Using Context to Enable Semantic Mediation in Web Service Communities

Michael Mrissa and Philippe Thiran
PReCISE Research Center
University of Namur
Namur, Belgium
mmrissa@fundp.ac.be
pthiran@fundp.ac.be

Chirine Ghedira and Djamal Benslimane
LIRIS laboratory
Lyon 1 University
Lyon, France
cghedira@liris.cnrs.fr
dbenslim@liris.cnrs.fr

Zakaria Maamar
College of Information Technology
Zayed University
Dubai, U. A. E.
zakaria.maamar@zu.ac.ae

ABSTRACT

The use of communities provides a scalable solution for gathering and managing functionally-equivalent Web services. In order to ensure single access to the community, a community uses a common interface that acts as a proxy and selects other Web services in the community. However, Web services adopt different semantics for representing the data they receive and send. These semantics must be adapted to conforming to the community semantics. In this paper, we present a solution to this problem. Our solution is based on the use of context in order to explicitly describe semantic discrepancies within a community. We rely on a semantic annotation of WSDL descriptions to describe the semantics attached to Web services, and we provide mediation mechanisms at the community level to handle semantic heterogeneities between Web services and the community. We validate our solution through implementation and experimentation over a test community and show the limitations of our approach.

Categories and Subject Descriptors

D.2.12 [**Software Engineering**]: Interoperability

General Terms

Design, Languages, Standardization

Keywords

Web services, semantics, context, mediation, community

1. INTRODUCTION

Web services are remotely accessible software components providing a specific functionality. They rely on XML-based protocols and languages for supporting message exchange (SOAP [6]), service description (WSDL [8]) and discovery (UDDI [21]). The main advantage of Web services is their capacity of being composed. A composition consists in combining several Web services into the same business process, in order to address complex user's needs that a single Web services could not satisfy.

The common practice is to select Web services depending on their characteristics, such as availability, price, efficiency, etc. Gathering Web services into communities facilitates the selection process by enabling a centralized access to several functionally-equivalent Web services via a unique endpoint. Several research works propose to use communities for easing the management and access to Web services [2, 3, 11, 12, 17, 20].

However, several heterogeneities between Web services and the community hamper straightforward integration. At the semantic level, discrepancies between the data representations of Web services and the data representation of the community must be resolved in order to allow transparent invocation of services via the community.

In this paper, we address this problem by proposing a semantic mediation mechanism for Web service communities. This mechanism relies on a context-based model for data representation and WSDL annotation proposed in previous works [14]. It has for purpose to solve semantic discrepancies and enable seamless invocation of Web services in the community.

This paper is organized as follows: section 2 introduces the notion of community for gathering Web services and the context-based model and WSDL annotation we rely on for describing data semantics. Section 3 discusses how the deployment of a semantic mediation module helps solving semantic inconsistencies of data between Web services and communities. Further details on the functioning of the mediation module and how it takes advantage of our context-based model are given in this section. Section 4 presents related work on semantic mediation in communities, and section 5 discusses the limitations of our work and presents some insights for future work.

2. GENERAL ARCHITECTURE

2.1 Communities of Web services

A community is typically a group of people living together or united by shared interests, cultural, religious or political reasons. In the domain of Web services, communities help gather Web services that provide a common functionality, thus simplifying the access to several Web services through a unique communication endpoint, that is the access point to the community.

CSSSIA 2008, April 22, Beijing, China.

In previous work, we proposed an approach that supports the concepts, architecture, operation and deployment of Web service communities [17]. The notion of community serves as a intermediary layer to bind to Web services. A community gathers several *slave* Web services that provide the same functionality. The community is accessed via a unique *master* Web service. Users bind to the master Web service that transparently calls a slave in the community. Our previous work details the management tasks a master Web service is responsible for. Such tasks include registering new Web services into the community, tracking bad Web services, removing ineffective Web services from the community, and so on.

A *master* Web service represents the community and handles relationships with slave Web services using a specific protocol. In our previous work we rely on a slightly extended version of the Contract Net protocol (CNProtocol) [19], as illustrated in Fig. 1:

1. The master Web services sends a call for bid to the slave Web services of the community.
2. Slave Web services assess their current status and availability to fulfil the resquest of the master Web service, and interested Web service reply to the call.
3. The master Web service examines the received proposals and chooses the best Web services according to its preferences (QoS, availability, cost, fairness...). It notifies the winner slave Web service.
4. Slave Web service that answered the call for bid but were not selected are notified too.

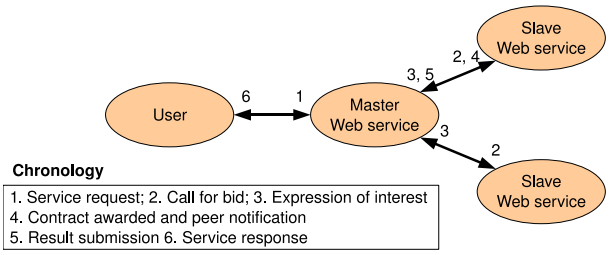

Figure 1: Contract-Net protocol interactions

In this paper, we provide a context-based semantic mediation architecture for such communities. Indeed, the applicability of our mediation proposition goes beyond the scope of this paper, however we specifically focus here on its deployment with communities as presented in [17]. Semantic mediation is performed between the community master and slave Web services. Our semantic mediation architecture relies on a context-based data representation and context annotation to WSDL.

2.2 Context representation

The notion of context for describing the semantics of data that Web services exchange has been introduced in previous work [14]. Context involves pushing data up to the level of *semantic objects*. A semantic object is a data object with an explicit semantics described through its context. A semantic object S is a tuple $S = (c, t, v, C)$ that holds a concept c described in a domain ontology, a type t that is the XML Schema type of the data, a value v that is the data itself and a context C that is a set of semantic objects called modifiers.

Modifiers are semantic objects that participate in a context. Therefore, context is seen as a tree of modifiers where the leaves are modifiers with a *null* context. Modifiers can be of type static or dynamic. The main characteristic of dynamic modifiers is their capacity to have their value inferred from the values of static modifiers. For instance, as shown in Fig. 2, if a *price* semantic object is attached to the country *France* and to the date 15/05/2005, then it can be inferred that the *currency* modifier, which describes the currency of this price, has *Euro* as a value, making *currency* a dynamic modifier. Static modifiers have to be made explicit in order to describe the meaning of the semantic object.

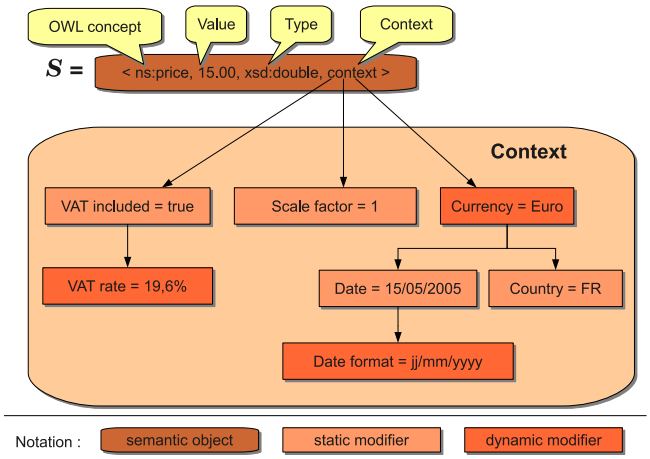

Figure 2: Description of the *price* semantic object

2.3 Domain and context ontologies

A major characteristic that comes with our context-based model is the attachment of context ontologies to concepts of domain ontologies. The use of context comes from a simple assumption: we noticed the difficulty for Web services to adhere to several communities. In effect, each community follows a specific ontology that is not always compliant with other communities' ontologies, and Web services already follow, either implicitly or explicitly, their providers' local semantics.

Therefore, the adhesion of a Web service to a new community requires either a hard-coded change in the service implementation or the design of a wrapper that acts on behalf of the original Web service, in order to comply with the community's ontology. Such tedious requirement applies to each new community a Web service wishes to adhere to. Our context-based model has for objective to ease the task of Web service providers when adhering to new communities, by reducing the domain ontology to the minimum, and providing additional context ontologies to handle the different local semantics of service providers.

We deem appropriate to limit the application of domain ontologies to the concepts and relationships that can be devised with a top-down approach. Experts in the knowledge domain should specify domain ontologies and limit to the minimum the further description of domain concepts. Thus, domain ontologies should be the most similar and the burden when adhering to a community should be limited. An ideal

situation is the design of a unique domain ontology that all the communities could adopt, although such a situation is not likely to reach in an open world like the Internet.

However, the semantic heterogeneities between providers and communities still need to be handled. This is where context ontologies come into play. A context ontology is attached to each concept of the domain ontology. Context ontologies have for purpose to describe all the different properties of domain concepts that are not described in the domain ontology. In order to populate the context ontology, a bottom-up approach is adopted. The contexts of domain concepts are updated by service providers when they adhere to the community, as well as by the community maintainer. Service providers should make explicit the semantics they adopt via the context ontology. As well, service providers and the community maintainer should describe the links between the different context representations that populate context ontologies.

Separation of concerns has proven to be an efficient way to solve complex problems in the field of software engineering, and the separation we propose between domain and context ontologies follows such well-established practice in order to facilitate semantic interoperability between Web services.

2.4 Context annotation to WSDL

Propelling input and output data of Web services (described in the WSDL documents) to the level of semantic objects requires additional information. In WSDL documents, `<message>` elements describe data exchanged for an operation. Each message consists of one or more `<part>` elements. We also refer to `<part>` elements as "parameters" in the rest of this paper. Each parameter has a `<name>` and a `<type>` attribute, and allows additional attributes.

In [15], we developed an annotation that enriches input and output parameters contained in WSDL documents with a concept from a domain ontology and static modifiers from the context ontology attached to the concept. Our annotation takes advantage of the extension proposed in the WSDL specification [8], so that annotated WSDL documents operate seamlessly with both classical and annotation-aware clients. `<part>` elements are annotated with a `context` attribute that describes the names and values of static modifiers using a list of qualified names. The first qualified name of the list specifies the domain ontology concept of the semantic object (c). Additional elements refer to context ontology concept instances to describe the names and values of static modifiers. These concept instances are OWL individuals, thus they allow specifying the name and value of context attributes at the same time.

With the help of such annotation, a value v and its data type t described in the WSDL document are enriched with the concept c and the necessary modifiers to define the context C, thus forming a semantic object (c, v, t, C). To keep the paper self-contained, we overview a simplified structure of the annotated WSDL document in Listing 1.

A context C is populated at runtime, using logical rules. Logical rules infer the values of dynamic modifiers from the information provided by static modifiers of the WSDL annotation. Using rules offers several advantages: rules are easily modifiable, making this solution more adaptable to changes in the semantics. Often-changing values of dynamic modifiers could not be statically stored, so using rules simplifies the annotation to WSDL. Furthermore, rules separate application logic from the rest of the system, so updating rules does not require rewriting application code. In this paper, we rely on our annotation and rule-based mechanisms in order to provide the semantic information required to perform semantic mediation within a community.

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<wsdl:definitions ...>
...
<wsdl:message name=''checkPriceReq''>
  <wsdl:part name=''price'' type=''xsd:double''
  ctxt:context=''dom:Price ctx1:ScaleFactorOne
  ctx1:dateValue ctx1:VATIncluded ctx1:France''/>
</wsdl:message>
...
</wsdl:definitions>
```

Listing 1: Annotated WSDL Snippet

3. SEMANTIC MEDIATION FOR WEB SERVICE COMMUNITIES

Our mediation architecture for Web service communities is built on a master Web service that contains a mediation module. This mediation module enables the master Web service to handle incoming requests from outside the community.

Thanks to the mediation module, the master Web service can act as a mediator. Upon reception of a user's request, it uses the mediation module to convert the message into the slave Web service's semantics. Upon reception of an answer from a slave Web service, the master Web service uses the mediation module again to convert the message into the semantics of the community before sending it back to the user. Our master Web service is also responsible for other tasks, such as selecting a slave Web service upon reception of a request or managing the community, as described before.

3.1 Accessing the community

Typically, a user that wishes to interact with a community for fulfilling his/her goals discovers and selects the WSDL file of the community via a UDDI registry. Then, the client program uses this WSDL file to build a query and send it to the community endpoint, i.e. the master Web service. The latter has for purpose to handle the interactions with the client in order to hide the community complexity. Then, a community-based architecture is completely transparent from the user's point of view.

However, the master Web service does not implement the community functionality itself. Its role is to select one of the slave Web services that belong to the community according to the user's preferences, and to forward the client's request to this slave Web service. Then, it must forward back the answer from the slave Web service to the client. We detail the functioning of the master Web service hereafter.

3.2 Details of context-based mediation

When it comes to the mediation concern, our master Web service behaves as a proxy for the community. It handles and transfers incoming requests from users and outgoing answers from slave Web services. On reception of an incoming request, it uses the mediation module to perform the following actions in order to solve semantic heterogeneities using our context-based approach. The mediation module contains several components:

- an interface to communicate with the master Web service,
- a core component called *mediator* that orchestrates the different steps of the mediation process described below,
- a component called $WSDLcontextreader$ to read WSDL annotations,
- repositories for domain and context ontologies for the community,
- a rule engine and a knowledge repository to store rules for data conversion and context building.

All these components participate to the semantic mediation process in the following way, as illustrated in Fig. 3:

1. Reading WSDL annotation
   (a) It selects a slave service and fetches its WSDL description.
   (b) Then, it parses the input and output parameters of the selected WSDL operation of this slave Web service.
   (c) For each parameter, it extracts the domain concept and static modifiers contained in the WSDL annotation. We assume our semantic mediation module is configured for a specific community and already has in-memory representations of the input/output parameters of the community as semantic objects, so there is no need to fetch the WSDL file proposed by the community.
2. Identifying domain vocabulary
   (a) It communicates with the domain ontology to identify the domain concepts contained in the annotation.
   (b) If the terms are not found, an exception is raised, otherwise the next step is confirmed.
3. Identifying context
   (a) It communicates with the context ontology to identify the modifiers contained in the annotation.
   (b) If the terms are not found, an exception is raised, otherwise the next step is confirmed.
4. Building semantic objects
   (a) Using the information contained in the WSDL annotation, our mediation module converts the annotated WSDL parameters into semantic objects.
   (b) It interacts with a rule engine in order to infer the values of dynamic modifiers available in the context. Sometimes not all dynamic modifiers can be populated. In such case, data semantic conversion is still possible over a limited context.
5. Performing data conversion
   (a) At this stage, our mediation module possesses two in-memory semantic objects that have different contexts, and it needs to convert data from the context of the community to the context of the slave Web service. To do so, it interacts with a knowledge repository that stores conversion rules and enables data conversion from the community semantics to the slave Web service's semantics.
   (b) If the conversion is not possible, an exception is raised, otherwise data is forwarded to the slave Web service.

On reception of outcoming answer, the task of our mediation module is reversed:

1. It reads the WSDL description of the slave service it interacts with, identifies the domain and context information contained in the annotation, and builds semantic objects.
2. It interacts with the rule engine that converts data from the slave Web service's semantics back to the community semantics.
3. It raises an exception if the conversion does not succeeds or forwards the converted data back to the client of the community.

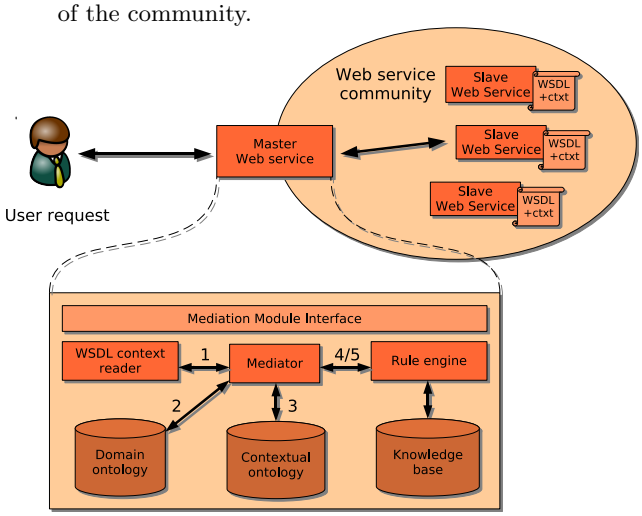

Figure 3: Overview of the mediation process

3.3 Implementation & case study

A proof-of-concept prototype has been developed under the Java™JDK 1.5 platform and Netbeans 6™development environment. Our master Web service currently handles the semantic mediation and forwarding of user calls and answers from slave Web services. The mediation module has been implemented using the Jena 2 API for querying domain and context ontologies. Thanks to the Jena API, in-memory representations of ontologies are built, and queries on these representations allow identifying the terms used in the annotation. Concerning the mediation process, a Drools rule engine is used to store conversion rules and rules that help infer the values of dynamic modifiers of context.

We illustrated the developpement of semantic mediation with the *price* example presented in section 2.2. We consider the following case study: a European community provides

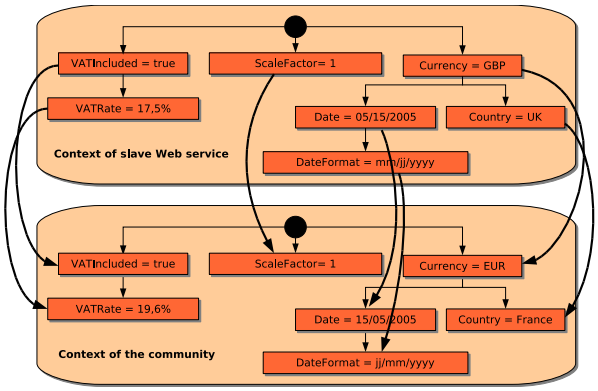

Figure 4: Converting between contexts

a *FlightBooking* functionality. As input data, users send information about the trip: departure city and country, arrival city and country, outward journey date and return date. In return, they receive a trip proposal that includes the price of the trip corresponding to the demand.

The community has adopted the following context for data representation: dates are in the $dd/mm/yyyy$ format where dd is the day, mm is the month and $yyyy$ is the year of the date, and city and country are sent as a $city;country$ string, and prices are described in Euro, VAT included, and with a scale factor of 1.

However, an English Web service joins the community. This Web service was developed before joining the community, therefore it adopted the local semantic of use in its country: dates are under the $mm/dd/yyyy$ format and prices are in Pounds, VAT included and with a scale factor of 1.

On reception of the user request, the master Web service selects our English slave Web service. It downloads its WSDL file that contains the annotation. When comparing the input/output parameters of the community and of the slave Web service, the mediation module identifies the $Date$ and $Price$ concepts in the domain ontology, and the corresponding names and values of static modifiers as OWL individuals: $UKdateformat$ from the context ontology attached to the $Date$ concept, and UK, $dateValue$ and $VATIncluded$ from the context ontology attached to the $Price$ concept. These ontologies are referenced in the WSDL namespace.

Then, it sends the static modifiers to the rule engine that returns additional information: $dateformat$ modifier of the $Date$ concept gets the $mm/dd/yyyy$ value, and for the $Price$ concept, the $currency$ modifier gets the $Pound$ value. Finally, the mediation module is able to convert data from one semantic context to another using again the rule engine and conversion rules stored in the knowledge repository, as shown in Fig. 4.

4. RELATED WORK

To the best of our knowledge, there are no existing works on semantic mediation within the context of Web service communities. However, our work is inspired by several papers on semantic mediation for Web services and communities of Web services, that we detail hereafter.

4.1 Semantics and mediation for Web services

Semantic description and mediation for Web services is a very active research topic, as prove the many works on the subject [1, 4, 5, 7, 9, 10, 13, 16]. In the following, we describe the most important works that inspired us for this paper.

In [16], Nagarajan et al. classify the different semantic interoperability concerns that apply to Web services. They distinguish several aspects that are particularly useful for the purpose of semantic mediation.

OWL-S [10] is a language for semantic description of Web services, relying on the OWL language [18], OWL-S enables explicit semantic description of Web services input an output parameters via references to concepts described in OWL domain ontologies. With [13], Miller et al. propose an annotation to the standard description language WSDL in order to facilitate the semantic description of Web services. However, OWL-S is a full language and does not offer the benefits of WSDL annotation, and the WSDL-S annotation typically relies on domain ontologies and does not support additional context attributes nor the use of context ontologies.

Two important works rely on the WSMO framework for semantic Web services [1]. With IRS-III [7], Cabral and Domingue propose a WSMO-based architecture to establish correspondences between service descriptions thanks to a mediator Web service. With WSMX [9], Haller et al. propose a similar solution that is a part of the WSMO framework (WSMX is the reference implementation of WSMO). In these works, the semantics of the terms used are encoded into the WSMO description files of the services. Semantic heterogeneities between Web services are solved by reasoning on the content of a common domain ontology that has for purpose to explicitly describe reference vocabulary. The mediation process is not about converting data but more about matching the semantic description stored in the ontologies.

In [4], Dogac et al. propose an interoperability framework for the healthcare domain. This framework relies on the notion of archetype to describe data semantics. An archetype is a formal way to describe domain concepts via constraints on data. Data instances are constrained instances of a reference domain model. This work is similar to our context-based approach in the sense that a common agreement is made on a domain concept, and the different views of Web services are represented under the form of constraints over the instances of these domain concepts. However, the work of Dogac et al. requires the domain concept to encompass all the different views of Web services, which is feasible in the healthcare domain where predefined models are agreed on, but not in a more general context as presented in this paper.

In [5], Bowers and Ludäscher propose a semantic mediation framework for scientific workflows. Their work relies on the notion of semantic type and structural type, defined on a global ontology shared by all the users. The semantic type corresponds to the abstract concept that characterizes data, and the structural type is the schema that describes data structure. For a single semantic type, the objective is to adapt the different structural data representations of Web services. This paper relies on semantic matching before performing structural-level data mediation. In the present paper, we propose context-based, semantic-level data mediation for Web services.

4.2 Communities of Web services

Several works gather functionally-similar Web services into communities that are accessed via a common interface. Benatallah et al. propose such a solution with SELF-SERV [2]. In this work, several mediators establish correspondences between the community interface and Web services that implement the functionality of the community.

Benslimane et al. [3], also group Web service into communities. The community is accessed via an interface implemented as an abstract Web service that describes the provided functionality in an abstract fashion and a set a concrete Web services that implement the functionality. A generic driver called Open Software Connectivity (OSC) handles the interactions between clients and the community.

Building upon this work, Taher et al. [20] address the problem of Web service substitution in communities. Web service substitution consists in replacing a disfunctioning or non-responding Web service with a functionally equivalent one, in order to find an alternative way to enable a composition in case of exception. Substituting a service with another requires the mediation of communications between the replacing service and the original client. Mediator Web services communicate with the concrete Web services that implement the functionality, each mediator connects to a specific service.

In Taher et al.'s work, Web service selection is performed according to a set of QoS criteria (speed, reliability, reputation, etc.). The community also takes in charge of administrative tasks such as addition and suppression of services to and from the community.

Medjahed proposes a community-based architecture for semantic Web services in [12]. In this work, communities gather services from the same domain of interest and publish the functionalities offered by Web services as generic operations. A community ontology is used as a general template for describing semantic Web services and communities. A major advantage of this work that relates to our proposal is the peer-to-peer community management solution that addresses the problems of centralized approaches.

5. CONCLUSION

Most works on communities either impose a unique ontology that Web services must bind to, or require users to adapt to the semantic requirements of each Web service utilized in the community. In this work, we present a compromise between these approaches by using a context-based approach that separates the shared knowledge from the local contexts of the Web service providers. We provide the appropriate mechanisms to handle semantic data discrepancies between Web services and communities and enable seamless interoperation at the semantic level. Short-term future work includes studying other aspects related to mediation in Web service communities such as transaction level or security aspects.

However, we noticed from our experimentation that the weakest part of our solution resides in updating the context ontologies. Indeed, it is required that providers correctly update the context ontologies, by explicitly describing their own context representation, but also the relationships between their context and the contexts of other providers. Having such knowledge is a hard constraint on providers, and on a large scale it is an assumption that is difficult to defend. Therefore, long-term future work includes studying how to reduce this difficulty by proposing advanced reasoning mechanisms that could (semi-)automatically interconnect the different contexts of providers.

6. REFERENCES

[1] S. Arroyo and M. Stollberg. WSMO Primer. WSMO Deliverable D3.1, DERI Working Draft. Technical report, WSMO, 2004. `http://www.wsmo.org/2004/d3/d3.1/`.

[2] B. Benatallah, Q. Z. Sheng, and M. Dumas. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.

[3] D. Benslimane, Z. Maamar, Y. Taher, M. Lahkim, M.-C. Fauvet, and M. Mrissa. A multi-layer and multi-perspective approach to compose web services. In *AINA*, pages 31–37. IEEE Computer Society, 2007.

[4] V. Bicer, O. Kilic, A. Dogac, and G. B. Laleci. Archetype-based semantic interoperability of web service messages in the health care domain. *International Journal of Semantic Web and Information Systems (IJSWIS)*, 1(4):1–23, October 2005.

[5] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In E. Rahm, editor, *DILS*, volume 2994 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[6] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1. Technical report, The World Wide Web Consortium (W3C), 2000. `http://www.w3.org/TR/SOAP/`.

[7] L. Cabral and J. Domingue. Mediation of semantic web services in irs-iii. In *First International Workshop on Mediation in Semantic Web Services (MEDIATE 2005) held in conjunction with the 3rd International Conference on Service Oriented Computing (ICSOC 2005), Amsterdam, The Netherlands.*, December 12th 2005.

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3c note, The World Wide Web Consortium (W3C), March 2001. `http://www.w3.org/TR/wsdl`.

[9] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx - a semantic service-oriented architecture. In I. C. Society, editor, *ICWS*, pages 321–328. IEEE Computer Society, 2005.

[10] D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In J. Cardoso and A. P. Sheth, editors, *SWSWPC*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42. Springer, 2004.

[11] B. Medjahed and Y. Atif. Context-based matching for web service composition. *Distrib. Parallel Databases*, 21(1):5–37, 2007.

[12] B. Medjahed and A. Bouguettaya. A dynamic foundational architecture for semantic web services.

Distributed and Parallel Databases, 17(2):179–206, 2005.

[13] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. WSDL-S: Adding Semantics to WSDL - White Paper. Technical report, Large Scale Distributed Information Systems, 2004. `http://lsdis.cs.uga.edu/library/download/wsdl-s.pdf`.

[14] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar. A context model for semantic mediation in web services composition. In D. W. Embley, A. Olivé, and S. Ram, editors, *ER*, volume 4215 of *Lecture Notes in Computer Science*, pages 12–25. Springer, 2006.

[15] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar. Towards Context-based Mediation for Semantic Web Services Composition. In *Proceedings of the Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'2006)*, San Francisco, California, July 2006.

[16] M. Nagarajan, K. Verma, A. P. Sheth, J. Miller, and J. Lathem. Semantic interoperability of web services - challenges and experiences. In *ICWS*, pages 373–382. IEEE Computer Society, 2006.

[17] S. Sattanathan, P. Thiran, Z. Maamar, and D. Benslimane. Engineering communities of web services. In G. Kotsis, D. Taniar, E. Pardede, and I. K. Ibrahim, editors, *iiWAS*, volume 229 of *books@ocg.at*, pages 57–66. Austrian Computer Society, 2007.

[18] G. Schreiber and M. Dean. Owl web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.

[19] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers*, 29(12):1104–1113, 1980.

[20] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar. Towards an approach for web services substitution. In P. Ghodous, R. Dieng-Kuntz, and G. Loureiro, editors, *IDEAS*, pages 166–173. IOS Press, 2006.

[21] UDDI Specification Technical Commitee. Universal Description, Discovery, and Integration of Business for the Web. Technical report, October 2001. `http://www.uddi.org`.