Bringing Together Semantic Web and Web Services

Joachim Peer

Institute for Media and Communications Management
University of St. Gallen
Blumenbergplatz 9, 9000 St. Gallen, Switzerland
joachim.peer@unisg.ch

Abstract. There are two major ongoing efforts to advance the World Wide Web. On one side there is the Semantic Web research, on the other side is the Web Service research. Both activities aim to make content on the web accessible and usable not only for humans but also for machines in order to create a foundation for intelligent automated services and business processes. These two efforts are highly complementary, and there is work in progress towards a unification of them. This paper contributes to this process of unification by presenting a method of connecting Web Services descriptions with Semantic Web ontologies.

1 Introduction

The World Wide Web is built for human use rather than for use by machines. At the moment we can identify two efforts being undertaken to make information on the web better accessible to machines. One of the efforts is the Web Service initiative, the other effort is the Semantic Web research.

It is the objective of both of these efforts to create a "next generation web", which provides new means for machines to use information on the web. While the Web Service effort is primary focused on syntactical standardisation of data exchange and service publishing, the Semantic Web effort is focused on expressing explicit semantics on the web.

This paper shows that the Semantic Web and Web Services are different but complementary approaches which can be combined to enhance the current web. Furthermore this paper will demonstrate that the convergence of these two efforts can be achieved with technology already available today. As a first step to validate this assumption the paper will present a concept of semantic annotations of WSDL documents.

This paper is structured as follows: Section 2 presents the existing background for this paper by describing the two approaches to provide a "next generation web" as well as the needs for their combination and the gap existing between them. Section 3 describes an approach to bridge this gap and illustrates it by an example. The paper concludes with a summary of findings and an outlook of activities to implement the suggested solution.

I. Horrocks and J. Hendler (Eds.): ISWC 2002, LNCS 2342, pp. 279–291, 2002.

2 Foundations

2.1 Web Services

Web Services are self contained, self described modular applications, which can be published, found and used on the web [12]. The infrastructural basis of the Web Services concept is formed by a small set of XML based standards.

Web services are important cornerstones of emerging architectures and application frameworks like Microsoft.NET, IBM Dynamic eBusiness and Sun ONE. Even eCommerce architectures like eSpeak or ebXML have been adapted to fit into the Web Service infrastructure. These and many other ongoing activities indicate the industrial backing of the Web Service efforts. The development of Web Service related efforts is coordinated by the World Wide Web consortium and by industrial consortia like the Web Service Interoperability Group.

The original motivation behind these activities is to enhance interoperability between heterogeneous information systems. Interoperability is primarily needed by two majors areas of application: on the one hand it is needed for enterprise application integration (EAI) in order to connect separated systems quickly and at low costs; on the other hand, interoperability is needed by business to business (B2B) integration, to reduce costs and enhance flexibility of cooperation.

The current Web Service architecture is built around a small set of de facto standards for message transfer, Web Service description and Web Service discovery:

Web Service Standards. The standard for the exchange of messages between Web Services is called Simple Object Access Protocol (SOAP) [22]. The SOAP specification defines an XML schema for a container which holds messages to be transported; it also defines a set of encoding rules and a convention for the representation of remote procedure calls (RPC) and responses.

The concept behind SOAP is not new. Many of its ideas are derived from RPC-related techniques like DCOM, CORBA or RMI. The key benefit of SOAP is its platform independence which its predecessors could not provide. SOAP is neither bound to Windows, nor to Java nor to some implementations of Object Request Brokers. This makes SOAP the preferred communication protocol in the world of Web Services.

The current standard of the description of Web Services is the Web Service Description Language (WSDL) [23] which is currently under review at the World Wide Web consortium. The syntax of WSDL is defined in XML Schema. If SOAP is to be compared with RPC related techniques such as CORBA or RMI, WSDL has to be compared with Interface Definition Languages (IDL). The technical requirements and details needed when accessing a Web Service are described by a WSDL document. A WSDL document describes the location of a web service, its available operations and their associated messages and data types as well as the format of their result values.

This paper will use some elements of the WSDL terminology; therefore the most important elements of WSDL documents will be described briefly: The

XML Schema types of message parameters may be defined using a `<types>` element. A `<message>` element is needed to compose such data types into messages. Messages need to be grouped into operations, which may define an `<input>`, an `<output>` and a `<fault>` message. Depending on the presence or absence of input- and output messages, operations may be defined as one-way messages, request/response messages, solicit/response messages or notifications. Operations are grouped within a specific `<portType>` element. All WSDL elements described so far allow the definition of web service signatures in an abstract and portable manner. To make these abstract operations available via a defined method of transport, `<binding>` elements are used which declare message formats and protocol details for operations and messages of a particular portType. An individual endpoint address of a binding is defined by a `<port>` element, which in turn may be referenced by a `<service>` element. WSDL documents can incorporate information from other WSDL documents using an `<import>` element. This allows modularization of WSDL descriptions and stimulates the reuse of common abstract messages, operations and data types. This way WSDL enables the standardization of abstract Web Service descriptions.

In order to help potential users to discover Web Services for particular tasks, infrastructures like "Yellow Pages" are needed. Various different architectures are proposed for this kind of service. The most important concept is the registry-based UDDI (Universal Description, Discovery and Integration) framework, which enables the registration of businesses and services using various kinds of taxonomies (e.g. UNSPSC, SIC, NAICS). Web Services that are not enlisted in registries may be published on Web Pages using Web Service Inspection Language [9] documents, which may be processed by search engines.

Limitations of Web Services. The Web Service standards described above are technical conventions which allow parties to easily exchange information in a standardized manner. These standards solve many problems on the technical level but the semantics of Web Services and Web Service descriptions as a whole are not addressed by them. The following fragment of a Web Service description illustrates this fact:

```
<message name="getTemperatureRequest">
  <part name="zipcode" type="xsd:string"/>
</message>
<message name="getTemperatureResponse">
  <part name="temperature" type="xsd:float"/>
</message>
<portType name="TemperatureServicePortType">
  <operation name="getTemperature" >
    <input message="tns:getTemperatureRequest"
           name="getTemperatureRequest"/>
    <output message="tns:getTemperatureResponse"
           name="getTemperatureResponse"/>
  </operation>
</portType>
```

An agent capable of WSDL can process this data structure and is able to interpret that the service offers messages called `getTemperatureRequest` and `getTemperatureResponse`. It can also interpret that these messages are used by a request/response operation called `getTemperature`. Furthermore it can determine the name and XML Schema types of the parameters `zipcode` and `temperature`. But it will not be able to figure out the actual meaning of these operations, e.g. that the service described will take Zone Improvement Plan (ZIP)-code of a North American city and will return the temperature of the air in that city expressed in units of Fahrenheit.

An agent is only able to interpret and process such information correctly if its internal model of the domain is in some way connected to the hidden semantics of the WSDL description. This kind of connection can be achieved if both the internal model of the agents and the description of the web service are related to a shared conceptualization.

A simple kind of a shared conceptualization is a domain-specific *standard*, which assigns a certain semantic meaning to a specific syntactical structure. The modularization of WSDL allows to define such standards; e.g. domain specific port types (and its supporting operations, messages and data types) may be multilaterally declared as a standard an may be incorporated into individual web service descriptions using WSDL `<import>` statements and can be bound to a specific protocol and physical address using `<binding>` and `<port>` elements.

As a result, agents programmed against that domain-specific standard descriptions can interact with every Web Service compliant to that standard; the agents' internal model is connected to the semantic meaning represented by the standard. This kind of connection is typically established by a developer who interprets the standard and incorporates its meaning into the agent she/he is implementing.

The benefits of this concept are the simple means for implementation and the predictable behaviour of automatic agents. The disadvantages of this concept can be described as follows: agents programmed against a specific set of Web Service interfaces can only solve a very restricted set of problems using a very restricted set of problem solving methods; they are basically static and inflexible.

If the semantics of such standards would be described explicitly in a machine interpretable manner, agents could gather information about purpose and usage of services at runtime and would be able to behave in a more flexible manner. This problem area is the object of investigations undertaken by Semantic Web research.

2.2 Semantic Web

The Semantic Web can be defined as "an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [1].

The core concept of the Semantic Web is the representation of data in a machine interpretable way. The basic means for this purpose is provided by the Ressource Description Framework (RDF) [18]. RDF allows the representation of

data as triples of subjects, predicates and their values, thus providing a basic and widely applicable binary propositional language.

To build more complex models, a Frame-like technique [15] is needed to separate the universe of discourse in classes, properties and their instanciations thus providing stronger modelling capabilites. These are provided by the RDF Schema standard [19].

The use of RDF and RDF Schema enables the construction of semantic nets [17] which can describe all kinds of resources. However, the problem with semantic nets is that they tend to lead to very complex graphs with sub-optimal reasoning behaviour.

A proposed solution to this problem is to restrict the expressiveness of semantic nets. This topic is well studied by the research field of Description Logics. Description logics restrict semantic nets to adhere to a controlled set of epistemological constructors ("primitives") which may be used as building blocks defining complex structures (ontologies[1]).

The set of constructors supported by a Description Logics system determines its reasoning behaviour. These relationships are well studied not only in theory but also in concrete systems like FaCT [8] or CLASSIC [2].

Efforts to leverage the benefits of Description Logics to the Semantic Web have led to the creation of the ontology language DAML+OIL. The syntax and some basic semantics of DAML+OIL are defined using RDF Schema. The precise semantics of its constructors are defined by axiomatic semantics expressed in first order logics [3] as well as by model theoretic semantics [7].

Applying Semantic Web Technology to Web Services. Adding semantic information to syntactical Web Service definitions can help an automatic agent to better interpret the purpose and usage of Web Services, thus leading to a higher level of flexibility. However, this assumption is based on two premises (a) that the referenced ontologies are built in a way that supports automatic reasoning and (b) that the agent is able to relate the concepts provided in the ontology to concepts of its internal model of the world.

As it has been discussed above, premise (a) can be fulfilled by using Description Logics based ontology formalisms like DAML+OIL. However, premise (b) can only be satisfied if the elements used in the referenced ontology are derived from concepts, which are common to all parties involved. This calls for standardised top level ontologies and agreements of the semantics expressed by that ontologies.

A DAML+OIL based framework providing such well defined ontologies for semantic markup of Web Services is DAML-S [12,14], which is currently under development and available as version 0.6. DAML-S aims to provide a set of ontological constructs which enables agents to automatically discover, evaluate and invoke Web Services, potentially as a part of an overall task (workflow). Addi-

[1] This paper uses the definition provided by Gruber [6], who defines an ontology as a "formalization of a conceptualization".

tional capabilities like monitoring of Web Services are supposed to be supported by future versions of DAML-S.

DAML-S consists of a couple of top-level ontologies containing constructs to describe various aspects of services needed to express its aim and usage. They are expressed by a "ServiceProfile", a "ServiceModel" and a "ServiceGrounding".

A ServiceProfile contains information needed to get a high level overview of the purpose of a service, its general input and output, its requirements (preconditions like membership or financial liquidity) and the effects of its execution. Moreover a ServiceProfile may provide non-functional aspects of the service like guaranteed levels of quality and it can also provide some information about the service provider. This kind of data is needed by an agent to quickly determine the applicability of a particular service for a particular task.

A ServiceModel provides a more detailed description of the operations provided by the Web Service. It allows the description of a Web Service in terms of a set of processes and the input, output, preconditions and effects of each of the processes. The process structure may be built in a recursive way, i.e. services may consist of a set of sub-services. ServiceModels may also contain statements about the runtime behaviour and interaction patterns of processes by defining workflow constructs like conditional switches, loops or parallel executions.

This kind of modelling enables both the fine-grained description of Web Services on the lower level of operations and message-parameters as well as the description of a Web Service and its behaviour as a whole.

The Remaining Gap. In contrast to the ServiceProfile and ServiceModel the proposed ontology for a ServiceGrounding does not exist yet. Therefore it is currently not possible to exploit the benefits of the semantic description of Web Services. This paper provides an approach that can provide that missing link for the domain of Web Services.

3 Proposed Solution

This paper proposes a method to bridge the gap between the semantic descriptions of Web Services (i.e. DAML-S ServiceProfiles and ServiceModels) on the one side and the technical descriptions (i.e. WSDL documents) on the other side.

The motivation behind our approach is as follows: all the technical aspects of Web Service usage are already described in WSDL documents. Instead of modelling all of that information in a meta data language, we propose to use the data in the WSDL document directly. The bridge between WSDL descriptions an Semantic Web ontologies is defined within the WSDL document.

To bridge between the structural WSDL definition and its intended meaning, we need to map two kinds of structural data to its semantics:

- Message Parts, and indirectly their respective data types
- Operations

In order to map structural data types to their semantics, a flexible data mapping mechanism is required. Such a mechanism can be found in the Meaning Definition Language (MDL) [24]:

3.1 Meaning Definition Language

MDL is an XML based language which allows to explicitly define how the structures of an XML data type conveys meanings. The meanings referenced may be contained in ontologies expressed by languages like DAML+OIL.

Basically, MDL defines the semantic information carried by nodes of an XML Schema definition. Nodes are referenced using XPATH [21] expressions. The meta model of the MDL is based on a triple of objects, properties and associations. This meta model leads to a widely applicable way of mapping structure to meaning (i.e. classes, properties and associations defined in DAML+OIL ontologies): To define how an XML Schema definition represents an instance of a *class*, the XPATH of at least one node type needs to be provided; to define how a schema represents a *property*, the XPATHs of at least two node types need to be provided; to define how a schema represents an *association*, the XPATHs of at least three nodes types need to be provided [24].

XML documents are hiding information not only inside their structure, but also inside the actual values of related elements and attributes. For instance, a certain XML element with XPATH "/school/pupil" may be assigned to a class "mature-student" or to a class "young-student" in dependence of a related attribute "age". To handle these and much more complicated cases, MDL provides several ways to incorporate conditions, using XPATH-compliant syntax. The listing below illustrates this technique, demonstrating how to define the meaning of elements of an XML Schema by mapping them to ontology elements like "mature-student" and "young-student":

```
<element context = '/school/pupil>
 <me:object class= 'mature-student' >
  <me:when objectToLeftValue = '@age'
           test = '>' rightValue = '30'/>
  <me:inclusion>
    <me:condition assoc="attends"
    obj1="mature-student" obj2="school" />
  </me:inclusion>
 </me:object>
 <me:object class= 'young-student' >
  <me:when objectToLeftValue = '@age'
           test = '<=' rightValue = '30'/>
  <me:inclusion>
   <me:condition assoc="attends"
    obj1="young-student" obj2="school" />
  </me:inclusion>
 </me:object>
</element>
```

A very useful property of MDL mappings is their bi-directionality: a valid MDL mapping between the elements of an XML Schema type and the elements of an ontology contains sufficient information to transform ontological data to its respective XML representation as well as to extract the meaning of a given XML document.

This feature is of great importance for the overall concept of the integration of Web Services with the Semantic Web. An agent capable of processing DAML+OIL may use MDL to transform data from a DAML+OIL repository into the XML grammar required by the service it wants to interact with. In case of a service response, the agent may use MDL to automatically generate DAML+OIL representations from the XML-based service-output.

3.2 Hook-Ups

The actual application of data mappings is performed by links defined inside the WSDL document. We will call these links "Hook-Ups". Hook-Ups must not violate the structural integrity of a WSDL file, because SOAP containers, development environments and common Web Service applications dealing with WSDL documents would be unable to process such documents. Therefore we use an annotation method as used by systems like SHOE [11] or Ontobroker's HTMLA [4] and use an extension mechanism provided by the hosting data format. In the case of WSDL, which is based on XML, we can use the extensions mechanism provided by XML namespaces [20].

Hook-Ups for Message Parts and Data Types. For combining message parts with their respective meaning we propose to use a namespace-qualified Hook-Up "semantic:schema-adjunct" which refers to an external MDL definition (called "adjunct" in MDL jargon). Alternatively, the MDL definition may be part of the XML schema definition inside the `<types>` element of the WSDL document. The standardized XML Schema element `<appinfo>` could be used for this purpose. However, this option has the disadvantage of ignoring that the meaning of an XML Schema type may change with the context (e.g. message type) it is associated with. Therefore this paper proposes to define the reference to a schema-adjunct always in correspondence to the `message` it is used in (as it will be shown in Fig.1 below).

Hook-Ups for Operations. Semantic Hook-Ups for message-parts as described above already provide an important part of the semantics of an operation. However, this is not a sufficient description yet. We also need to map the semantics of an operation as a whole to its respective meaningful pendant in a (DAML-S) ontology.

For this reason this paper proposes to provide a WSDL `<operation>` element with a Hook-Up by introducing an attribute `<operation>` in the newly created namespace.

3.3 An Example

We will now illustrate this technique by annotating the WSDL code fragment already introduced in Sect. 2.1. The semantic annotations adhere to the namespace prefix `semantic`.

In this example we define the meaning of the operation "getTemperature" by adding a Hook-Up `<semantic:operation>` referring to the DAML+OIL construct which defines the meaning of the operation (in this case, this meaning is provided by the DAML+OIL class http://another.org/agentTasks2344.daml#MeasuringTemperature).

Message parts contained in this operation are linked to their meaningful pendants, a DAML-S process-input "zip_code" and a DAML-S process-output "temperature".

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE uridef[
  <!ENTITY ont1 ‘‘http://some.org/classificationschemate.daml">
  <!ENTITY ont2 ‘‘http://some.other.org/nature.daml">
  <!ENTITY ont3 ‘‘http://another.org/agentTasks2344.daml">
  <!ENTITY adj ‘‘http://another.org/adjunct ‘‘>
]> <definitions name="DemoTemperatureService"
 targetNamespace="http://weatherstation.org/temperature"
 xmlns:tns="http://weatherservice.com/temperature"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
 xmlns:semantic="http://some.org/spec/"
 xmlns="http://schemas.xmlsoap.org/wsdl/"

  <message name="getTemperatureRequest">
    <part name="zipcode" type="xsd:string"
     semantic:type="&ont1;#ZipCode"
       semantic:schema-adjunct="&adj;_zipcode.mdl"/>
  </message>
  <message name="getTemperatureResponse">
    <part name="return"
    semantic:type="&ont2;#CelsiusTemperature"
    semantic:schema-adjunct="&adj;_temperature.mdl"/>
  </message>
  <portType name="DemoTemperatureServicePortType">
    <operation name="getTemperature"
      semantic:operation="&ont3;#MeasuringTemperature" >
        <input message="tns:getTemperatureRequest"/>
        <output message="tns:getTemperatureResponse"/>
      </operation>
  </portType>
  <--  rest of the WSDL document (binding, service - element) -->
</definitions>
```

A graphical representation of this semantic annotation is shown in Fig. 1:

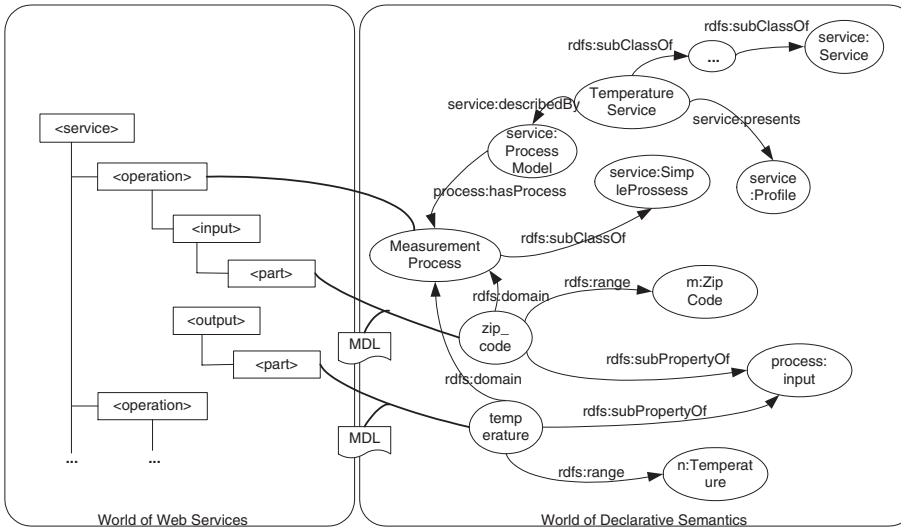

Fig. 1. Linking a WSDL document to its meaning

The annotated WSDL description may be referenced by a t-Model entry of a UDDI registry and it may be used by agents which are not capable of drawing logical inferences, thus providing necessary backward compatibility to existing solutions.

The DAML-S descriptions the WSDL annotations refer to may be stored at any other place on the web and may be found and processed by (Semantic) Web Crawlers and made accessible for search engines like ASCS (e.g. Agent Semantic Communication Service [10]). Such search engines or similar tools for semantic reasoning may be used to query DAML-S descriptions to find the Web-Services (and their WSDL descriptions) which fit the needs for a particular process or task.

If the search was successful and an URL of an annotated WSDL description of the matching service or services was found, then the agent can use the annotated WSDL document to gather the technical details needed for calling the services' operations and to transform DAML+OIL instances to XML elements used by the Web Service. The XML based result values returned by the Web Service may then be transformed back into DAML+OIL-instances, and may used for further processing by the agent, eventually triggering additional service calls.

4 Related Work

The idea of combining Semantic Web techniques with the concept of Web Services was presented in several publications, with different focus. A very influential work is DAML-S [13,14], which establishes a framework for reasoning over web service meta data. The relationships between WSDL and DAML-S were also briefly discussed in [13], where the authors state that WSDL differs from DAML-S, because DAML-S covers all aspects of service descriptions, whereas WSDL is primarily focussing on service grounding. The present paper is based on this finding and suggests a way how agents capable of DAML-S can interact with WSDL-based service groundings.

A proposal with some similarities to the ideas described in the present paper can be found in [16]. This paper demonstrates how WSDL descriptions can be expressed using RDF, thus providing a bridge between Semantic Web and Web Services. The concept presented in [16] has the advantage that all elements of (RDF encoded) WSDL descriptions are identified by an URI, thus allowing other RDF documents to assert statements about them. This way the semantic Hook-Ups could be defined completely outside the WSDL document. However, the disadvantage of such a method is that RDF encoded WSDL documents can not be considered as valid WSDL. This requires to maintain both an XML encoded and an RDF encoded version of the WSDL document and to keep both in sync.

5 Summary and Outlook

Semantic Web research and the Web Service initiative are different but complementary approaches to enable automatic agents accessing information on the web in a better way than on the classical human-centric World Wide Web.

Furthermore this paper argues that the convergence of these two efforts can be achieved with technology available today. As a first step to validate this assumption the paper presented a concept of semantic annotations of WSDL which link the structural elements of WSDL documents to semantics contained in DAML-S models. This concept contributes to the improvement of the current usage of Web Service technology.

However, the assumptions presented by this paper are not tested yet. The next step will be to build a prototype which actually demonstrates the concepts presented in this paper in order to test this approach. The challenge of this project lies in combining existing technologies such as WSDL-, UDDI- and SOAP-APIs, XML schema validators, MDL interpreters, RDF parsers and DAML+OIL engines.

Acknowledgements

The author would like to thank Izabella Mierzejewska, Markus Greunz and Rolf Grütter for their valuable inputs and support.

References

1. Berners-Lee, T.; Hendler, J.; Lassile, O.: The Semantic Web. *Scientific American*, Vol. 5/01, 2001. (http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html)
2. Borgida, A.; Brachman, R.J.; McGuinness D.L; Alperin Resnick, L.: CLASSIC: A Structural Data Model for Objects. in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* , pp 59-67, 1989.
3. Fikes, R.; McGuinness, D.: An Axiomatic Semantics for RDF, RDF Schema, and DAML+OIL. Stanford University, 2001. (http://www.daml.org/2001/03/axiomatic-semantics.html)
4. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer Verlag, 2001.
5. Festa, P.: Wanted: Web Service Standards. Online Magazine Article, 2001. (http://zdnet.com.com/2100-1104-835247.html)
6. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Technical Report, Stanford University, 1997.
7. van Harmelen, F.; Patel-Schneider, P.; Horrocks, I.: A Model Theoretic Semantics for DAML+OIL. 2001. (http://www.daml.org/2001/03/model-theoretic-semantics.html)
8. Horrocks, I.: The FaCT system. In H. de Swart (editor), Automate Reasoning with Analytic Tableaux and Related Methods: *International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pp 307-312, Springer-Verlag, 1998. (http:/www.cs.man.ac.uk/˜horrocks/Publications/download/1998/t98-paper.ps.gz)
9. International Business Machines Corporation: Web Services Inspection Language (WS-Inspection) 1.0. Specification, 2001. (http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html?dwzone=webservices)
10. Lee, J.; Pease, A.; Barbee; C.: Experimenting with ASCS Semantic Search. Project Report, 2002. (http://reliant.teknowledge.com/DAML/DAML.ps)
11. Luke, S.; Spector, L.; Rager, D.: Ontology-Based Knowledge Discovery on the World Wid Web. In: *Proceedings of the Workshop on Internet-based Information Systems at the AAAI-96*, Portland, Oregon, USA, August 4-8, 1996.
12. Martin, J.: Web Services: The Next Big Thing. XML Journal 2, 2001. (http://www.sys-con.com/xml/archivesbad.cfm)
13. Martin, D.; Burstein, M.; Ankolenkar, A.; Paolucci, M.; Payne, T.; Sycara, K; Lassila, O.; McIlraith, S.; Son, T.C; Zeng, H.; Hobbs, J.; Narayanan, S; McDermott, D.: DAML-S: Semantic Markup for Web Services. White paper, 2001. (http://www.daml.org/services/daml-s/2001/10/daml-s.pdf)
14. McIlraith, S.A.; Son, T.C.; Zeng, H.: Semantic web services. *IEEEIntelligent Systems*, 16(2), March/April, 2001.
15. Minsky, M.: A Framework for Representing Knowledge. MIT-AI Laboratory Memo 306, June, 1974. (http://web.media.mit.edu/ minsky/papers/Frames/frames.html)
16. Ogbuji, U.: Supercharging WSDL with RDF - Managing structured Web Service metadata. IBM developerWorks article, 2000.
17. Quillian, M.R.: Semantic memory. In: Minsky, M(editor), S*emantic Information Processing*. M.I.T. Press, 1968.
18. World Wide Web Consortium: Resource Description Framework (RDF) Model and Syntax Specification. Specification, 1999. (http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/)

19. World Wide Web Consortium: Resource Description Framework (RDF) Schema Specification 1.0. Specification, 1999. (http://www.w3.org/TR/2000/CR-rdf-schema-20000327)
20. World Wide Web Consortium: Namespaces in XML. Spezification, 1999. (http://www.w3.org/TR/1999/REC-xml-names-19990114)
21. World Wide Web Consortium: XML Path Language (XPath) Version 1.0. Spezification, 1999. (http://www.w3.org/TR/xpath)
22. World Wide Web Consortium: SOAP Version 1.2 Part 0: Primer. 2001 (http://www.w3.org/TR/soap12-part0/)
23. World Wide Web Consortium: Web Services Description Language (WSDL) 1.1. 2001 (http://www.w3.org/TR/wsdl)
24. Worden, R.: A Meaning Definition Language. White paper, 2001. (http://www.charteris.com/mdl/MDLWhitePaper.pdf)