

# Web service description, advertising and discovery: WSDL and beyond

Wilfried Lemahieu

***Abstract.** This paper discusses the predicaments surrounding web service description, advertising and discovery. It provides an overview of the various technologies involved and advocates how current state-of-the-art standards such as UDDI and WSDL can be enhanced by means of concepts from the semantic web vision such as RDF, RDF Schema and DAML+OIL. Moreover, it is shown how semantic metadata specifications can be considered as (a) layer(s) above syntactic metadata modeled by means of XML schemas. Finally, suggestions are made for further research towards the enhancement of current web ontology efforts, seen in the light of web service description.*

## 1. Introduction

### 1.1. Towards structured web documents

A key factor in the World Wide web's initial success was undoubtedly its *simplicity*. It allowed in a very natural way to interconnect chunks of information that were dispersed around the world. The latter was accomplished by means of the *hypertext* paradigm: the information was represented in (static) HTML pages and interrelated by means of embedded links to other pages. The HTML specification was primarily targeted at a *human* audience: its main purpose was specifying a document's *page layout*. Obviously, the way in which a document is visualized is an utterly important factor in a human reader's ability to *comprehend* the information it contains.

The concept of having a world-wide network of information at the fingertips from a single client application, the web browser, was so appealing, that the demand arose to disclose other types of information from this same "universal client" as well. However, whereas the web's initial objective entailed the (stateless) sharing of data between *human beings* by means of

publishing and consulting *static*, interlinked information, ambitions were adjusted to reach far further than that. The web browser was now considered as the client interface by which end users could interact with *dynamic*, session-based applications that were able to perform complex transactions on data stored in heterogeneous resources. Moreover, a “user” was no longer necessarily human: web data was to become *machine-processable*, such that applications could interact with one another in an automated way and “understand” the data being exchanged, without human intervention.

The property of the original web language HTML being confined to a fixed tag set, which in addition focused layout rather than structure, was nearly prohibitive to automated document processing other than presenting it on screen. All this changed with the introduction of XML, which allowed for a document’s tag set to be freely defined, with the tags comporting a *structural* connotation, rather than a layout one. In this way, markup dealt with what an element *meant* instead of how it was to be visualized. At last, computers were able to better “understand” the information contained in web documents, provided they knew how to handle the corresponding tags. Moreover, the use of DTD’s and, even better, XML schema’s, allowed to *prescribe* the structure of a certain type of documents, such that all instances of such type could be checked for their validity and correspondence to the schema. As a consequence, computers were able to know in advance what kinds of documents could be expected, hence they were prepared to deal with them, provided they had access to the corresponding schema definitions.

### ***1.1. Web services and the semantic web: dichotomy or ... ?***

Whereas the acceptance of XML is virtually unanimous, opinions are much more divided with regard to how its undeniable benefits are to be exploited in present web development efforts. On the one hand, there is the view of many commercial enterprises, which look upon XML as the enabling technology for a web that consists of *services*, which interact by means of the web equivalent of remote procedure calls. On the other hand, there is the vision of the *semantic web*, which looks upon the web as a colossal repository of *knowledge*, which can be reasoned upon by intelligent web agents. This paper arguments how both visions, one industry-driven and somewhat pragmatic, the other still ill-defined and premature, should not necessarily contradict one another, but may actually gain from each other’s insights.

First, it is shown how current state-of-the-art web service technology can be considered as a web variant of distributed object architectures such as CORBA and DCOM. Thereafter, accompanying web service description

technologies are discussed, such as UDDI and WSDL. It is then advocated how semantic web concepts can be applied successfully to enhance existing web service description technologies. First the *semantic* layer is discussed, based on RDF and RDF Schema. Thereafter, a brief overview is provided of current attempts to standardize the *ontological* layer, with emphasis on the potential contributions to web service description. A last section formulates conclusions and suggests issues for further research towards the enhancement of current web ontology efforts, seen in the light of web service description.

## 1. Towards a web of services

### 1.2. Introduction

The evolution from a web of static pages to web services is closely related to the way in which the web has been exploited by companies to do business. At first, companies were just “present” on the web. It was used for unidirectional communication by means of static HTML pages, which provided product information and a general overview of the company. In a second stage, company web sites acquired a measure of interactivity. They were mainly targeted at B2C communication, not only informing customers about the company, but also allowing them to directly purchase products or services through the web site. Techniques involved were e.g. CGI scripts, servlets etc. Note that, in this stage, the end user was still supposed to be a human being.

In a third phase, it became obvious that the real value of doing business on the web was primarily to be found in the automation of B2B interaction. Initially, this was mainly accomplished by replacing paper documents such as product catalogues, purchase orders, invoices etc. by an electronic counterpart. XML was utterly useful as data format for this purpose, as it allowed annotating the documents’ content with tags that described the *meaning* of each element, such as price, quantity, delivery address etc. Moreover, interacting companies could mutually agree on a DTD or XML schema for each document type involved in their transactions, such that the information embodied in a valid document could be very easily processed automatically: an information system knew what it could expect. Although several efforts exist to define standard XML tag sets for various kinds of business documents (e.g. RosettaNet [21]), this approach was mainly taken by companies that were already long standing business partners and already possessed established interaction protocols, e.g. by means of EDI, fax or paper. The XML documents were just an electronic version of these existing document types. Most important was that many “standard” interaction

patterns could now be automated: where possible, communication was conducted between computers instead of between humans.

However, instead of being document-based, automated B2B interaction can be accomplished by means of another paradigm as well. The latter originates in the distributed object architectures that are well established as an inter-application interaction mechanism in a single enterprise and which are essentially object-oriented variants of remote procedure calls. However, although techniques such as DCOM, RMI and CORBA are successful on the local network, they largely fail when transposed to a web environment. They are rather unwieldy, entail too tight a coupling between components and above all conflict with existing firewall technology. However, all of this changed when a simple, lightweight RPC-based mechanism was devised, that was independent of Internet wire protocols and internal application architectures: SOAP (Simple Object Access Protocol) [23]. SOAP uses XML messaging over plain HTTP, thus avoiding firewall problems (asynchronous communication can also be accomplished via SMTP). Although SOAP supports document-based as well as RPC-based interaction, the latter seems to become the predominant technology for B2B communication. Hence SOAP is basically a technology to allow for RPC over the web. It consists of a very simple request/reply mechanism: the client marshals a method call and parameters into an XML document, which is sent to the server. The server parses the document and marshals the result into another XML document, which is then returned to the client.

*Web services* can then be defined as self-contained, modular software components that expose specific business functionality on the Internet, such that other applications can use them by means of established web protocols and data formats such as HTTP and XML. SOAP is becoming the de facto standard for web service interaction. Enterprises publish only a limited set of services instead of entire applications. This approach closely resembles distributed object architectures such as CORBA, where a limited set of methods on an object's public interface can be called remotely, with the actual implementation details being *hidden* from the outside world. The biggest difference is SOAP having a much more lightweight implementation, which is far less obtrusive to existing web standards and firewall protocols.

Such service-based interaction allows for integrating and aggregating information systems (and business processes) along the entire value chain into an ad-hoc distributed environment of loosely coupled components that interact by requesting services from one another. A service can provide information, e.g. a weather forecast service; it may have an effect in the real world, e.g. an online flight booking service, or both.

## ***1.2. Advertising and discovering web services***

Obviously, a key requirement for a successful web service is that potential users are able to discover and use it. If interaction should be accomplished in an automated way, other applications should be able to *find the right service* and *obtain the information necessary to interact with it*. Hence an automated advertising and discovery mechanism is needed. In its most basic form, this could be seen as publishing a simple text document describing the service on the web. However, traditional text-based searches are unsatisfactory because they are largely inaccurate, e.g. the same search term may have a different meaning in different contexts, synonyms are not taken into account etc. To facilitate and improve web service advertising, discovery and invocation, SOAP has spawned two other technologies: UDDI (Universal Description, Discovery and Integration) and WSDL (Web Service Description Language).

### ***1.1.1. UDDI***

UDDI [24] defines a framework that allows organizations to discover services offered by other organizations as well as describe their own services and explain how they want to conduct business over the Internet. At the heart of UDDI sits the XML-based virtual *UDDI Business Registry directory*. The latter can be seen as a physically distributed, but logically centralized collection of registries in which the entries describe business entities and the web services they offer.

The UDDI specification is described by means of XML Schema. The information provided contains four levels: the top level element is the *Business entity*, which provides general data about a company such as address, a short description, contact information and other general identifiers. This information can be seen as the “white pages” of UDDI. Associated with each business entity is a list of *Business services*. These contain a description of the service and a list of categories that describe the service, e.g. purchasing, shipping etc. Categories of services can be further specialized according to standard taxonomies such as industry branch, product, geographic location, etc. The latter can be considered the “yellow pages” of UDDI. Within a business service, one or more *Binding templates* define the “green pages”: they provide the more technical information about a web service. The latter is required by application programs that intent to use the service, e.g. the web address to contact the service, specification of types of input and output etc. They also associate a service with a *Service type*. The latter is defined by a *tModel*, which contains pointers to low-level technical specifications for the service type such as message protocols, the

exact format of possible input and output etc. The latter is conceived as an opaque set of identifiers that refer to the actual information sources that provide technical details. In this way, many companies can provide web services that implement the same service type and are compatible with the same high-level specifications.

UDDI can be seen as the “telephone directory” of web services. As such it represents a service itself, with information about other services as “merchandise”. Direct interaction with the UDDI registry is possible via SOAP. UDDI defines two API’s: an *Inquiry* API and a *Publisher’s* API. However, the majority of the queries are likely to be conducted through existing Internet search portals and marketplaces, which use the UDDI repository as a data source. In this way, UDDI can be seen as a complement to these, rather than a substitute. The second API can be called by a business to register the services that it wants to expose for use by other businesses.

#### 1.1.2. WSDL

Whereas UDDI provides a “forum” for advertising web services, it is complemented by WSDL [29], which builds upon XML schema to define an actual XML *vocabulary* for such description. WSDL describes a web service by means of the *format* of a request to the service. It looks upon the web (services) as a collection of *endpoints* or *ports*. Endpoints interact by exchanging and operating on *messages*. A message defines the data format to a single request or response in the communication. Each possible request or response is described by a message element. A message may contain either procedure-oriented or document-oriented information: it may represent a procedure call and its parameters or an actual document being exchanged. WSDL makes use of XML schema for low-level data typing. *Operations* define abstract definitions of an action supported by the service. A *port type* groups an abstract set of operations that make out a single logical operation, as supported by one or more endpoints.

*Reuse* of service definitions is facilitated by initially defining messages (i.e. descriptions of the data being exchanged) and port types (i.e. collections of operations) on an abstract level, after which they can be associated with a concrete network protocol and message format by means of a *binding*. A *port* is defined as the combination of such binding and a network address. Finally, a service is perceived as a collection of ports. Hence a service can be seen as the mapping of port types and bindings to the URI where the service can be called. In this way, the same service (i.e. with the same description) can be offered from multiple URI’s.

Recently, WSCL (Web Services Conversation Language) [28] has been developed as a complement to WSDL. Whereas the latter specifies how to send messages to a service, it does not state the order in which such messages are allowed to be sent. This issue is addressed in WSCL, which defines legal sequences of document exchange between web services. Other emerging standards, which partially overlap with UDDI and WSDL are ebXML [6] and E-speak [7].

### **1.2. Evaluation**

The technologies described previously each attribute their piece to the jigsaw of automated web service interoperability. The latter can be depicted in layers. At the lowest level, XML is irrefutable as the standard for data encoding and formatting. SOAP packages these data and transfers it from system to system. Above that, SOAP is bound to an actual web protocol such as HTTP or SMTP. At still a higher level, UDDI offers a forum for companies to describe the services they offer and inquire about services made available by other companies. WSDL provides a language for such descriptions.

Although not all specifications in the scheme have already attained the status of W3C *recommendation*, industry commitment to them is already impressive. This can be explained by the fact that, whereas these technologies are quite novel and maybe didn't even get the time to fully mature, they can be interpreted as web equivalents of well-proven concepts from the distributed object world. E.g. UDDI is related to the concept of CORBA interface repositories, whereas WSDL can (to a certain extent) be considered the IDL of web services.

However, whereas LAN-based technologies such as CORBA still pertain to a manageable number of services, transposing a similar approach to a world-spanning environment such as the web may result in very poor searching performance, especially considering the expected growth rate of the number of services. UDDI only accommodates for a rough, first-level filtering of available services; a more detailed selection will have to be carried out by means of direct communication with the service provider. Obviously, this is unacceptable when the number of initial search hits is huge and one has very detailed requirements about the specifications of the service to be used. Also, whereas traditional distributed object technologies base their search criteria on the *type of interface* an object implements, this may be totally inadequate for web service selection. Web services not only deliver some kind of return value to a request, they may also have an effect in the real world. This effect may be a much more important search criterion; the web service's "interface" will only be a secondary facet.

Still, WSDL describes a web service by means of the *format* of possible request and responses. UDDI provides a categorization mechanism according to “real-world” criteria such as industry branch, product type and geographic location, but it is in no way destined at discovering services based on fine-grained specifications of what is required from the service. Therefore, we advocate that resource description technologies that are categorized under the denominator “*Semantic Web*” may very well fit into the framework of *web services* as well. Moreover, it will become clear that, whereas the vision of the semantic web has come no way near its full potential yet, this does not prevent some of its techniques to be already utterly rewarding in the web of services that exists today.

## 1. Attributing semantics to Web resources

### 1.3. Introduction

The approaches sketched above make use of XML as a standardized way of exchanging structured data. XML allows for self-defined tags, which carry a domain-specific meaning. Also, the capabilities of DTD’s and XML schemas to define document types allow the receiving system to automatically check whether a document satisfies the specifications of its type. Hence the metadata are used in a *prescriptive way*, to check for document validity. Initially, such document types reflected a *business reality*, i.e. they were modeled after their paper-based predecessors such as purchase orders and invoices. However, with an RPC-based approach gaining over pure document-based interaction, XML is used in short messages that describe method calls to web services. Obviously, the corresponding XML schema definitions are modeled accordingly.

More or less simultaneously with the emergence of web services, the concept of the *semantic web* was born. Once more, the web is envisaged as a medium for *machine-to-machine* interaction but the approach taken is a radically different one, although it equally has XML as a catalyst. This view keeps closer to the original *document-oriented* nature of the web. The document notion is retained, although abstraction is made from how such document is stored physically. Documents are not necessarily static; they can be stored statically as real documents, they can be published from data in e.g. a relational database or they can even be entirely virtual, such as XML-based dynamic views over relational database data. The only requirement is that they can be identified by a URI. Regardless of their origin, to make such documents machine-processable, semantic metadata are to be added, such that computers can better “understand” the documents.

The latter lead to the vision of a semantic web: a web of interconnected documents that are semantically sufficiently rich to allow for fully automated searching and processing [1]. Here also, XML was considered a godsend because the self-defined tags allowed for attributing semantics to *document content*. However, it soon became clear that self-defined tags alone were not sufficient for semantically rich document description. The same tag can have a completely different *meaning*, depending on the domain in which it is applied. For instance, a <client> tag may refer to a customer or to a part of a software architecture. On the other hand, different tags may actually be synonyms and should therefore be acknowledged as referring to the same real-world concept. Consequently, XML schema's can be deemed as inappropriate to attribute a semantic meaning to tags, as they mainly formalize the *document structure*, i.e. how elements are to be nested in one another, but not what these elements *mean*.

Therefore, apart from XML schema, another standard was developed for attributing metadata to web documents. Called RDF (Resource Description Framework), the standard has since reached recommendation status with the W3C. RDF can be seen as a very lightweight *ontology language*. An ontology can be defined as “*a shared understanding of some subject area which helps people or processes achieve better communication, interoperability and effective reuse. The ontology embodies a conceptualization - definitions of entities, their attributes and relationships that exist in some domain of interest. The conceptualization is explicitly represented.*” [25]. The next two sections provide an overview of RDF and its complementary standard, RDF Schema.

### **1.3. RDF**

RDF [13] is based on the extensive research efforts conducted in the field of *knowledge representation* upon such topics as semantic nets, frame systems and logic languages. Developed as a standard for capturing *semantic metadata*, it offers a syntax and data model to specify semantics to web resources with the explicit requirement in mind that these specifications should be machine processable.

RDF's data model consists of three building blocks. *Resources* may be anything: entire web documents, specific parts of documents, collections of documents but also real world objects such as persons, countries... anything that can be identified by a URI. *Properties* describe a resource by relating it to another resource or atomic value. This is accomplished in *statements*, which consist of the combination of a specific resource (the statement's *subject*), a named property (the *predicate*) and a value (the *object*), which may be a resource or a literal. The (subject, predicate, object) triple is RDF's

basic modeling primitive. Moreover, the concept of *reification* allows for a statement to be denoted as a resource itself, which can then become the subject (or object) of new statements. Because of this possibility of *meta-statements*, i.e. statements about statements, RDF's underlying data model can be looked upon as a labeled hyper-graph. As a consequence, a limited degree of automated *reasoning* is possible over RDF models.

An RDF specification is to be considered as metadata in that it describes another document (or resource in general). However, although XML schema is also targeted at specifying metadata to (XML-) documents, its purpose is very different. Instead of being concerned with document *structure*, RDF really deals with a document's *semantics* and *meaning*. Also, because RDF is graph-based and XML is tree-based, semantic relations between concepts can be modeled much more naturally in RDF than in XML. The same RDF specification can be represented in different ways in XML, e.g. a relation between a *book* and its *authors* can be denoted in XML as a <book> tag being nested in an <author> tag or vice versa. Furthermore, whereas XML Schema is *prescriptive* about a document's (structural) properties, RDF is in the first place *descriptive*: it attributes semantic properties to a document. These do not restrict the document in any way, but may assist in its interpretation. Finally, whereas XML schema is explicitly targeted at XML tags, RDF adds semantics to arbitrary resources without making any assumptions about their structure. Moreover, RDF metadata are stored separately from the document they describe, i.e. RDF does not (necessarily) relate document structure to conceptual terms. It can be applied both to structured information and to unstructured information. If a document is not written in XML but in e.g. HTML or a word processor format instead, its semantics can still be described in RDF. However, if the data document is not structured, semantics can only describe the document in its entirety. An XML document on the other hand, can be structured to reflect the semantic specifications laid down in RDF. For that purpose, the *descriptive* metadata in RDF can be translated to a *prescriptive* XML schema, such that XML documents that are instances of the schema comply with the semantics laid down in RDF. In this way, semantics can be attributed to a document's *elements*, instead of to the document as a whole.

A very important asset is that XML can be used as *serialization format* for an RDF description, i.e. an RDF specification can be written down in XML. As a result, it becomes very easy to exchange RDF metadata over the web as XML documents. It is worth noting that an RDF specification can result in two different kinds of XML documents: one may have the RDF model itself as content whereas the other may have data that *comply with* this RDF model as content.

Although a few of RDF's properties (e.g. its awkward serialization model) are vehemently criticized by some members of the W3C community, it appears to have become the standard for semantic modeling in the web. In our opinion, its two strengths are in that an RDF specification can be serialized to an XML document and the fact that RDF is very easily extensible, to overcome its initial weaknesses. A fundamental extension is RDFS (RDF Schema), as discussed in the next section.

### **1.3. RDF Schema**

RDF offers only very basic modeling primitives. RDFS [2] extends RDF by introducing a set of pre-defined concepts, of which the semantic meaning is specified externally. They allow for capturing *typing* semantics, comparable to an object-oriented model, which cannot be expressed by RDF alone. The concepts are, among others, the *class* resource, the *subclass* resource, the *subclass-of* property and the *instance-of* property as well as domain and range *constraints* to properties. RDFS is itself specified in RDF syntax. Moreover, it has the peculiar property of using its own constructs to define itself, e.g. the *subproperty-of* property is defined as, indeed, an instance of *property*.

RDF Schema takes a thoroughly different position towards RDF than the XML Schema specification does towards XML: XML Schema *prescribes* the order and combination of tags in an XML document. RDFS, conversely, does not constrain the syntactical properties of an RDF description in any way, but only *describes* and *extends* the description with additional semantics about the resources and properties used. On the other hand, an RDF Schema can be translated to an XML schema to *prescribe* a document type, of which the instances will comply with the RDF Schema specifications. The semantics expressed in this way will be much richer than with RDF alone, because of the abstract typing and constraint information that can be added by means of RDFS. Finally, because RDFS is written "in" RDF, it is obvious that an RDFS specification can be serialized by means of XML, just as an RDF specification can.

### **1.3. Evaluation**

Attributing semantic metadata to documents is essential if these documents are destined for automated processing. Although XML schema is a tremendous improvement over DTD's, its use remains largely restricted to modeling syntactic properties, not semantic ones. However, purely structural metadata is unsatisfactory as a basis for mutual understanding

between information systems. Indeed, data should not only be provided in a structured way, a remote system should also be able to interpret and process them. There is no doubt about XML's role as the universal syntax for web-based data exchange, but without proper semantic grounding, the *interoperability problem* remains: for the same kind of data, one is able to compose numerous different XML document types as DTD's or XML schema's, each with a totally different structure. Although it is possible to transform one such scheme into another by means of XSLT, defining such mappings may be a complex and laborious task, which can hardly be automated [18]. Obviously, this problem becomes even more stringent in the case of fast evolving schemas. Therefore, intelligent interaction on the web should better not be based on XML document structure, but on the *semantics* of the concepts embodied within a document [27].

Attributing semantic metadata to documents is essential if these documents are destined for automated processing. However, whereas semantic metadata is crucial to automated *document* processing, this is also true for *web services* that are destined for automated interaction. Whereas the semantic web vision was in origin document-oriented, the following section argues how RDF(S) is equally effective for describing web resources representing services that are to "understand" one another in order to be able to collaborate effectively. Semantic interoperability between web services can be based on a common vocabulary described in RDF, attributing semantics to both the services themselves and the messages/operations they use to interact. The XML syntax is still present at a lower level, to provide a serialization format to both the RDF metadata and to encode the messages used for interaction. As to the latter, XML has the advantage that the message structure will actually reflect the specification at a semantic level; such that semantic concepts can be extracted far more easily than if e.g. HTML is used. RDF(S) based descriptions can enhance web services in multiple ways, as discussed in the subsequent sections. The delineation of topics is loosely based on [17].

## **1. A semantic description of Web services**

### ***1.4. Description, advertising and automatic discovery of web services***

The primary goal of the semantic web is to facilitate automated searching and processing of documents, because the metadata allow for computers to better "understand" them. The latter is, however, not restricted to static documents, but can be applied to web services in two ways. First, it allows for describing the input and output parameters to the service. This can also be accomplished by means of WSDL, but RDF allows for a semantically

much richer description: the parameters themselves can be related to other resources for further specification, instead of only stating the parameters' *type*.

However, as already said, specifying a service solely in terms of its input/output data types only offers a very incomplete picture. Especially since the ultimate goal of many web services is to provoke changes in the *real world*, e.g. debiting a credit card in exchange for the delivery of a book at a certain address. The latter cannot be described adequately in terms of input/output. Particularly in such case, it is important to be able to describe what the service *does*. Again, RDF(S) is utterly appropriate for this kind of specification. In general, web service specification can be accomplished in RDF by depicting the service in a semantic network with the service itself at the "center". The web service is represented as an RDF *resource*, which is semantically connected to other resources. Some will represent input or output data specifications; others will represent descriptions of how the service influences the real world. Obviously, both (and possibly other) kinds of properties can be combined in a single search. Attributes describing a web service can in their turn be related to additional resources, e.g. to find synonyms, to discern between different possible meanings in different contexts etc. This is the best way to guarantee that service provider and service user effectively agree on the meaning of the terms used in the service description. Finally, the full potential of RDF Schemas allows for web service classification that is far subtler than the standard taxonomies of UDDI: superclass/subclass relationships and subproperties can be used to refine search criteria and *reasoning* on the specifications may even result in categorizations being derived that were not intended originally.

Applying semantic web techniques to encode the properties and capabilities of web services allows for more accurate descriptions, which can be used to enhance the quality of results produced by search engines and matchmaking systems, e.g. [11]. As such, it will become easier to find the required service, especially if the number of services will mount dramatically over the coming years and the need for more precise search criteria becomes stringent. Importance of the latter is not restricted to B2B interaction, but applies to B2C as well. An example is the development of intelligent agents that interact with web services on behalf of a human customer e.g. to book a flight, schedule an appointment etc.

#### **1.4. Automatic service invocation**

Once the appropriate web service has been selected, an even more complex problem is the ability to invoke it and interact with it in an automated way, without human intervention. For that purpose, potential users need to be

able to retrieve information about how a request to the service is to be conducted. This will again include details of the input that is to be provided and the output that can be expected. At this stage, also a lower-level, syntactical description of the exact message *formats* will be indispensable. Whereas such specification will be quite straightforward for very simple services such as currency conversion services, they may become considerably more complex in the case where multiple types of inputs and intermediate conditions may result in a diverse range of possible types of output, e.g. an online car rental service. In such case, an explicit description of the service's *logic* may be required. Also, "real world" properties, which cannot be considered input or output parameters but which may definitely affect the outcome of a transaction should be taken into consideration, e.g. whether or not a product is in stock.

Evidently, such kind of specification goes much further than the mere "interface" definition that can be provided by means of WSDL. Again, semantic relationships modeled by means of RDF prove their value. For instance if the value of certain required input parameters is not known to the calling application or agent, the semantic network of interrelated concepts may suggest alternatives or may provide information about where the needed input can be obtained or how it can be inferred.

#### ***1.4. Automatic web service composition and interoperation***

Taking the latter issue a bit further, we can imagine an agent that is provided with a high-level description of a complex task and autonomously conceives a strategy to accomplish it. The latter involves devising selection criteria to find services that may be of assistance in accomplishing the task. Searches are to be issued according to these criteria and the services that will actually be used are to be selected. In accordance with the specifications of the selected services, the agent should be able to combine them and interact with them in an appropriate manner. Again, service descriptions should not only entail input/output formats, but also specifications about the effect(s) on the real world. For that purpose, a process ontology can be specified, which describes individual and composite programs as either atomic or composite processes, such that their invocation and composition can be automated. Again, the typing system of RDFS allows for such services to be specified at a high level of abstraction and specialized and refined for concrete situations. For instance a high-level specification of a service ontology for a generic sale over the web can be specialized and extended with particular, customized features, for specific cases such as an online bookstore, a flight booking service etc.

#### **1.4. Evaluation**

An early attempt at an RDF-based standard for web service description is RSS (RDF Site Summary) [22]. However, as a first step, service providers can port their existing metadata regarding their services to RDF. The standards for service advertisement discussed in section 1.2, such as UDDI, WSDL, ebXML and E-speak can be reused quite easily in an RDF context. Transformation schemas exist for representing their specifications by means of RDF primitives, e.g. [19]. These RDF-based specifications can then be serialized as XML documents. The combined specifications provide an embryonic semantic network, which can gradually be enriched with further metadata. In this way, a web service becomes a central resource in a web of other resources, some of which provide access points to the service, others contain static documents that describe the service with metadata such as name, description, cost, message schema, etc to cater for automated discovery and invocation. The corresponding service descriptions could be automatically incorporated into RDF-aware search engines and classification systems.

### **1. A layered semantic Web**

#### **1.5. RDFS specifications as the semantic layer**

XML schema and RDF(S) should not be seen as substitutes, but rather as complementary specifications. [12] and [20] make the comparison to relational database design: the RDF model fulfills the same role as the *Entity-Relationship* model, capturing real-world semantics as truthfully as possible. An XML Schema can be compared to a logical database model, modeling the logical data structure based on these conceptual specifications. Finally, the XML code itself is comparable to the relational data.

However, although the previous sections already showed the clear advantages of attributing semantic specifications to web resources, the complete semantic web vision is even more ambitious than that. The entirety of the semantic web is still quite ill-defined, the most complete overview to date can be found in [1]. Here, the RDFS specification is in itself considered as a semantic layer above the structural layer based on XML schema definitions, but with still some other layers above it. At the lowest level are the actual documents, which are preferably made up in XML, although this is not strictly required. Namespaces allow for unique identification of tags. The layer above describes the document types at a *structural level* by means of XML schema definitions. A third layer describes the resources at a *semantic level*, by means of RDF and RDFS. Above the semantic layer sits

the *ontology* layer. Above the ontology layer are still other layers situated: a *logic* layer, a layer of *proof* and a layer of *trust*. The three uppermost layers have not been realized to date and merely exist as abstract ideas. Current research primarily focuses on the ontology layer. Although these efforts are still far from standardization, the insights gained may already be pertinent to web service description.

### 1.5. *Beyond the semantic layer*

Indeed, RDFS provides a means to define semantics to web resources and can in itself be seen as a very basic ontology modeling language. However, RDFS lacks expressivity to be truly adequate for full-fledged ontological modeling and reasoning [3]. It does not define necessary or sufficient conditions for class membership, nor equivalence and disjointness of classes. Moreover, constraints on properties are restricted to domain and range. Its biggest lacuna, however, is the fact that its own semantics remain under-specified [17]. Therefore, current research is targeted at defining more rich ontological languages, based on XML markup. Earlier efforts such as SHOE [15] and Ontobroker [8] were specified directly above HTML. More recently, it is envisaged to *extend* RDFS rather than replace it. Indeed, RDFS is very easily extensible: in the same way as RDF Schema is used to define itself, it can be used to define other ontology languages. In this next section we will discuss the most mature of these languages, called DAML+OIL and indicate their relevancy to web service description. An overview of other approaches is provided in [5].

DAML+OIL [26] is the result of a merger between two existing ontology languages: DAML-ONT (DARPA Agent Markup Language) [16] and OIL (Ontology Inference Layer) [10]. DAML+OIL defines extension to RDF, such that a full-fledged knowledge representation language can be expressed in it. For that purpose, formal semantics, borrowed from description logic, are combined with frame based modeling primitives and a syntax based on XML. DAML+OIL extends RDF with efficient reasoning support, formal semantics and enriched modeling primitives. As to the latter, where RDF properties could only be constrained in their domain and range, DAML+OIL allows for defining additional property constraints, such as cardinalities, inverse and qualifiers such as “transitive” and “symmetric”. DAM+OIL primitives can be defined by means of RDF Schema, such that any DAM+OIL ontology can be expressed in RDF syntax. As a result, DAML+OIL is fully backward compatible with RDF(S): ontologies written in DAML+OIL are valid RDF documents. They can still be partially interpreted by any RDFS-only processor [3].

DAML-S [4] is built on top of DAM+OIL. Explicitly focused on web services, it defines several ontologies in the DAM+OIL markup language to enhance web service description. Automated *discovery* is facilitated by means of a markup language for encoding properties and capabilities of a web service. Moreover, a process ontology is defined to cater for automated service *invocation*. Finally, automated service *composition* and *interoperation* is supported by means of *preconditions* and *effects*, which define respectively the prerequisites and consequences of using an individual service. Further research envisages the development of DAML-L, a logical language, which is to provide support for rules and reasoning.

The DAML family of languages (and DAML+OIL in particular) can be considered as the basis on which the Web Ontology (WebOnt) working group endeavors to come to a standardized web ontology language. According to its charter [9], such language is to extend RDFS to allow for more complex relationships between entities, including a means for inferring implicit class membership and a well-defined model of specialization and property inheritance and overriding. Applied to web service description, such language will allow services to truly interact on a semantic instead of a syntactic level. Again, ontologies can be put to use in three ways: to facilitate automated *discovery* of a service, to facilitate automated *invocation* and to facilitate *automated composition and interoperation* between multiple services. Future layers above this ontology layer may allow for *reasoning* and *proof*. In this way, agents will be able to use assertions from around the web to derive and prove new knowledge, e.g. to ascertain that a given transaction has been completed in a satisfactory way. Finally, a *trust* layer, based on encryption technologies, may allow for agents to trust assertions of other parties, instead of re-inferring all knowledge necessary to complete a transaction from scratch. However, actual implementation of the latter layers is by no means a prerequisite for the semantic and ontology layers to be valuable in the web service environment of today.

## 1. Conclusions and considerations

Whereas the previous section already summarized the value of the semantic web vision to web service description, this final section suggests a few topic to be taken into consideration with regard to a possible standard ontology language for the web.

### ***1.6. Classification or constraint checking ?***

As already said, RDF(S) primarily *describes* resources instead of *prescribing* constraints to them. In general, an ontology provides a domain theory rather than data structures. Moreover, RDFS is *property centric* in that properties are not defined as attributes of objects, but are considered first class objects themselves. A class instance is just a resource referred to by a URI, without any value or state; it doesn't need any properties to be a valid instance of its class. Also, object classes are not allowed to use the same property name with different domain and value constraints.

Although it can be seen as a defining feature of the Web that “anyone can say anything about anything” and although it is obviously impossible to enforce global integrity constraints that span the entire web, we think it's important at least to be able to enforce *local* consistency on e.g. a single service description. For that purpose, RDFS will have to be extended with the possibility of enforcing such constraints. The latter is already acknowledged in the DAML+OIL specification. On the other hand, because there are no “absolute” truths in the web, the concepts of proof and trust may become utterly important when they are actually realized.

### ***1.6. Inherit-and-override or copy-and-paste ?***

Even if the data cannot be kept consistent, it should at least be possible to enforce consistency upon the metadata. The most important relationship in an ontology is arguable the *is-a* relationship. However, the semantics surrounding the concepts of inheritance and overriding that result from an *is-a* relation are very poorly defined in RDFS, XML Schema and (to a lesser extent) in DAML+OIL. This is very unfortunate, because the ability to *reuse* existing service descriptions depends largely on the ability to “borrow” functionality from a higher-level specification and adapt it to one's own particular needs. For instance the service description for *buying an airline ticket* can be seen as a specialization from a more general *buy-ticket* service description, which may in its turn be based on a highly generic *purchase* service description. However, allowing inheritance seems quite dangerous if there is no way to ascertain that the subclass specifications are still consistent with the higher-level specifications or, conversely, if properties cannot be overridden at all to accommodate for one's own particular needs. A well-defined model of property inheritance is required, as also stated explicitly in the WebOnt Working Group charter [9]. Note that, as to ontologies, the concepts of inheritance and overriding are even more intricate because of the possibility of multiple inheritance and classifications being derived implicitly.

A related issue is the concept of *property* subtyping, such that a subproperty models a more specific relationship than the “parent” property. Also here, formal overriding semantics are to be declared. We already addressed the issues of property inheritance and property subtyping in depth in the context of “link subtyping” in a hypermedia environment [14], which describes exactly the same problem, be it that the terms “node type” and “link type” are used instead of “class” and “property”.

### 1.6. *Specification of behavior*

A last remark to be made is with regard to specifying web service behavior. Current web service description standards are still primarily targeted at stipulating data structures. They remain utterly informal with respect to the specification of *operations*, maybe except for the efforts made in DAML-S. One of our current research issues is the applicability of UML use-case scenarios to modeling web service behavior, especially concerning its effects on the “real world”.

## 1. References

- [1] T. Berners-Lee, M. Fischetti and T. Dertouzos, Weaving the web: The Original Design and Ultimate Destiny of the World Wide web by its Inventor, Harper, San Francisco, 1999.
- [2] D. Brickley and R. Guha Eds., Resource Description Framework Schema Specification, W3C Candidate Recommendation, 2000.
- [3] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks, Enabling knowledge representation on the Web by Extending RDF Schema, Proceedings of the Tenth International World Wide Web Conference, Hong Kong, 2001.
- [4] DAML-S official website, <http://www.daml.org/services/damls/2001/05/>, 2001.
- [5] M. Dimitrov., XML standards for Ontology Exchange, Proceedings of “OntoLex 2000: Ontologies and Lexical Knowledge Bases”, Sozopol, Bulgaria, 2000.
- [6] ebXML official website, <http://www.ebXML.org>, 2001.
- [7] E-Speak official website, <http://www.e-speak.hp.com/map.shtm>, 2001.
- [8] D. Fensel, S. Decker, M. Erdmann and R. Studer, Ontobroker: Or How to Enable Intelligent Access to the WWW, Proceedings of the eleventh International Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, 1998.

- [9] J. Hendler Ed., Web Ontology (WebOnt) Working Group Charter, 2001.
- [10] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. Van Harmelen, M. Klein, S. Staab, and R. Studer: The Ontology Interchange Language OIL. Technical Report, Free University of Amsterdam, 2000.
- [11] G. Karvounarakis, V. Christophides, D. Plexousakis and S. Alexaki, Querying Community web Portals, yet to appear, 2001.
- [12] M. Klein, D. Fensel, F. Van Harmelen and I. Horrocks, The relation between ontologies and XML schemata, Proceedings of the fourteenth European Conference on Artificial Intelligence, Berlin, 2000.
- [13] O. Lassila and R. Swick Eds., Resource Description Framework Model and Syntax Specification, W3C Recommendation, 1999.
- [14] W. Lemahieu, MESH: A Model-Based Approach to Hypermedia Design, in: Q. Chen (ed.) Human Computer Interaction: Issues and Challenges, Idea Group Publishing, Hershey, PA, 2001.
- [15] S. Luke, L. Spector, D. Rager and J. Hendler, Ontology based web agents, Proceedings of the First International Conference on Autonomous Agents, Marina Beach, CA, 1997.
- [16] D. McGuinness, R. Fikes, L. Stein, and J. Hendler, DAML-ONT: An Ontology Language for the Semantic web, to appear in D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster Eds., The Semantic web: Why, What, and How, MIT Press, 2001.
- [17] S. McIlraith, T. Son and H. Zeng, Mobilizing the Semantic web with DAML-Enabled web Services, Proceedings of the Second Int'l Workshop on the Semantic Web, Hongkong, 2001.
- [18] S. Melnik and S. Decker, A Layered Approach to Information Modeling and Interoperability on the Web, Proceedings of the ECDL 2000 Workshop on the Semantic Web, Lisbon, Portugal, 2000.
- [19] U. Ogbuji, Supercharging WSDL with RDF, IBM developerWorks paper, 2000.
- [20] R. Rami and B. Nabila, A Translation Procedure to Clarify The Relationship Between Ontology and XML Schema, Proceedings of the International Conference on Internet Computing, IC'2001, Las Vegas, USA, 2001.
- [21] RosettaNet Implementation Framework RNIF, version 2.0, RosettaNet Consortium, <http://www.rosettanet.org>, 2001.
- [22] RSS official website, [www.purl.org/rss/1.0/](http://www.purl.org/rss/1.0/), 2001.
- [23] Simple Object Access Protocol (SOAP) 1.1, W3C Note, 2000.
- [24] UDDI Technical White Paper, Ariba, IBM Corporation and Microsoft Corporation, 2000.
- [25] M. Uschold and M. Grüninger: Ontologies: Principles, methods and applications, Knowledge Engineering Review, 11(2), 1996.

- [26] F. van Harmelen, P. Patel-Schneider and I. Horrocks, Eds., Reference description of the DAML+OIL ontology markup language (revision 4.2) 2001.
- [27] H. Wache, T. Vögele U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner Ontology-based Information Integration: a survey, yet to appear, 2001.
- [28] Web-Services Conversation Language (WSCL) 1.0 specification, [www.e-speak.hp.com/specifications/wscl.shtm](http://www.e-speak.hp.com/specifications/wscl.shtm), 2001.
- [29] Web Services Description Language (WSDL) 1.0. specification, <http://msdn.microsoft.com/xml/general/wsdl.asp>, 2001.