# A Mediation Framework for Web Services in a Distributed Healthcare Information System

Michael Mrissa
*Université Bordeaux 1*
*Bordeaux, France*
*michaelmrissa@myway.com*

Djamal Benslimane and Chirine Ghedira
*Université Claude Bernard Lyon 1*
*Lyon, France*
{*djamal.benslimane,chirine.ghedira*}*@iuta.univ-lyon1.fr*

Zakaria Maamar
*Zayed University*
*Dubai, United Arab Emirates*
*zakaria.maamar@zu.ac.ae*

## Abstract

*Conceptualizing distributed Healthcare Information System is an important step toward the enhancement of clinical decision support system. In this paper, we propose a semantic mediation of Web Services interfaces for distributed healthcare system. Our proposal is an approach based on Web Services technology and their mediation in a Peer to Peer environment. This approach constitutes the foundation for the set-up of a mediation framework built around the JXTA P2P architecture applied to cardiology domain in collaboration with the National Institute of Health and Medical Research (INSER ERM 107). To achieve our goal, we used the OWL-S language as a means of describing semantics of Web Services interfaces, and the JXTA distributed architecture.*

## 1. Introduction

Retrieving medical information and conceptualizing distributed healthcare systems still are difficult tasks, especially in the cardiology domain. Health professionals are faced to a large amount of data, with the concern of a seamless and rapid access to patients' information without the constraints of their semantic and location diversity. A patient's computer record is composed of heterogeneous data with various granularity or aggregation levels that may be displayed according to different modalities. Data may directly come from clinical observations or result from several processing steps performed on biosignals and images captured by different means. Information overload, complexity and heterogeneity of cardiology data on the one hand, and the proprietary basis of most existing healthcare systems on the other hand, lead to the need of more flexible solutions for delivering information to the user. Medical information must be available and shareable among different physicians consulting a patient record, compliant with each physician's clinical practices, and specific to each patient or situation (Historical data for instance, can be presented through graph-based or text-based interfaces). To fulfill these requirements, we propose an approach based on Web Services technology and their mediation, since the technological development of this paradigm aims at improving inter-platform communication through well-defined standards, especially in a dynamic environment. This approach is a mediation framework applied to the cardiology domain, built on a distributed P2P architecture, more precisely JXTA-based. This framework is based on the OWL-S language as a means of describing semantics of Web Services interfaces, in order to solve heterogeneity problems.

The rest of the paper is organized as follows. In section 2 we briefly introduce the Web Services, Semantic Web and P2P domains and we review different approaches for a semantic enhancement of Web Services. Then, we present a running example and propose a classification of Web Services interface heterogeneities in a composition, before developing a solution based on OWL-S for a dynamic mediation of semantic Web Services. In section 3 we outline our framework for a semantic Web Services integration in a P2P environment, and we illustrate the example introduced previously. We conclude with a review of our contribution and some propositions for future improvements of our architecture and mediation proposal.

## 2. Mediation for Medical Web Services

While building our approach for a mediation architecture in the domain of medical information systems, three main areas were mentioned : Web Services, the semantic Web, and P2P. In this section we provide some background knowledge related to these domains. Then, we introduce a running example that illustrates our contribution and provides some support for our classification of interface heterogeneities. Lastly we present our work for a semantic mediation of Web Services interfaces.

## 2.1. Background Knowledge

**2.1.1. Web Services, Definition and Architecture.** Despite several definitions from major vendors, the World Wide Web Consortium (W3C) provides a quite straightforward definition to Web Services :

> *"A Web Service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered by XML artefacts and support direct interactions with other software applications using XML based messages via Internet based protocols."*

Web Services architecture is built on three major actors (see figure 1). Providers supply all the implementation of Web Services. They publish Web Services on the Internet, in registries that store Web Services descriptions (WSDL) and provide a centralized repositories for discovery. Clients find Web Services (generally by querying one of these registries) and query them by sending XML requests.

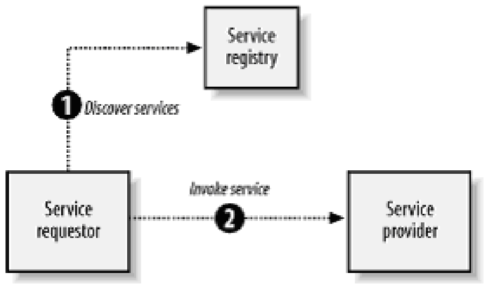
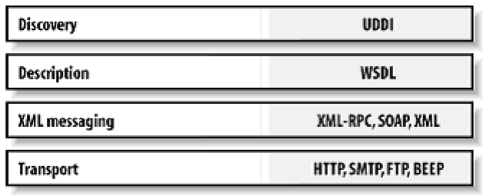


**Figure 1. Web services architecture [9]**

More precisely, Web Services architecture relies on the four main layers of its protocol stack (see figure 2) :



| | |
|---|---|
| Discovery | UDDI |
| Description | WSDL |
| XML messaging | XML-RPC, SOAP, XML |
| Transport | HTTP, SMTP, FTP, BEEP |

**Figure 2. Web services protocol stack [9]**

- The **transport layer** is the means for message transmission over the network. Its *"de facto"* standard protocol is HyperText Transfer Protocol (HTTP).
- The **messaging layer** provides a means for encoding messages in a common XML format. XML-RPC and SOAP are currently the standards for this layer.
- A Web Service is described in the **description layer**, and the standard protocol for this task is Web Service Description Language (WSDL).
- The most famous architecture for Web Services **discovery** is Universal Description, Discovery, and Integration (UDDI). It consists of a set of registries accessible from the Internet via an API, and it suffers from problems due to its client server architecture. UDDI providers use registry replication for maintaining a good availability under high load.

There are two important aspects motivating the use of Web Services. The first one is the use of XML language as a building block. It provides application-independence, as this language is not bound to any kind of proprietary solution for interpretation (XML parsers are widely available). The second one is the vision of a Web Service as a component that brings out the idea of a cross-platform technology, accessible in a standardized manner, and supporting composition or aggregation.

**2.1.2. Semantic Web, History and State of The Art.** The term "Semantic Web", firstly introduced by Berners-Lee et al. [5], is about the means to make data machine-understandable, since data semantics (meaning of data) are not explicitly represented on the Web. Additional structured information, called metadata, are so essential in order to explicitly describe them. Several initiatives have been undertaken for describing semantics of Web resources, firstly a generation of non-XML languages, then XML Schema and XML-based ontology languages [9]. Resource Description Framework (RDF) is the most important XML-based attempt for a semantic description of Web resources. This W3C project defines a grammar (RDF Syntax) and a way to structure information (RDF Schema) for describing domain specific knowledge with ontologies. Nowadays, new languages like OWL extend RDF principally because it still lacks from several features for describing information.

While Semantic Web is interested in describing static information available on the Internet in a machine-understandable way, Web Services are concerned with interworking between applications via the Web in order to make it dynamic. Web Services represent an interesting domain of application for the use of semantics. There are several ways of inserting semantics in Web Services. We present hereafter the important work in field.

A well-known approach consists in using a dedicated language. Recently, the DAML-S Coalition created OWL-S (previously named DAML-S), a language built on the OWL Candidate recommendation from the Web-Ontology Working Group. OWL-S stands for Web Ontology Language for Services, and defines a set of classes and properties for describing Web Services semantics, focusing on their properties and capabilities [1, 13, 14]. The goal of OWL-S is to allow a better automation of Web Services-related tasks like discovery, invocation, composition and interoperation. W3C defines OWL-S as:

> *"...a OWL-based Web Services ontology, which supplies Web Services providers with a core set of markup language constructs for describing the properties and capabilities of their Web Services in unambiguous, computer-interpretable form."*
>
> (OWL-S Release 1.0)

Another way to semantically enhance Web Services description is to annotate their description (WSDL) files. With the help of the extensibility support of WSDL, files can be extended with semantic information [25]. If using Java, Web Services source code can be annotated, as usually WSDL files are issued from Web Services code (Java2WSDL tool).

A third idea consists in registering semantic constructs in UDDI. Particularly, Verma et al. [26] present a P2P solution for a decentralized localization of Web Services description repositories. UDDI data structures can be used to store semantic details of Web Services [25].

Two other approaches can be mentioned for semantically describing Web Services. A first one consists in using RDF in conjunction with WSDL, or some other standards, to describe a Web Service. This approach can be effective because it allows describing WSDL in RDF syntax, which makes it compatible with existing RDF-based systems [22]. A second approach aims at automatically generating metadata [12], using machine learning and clustering techniques in order to attach semantic metadata to Web Services, clustering techniques serving to categorize Web Services into the classifications included in WSDL.

Other groups [6, 15] are interested in defining a formal framework for allowing a better relation inclusion and understanding between global properties of a Web Services and local properties of its components. Their motivation is to develop techniques of composed Web Services properties checking and synthesis, starting from the properties of their components.

**2.1.3. The JXTA P2P Framework.** The JXTA (for "JuXTApose") project [21], created and sponsored by Sun Microsystems, is an open source implementation of the P2P paradigm, based on original ideas like peergroups and relying on famous concepts like UNIX-style pipes for point-to-point communication. Moreover, the use of ubiquitous protocols like TCP/IP for local network and HTTP for inter network communication makes of JXTA a standardized specification for P2P applications. As quotes one of the creators of JXTA, independence is a primordial characteristic of this specification:

> *" Java implies platform independence, XML implies application independence, JXTA implies network independence."*
>
> (Bernard Traversat)

Since its birth, JXTA has grown and now comes with many features added to a standard set of core services, a complete binding for Java, and a complete C implementation. Moreover, the management of peergroups, the publishing of services within a same group, and the concept of group service (a service automatically provided by all the peers in the group) supply a rich framework for developing P2P applications.

The use of "rendezvous peers" and "relay peers" allows communication to pass through firewalls and, like a Virtual Private Network, allows devices from two separate local networks to get connected as if they were in the same. It is also possible to integrate existing services like CORBA, RMI or others in a JXTA envelope and to publish them through the JXTA network. This capacity of integration allows a good reuse of previous systems, adding transparency to the implementation of existing services, which are seen as normal JXTA services. JXTA security is based on a built-in model allowing encrypted and authenticated (SSL) communication between two peers. This specification for security potentially supports any other choices of implementation. Algorithms for encoding information and passwords can be chosen by the developer. Security is an important feature in the medical environment, where data confidentiality is a major constraint, so the reliability of JXTA's security model is an important advantage in the medical domain. Also, integration in open source IDE like Eclipse is quite straightforward thanks to a special plug-in.

- JXTA Peers

JXTA defines a peer as "any digital device connected to a network" [21]. Peers can belong to different JXTA peergroups (basically a group of peers), depending on users' choice. There are different kinds of peers. Rendezvous peers provide a landmark for other peers. By default, JXTA uses known rendezvous peers that act as bootstrapping peers, providing the lists of peers they are connected to, and thus initiating the discovery process. Relay peers are special peers that act like bridges in a physical network. A relay that interconnects two local networks will forward all the requests sent over one network to the other one, thus enabling both network to communicate seamlessly with each other. As stated in Oaks et al. [21], most of the time dedicated peers act both as a relay and as a rendezvous peer. It is the network administrator's choice to determine the ratio of peers acting as relays and/or rendezvous peers.

- Peergroups

A peergroup in JXTA is simply a group of peers, and giving a meaning to a peergroup is left to application developers. In practice, group classification limits the range of resource search and selection. There is a default "NetPeerGroup" that includes all the peers from all the peergroups (except if there is no link at all between groups) and this default peergroup virtually allows any peer to establish a dialog with any other peer. A group also has a membership policy, and once again it is the developer's role to define the level of security or requirements needed for a peer to joining a peergroup. As a consequence, the network is no more fragmented by firewalls and network topologies, but by peergroup membership policies. Members of a peergroup can access services provided in this particular peergroup, whether they provide these services or not.

- Pipes

The idea of pipes is adapted from UNIX pipes, and used to provide end-to-end unidirectional communication between peer processes. A pipe is basically a communication channel between two JXTA endpoints. It supports secured communication if necessary, using encryption algorithms such as SHA-1 or RSA [21]. Bidirectional pipes are also implemented in the Java bindings of the JXTA specification.

- Advertisements

JXTA also introduces the concept of advertisements, that describe available network resources like pipe endpoints, services (for instance WS), peers and other peergroups. There are six basic classes of advertisements, and it is possible to create specific subclasses describing more particular objects. Advertisements are published over the network to participants of a peergroup in a decentralized manner [21]. The architecture of JXTA is based on the distribution of advertisements. Advertisements are stored in each peer local cache and accessed when needed. They are the standard XML-based publication format defined in JXTA, though different from WSDL. There is a level of overlapping between the Web Services protocol stack (using UDDI) and JXTA's architecture at the publication and discovery level. Based on the latter background, we provide a framework for Web Services integration into the JXTA network applied to the medical domain through a running example presented hereafter.

### 2.2. Running Example

In this work, we consider the case of a physician that needs to check historical data about a patient. For instance, the physician needs to know if the patient has any allergy, or heart problems, in which case some categories of medication are not recommend, or particularly dangerous. Results and dates of last analysis may also provide important information. Our goal is to automate the process of gathering that patient's information and to accurately display it to the physician. For this purpose, following steps are required:

- Firstly, find available medical centers providing Web Services that give patients' historical data (see figure 3).
- Secondly, ask each medical center for descriptions of its Web Services (how to query them) and for a semantic description of the vocabulary used for communicating (what is the exact meaning of these descriptions).
- Thirdly, automatically call Web Services, interpret and then merge (or put together in some way) their response messages for user display.
- Finally, when receiving response messages from different Web Services, our client program must adapt its behaviour and accurately display the information.

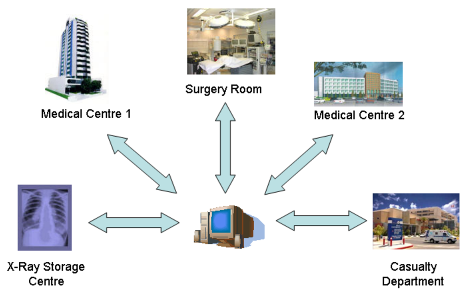


**Figure 3. Accessing Medical Web Services**

In our example we have two kind of Web Services. The first kind of Web Services, when receiving a client's insurance number, returns data about a patient's historical background, and the second kind returns patient's chest X-ray images. So, these Web Services take a non negative non null integer as an input parameter, and they return the date the answer was generated, the type of answer (X-ray images or some historical data), and data themselves. When a Web Service returns an image, additional information specifies units of the image.

As mentioned before, these Web Services are frequently heterogeneous. We propose hence a classification of their interface heterogeneities. However, this classification is mainly adapted from another contribution to be submitted. In the context of our medical Web Services, data conflicts can be described as follows:

- The semantic level

We detail three kinds of semantic conflicts: label, unit and value conflicts. Label conflicts reside in the vocabulary used for data description. For instance, a label conflict occurs when a French Web Service uses the word ”DONNEES” in order to name some data parameter, whereas an English Web Service uses the word ”DATA”. In this case, two different words have the same meaning, but this conflict also occurs when the same word means something different in another language.

Unit conflicts refer to different units used in a same description. For example a ”UNIT” parameter describing the unit used in images can be expressed in inches or centimeters. This problem becomes more important with discrete data like temperature where units and scales can change (Celsius and Fahrenheit degrees for example).

A value conflict occurs when different values have the same meaning. This is different from the label conflict described above because it concerns the content of a parameter and not its name. For instance values addressing a patient’s gender can be ”masc” or ”fem” in a Web Service, and ”m” or ”f” in a second.

- The structural level

Structural conflicts occur when parameters are structured in different ways. In our example, a first service returning some historical data could have three output parameters, whereas another one only has two. There is a mismatch in the structure of the information, and in the domain of Web Services this is characterized by the number of input and output parameters.

Merging conflict is a particular case of structural conflict. A merging conflict exists in the case of several parameters describing the same entity on a service, and of a single parameter describing an entity on another Web Service. For instance a Web Service may describe data type (historical data or image) in an independent parameter called ”TYPE”. Another Web Service can describe data type inside another parameter, separating data type from the other parameter with a special character like a coma. Thus parameters need to be merged in order to match those of the second Web Service.

- The syntactic level

Whereas in traditional data exchange, syntax conflicts usually happen between proprietary data storage methods, in the context of Web Services, the standard use of XML as an independent encoding language hides all conflicts that could happen. This is one of the advantages of using standard Web Services, relying on HTTP, SOAP and XML. However, it is noticeable that Web Services can use non-XML syntax languages, but then they are limited to a local organization.

### 2.3. Medical Web Services Interface Mediation

A common agreement like an ontology provides a support for solving input/output heterogeneities listed above. We consider the OWL-S service-oriented ontology language as a reliable answer to such problem, as ontologies represent widely and freely distributed references, inspired from the Healthcare Information System Architecture (HISA) work item of CEN/TC 251. Using an ontology allows describing a domain knowledge, and precisely specifying data meaning and structure. Medical centers willing to join a peergroup providing medical Web Services must first agree on a common ontology, and give a OWL-S description for each Web Service provided. Then, when invoking a Web Service, mappings from its WSDL file to the common OWL-S ontology can be generated, and exchanged data are interpretable by all the peers of the community (see figure 4). Thus, depending on semantic descriptions of Web Service interfaces, answers can be differently interpreted, processed and accurately displayed to the user.

In our case, if a Web Service provides heart characteristics images, they are displayed correctly in the user interface. If another Web Service provides some historical data about the patient, they are merged with other historical data received from other medical centers. In the end, user interface will provide a global view of every kind of historical data gathered from medical centers.

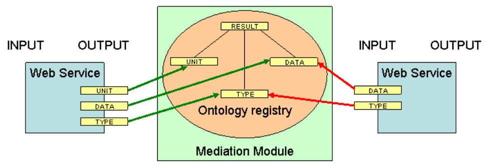


**Figure 4. Interface mappings to ontology**

## 3. Medical Web Services Integration in a P2P Framework

The approach presented above, which is built on the use of ontologies, forms a basis for medical Web Services composition and aggregation. In this section we integrate this infrastructure into the JXTA framework, and illustrate our approach with the example introduced before.

### 3.1. Discovery

In the case of our running example, a physician's application accesses distant Web Services. Those latter are typically available through UDDI. However, in our architecture, Web Services are discovered through the JXTA P2P network, so that the end-user application does not use UDDI for discovery. As network resources (and Web Services advertisements) are spread over the network in a decentralized manner, all the disadvantages of centralized UDDI architecture are solved.

Concerning the semantics of Web Services, we assume that each JXTA peergroup is associated with a particular ontology. For instance, our medical centers are all part of a peergroup with an ontology of Web Services that describes historical data and supports encoded images as an output parameter. As a consequence, limits of Web Services search and selection are no more physical limits of the network, but rather they are constraints stated in our peergroup definition, like membership fee or identification.

This independence from the physical layer is an important advantage in medical information systems, where locations of hospitals and emergency centers (and Web Service providers in general) form an unstructured network. JXTA provides end users with a list of available peergroups, so physicians can query for medical peergroups and register the peergroup they want. In our example, the physician registers in the peergroup providing patients' historical and heart-related data from medical centers. Different selection criteria may apply at these stages (peergroup selection and registration), but this is out of our scope, and described in Oaks et al. [21].

### 3.2. Publication

For Web Service publication we define a special "gateway peer", that is a special featured peer able to encapsulate both WSDL and OWL-S descriptions in a JXTA advertisement, and to publish the latter into the peergroup (see figure 5). A reliable solution consists in installing a gateway peer for each medical center, and registering it into the JXTA peergroup for publishing Web Services advertisements at regular intervals (if a timeout has been set for advertisement availability). Gateway peers form a link between available Web Services (that are typically discovered by accessing UDDI registries) and JXTA peergroups in which discovery is decentralized. Since gateway peers can be installed by virtually anybody, any kind of existing Web Services may be made available in our peergroup, since it gets the right to register from the peergroup's administrators. A proof-of-concept implementation of a gateway peer has been developed and tested for publication through a JXTA network. Ongoing work is aiming at an easily deployable component that could be installed on any JXTA peer, making it a gateway peer.

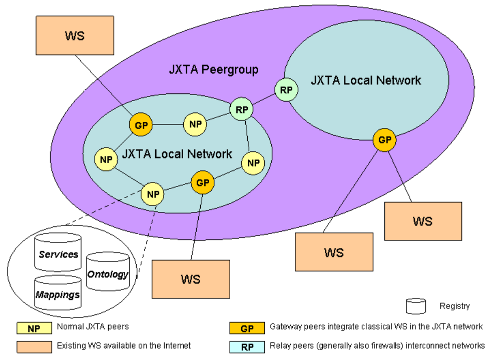

WS
JXTA Peergroup
JXTA Local Network
RP
RP
GP
NP
NP
JXTA Local Network
NP
GP
NP
GP
WS
WS
WS
Services
Ontology
Mappings
Registry
NP Normal JXTA peers
Existing WS available on the Internet
GP Gateway peers integrate classical WS in the JXTA network
RP Relay peers (generally also firewalls) interconnect networks

**Figure 5. Web service integration**

### 3.3. Invocation

There are several strategies for Web Service invocation from the JXTA network. Administrators can decide that some specialized peers only are able to invoke Web Services, or they can choose that all the peers can invoke Web Services independently. In our architecture, we chose to allow all the peers in the community to seamlessly invoke Web Services. For that purpose, we use a generic Web Service client engine that allows Web Service invocation from a WSDL description. This invocation engine extracts input and output parameters needed for Web Service invocation, and asks user if additional information is necessary. Then it builds and sends requests for invoking Web Services. It is a "group service" in the JXTA community, in other words it is a service that is shared between all the peers, and each new peer joining the community downloads the piece of code (JXTA module) needed for executing this service. Thus, every peer is able to directly invoke Web Services, avoiding bottleneck problems that appear when using specialized peers, and taking full advantage of the P2P community. Moreover, our peers are able to accurately interpret and process results received, thanks to the OWL-S description, that gives semantic information about the Web Service.

### 3.4. Application

In this section we detail the complete process of our architecture (Figure 6) using our example presented in section 2.2. A physician invokes several medical centers for gathering a patient's data. Firstly, the JXTA peer searches in its registry for resource descriptions corresponding with its needs, in our case Web Services providing historical medical information. Then a list of available Web Services is built up, and OWL-S descriptions are compared with the common ontology, which is present in each peer. After checking the validity of each Web Service description, the peer knows if it can interpret answers delivered by Web Services. Interface mappings of each Web Service are stored in a local registry, and unmatching Web Services are ignored. At this time, the peer uses its invocation engine for dynamically calling each Web Service endpoint, asking the end user (our physician) for additional information if required. When receiving different answers from Web Services, the peer classifies historical data and images, and processes them for final display.

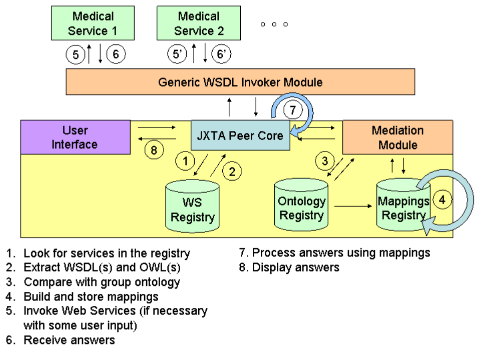

Medical Service 1
Medical Service 2
5
6
5'
6'
Generic WSDL Invoker Module
7
User Interface
JXTA Peer Core
Mediation Module
8
1
2
3
WS Registry
Ontology Registry
Mappings Registry
4
1. Look for services in the registry
2. Extract WSDL(s) and OWL(s)
3. Compare with group ontology
4. Build and store mappings
5. Invoke Web Services (if necessary with some user input)
6. Receive answers
7. Process answers using mappings
8. Display answers

**Figure 6. Detailed web service invocation**

## 4. Conclusion and Future Work

In this paper we outlined a complete infrastructure, combining OWL-S based mediation for semantically described Web Services, and their integration into a JXTA P2P platform, applied to the medical domain. We proposed a classification of interface heterogeneities, and introduced the idea of a specialized peer (called gateway peer) and of a generic client engine for respectively Web Services publication and invocation, as building blocks for medical Web Services P2P integration model. The descriptive capability and extensibility of OWL-S provides a good support for precise descriptions of Web Services characteristics and properties. Furthermore, the flexibility of P2P model frees the network from its physical architecture and provides a good dynamic support for medical Healthcare information system.

The use of a medical ontology as a common agreement for describing Web Services data semantics is reliable, and can be extended to other Web Service semantics, like QoS, functional and execution semantics. Also, using several ontologies is a perspective to a larger subject encompassing ontology merge and translation.

## References

[1] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web, 2002.

[2] K. Anyanwu, A. Sheth, J. Cardoso, J. Miller, and K. Kochut. Healthcare Enterprise Process Development and Integration. Technical report, LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, 1999.

[3] B. Benatallah, M.-S. Hacd, C. Rey, and F. Toumani. Semantic Reasoning for Web Services Discovery. In *WWW2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary*, May 20, 2003.

[4] B. Benatallah, B. Medjahed, A. Bouguettaya, A. Elmagarmid, and J. Beard. WebBIS: a system for building and managing Web-based virtual enterprises. In *Proceedings of the 1st workshop on Technologies for E-Services, in cooperation with VLDB2000, Cairo, Egypt*, September 2000.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *The Scientific American*, May 2001.

[6] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: a new approach to design and analysis of e-service composition. In *Proceedings of the twelfth international conference on World Wide Web*, pages 403–410. ACM Press, 2003.

[7] CEN/TC 251/ENV 12967-1 Medical Informatics. *Healthcare Information System Architecture Part 1 (HISA)*, page 130. Healthcare Middleware Layer: European Commitee for Standardization, 1998.

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. *W3C*, March 2001.

[9] A. Dogac, G. Laleci, S. Kirbas, Y. Kabak, S. Sinir, and A. Yildiz. Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain. Technical report, METU-SRDC, November 2003.

[10] D. Fensel and C. Bussler. The Web Service Modeling Framework Web ServicesMF. Technical report, Vrije Universiteit Amsterdam, 2002.

[11] M. Gudgin, M. Hadley, J.-J. Moreau, and H. F. Nielsen. SOAP Candidate Recommendation Version 1.2. *W3C*, 2002.

[12] A. Heß, , and N. Kushmerick. Learning to attach semantic metadata to Web Services. In *International Conference on Web Services (ICWeb Services'03)*, 2003.

[13] I. Horrocks. DAML+OIL: A reason-able web ontology language. In *Extending Database Technology*, pages 2–13, 2002.

[14] I. Horrocks, P.F.Patel-Schneider, and F. . van Harmelen. Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 792–797, 2002.

[15] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: a look behind the curtain. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–14. ACM Press, 2003.

[16] E. Leclercq, D. Benslimane, and K. Yétongnon. ISIS: A Semantic Mediation Model and an Agent Based Architecture for GIS Interoperability. In *Proceedings of the International Database Engineering and Applications Symposium, Montreal, Canada*, 1999.

[17] Z. Maamar and N. C. Narendra. Ontology-based Context Reconciliation in a Web Services Environment: From OWL-S to OWL-C. In *Proceedings of The 2nd International Workshop on Web Services and Agent-based Engineering (Web ServicesABE'2004) held in conjunction with The 3rd International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'2004), New York, USA*, July 2004.

[18] A. Mädche and S. Staab. Services on the Move - Towards P2P-Enabled Semantic Web Services. In *Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January 2003*. Springer, 2003.

[19] D. Martin, M. Burstein, O. Lassila, M. Paolucci, T. Payne, and S. McIlraith. Describing Web Services using OWL-S and WSDL. Technical report, DAML-S Coalition, October 2003.

[20] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *Proceedings of the eleventh international conference on World Wide Web*, pages 77–88. ACM Press, 2002.

[21] S. Oaks, B. Traversat, and L. Gong. *JXTA in a Nutshell*. In a Nutshell. O'Reilly, September 2002.

[22] U. Ogbuji. Supercharging WSDL with RDF Managing structured Web service metadata. *IBM Developer Works*, 2000. http://www-106.ibm.com/developerworks/library/ws-rdf/?dwzone=ws.

[23] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Importing the Semantic Web in UDDI. In *Proceedings of E-Services and the Semantic Web Workshop*, 2002.

[24] K. Sivahanmugam, A. Sheth, J. Miller, K. Verma, R. Aggarwal, and P. Rajasekaran. *Databases and Information Systems*, volume Praktische Informatik I, chapter Metadata and Semantics for Web Services and Processes, pages 245–271. Festschrift zum 60. Geburtstag von Gunter Schlageter, W. Benn, P. Dadam, S. Kirn and R. Unland Editors, Hagen, Germany, 2003.

[25] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding Semantics to Web Services Standards. In *International Conference on Web Services (ICWeb Services'03)*, 2003.

[26] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S Web ServicesDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *To be published in the Journal of Information Technology and Management*, 2004.

[27] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S Web Services Composition Using SHOP2. In D. Fensel, K. Sycara, , and J. Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, pages 195–210, 2003. number 2870 in Lecture Notes in Computer Science.