Context-based Semantic Mediation in Web Service Communities

Michael Mrissa, Stefan Dietze, Philippe Thiran, Chirine Ghedira, Djamal Benslimane and Zakaria Maamar

Abstract Communities gather Web services that provide a common functionality, acting as an intermediate layer between end users and Web services. On the one hand, they provide a single endpoint that handles user requests and transparently selects and invokes Web services, thus abstracting the selection task and leveraging the provided quality of service level. On the other hand, they maximize the visibility and use rate of Web services. However, data exchanges that take place between Web services and the community endpoint raise several issues, in particular due to semantic heterogeneities of data. Specific mediation mechanisms are required to adapt data operated by Web services to those of the community. Hence, mediation facilititates interoperability and reduces the level of difficulty for Web services to join and interact with communities. In this chapter, we propose a mediation approach that builds on (1) context-based semantic representation for Web services and the community; and (2) mediation mechanisms to resolve the semantic heterogeneities occuring during data exchanges. We validate our solution through some experiments as part of the WSMO framework over a test community and show the limitations of our approach.

Keywords Context · Community · Mediation · Semantics · Web services · WSMO

1 Introduction

The service-oriented paradigm is gaining momentum as a way to interconnect applications across distributed organizations. Especially on the World Wide Web, the advent of Web services provides a framework to interconnect applications. Web services are remotely accessible software components providing a specific functionality. They rely on well-defined Web protocols such as HTTP for transport, which are coupled to some XML-based languages for supporting message

---

M. Mrissa (✉)
PReCISE Research Center, University of Namur, Belgium
e-mail: mmrissa@fundp.ac.be

exchange (SOAP [7]), functional service description (WSDL [10]), and discovery (UDDI [27]). The main advantage of Web services is their capacity of being composed. A composition consists of combining several Web services into the same business process, in order to address complex user's needs that a single Web service could not satisfy.

Due to the increasing number of services available on the Web, the discovery and selection steps are becoming of major importance. They will later determine the relevancy of a composition for fulfilling a specific goal in a specific situation and also contribute to its successful achievement. As stated in the work of Al-Masri et al. [1], the common practice nowadays is to manually search for and select Web services depending on several Quality of Service (QoS) characteristics such as availability, price, efficiency, reliability, etc.

Gathering Web services into communities facilitates the discovery and selection tasks by providing a centralized access to several functionally-equivalent Web services via a unique endpoint. Community-based frameworks thus enhance Web services availability and improve the success rate of compositions. They also improve the confidence level as they select independent Web services according to a set of criteria (price, availability, speed, response quality, etc.). Several research works propose to use communities for easing the management and access to Web services [3, 4, 15, 16, 23, 26].

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. Indeed, when a user request is sent to the community endpoint, the semantics of this request is organized according to the community semantics. This request is forwarded to some Web service of the community. However, each Web service already has its own semantics chosen at design time, prior to subscribing to the community. A mediation is required between the semantics of the Web services that answer the request and those of the community endpoint.

In this chapter, we address the interoperability problems raised by semantic heterogeneities in Web services communities. Our motivation, and what makes the originality and main contribution of this chapter, is to demonstrate how context-based mediation is relevant to the domain of Web service communities, and to illustrate the feasibility of our proposal via a concrete implementation. To do so, we develop a mechanism for semantic mediation that relies on a context-based model for data representation proposed in previous work [19]. We show in the following how to solve semantic discrepancies and enable seamless invocation of Web services in communities, and we illustrate our work by implementing our proposal within the WSMO framework.

This chapter is organized as follows: Sect. 2 introduces the notion of community for gathering Web services and summarizes the context-based model we rely on for describing data semantics. Section 3 discusses how the deployment of a semantic mediation module helps solve 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. Moreover, an implementation alternative based on the WSMO framework is proposed in Sect. 4. Section 5 presents related work on semantic mediation in communities, and Sect. 6 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 Web services via a unique communication endpoint, that is the access point to the community.

In previous work, we proposed an approach that supports the concepts, architecture, operation and deployment of Web service communities [23]. The notion of community serves as a means for binding 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. On top of forwarding calls back and forth between users and slave Web services, the master also manages its community at a higher level. Our previous work details the management tasks a master Web service is responsible for. Such tasks include among other things registering new Web services into the community, tracking bad Web services, and removing ineffective Web services from the community.

A *master* Web service represents the community and handles users' requests with slave Web services with the help of a specific protocol. In our previous work we rely on a slightly extended version of the Contract Net protocol (CNProtocol) [25], as illustrated in Fig. 1:

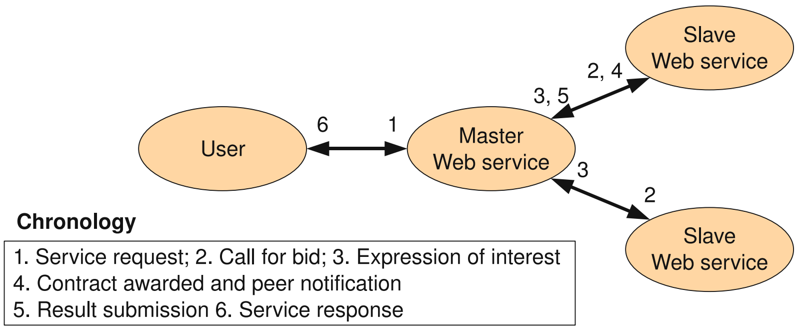
Slave Web service
2, 4
3, 5
6
1
User
Master Web service
3
2
Slave Web service
Chronology
1. Service request; 2. Call for bid; 3. Expression of interest
4. Contract awarded and peer notification
5. Result submission 6. Service response

Fig. 1 Contract-Net protocol interactions

1. The master Web service sends a call for bids to the slave Web services of the community.
2. Slave Web services assess their current status and availability to fulfil the request 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…). Then, it notifies the winner slave Web service.
4. Slave Web service that answered the call for bids but were not selected are notified too.

In this chapter, we provide a context-based semantic mediation architecture for Web service communities. Indeed, the applicability of our mediation proposition goes beyond this domain. However, we specifically focus here on its deployment with communities as defined in [23], where semantic mediation is performed between the community master and slave Web services.

2.2 Context Representation

The notion of context has for a while been a hot topic in the database field [12], and has been specifically adapted in previous work to the description and mediation of data semantics for Web services [19]. In this specific work [19], context-based representation of data semantics is particularly relevant and specifically designed to data exchange between Web services engaged in a composition. It distinguishes two concerns at different levels, where existing knowledge representation approaches see one concern only.

At the conceptual level, context-based approach encompasses matching the different domain concepts used by the actors involved in the data exchange. While semantic differences (such as different granularities of the domain concepts) may hamper straightforward matching between world representations, we assume that correspondences can be established between the domain concepts used.

At the contextual level, context-based approach encompasses the description and mediation of the different data interpretations attached to the domain concepts, which are intrinsically related to the (heterogeneous) local assumptions of the actors involved. At this level, more complex conversion rules are required to enable accurate data conversion and interpretation.

Thus, while typical approaches on knowledge representation visualize domain concepts with attached properties, and perform data mediation in an all-in-one fashion, context-based approach distinguishes two concerns and simplifies the mediation steps by dividing the complex mediation task into two distinct subtasks. The first subtask is to interconnect domain concepts at the semantic level, and the second subtask is to mediate data between different contexts using conversion rules.

Context description 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 May 15, 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.

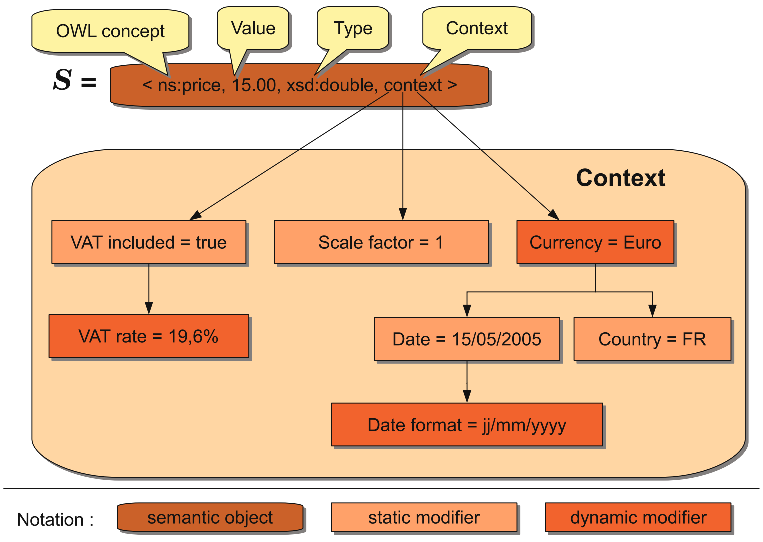

Fig. 2 Description of the *price* semantic object

2.3 Domain and Context Ontologies

The use of context comes from a simple assumption: Web services adhere to communities with difficulty. Several reasons such as strategic and economic aspects are involved, but are out of the scope of this chapter. Another reason is semantic incompatibility. In fact, each community follows a specific knowledge representation that is not always compliant with other communities' knowledge representations, and Web services already have, 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 knowledge representation. 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 they decide to adhere to new communities, by scaling domain ontologies down to the minimum, and providing additional context ontologies to handle the different local semantics of service providers. To do so, our context-based model makes the distinction between domain knowledge and context knowledge.

Domain knowledge includes main concepts that are assumed to be (or should be) common to all parties. For instance, the concept of *price* might be part of a domain knowledge and included in domain ontologies. We deem appropriate to limit the application of domain ontologies to the concepts and relationships that can be devised in a top-down way. Experts in the knowledge domain should specify domain ontologies and limit to the minimum the extra 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 communities could adopt, although such a situation is not likely to happen in an open world like the Web.

Context knowledge includes all the knowledge (i.e. properties, features, etc.) attached to the concept of *price*, which is useful for a correct interpretation when one needs to understand an instance of *price*. This is where context ontologies come into play. A context ontology is attached to each concept of the domain ontology. Context ontologies 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 their semantics explicit via the context ontology. In addition Web 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 of concerns we propose between domain and context ontologies follows such a well-established practice in order to facilitate semantic interoperability between Web services.

2.4 Context Annotation of 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 chapter. Each parameter has a `<name>` and a `<type>` attribute, and allows additional attributes.

In [20], we proposed a WSDL 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 [10], 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 this chapter self-contained, we provide a simplified structure of the annotated WSDL document in Listing 1.

```
<? 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

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 of WSDL. Furthermore, rules separate application logic from the rest of the system, so updating rules does not require rewriting application code. In this chapter, we rely on our annotation and rule-based mechanisms in order to provide the semantic information required to perform semantic mediation within a community.

Our annotation of WSDL is a way to add semantics to the standard description language for Web services. While in our implementation (Sect. 4) we use another language (OCML) for describing Web services, our WSDL annotation allows existing Web services to comply with the requirements of context-based representation with few simple changes in the original WSDL file of the Web service.

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 earlier.

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 handles 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 on Context-Based Mediation

When it comes to the mediation aspect, 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 *WSDL context reader* 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.

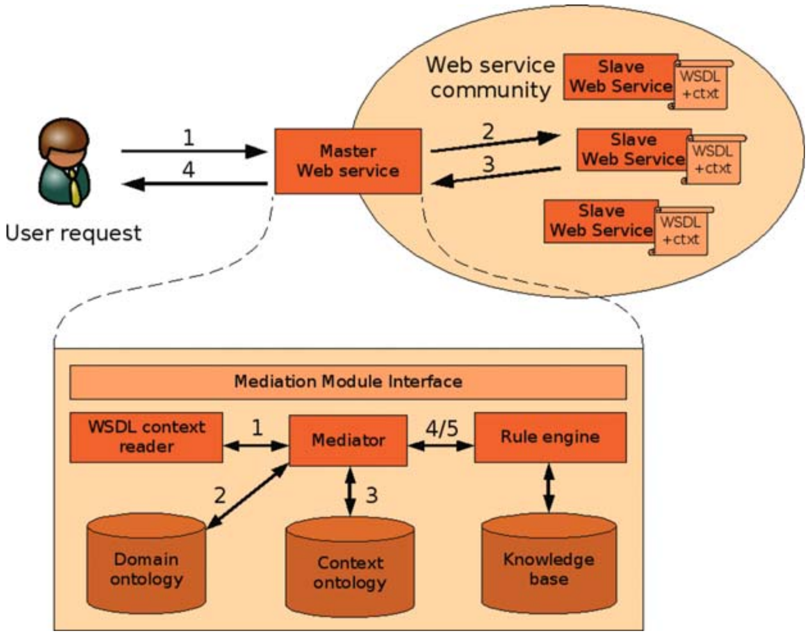

Fig. 3 Overview of the mediation process

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

1. Reading WSDL annotation

    a. It selects a slave Web 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 a 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 needs now 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 an outgoing 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 succeed or forwards the converted data back to the client of the community.

4 Interoperability with Semantic Web Services Frameworks: The Case of WSMO

In order to demonstrate the applicability of our approach, we consider the implementation of the conceptual approach proposed in this chapter through established Semantic Web Service (SWS) frameworks. Particularly, we discuss in the following its implementation through the established Web Service Modelling Ontology (WSMO) framework.

4.1 Semantic Web Services and WSMO

SWS frameworks aim at the automatic discovery, orchestration and invocation of distributed services for a given user goal on the basis of comprehensive semantic descriptions. SWS are supported through representation standards such as WSMO [2] and OWL-S [14]. In this chapter, we refer to the Web Service Modelling Ontology (WSMO), a well established SWS reference ontology and framework. The conceptual model of WSMO defines the following four main entities:

- Domain Ontologies provide the foundation for describing domains semantically. They are used by the three other WSMO elements. WSMO domain ontologies not only support Web services related knowledge representation but semantic knowledge representation in general.
- Goals define the tasks that a service requester expects a Web service to fulfill. In this sense they express the requester's intent.
- Web service descriptions represent the functional behavior of an existing deployed Web service. The description also outlines how Web services communicate (choreography) and how they are composed (orchestration).
- Mediators handle data and process interoperability issues that arise when handling heterogeneous systems.

WSMO is currently supported through several software tools and runtime environments, such as the Internet Reasoning Service IRS-III [9] and WSMX [28]. IRS-III is a Semantic Execution Environment (SEE) that also provides a development and broker environment for SWS following WSMO. IRS-III mediates between a service requester and one or more service providers. Based on a client request capturing a desired outcome, the goal, IRS-III proceeds through the following steps utilizing the set of SWS capability descriptions:

1. Discovery of potentially relevant Web services.
2. Selection of a set of Web services which best fit the incoming request.
3. Invocation of selected Web services whilst adhering to any data, control flow and Web service invocation constraints defined in the SWS capabilities.
4. Mediation of mismatches at the data or process level.

In particular, IRS-III incorporates and extends WSMO as core epistemological framework of the IRS-III service ontology which provides semantic links between the knowledge level components describing the capabilities of a service and the restrictions applied to its use. IRS-III utilizes OCML [18] as knowledge modelling language to represent WSMO-based service models.

4.2 Implementing Context-Based Mediation Through WSMO and IRS-III

Particularly with respect to mediation, the use of WSMO and IRS-III provides several advantages, since mediation is an implicit notion of WSMO with explicit support through dedicated mediators [8]. Mediators are often classified as OO-, GG-, WG- and WW-mediators [22]. Whereas OO-mediators resolve mismatches between distinct WSMO ontologies, GG-mediators refine a WSMO goal. WG-mediators mediate between heterogeneous goals and SWS specifications whereas

WW mediators resolve heterogeneities between distinct Web service implementations. Whereas OO-, GG-, and WG-mediators primarily support SWS discovery, WW mediators are related to orchestration and invocation. However, mediation usually involves a set of multiple different mediators. For instance, a WG-mediator usually requires OO-mediation as well in order to align distinct vocabulary formalizations. Therefore, we use the generic term mediator throughout the remaining sections instead of explicitly separating between different sorts of mediators.

The conceptual model of WSMO is well-suited to support the conceptual approach proposed in this chapter, since the representation of distinct service provider perspectives, i.e., their contexts, is an implicit element of WSMO. Moreover, distinct contexts are addressed through the notion of mediation, which aims at resolving heterogeneities which will definitely occur between distinctive Web service implementations and representations. Particularly, regarding the conceptual approach proposed in this chapter, we propose an implementation of the mediation scenario (Sect. 3) utilizing WSMO and IRS-III as follows:

1. Representation of slave Web services as SWS following WSMO.
2. Representation of context ontologies utilized by each provider as WSMO ontologies associated with each SWS.
3. Aligning the WSMO ontologies to a common upper-level domain ontology.
4. Representation of a common WSMO goal expressing the community request.
5. Description of WSMO mediators linking WSMO SWS to the WSMO goal by defining appropriate mediation rules.

Given these semantic representations, IRS-III is able to select appropriate services, which are orchestrated and invoked, whereas the mediator resolves heterogeneities at runtime. In that, by referring to the description of the functionalities of the master Web service (Sect. 3.2), it can be stated, that the built-in reasoning of IRS-III facilitates steps 1–4 as proposed in Sect. 3.2, while the implemented WSMO mediator (Med.1 in Fig. 4) aims at resolving data heterogeneities (step 5 of Sect. 3.2). Figure 4 depicts the created WSMO goal, mediator and SWS descriptions which implement the use case described in Sect. 3.

Please note, that the context ontologies, representing the context of each Web service provider are now explicit parts of the WSMO SWS descriptions, i.e., WS.1 and WS.2, whereas the domain ontology is implemented as an independent upper-level WSMO ontology, which is derived for certain contexts through the WSMO SWS descriptions, respectively the context ontologies. Indeed, the context and domain ontologies as mentioned in Sect. 2.2 are not physically separated, but the context-based approach still holds as the two levels (domain concept matching and data interpretation/conversion) remain clearly distinct via WSMO ontologies and SWS descriptions. Apart from that, we would like to point out that following the proposed WSMO-based approach, the previously introduced slave services are now supported through WSMO-compliant SWS, whereas the functionality

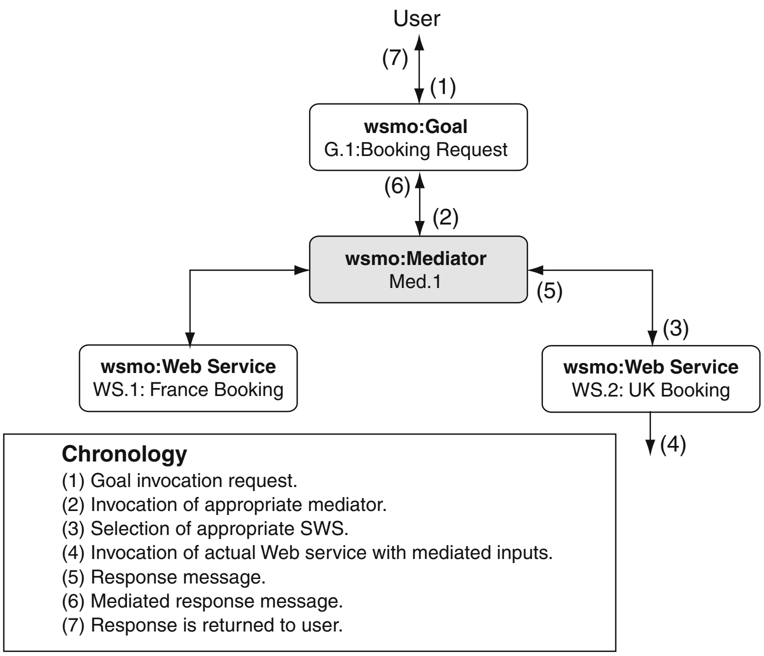
User
(7)
(1)
wsmo:Goal
G.1:Booking Request
(6)
(2)
wsmo:Mediator
Med.1
(5)
(3)
wsmo:Web Service
WS.1: France Booking
wsmo:Web Service
WS.2: UK Booking
(4)
Chronology
(1) Goal invocation request.
(2) Invocation of appropriate mediator.
(3) Selection of appropriate SWS.
(4) Invocation of actual Web service with mediated inputs.
(5) Response message.
(6) Mediated response message.
(7) Response is returned to user.

Fig. 4 WSMO goal linked to Semantic Web Services through common mediator

of the master Web service is provided by the WSMO reasoning mechanisms of IRS-III together with the mediation rules defined in the corresponding WSMO mediator.

Referring to the previously given definition of a semantic object $S = (c, v, t, C)$, c is defined as part of the upper-level WSMO domain ontology, while t is defined as part of the WSMO SWS, describing on the one hand the XML binding of the actual input message of the Web service and its expression following the used modelling language (OCML) on the other hand. The value v represents the actual value of an input parameter chosen to invoke the WSMO goal while the context C is represented as part of the WSMO SWS descriptions. Moreover, please take into account, that the WSMO descriptions proposed above not only enable the mediation between distinct data formats, i.e., currencies and date formats, but also the selection of the most appropriate Web service, either WS.1 or WS.2, based on the input values used to invoke the goal G.1. Particularly, the requested departure country is utilized with this respect, i.e., WS.1 is selected in case the requested booking departs from any European country other than UK, while in the case of UK, WS.2 is automatically invoked. The Web service selection is based on the semantic capability descriptions provided through WSMO. Listing 2 shows a simple SWS capability of WS.2 enabling the selection proposed above:

```
wsmo:WebService WS.2

(DEF–CLASS Get–UK–BOOKING–REQUEST–WS–CAPABILITY
          (CAPABILITY)
          ?CAPABILITY
          ((USED–MEDIATOR :VALUE GET-BOOKING–REQUEST–MED)
                      (HAS–ASSUMPTION
                      :VALUE
                      (KAPPA
                        (?WEB–SERVICE)
(= (WSMO–ROLE–VALUE ?WEB–SERVICE HAS–DEPARTURE–COUNTRY) ”Uk”)))
          (HAS–NON–FUNCTIONAL–PROPERTIES
:VALUE
Get–UK–BOOKING–REQUEST–WS–CAPABILITY–NON–FUNCTIONAL–PROPERTIES)))
```

Listing 2 SWS capability description of WS.2

It can be summarized, that basing the conceptual approach proposed in this chapter on a common SWS framework, i.e., WSMO, and an established execution environment and reasoning engines, namely IRS-III, provides the opportunity of reusing predefined functionalities related to SWS representation, discovery and orchestration. Particularly, it could be shown, that the WSMO-based implementation enabled the reuse of IRS-III in order to deal with the master service functionalities 1-4 (Sect. 3.2). Moreover, the approach of aligning distinct context representations, being implicit part of SWS descriptions, to a common upper-level ontology is well-suited to facilitate interoperability between distinct SWS representations [11].

5 Related Work

To the best of our knowledge, there are no existing works on semantic mediation within the context of Web service communities as defined in [23]. However, our approach is inspired by several works on semantic mediation for Web services and communities of Web services, that we detail hereafter.

5.1 Semantics and Mediation for Web Services

Semantic description and mediation for Web services is a very active research topic, as the many works on the subject [2, 5, 6, 8, 13, 14, 17, 21] prove. In the following, we describe the most important works that inspired us for this chapter.

In [21], 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 [14] is a language for semantic description of Web services, relying on the OWL language [24], OWL-S enables explicit semantic description of Web services input an output parameters via references to concepts described in OWL domain ontologies. With [17], 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.

With WSMX [13], Haller et al. propose a solution that is a part of the WSMO framework (WSMX is the reference implementation of WSMO). In this work, 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 [5], 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 chapter.

In [6], 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 typical semantic matching methods before performing structural-level data mediation. In the present work, we propose context-based, semantic-level data mediation for Web services.

5.2 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 [3]. In this work, several mediators establish correspondences between the community interface and Web services that implement the functionality of the community.

Benslimane et al. [4], also group Web services 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. [26] address the problem of Web service substitution in communities. Web service substitution consists of replacing a non-functioning 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 is also in charge of administrative tasks such as addition and suppression of services to and from the community.

Medjahed and Bouguettaya proposes a community-based architecture for semantic Web services in [16]. 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.

6 Conclusion

Communities facilitate the discovery and selection of several functionally-equivalent Web services. Nevertheless, researches on communities usually impose a unique ontology that Web services must bind to, or require users to adapt to the semantic requirements of Web services belonging to the community. In this work, we present a trade-off between these approaches by using a context-based approach that separates shared knowledge from the local contexts of Web service providers. We demonstrate the significance of our proposal and develop mediation mechanisms that handle semantic data discrepancies between Web services and communities, thus enabling seamless interoperation at the semantic level. Short-term future work includes studying other aspects related to mediation in Web service communities such as transactional or security aspects.

However, we noticed from our experimentation that creating and using several ontologies is a difficult task, the hardest task being the update of context ontologies. Indeed, it is required that providers correctly update their context ontologies with their own context representations, but also they must provide the correspondences between their context and the contexts of other providers. Making such knowledge explicit is a hard task for providers, particularly on a large scale.

In that way, it would be interesting to find suitable solutions to avoid such constraints on providers. Therefore, long-term future work includes studying how to reduce this task by proposing advanced reasoning mechanisms that could help interconnect the different contexts of providers.

References

1. E. Al-Masri and Q. H. Mahmoud. Investigating web services on the world wide web. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 795–804. ACM, 2008.
2. 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/`.
3. B. Benatallah, Q. Z. Sheng, and M. Dumas. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.
4. 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, Washington, DC 2007.
5. 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.
6. 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.
7. 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/`.
8. 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.
9. L. Cabral, J. B. Domingue, S. Galizia, A. Gugliotta, B. Norton, V. Tanasescu, and C. Pedrinaci. IRS-III: A broker for semantic web services based applications. In *Proceeding of the 5th International Semantic Web Conference (ISWC2006)*, Athens, GA, USA, 2006.
10. 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`.
11. S. Dietze, A. Gugliotta, and J. Domingue. A semantic web service oriented framework for adaptive learning environments. In E. Franconi, M. Kifer, and W. May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 701–715. Springer, 2007.
12. C. H. Goh, S. Bressan, S. Madnick, and M. Siegel. Context interchange: new features and formalisms for the intelligent integration of information. *ACM Transactions on Information and Systems*, 17(3):270–293, 1999.
13. 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 Washington, DC, 2005.
14. 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 Berlin, 2004.
15. B. Medjahed and Y. Atif. Context-based matching for web service composition. *Distrib. Parallel Databases*, 21(1):5–37, 2007.
16. B. Medjahed and A. Bouguettaya. A dynamic foundational architecture for semantic web services. *Distributed and Parallel Databases*, 17(2):179–206, 2005.
17. 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`.

18. E. Motta. An overview of the ocml modelling language. In *Proceedings KEML98: 8th Workshop on Knowledge Engineering Methods & Languages*, pages 21–22. Karlsruhe, Germany, 1998.
19. 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, Berlin, 2006.
20. 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.
21. 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 Washington, DC, 2006.
22. M. Paolucci, N. Srinivasan, and K. Sycara. Expressing WSMO mediators in owl-s. In *Proceeding of the Semantic Web Services Workshop (SWS) at the 3rd International Semantic Web Conference (ISWC)*, Hiroshima, Japan, 2004.
23. 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, Wien, 2007.
24. G. Schreiber and M. Dean. Owl web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.
25. 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.
26. 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, Amsterdam, 2006.
27. UDDI Specification Technical Commitee. Universal Description, Discovery, and Integration of Business for the Web. Technical report, October 2001. `http://www.uddi.org`.
28. WSMX Working Group. The web service modelling execution environment, 2007. `http://www.wsmx.org/`.