy can be achieved.
2. **categorization:** this refers to techniques that perform discovery on the basis of a categorization. Mostly, the Web services are annotated with concepts of a domain ontology, and the taxonomic structure of the ontology serves as the categorization scheme – e.g. all the Web services annotated with the concepts `car`, `train`, and `plane` are organized in the category `vehicle` as the common super-concept in the ontology. Although the actual retrieval accuracy is similar to the keyword-based techniques, the categorization allows to browse and pre-filter potential candidates for a more detailed inspection. A SWS system that realizes this approach is presented in [67].
3. **matchmaking input / output:** this refers to techniques that consider the compatibility of the in- and outputs. In principle, a Web service is considered to be usable if the requester can provide all required inputs, and if the outputs of the Web service satisfy those expected by the requester. This allows to achieve a significantly higher retrieval accuracy than the previous techniques, and is most commonly applied in existing SWS environments.
4. **matchmaking precondition / effect:** the next group of discovery techniques does not only consider the in- and outputs but also further conditions that are defined in terms of preconditions and effects. We here need to further distinguish **light-weight** techniques wherein the pre- and post-execution constraints are considered as isolated logical formulae, and **heavy-weight** techniques wherein the functional descriptions are considered as a coherent formal specification. Naturally, the latter can achieve a higher retrieval accuracy because the relationship between preconditions and effects is considered as well. An exhaustive discussion on this is provided in [36].
5. **goal-based:** the last group of automated Web service discovery techniques follows the goal-based approach promoted by the WSMO framework (see above). Therein, client requests are associated to generic goal descriptions. Apart from a high retrieval accuracy, this allows develop efficient and scalable discovery techniques by separating design- and runtime operations [63].

4.2 Selection and Ranking

This encompasses SWS techniques for determining the usability of Web services with respect to non-functional aspects. This includes quality-of-service information, and in particular data security and usage rights. The benefit of semantic techniques for this is that – on the basis of respective ontologies – a more precise and serviceable processing of quality requirements and usage policies can be achieved than with conventional techniques.

While Web service discovery as discussed above is concerned with *what* a Web service does, the techniques that we consider here are concerned with quality and usage conditions. The former aspect relates to the operational reliability of a Web service as a software artefact (i.e. regarding the availability, robustness, and execution performance) as well as the quality of the provided business service. These parameters are usually described in terms of respective time and quality measurements. The latter aspect is concerned with access rights and data security, which becomes in particular relevant when Web services are applied in IT systems for intra- and inter-organizational communication.

The approach for handling both quality-of-service as well as usage rights of Web services by semantic techniques is to reason upon *policies* that are defined on the basis of respective domain ontologies [57]. For example, if the access to an electronic journal might only be granted for members of a specific department of a university, this can be checked by the user profile of a requester; moreover, the specification of such conditions in an ontology makes the usage rights more transparent for the involved parties. Upon this, techniques are developed for automatically selecting a Web services which conforms with the relevant policies, or to determine a priority list of the usable candidates with respect to the client requirements (e.g. [69,70]).

4.3 Behavioral Compatibility

The third group of SWS techniques is concerned with determining whether the communication between the requester and the provider can be conducted successfully. This is necessary in order to ensure that the actual consumption of the Web service can be carried out successfully.

This problem does not occur within conventional Web service technologies, because the client needs to explicitly trigger every outgoing SOAP message. However, in order to do this the developer of a client application must manually implement the correct communication behavior before the Web service can be used. In the context of Semantic Web services, the aim is to automatically execute the suitable Web services after they have been detected. For this, the communication behavior expected by the client and the one supported by the Web service must be compatible.

This can be checked automatically on the basis of the formally described interfaces - i.e. the OWL-S service model or the choreography and orchestration interfaces in WSMO (see above). Although this problem has only received little attention in the research community so far, existing approaches apply *conformance*

testing techniques from the field of formal process management(e.g. [45,68]). In a nutshell, the behavioral compatibility is considered to be given if (1) the in- and outgoing messages of the requester and the provider are compatible, and (2) there exists at least one possible sequence of message exchange that can be carried out between the involved parties.

4.4 Composition

The aim of Web service composition is to automatically combine several Web services in order to obtain a more complex functionality. The surplus value of Web service composition techniques is that new functionalities can be created that are not provided by the actually existing Web services, which is hardly achievable without any automation support.

The overall task for Web service composition is as follows: given a client request that can not be solved by a single Web services, an executable combination of several Web services which is suitable to solve the client request. A lot of research works address this challenge, applying different techniques for the composition problem. In general, we can distinguish composition techniques on two levels. The first one is considers the functionalities of the Web services for determining a suitable execution order, and hence is referred to as *functional composition*. The respective techniques work on the formal functional descriptions – i.e. OWL-S service profiles or WSMO capabilities – and mostly apply AI planning techniques for the composition task (e.g. [50,72,35]). The second level is concerned with the communication behavior in a composition of Web services. The aim is to ensure that the interaction between the client and the composed Web services can be conducted successfully – i.e. the problem of behavioral compatibility as discussed above within a composition of Web services. This is commonly referred to as *behavioral composition*, and most approaches apply formal workflow or process management techniques for this (e.g. [8,2]).

To leverage automated Web service composition within SWS environments, both types of composition techniques need to be integrated in order to attain executable compositions of Web services for solving a given client request. For this, [66] presents an approach wherein at first functional composition is applied to create a skeleton of a composition that is suitable for solving the client request, and in a second step its executability is verified by behavioral composition techniques. Besides, recent approaches consider Web service discovery and composition as interleaved operations: composition is only needed if a directly usable Web service can not be discovered, and discovery techniques are used to find the candidates during the composition procedure (e.g. [10]).

4.5 Mediation

In the context of Semantic Web services, mediation refers to the handling and resolving potentially occurring heterogeneities which may hamper the interoperability between a requester and a provider. This becomes in particular important within open and distributed environments like the Web where requesters

and providers can be expected to use different data representation formats, incompatible terminologies, or expose business processes that are not compatible a priori. The main merit of SWS technologies is that such heterogeneities can be handled on the semantic level, i.e. by domain independent mediation techniques that allow to properly resolve and handle the mismatches [27].

WSMO is the only SWS framework that encompasses mediation as an integral part. It defines specific mediators for handling different types of heterogeneities, provides respective mediation techniques, and defines an integrated architecture for the specification and usage of mediators within SWS environments [62]. The most relevant mediation techniques are (1) the *data level mediation* which is concerned with mismatch handling on terminologies, domain knowledge, and representation formats [52], and (2) the *process level mediation* which is concerned with handling incompatible communication behaviors and business processes of requesters and providers [16]. The other SWS frameworks presented in Section 3 do not consider mediation; in fact, they are merely concerned with the semantic description of Web services while remaining orthogonal to all other aspects that occur to be relevant for the employment of semantic technology in SOA systems. However, existing techniques for heterogeneity handling can be employed, e.g. ontology-based data integration techniques that have been developed for the Semantic Web [56].

4.6 Automated Execution

The final aspect of SWS technology is the automated execution of Web services. Once the suitable Web services for solving a given request have been detected and all other relevant aspects have been checked, they should be executed automatically in order to minimize the need for human intervention.

For this, the semantic descriptions of the Web services need to be mapped to technologies that allow to carry out the actual information interchange. Commonly, this is achieved by mapping the semantic annotations to a WSDL description, which then allows to invoke and consume the Web services via SOAP as explained in Section 2. This usually also includes an explicit mapping between the XML data types used within SOAP messages and domain ontologies used for the semantic descriptions in order to facilitate the processing of the interchanged data on the semantic level.

This this supported by all SWS frameworks presented above. OWL-S and SWSF define the mappings in the service grounding element, which specifies the mapping of the domain ontology to an XML Schema definition and maps the service model definitions to WSDL operations. This is processed by the OWL-S Virtual Machine for automated execution [58]. The same approach is realized in WSMO: the mappings from the ontology definitions to XML as well as the mapping to WSDL operations is defined within the WSMO choreography interface description, and the WSMX execution component invokes the Web services via WSDL [38]. Within WSDL-S and SAWSDL, the mappings are defined explicitly by the references to a domain ontology within additional tags in the WSDL document, which can be processed by respective execution environments.

Summarizing, we have shown that there is a wealth of work on SWS techniques for the automated detection, usability analysis, composition, and execution of Web services. The individual solution vary in the achievable quality and the necessary efforts for their employment; the appropriate ones can be chosen for a specific application scenario. Moreover, there are open-source development and execution environments for Semantic Web services, e.g. the OWL-S IDE [61] and the WSMX system as the reference implementation of the WSMO framework [32]. However, most of the currently existing SWS technologies have been developed in the course of academic research, and their employment in real-world applications requires additional software development.

5 Conclusions and Outlook

In this article, we have provided an overview of the emerging concept of Semantic Web services (SWS) and the state-of-the-art in respective technology developments. The following summarizes the article, and discusses the potential as well as the challenges for the future developments of semantic SOA technologies.

5.1 Summary

The idea of Service-Oriented Architectures (SOA) is to employ Web services as the basic building blocks of future IT systems. For this, the initial Web service technology provides a standardized description language for the technical accessibility and the interfaces of Web services (WSDL), a communication protocol for the consumption of Web services by exchanging messages over the Web (SOAP), and a registry technology that allows to publish and search Web services (UDDI).

Although this allows to technical use Web services, the detection and usability analysis of suitable Web services for a specific client application is limited to manual inspection. To overcome this, the SWS approach develops techniques for the automated discovery, usability analysis, composition, mediation, and execution of Web services. These techniques work on rich formal descriptions of Web services that are defined on the basis of domain ontologies, i.e. formal knowledge models which ar propagated as the base technology for the Semantic Web.

We have explained the most prominent frameworks for Semantic Web services that have been submitted to, respectively published as recommendations by the W3C. The chronologically first approach is OWL-S, which semantically describes Web services by a service profile (the "who" and "what"), a service model (the "how"), and a grounding to WSDL for the execution. This has later been extended by the SWSF initiative wherein a more sophisticated formal process language for describing Web services has been developed. The second important framework is the Web Service Modeling Ontology WSMO, which defines a comprehensive framework for semantically enabled SOA technology. Going beyond the semantic annotation of Web services, WSMO propagates a goal-driven approach for Semantic Web services wherein clients request and consume Web

services on the basis of goals that abstract from technical details, and it considers mediation facilitates for the handling and resolving potentially occurring mismatches as an integral part. The third approach is the WSDL-S model which – in contrast to the other frameworks – defines the semantic annotation of Web services by extending WSDL descriptions with references to a domain ontology. A light-weight version of this approach is SAWSDL which supports the annotation of XML Schema and WSDL descriptions with ontology concepts. Although the obtainable support for the automated detection and usability analysis is fairly limited, SAWSDL is the only W3C technology recommendation for the Semantic Web services that exists at this point in time.

We then have explained the central SWS techniques for automating the Web service usage process by clients. The usually first processing step is *discovery*, i.e. the detection of the suitable Web services for a given client requests. This is commonly performed by matchmaking of the requested and the provided functionalities, and we have outlined several techniques for this. Next, the usability of the discovered candidates is inspected with respect to non-functional aspects such as quality-of-service criteria, data security, and usage rights, and finally the behavioral compatibility is tested in order to ensure the successful interaction between the client and the Web service. Techniques for automated composition allow to combine several Web services into more complex functionalities, and mediation techniques can be employed as auxiliary facilities to handle possibly occurring mismatches that may hamper the successful interaction. When the Web services for solving a client request have been detected, they are executed automatically by lowering the semantic descriptions to WSDL and XML.

5.2 Future Challenges

So far, we have explained the motivation and state-of-the-art in SWS technologies. We also have shown that significant improvements for both the quality of the usability analysis and the degree of automation can be achieved. However, the existing SWS technologies are mostly academic developments. With respect to this, the following discusses challenges for future developments in order to make SWS techniques employable in real-world SOA applications.

The pre-requisite for SWS techniques is the existence of appropriate semantic descriptions for the available Web services and all other related resources. Most of the existing SWS techniques focus on new functionalities and the achievable benefits under the assumption that the necessary resource descriptions are given. However, this occurs to not be the case, in particular when SWS technology shall be applied within existing systems. Thus, techniques for the semantic annotation of legacy systems occurs to be essential in order to assure the applicability of SWS technologies in real-world settings. This challenge as only received very little attention in the research community so far. It appears to be possible to adopt techniques for the ontology-based annotation of natural language texts for this; however, in general this can only be supported in a semi-automated manner due to the gap between syntactic and adequate semantic descriptions, and also the annotation of Web services is expectably much more complex.

Another concern related to the general applicability of SWS techniques is the extent to which they shall be employed such that a substantial benefit can be achieved while the effort and costs remain moderate. The initial Web service technology occurs to be not sufficient because it limits the Web service usage to manual inspection. The WSDL-S and SAWSDL approach occurs to be only a little bit better: the semantic annotation by additional tags in WSDL documents is relatively easy to realize, but on the other hand the obtainable benefits are only marginal. The OWL-S approach requires exhaustive descriptions of Web services on which significant quality improvements can be achieved; however, the employment an existing system occurs to be expensive. The WSMO approach allows to achieve the highest benefits because of the goal-based approach as well as the integrated mediation facilities, but its employment requires a comprehensive redesign of a SOA system. With respect to this, the aim for future research should be to identify the degree of employment for which the cost-benefit relation is optimal and then initiate respective technology standardizations.

A further aspect for the prosperous application of SWS technology is the provision of adequate tooling support. Although a remarkable number of graphical editors, APIs, and execution environments already exists, this occurs to still be not sufficient in order to properly support users in real-world SOA applications. In particular, expedient graphical user interfaces for managing Web services and their semantic descriptions as well as sophisticated validation services for the formal specifications occur to be desirable in order to better support end-users and system administrators. However, this can be considered as supplementary development efforts once the underlying technology exists.

To conclude, semantic techniques for the automated detection and usage of Web services as explained in this article occur to be capable and eligible to effectively support the idea of Service-Oriented Architectures. In fact, some "intelligence" occurs to be necessary in order to prosperously realize the SOA vision, and the employment of semantic technologies occurs to be a suitable promising approach for this. However, in order to leverage a successful deployment of such techniques within future SOA technology, it occurs to be evident to properly address the mentioned challenges.

Acknowledgements. The presented work has been supported by the European Commission under the projects SUPER (FP6 - 026850) and by the Austrian BMVIT/FFG under the FIT-IT project myOntology (Grant no. 812515/9284).

References

1. R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma. Web Service Semantics - WSDL-S. W3C Member Submission 7 November 2005, 2005. online: `http://www.w3.org/Submission/WSDL-S/`.
2. P. Albert, L. Henocque, and M. Kleiner. Configuration-Based Workflow Composition. In *Proc of 3rd International Conference on Web Services (ICWS-05), Orlando, Florida*, 2005.
3. V. Alexiev, M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen. *Information Integration with Ontologies.* Wiley, West Sussex, UK, 2005.

4. P. Allen. *Service Orientation: Winning Strategies and Best Practices*. Cambridge University Press, 2006.
5. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg, 2004.
6. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services version 1.1. Specification, IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, May 2003. online: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
7. S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, Martin. D., McIlraith. S., D. McGuinness, J. Su, and S. Tabet. Semantic Web Services Framework (SWSF). W3C Member Submission 9 September 2005, 2005. online: `http://www.w3.org/Submission/SWSF/`.
8. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic Composition of e-Services that Export their Behavior. In *Proc. of First Int. Conference on Service Oriented Computing (ICSOC)*, 2003.
9. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. A new form of Web content that is meaningful to computers will uleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, May 2001.
10. P. Bertoli, J. Hoffmann, F. Lecue, and M. Pistore. Integrating Discovery and Automated Composition: from Semantic Requirements to Executable Code. In *Proc. of the IEEE 2007 International Conference on Web Services (ICWS'07), Salt Lake City, USA*, 2007.
11. D. Booth and C. K. Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. Recommendation 26 June 2007, W3C, 2007. online: `www.w3.org/TR/wsdl20-primer`.
12. M. Brodie, C. Bussler, J. de Brujin, T. Fahringer, D. Fensel, M. Hepp, H. Lausen, D. Roman, T. Strang, H. Werthner, and M. Zaremba. Semantically Enabled ServiceOriented Architectures: A Manifesto and a Paradigm Shift in Computer Science. Technical Report TR-2005-12-26, DERI, 2005.
13. C. Bussler. *B2B Integration: Concepts and Architecture*. Springer, Berlin, Heidelberg, 2003.
14. L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, B. Norton, V. Tanasescu, and C. Pedrinaci. IRS-III – A Broker for Semantic Web Services based Applications. In *Proc. of the 5th International Semantic Web Conference (ISWC 2006), Athens(GA), USA*, 2006.
15. J. Cardoso and A. Sheth. *Semantic Web Services, Processes and Applications*. Semantic Web and Beyond. Springer, 2006.
16. 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.
17. E. Cimpian, A. Mocan, and M. Stollberg. Mediation Enabled SemanticWeb Services Usage. In *Proc. of the 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China*, 2006.
18. L. Clement, A. Hately, C. von Riegen, and T. (eds) Rogers. UDDI Version 3.0.2. UDDI Spec Technical Committee Draft, OASIS, 2004. online: `www.uddi.org/pubs/uddi_v3.htm`.

19. S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. *Electronic Commerce Research and Applications*, 4:345–361, 2005.
20. J. Davis, R. Studer, and P. Warren. *Semantic Web Technology. Trends and Research in Ontology-based System.* Wiley & Sons, 2006.
21. J. de Bruijn. Logics for the Semantic Web. In J. Cardoses, editor, *Semantic Web: Theory, Tools and Applications.* Idea Publishing Group, 2006.
22. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Deliverable D16.1 final draft 05 Oct 2005, WSML Working Group, 2005. Available at: `http://www.wsmo.org/TR/d16/d16.1/v0.2/`.
23. P. de Leenheer and T. Mens. Ontology Evolution: State of the Art and Future Directions. In De Leenheer P. de Moor A. Hepp, M. and Y. Sure, editors, *Ontology Management.* Springer, 2006.
24. T. Erl. *Service-Oriented Architecture (SOA). Concepts, Technology, and Design.* Prentice Hall PTR, 2005.
25. J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema. W3C Recommendation 28 August 2007, 2007. online: `http://www.w3.org/TR/sawsdl/`.
26. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and E-Commerce.* Springer, Berlin, Heidelberg, 2 edition, 2003.
27. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
28. D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domigue. *Enabling Semantic Web Services. The Web Service Modeling Ontology.* Springer, Berlin, Heidelberg, 2006.
29. A. Goméz-Peréz, 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.
30. Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5:199–220, 1993.
31. M. Gruninger and C. Menzel. The Process Specification Language (PSL) Theory and Applications. *AI Magazine*, 24(3):63–74, 2003.
32. 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), Orlando, Florida*, 2005.
33. A. Harth and S. Decker. Optimized Index Structures for Querying RDF from the Web. In *Proc. of 3rd Latin American Web Congress, Buenos Aires, Argentina, Oct. 31 - Nov*, 2005.
34. M. Hepp, P. de Leenheer, A. de Moor, and Y. Sure. *Ontology Management. Semantic Web, Semantic Web Services, and Business Applications.* Semantic Web and Beyond. Springer, 2007.
35. J. Hoffmann, P. Bertoli, and M. Pistore. Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty. In *Proc. of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07), Vancouver, Canada*, 2007.
36. U. Keller, R. Lara, H. Lausen, and D. Fensel. Semantic Web Service Discovery in the WSMO Framework. In J. Cardoses, editor, *Semantic Web: Theory, Tools and Applications.* Idea Publishing Group, 2006.

37. M. Klusch, B. Fries, and K. Sycara. Automated Semantic Web Service Discovery with OWLS-MX. In *Proc. of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8 - 12*, 2006.
38. J. Kopecký, D. Roman, M. Moran, and D. Fensel. Semantic Web Services Grounding. In *Proc. of the International Conference on Internet and Web Applications and Services (ICIW'06), Guadeloupe, French Caribbean*, 2006.
39. Jacek Kopecky. Semantic Annotations for WSDL and XML Schema. Talk at W3C track in the WWW 2007 Conference, Banff, Canada, 2007.
40. L. Lara, D. Roman, A. Polleres, and D. Fensel. A Conceptual Comparison of WSMO and OWL-S. In *Proc. of the European Conference on Web Services (ECOWS 2004), Erfurt, Germany*, 2004.
41. H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). W3C Member Submission 3 June 2005, 2005. online: `http://www.w3.org/Submission/WSMO/`.
42. L. Li and I. Horrocks. A Software Fframework for Matchmaking based on Semantic Web Technology. In *Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary*, 2003.
43. F. Manola and E. Miller. SPARQL Query Language for RDF. W3C Candidate Recommendation 14 June 2007, 2007. online: `www.w3.org/RDF/`.
44. E. A. Marks and M. Bell. *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology.* Wiley, 2006.
45. A. Martens. On Compatibility of Web Services. *Petri Net Newletter*, 65:12–20, 2003.
46. D. Martin. OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004, 2004. online: `http://www.w3.org/Submission/OWL-S/`.
47. D. Martin, M. Paolucci, and M. Wagner. Towards Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective. In *Proc. of the ESWC 2007 workshop OWL-S: Experiences and Directions, Innsbruck, Austria*, 2007.
48. D. McGuinness and F. van Harmelen. OWL Web Ontology Language - Overview. W3C Recommendation 10 February 2004, 2004. online: `http://www.w3.org/TR/owl-features/`.
49. 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.
50. S. McIlraith and T. C. Son. Adapting Golog for Composition of Semantic Web Services. In *roc. of the 8th International Conference on Knowledge Representation and Reasoning (KR '02), Toulouse, France*, 2002.
51. N. Mitra and Y. Lafon. SOAP Version 1.2 Part 0: Primer (Second Edition). Recommendation 27 April 2007, W3C, 2007. online: `www.w3.org/TR/soap12-part0/`.
52. A. Mocan and E. Cimpian. An Ontology-based Data Mediation Framework for Semantic Environments. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(2):66 – 95, April - June 2007.
53. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In *Proc. of the 21st Conference on Automated Deduction (CADE-21), Bremen, Germany, July 17-20*, 2007.
54. T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. A System for Principled Matchmaking in an Electronic Marketplace. In *Proc. of the 12th International Conference on the World Wide Web (WWW'03), Budapest, Hungary*, 2003.
55. B. Norton and A. Mocan. Reference Model for Semantic Service Oriented Architecture. Working Draft, 21 March 2007, OASIS, 2007.

56. N. Noy. Semantic Integration: a Survey of Ontology-based Approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004.
57. D. Olmedilla, R. Lara, A. Polleres, and H. Lausen. Trust Negotiation for Semantic Web Services. In *Proc. of the 1st International Workshop on Semantic Web Services and Web Process Composition at the ICWS 2004, SanDiego, California (USA)*, 2004.
58. M. Paolucci, A. Ankolekar, N. Srinivasan, and K. Sycara. The DAML-S Virtual Machine. In *Proc. of the 2nd International Semantic Web Conference (ISWC),Sandial Island, Florida*, 2003.
59. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. of the 1st International Semantic Web Conference, Sardinia, Italy*, 2002.
60. C. Preist. A Conceptual Architecture for Semantic Web Services. In *Proc. of the 2nd International Semantic Web Conference (ISWC 2004)*, 2004.
61. N. Srinivasan, M. Paolucci, and K. Sycara. CODE: A Development Environment for OWL-S Web services. In *Demonstration at 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan.*, 2004.
62. M. Stollberg, E. Cimpian, A. Mocan, and D. Fensel. A Semantic Web Mediation Architecture. In *Proceedings of the 1st Canadian Semantic Web Working Symposium (CSWWS 2006), Quebec, Canada*, 2006.
63. M. Stollberg, M. Hepp, and J. Hoffmann. A Caching Mechanism for Semantic Web Service Discovery. In *Proc. of the 6th International Semantic Web Conference (ISWC 2007), Busan, South Korea*, 2007.
64. M. Stollberg and B. Norton. A Refined Goal Model for Semantic Web Services. In *Proc. of the 2nd International Conference on Internet and Web Applications and Services (ICIW 2007), Mauritius*, 2007.
65. R. Studer, S. Grimm, and A. Abecker. *Semantic Web Services. Concepts, Technologies, and Applications*. Springer, 2007.
66. P. Traverso and M. Pistore. Automatic Composition of Semantic Web Services into Executable Processes. In *Proc. of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan*, 2004.
67. K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Journal of Information Technology and Management*, 6(1):17–39, 2005. Special Issue on Universal Global Integration.
68. T. Vitvar, M. Zaremba, and M. Moran. Dynamic Service Discovery through Meta-Interactions with Service Providers. In *Proc. of the 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria*, 2007.
69. L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-Based Service Selection and Ranking with Trust and Reputation Management. In *Proc. of the OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Cyprus*, 2005.
70. X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-aware Selection Model for Semantic Web Services. In *Proc. of the 4th International Conference on Service Oriented Computing (ICSOC), December, 2006, Chicago, USA*, 2006.
71. S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, 2005.
72. D. Wu, B. Parsia, Sirin E., J. Hendler, and D. Nau. Automating DAML-S Web Services Composition Using SHOP2. In *Proceedings of 2nd International Semantic Web Conference (ISWC 2003), Sanibel Island, Florida*, 2003.