A Context Model for Semantic Mediation in Web services Composition

Michael Mrissa[1], Chirine Ghedira[1], Djamal Benslimane[1], and Zakaria Maamar[2]

[1] Université Claude Bernard Lyon 1, Villeurbanne, France
`firstname.lastname@liris.cnrs.fr`
[2] Zayed University, Dubai, United Arab Emirates
`zakaria.maamar@zu.ac.ae`

Abstract. This paper presents a context-driven approach that aims at supporting semantic mediation between composed Web services. Despite the widespread adoption of Web services by the IT community, innovative solutions are needed in order to overcome the challenging issue that relates to the semantic disparity of exchanged data. Indeed, there is a lack of means for interpreting these data according to the contextual requirements of each Web service. The context-driven approach suggests two steps. The first step consists of developing a model for anchoring context to data flowing between Web services. In the second step, we use this model to support the semantic mediation between Web services engaged in a composition.

1 Introduction

In the field of service-oriented computing, Web services are now widely used to connect business processes. The suitability of Web services for composition allows answering complex users' needs. Composition involves interacting Web services to provide value-added business processes. However, efficient description and management of semantics of data are major requirements to the success of system interoperability. Particularly, composition requires understanding the semantics of the data exchanged between Web services. The Web services protocol stack (SOAP [1], WSDL [2], and UDDI [3]) achieves application level interoperability, but does not satisfy the requirements of semantic exchange. Recent initiatives propose languages and frameworks (e.g., OWL-S [4], WSMO [5], and WSDL-S [6]) that use ontologies[3] for adding explicit semantic descriptions to Web services, which are now referred to as *semantic Web services*.

However, these initiatives towards semantic Web services do not take into consideration the *context* of exchanged data. By context, we mean the collection of implicit assumptions that are required to obtain accurate data interpretation. We advocate that a semantic concept should be interpreted differently, depending on the context it relates to. In the domain of Web services composition,

[3] An ontology is defined as a shared description of a domain knowledge in [7].

context interpretation generally remains ignored, due to lack of explicit context descriptions. As a consequence, the adaptation of Web services to context changes is still performed manually, which reduces their availability and reliability. Explicit context description and management are required to meet the challenges of automatic semantic interpretation and data flow handling during Web services composition.

In this paper, we aim at *presenting a context-based approach for semantic reconciliation of Web services engaged in a composition.* To this end, we develop a model that supports explicit description of context, before deploying runtime mediation mechanisms between Web services, based on the contextual annotation of WSDL input and output message parts.

This paper is organized as follows. Section 2 suggests a motivating example to back the value-added of data context management to Web services composition. Section 3 presents a context-based model for Web services, supported by the definition of *semantic object*, prior to discussing the integration of this model into the Web services protocol stack. Section 4 presents a context- and rule-based mediation architecture for Web services composition. Section 5 overviews related work on mediation and semantics for Web services and context representation. Finally, Section 6 concludes the paper and sets guidelines for future work.

2 Motivating Example

We demonstrate with a simple booking example how context impacts the interpretation of data flow between Web services. The example concerns a trip to Japan. A rate-based attractive hotel provides a Web service for bookings. To judge the affordability of this hotel for an European passenger, the following composition occurs: *hotel booking* WS_1 calculates charges based on the number of booked nights, and *banking* WS_2 manages account payment.

From a technical perspective, WS_1 sends "price-yen" parameter and WS_2 receives "price-euros" parameter. Both parameters are WSDL message parts. Although different type systems can be used, we consider for illustration purposes that "price-yen" and "price-euros" parameters are in XML Schema type system [8], and are of type "double". These details show low-level data compatibility between Web services. In addition, "price-yen" and "price-euros" parameters both have particular semantics. WS_1 delivers a value in Yens, whereas WS_2 expects a value in Euros, and both bind to a "price" semantic concept available in a common ontology. Existing approaches to semantic description and mediation of Web services, to overview in Section 5, explicitly describe the correspondence between parameters for conversion requirements. Such approaches refer to shared ontologies to address structural and semantic heterogeneities.

Now, let us inject context into these parameters. WS_1 binds to "Japanese Hotel Booking" context, in which charges have a scale factor of 1000, prices do not include Value-Added Tax (VAT), dates for conversion rates are in Japanese format (yyyy.mm.dd). WS_2 binds to "French Banking" context, where charges have a scale factor of 1, prices include VAT, and dates for conversion rates are in French format (dd.mm.yyyy). This shows context heterogeneity exists too, so an agreement on the value interpretation must be reached through context reconciliation.

Composing Web services involves dealing with many different contexts, and enabling significant interactions requires dynamic and complex transformations to adapt data to these contexts. In a semantic composition, context heterogeneity is resolved in an ad-hoc way at the receiver Web-service level, if at all. This reduces Web services adaptability and overloads them with solving context heterogeneities. To conduct context-aware composition, the context of data must be explicitly described and a mediation mechanism must handle data flow. Our proposal is to annotate WSDL so that messages parts are propelled to the level of *semantic objects*, which are described in the following.

3 A Context-based Model for Web services

As aforementioned, we propose a model that describes the underlying semantics of data flow between Web services. This model takes advantage of the notion of *semantic object* given in [9], and focuses on context description for data exchange in Web services composition. In this section, we define the two fundamental elements of our model: *semantic object* and *context*. Afterwards, we discuss how semantic conversion is performed between semantic objects using *conversion functions*. Finally, we define the notion of *semantic and absolute comparison* between semantic objects.

3.1 Semantic Object

In the domain of semantic Web services, concern separation between data grounding and data abstract-view is required. Listing 1.1 illustrates this separation with an OWL-S Web service input description:

```
<!-- Abstract description -->
        <process:Input rdf:ID="InputLanguage">
                <process:parameterType  rdf:datatype="&xsd;#anyURI">
                        &this;#SupportedLanguage
                </process:parameterType>
                <rdfs:label>Input Language</rdfs:label>
        </process:Input>

<!-- Grounding description-->
        <grounding:WsdlInputMessageMap>
                <grounding:owlsParameter  rdf:resource="#InputLanguage"/>
                <grounding:wsdlMessagePart rdf:datatype="&xsd;#anyURI">
                        &groundingWSDL;#inputLanguage
                </grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
```

Listing 1.1. OWL-S Input Description Snippet

The abstract view binds the data to a conceptual description generally using an ontology language like OWL [10]. The grounding view describes the physical representation of data which generally follows XML Schema [8]. This separation allows different physical representations of the same concept, and strengthens the role of ontologies in the abstract representation of data semantics. In the rest of this paper, we refer to concept c as an individual, or fact, defined in a domain ontology. The notion of individual is detailed in the OWL recommendation [10].

Following a similar separation of abstract and grounding descriptions, we define a semantic object as a data object, i.e., a value v that is an instance of type t with "enough" meta-data for automatic interpretation. This meta-data includes a concept c, which describes the real world phenomena that the data object refers to, and a context C represented as a tree of meta-attributes. A semantic object $SemObj$ is a 4-tuple represented as follows:

$$SemObj =< c, v, t, C >,$$

where c is the concept that the semantic object $SemObj$ adheres to, value $v \in Dom(t)$ is the physical representation of v according to the domain of representation Dom of type t, and C specifies the context of $SemObj$. This context is a tree of semantic objects called *modifiers*. Such representation of an initial semantic object with additional semantic objects makes our context-based model self-describing. A formal definition of a context C is:

$$C = \{< c_1, v_1, t_1, C_1 >, \ldots, < c_n, v_n, t_n, C_n >\}, n \in \mathbb{N} \quad ,$$

where $< c_i, v_i, t_i, C_i >, 1 \leq i \leq n$, are modifiers that describe different semantic aspects of $SemObj$. Modifiers may also have a context, described in C_i, so it is possible to use recursive descriptions and to represent context in a tree.

3.2 Static and Dynamic Modifiers

On the basis of the definition presented above, we introduce the notion of *static* and *dynamic* modifiers. Values of static modifiers have to be explicitly specified, whereas values of dynamic modifiers can be determined by a function from the values of other (static or dynamic) modifiers. In Fig. 1, "*date format*" modifier is dynamic; its value can be inferred from the value of the "*country*" modifier. The relation of inference can be described as a rule, such as "If country is *France*, then date format is *dd.mm.yyyy*". Similar rules should be used for other countries. Further details on how rules support the proposed mediation architecture are given in Section 4. Formally, being given a modifier S and a context $Ctxt$ such that $S =< c, v, t, C > \in Ctxt$, then S is *dynamic* iff:

$$\forall v \in S, \exists f : \{Dom(t) \times \ldots \times Dom(t)\} \mapsto Dom(t) \wedge \exists \{S_1, \ldots S_i, \ldots, S_n\},$$
$$s.t.\ S_i =< c_i, v_i, t_i, C_i > \in Ctxt \wedge S_i \neq S \wedge f(v_1, \ldots, v_i, \ldots, v_n) = v.$$

Figure 1 shows a semantic object to be forwarded to banking Web service of Section 2. *ns:price* attribute refers to the concept of price described in a domain ontology, *55.00* is the value of type *xsd:double* flowing between the Web services. *Context* attribute is a list of modifiers that permit explicit interpretation of the inital semantic object. Here, the semantic object is in Euro, has a scale factor of 1, and includes a VAT of 19.6%. Additional parts of the context further describe the *Currency* modifier.

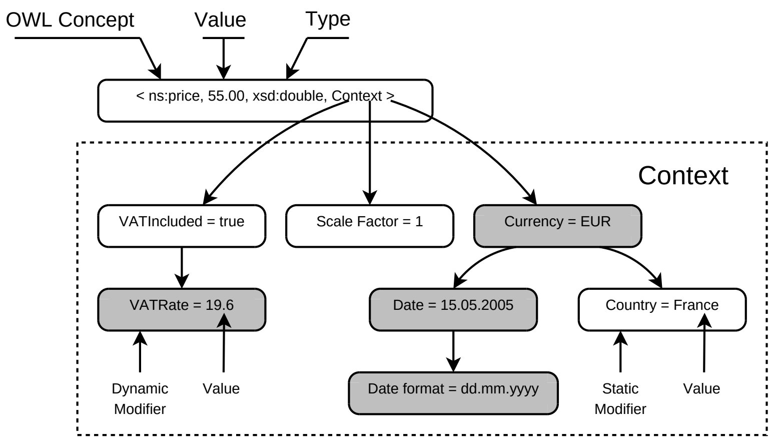


Fig. 1. Sample of a semantic object

3.3 Semantic Conversion of Semantic Objects

Adding context to data allows an explicit representation of the semantics of these data. Therefore, different semantic objects may describe the same information although they have different data and contexts. For example, let us have two simple semantic objects:

$$S_1 =< Price, 1, float, (currency = Euro) >$$
$$S_2 =< Price, 6.55957, float, (currency = French\ Francs) >$$

It is straightforward to note that S_1 and S_2 describe the same information (they are equal), because the exchange rate from French Francs to Euros is fixed at 6.55957 Francs for 1 Euro. Thus, a *conversion function* is required to change S_1 into French Francs or S_2 into Euros and show that S_1 and S_2 are equals.

Conversion functions enable mediation between semantic objects. They have several properties such as total, lossless, and order-preserving [11]. A total conversion function converts to and from any value of its domain of definition, e.g., distance unit conversion functions. An example of non-total conversion is precision conversion. Indeed, a precision conversion function can convert the value 1.25762 into a value with only one decimal of precision (1.2), but it cannot convert this result back to a better precision. In addition, a function is lossless if it can be applied several times on the same object without any loss of information. A function that compresses data files is lossless because the original content can later be extracted. However, a function that converts a BMP image into the JPEG format is lossy (loss due to image compression). A function is order-preserving when two semantic objects, once converted, conserve the order they had before. Temperature conversion functions between Celsius and Fahrenheit scales are order-preserving.

We distinguish two categories of conversion functions. *Context* and *type* conversion functions. *Context* conversion functions are related to the values that modifiers take. They change the interpretation of a semantic object and its value as well. They are stored as rules and may involve online access to other data sources. For example, currency rate conversion functions may call online currency rates providers for up-to-date rates. *Type* conversion functions only change the type t of semantic object (e.g., String2Float, Double2Integer). Such functions depend on the type system that is used to physically represent the semantic object. They can be part of a library associated with the type system, and are not prone to frequent changes.

3.4 Semantic Comparability of Semantic Objects

Since semantic objects can be converted into particular types and contexts, we introduce the notion of semantic comparability between semantic objects. Comparing semantic objects is a prerequisite to the semantic mediation to be introduced in Section 4. First, we consider two semantic objects $S_1 =< c, v_1, t_1, C_1 >$ and $S_2 =< c, v_2, t_2, C_2 >$ that refer to the same concept c. Let us have a relation ϕ (such as '<', '>' or '='), a context C (called target context), and a type t (called target type). Let us assume a conversion function $cvt(value, type, context)$ that consists of concatenating several conversions. This function converts v_1 and v_2 into type t and context C, such as $v'_1 = cvt(v_1, t, C)$ and $v'_2 = cvt(v_2, t, C)$. We state that S_1 and S_2 are semantically comparable with regard to type t and context C if v'_1 and v'_2 satisfy the relation $v'_1 \, \phi \, v'_2$. Therefore, if ϕ is the equality relation '=' we verify the equality of S_1 and S_2.

Second, we show that semantic objects that do not refer to the same concept, can still be compared relatively to the semantic aspects they have in common. For example, let us compare S_1 and S_2 such as:

$$S_1 =< ns : measurePrice, 10.00, float, (currency = euro, measureUnit = kg) >$$
$$S_2 =< ns : unitaryPrice, 15.00, float, (currency = euro, scaleFactor = 1) >$$

The first concept is the price of a measure in kilograms. The second concept is the unitary price that supports different scale factors. If v_1 and v_2 are compared according to context $C = (currency = euro)$ and type $t = float$, $v_1 < v_2$ is established. This example illustrates the possibility to perform a restricted comparison of these semantic objects although they refer to different concepts. We conclude that the semantic comparability of two semantic objects depends on the target context and the possibility of casting object types.

3.5 Absolute Comparability of Semantic Objects

Another aspect that turns out relevant for semantic mediation is the absolute comparison of semantic objects. It is reached when the semantic objects always verify a relation over a target context for all the possible values of the modifiers of this context. Let us consider two semantic objects S_a and S_b. Let be a relation ϕ (such as '<', '>' or '='), a target context $C = \{S_1, \ldots, S_n\}$ and a target type t. Let us consider a conversion function $cvt(value, type, context)$ that concatenates several conversions and converts v_a and v_b into type t and context C, such as $v'_a = cvt(v_a, t, C)$ and $v'_b = cvt(v_b, t, C)$. Then, we define S_a and S_b as absolutely comparable relatively to t and C if $v'_a \phi v'_b$ is verified, for all the possible values that the modifiers of C can take.

3.6 Context Integration into the Web Services Model

The context-based model described above meets the requirements for describing message parts of Web services as semantic objects. The concept of semantic object is intensionally described in a domain ontology, while context is extensionally described using additional meta-attributes. In addition, this model clearly distinguishes the data type t from the conceptual reference C of the semantic object. Then, existing mediation approaches to discuss in Section 5 can seamlessly adhere to our context representation. However, this model raises several questions about its integration into the Web services protocol stack.

Following Bornhövd's view [9], we advocate that a context description is always a subset of all the meaningful aspects of a concept, which are potentially infinite. However, Web service providers should be free to decide which subset of possible aspects is relevant to their application. Therefore, the vocabulary for context description cannot be added into the domain ontology. In such case the size of the latter would grow along with providers' needs. In effect, describing context as part of the domain ontology would require a specific subconcept for each possible combination of modifiers of a domain concept.

To overcome this problem, context ontologies are separated from domain ontologies so that they do not surcharge the latter. Context ontologies describe all the modifiers that Web service providers associate to a concept. Therefore, a context ontology is available for each concept of a domain ontology. Such context ontology should be extended according to Web service providers' requirements. In the following, we assume that Web service providers refer to the same context ontology when annotating Web services. Thus, our illustrative example relies on a single context ontology to put forward the importance of context.

As context ontologies provide shared vocabularies to specify structural and semantic representations of context, there is a need to extensionally specify context values into the descriptions of Web services. We propose a different solution for static and dynamic modifiers. In effect, values of static modifiers have to be specified to clarify the meaning of data. At the contrary, values of dynamic modifiers can be inferred from other parts of the semantic object. Therefore, we insert the description of static modifiers into WSDL, so that our approach is compliant with the standard Web services protocol stack. Descriptions of static modifiers provide the means for calculation of dynamic modifiers at runtime, using appropriate rules.

The use of context ontologies and WSDL annotations helps providers make explicit the context of data. It provides a scalable solution to integrate context into the Web service model. Also, it enables semantic mediation of data during the execution of a composition. In the next section, we present our solution for annotating descriptions of composed Web services, in order to make contextual

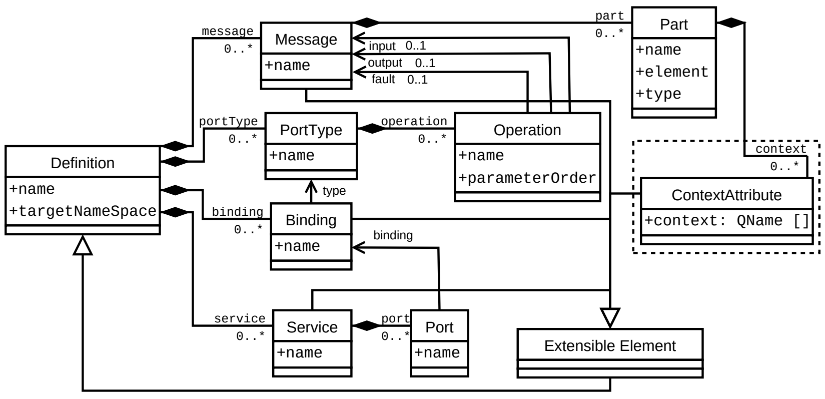


Fig. 2. Context in WSDL metamodel

information available at the execution stage of composition, before describing a service- and rule-based solution for context mediation.

4 Context management for Web Services

4.1 Annotating WSDL with Context

The use of the model described previously requires enriching the description of Web services with context, by annotating WSDL message parts, so that they can be described as *semantic objects*.

In WSDL descriptions, `<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 `<name>` and `<type>` attributes, and allows additional attributes. Our annotation takes advantage of such extension proposed in the WSDL specification [2], so that annotated WSDL operates seamlessly with classical and annotation-aware clients. To keep the paper self-contained, we overview a simplified structure of the WSDL metamodel including the annotation in Fig. 2.

We annotate `<part>` elements 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 ontology concept of the value (c). Additional elements refer to instances of static modifiers described in a context ontology. Listing 1.2 shows the proposed extension and corresponding namespaces in a WSDL file.

Relying on this annotation, a value v and its data type t described in WSDL are enhanced with the concept c and the modifiers necessary to define the context C, thus forming a semantic object $< c, v, t, C >$. To complete the context C, rules help infer the values of dynamic modifiers at runtime. This offers several advantages: rules are easily modifiable, making this solution adaptable to changes in the underlying semantics. Also, often-changing values of 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 the following, we detail our context mediation architecture, that integrates into composition as a Web service, and show its interactions with a rule-based inference engine.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost.../EuroBanking.jws"
   ...
   xmlns:ctxt="http://www710.univ-lyon1.fr/~mmrissa/context/context.xsd"
   xmlns:ctxt1="http://domain.ontology.org/Price.owl"
   xmlns:ctxt2="http://context.ontology.org/context/PriceContext.owl\#">
   ...
  <wsdl:message name="checkPriceReq">
    <wsdl:part name="price" type="xsd:double"
    ctxt:context="ctxt1:Price ctxt2:France
    ctxt2:VATIncluded ctxt2:ScaleFactorOne"/>
  </wsdl:message>
   ...
  <wsdl:portType name="EuroBanking">
    <wsdl:operation name="checkPrice" parameterOrder="price">
    <wsdl:input name="checkPriceReq" message="impl:checkPriceReq"/>
    <wsdl:output name="checkPriceResp" message="impl:checkPriceResp"/>
  </wsdl:operation>
 </wsdl:portType>
   ...
</wsdl:definitions>
```

Listing 1.2. Annotated WSDL Snippet

4.2 Context Integration and Mediation

Regarding the integration of context management capabilities into composition, we adopt a decoupled approach and deploy the context mediation functionality as a Web service. This solution presents three main advantages. First, the mediator Web service can be triggered via its WSDL interface by any remote composition, so it remains independent from composition languages and engines. Second, from composition point of view, it is straightforward to handle context. Composition designers invoke the mediation Web service between every two composed Web services. Third, data mediation is performed at runtime, so the operation of conversion is not statically stored. Instead, conversion rules dynamically infer the conversion between contexts. However, the scope of the mediator Web service is limited to data types specified in its WSDL description. To work out this problem, we generate at design time adapted WSDL description for accessing the mediator Web service.

The role of the mediator Web service is to convert data from the context of the Web service it originates (called source context) into the context of the Web service it is being sent to (called target context). With each exchanged message part, the mediator Web service carries out the following operations:

1. builds and populates source and target contexts using annotated data, ontologies and rules in order to determine context modifiers and their values;

2. examines heterogeneities between these contexts and establishes how data are converted using rules;
3. converts data to target context, or generates an error message if the conversion is not possible, and sends results to the appropriate target.

The mediator Web service includes five internal components. The context reader extracts context extensibility attributes from WSDL descriptions. Two repositories for context and domain ontologies respectively identify context structures and domain concepts. The rule engine infers the values of dynamic modifiers and performs data conversion. It communicates with the rule repository that stores the rules for inferring the operations of data conversion and the values of dynamic modifiers.

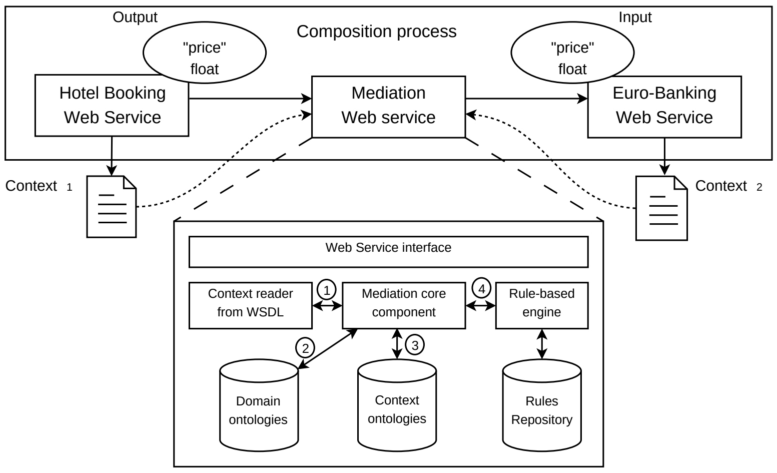


Fig. 3. Detailed View of the Mediator Web service

Figure 3 shows how the mediator Web service performs in the composition of Sect. 2. The numbers in this figure illustrate the chronology of operations that goes along the following description:

1. The mediator Web service generates an in-memory model of both WSDL descriptions and extracts context annotation for each message part concerned with the mediation process, in order to build contexts of parameters.
2. It identifies the first qualified name of each annotation as the concept of the parameter. Then, it checks that the concepts of both parameters match, i.e., that they verify a subsumption or equivalence relation. This is a simple approach to semantic matching but additional capacities can be integrated into the mediator. For a good survey on semantic integration techniques, see Noy's work [12].
3. It accesses the context ontology related to the domain concept matched, and gathers all its relative properties, as well as all its sub properties (i.e.

properties of its sub concepts in the context ontology) into a list of modifiers. With the following WSDL annotation attributes, the mediator affects values to static modifiers. Then, the values of dynamic modifiers are inferred by the rule engine, to build the context description.

4. First, the mediator determines the target context. It corresponds to the context of the banking Web service WS_2 in the example of Sect 2. For each modifier of the target context, the rule engine applies appropriate conversion to the data transmitted, so that the value of the source modifier matches the target context. If the value of the modifier is not convertible to the target context, an exception is thrown, and the mediator Web service returns a fault message. If the value of a modifier is missing, a general rule may affect a default value to this modifier. For example, a rule could set a default scale factor of 1 for prices. If such a rule is absent too, an exception is thrown, and the mediator Web service returns a fault message. If the mediation process is correctly performed, the data is converted into the target context and transmitted to the next Web service.

To operate properly, the rule engine connects to a rule repository. We assume that conversion rules are appropriately maintained, to benefit from advantages of decoupling business logic from the application[4]. For instance, considering our example in Section 2, being given V the values of parameters and SF their scale factors, the rule for managing scale factor modifiers should be stored in the rule repository as follows:

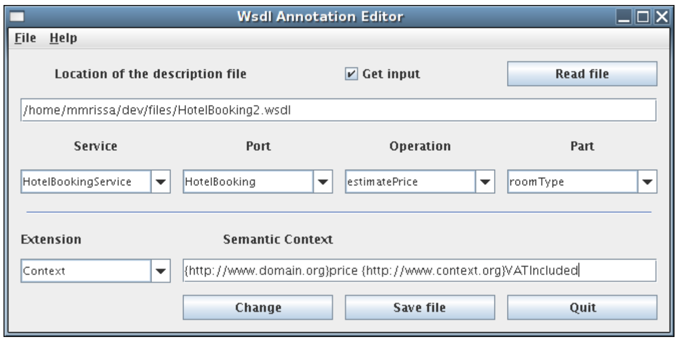
$$V_{target} = \frac{V_{source} * SF_{source}}{SF_{target}}$$

So, at execution time, the rule engine receives as input: “scalefactor”, 1000, 1 and the value v to convert, and performs the conversion to get the appropriate scale factor.

4.3 Implementation

A prototype has been developed as a proof-of-concept of the feasibility of this architecture under the Java^{TM} NetBeans environment. Figure 4 shows a snapshot of our graphical user interface to read/write context annotations from/to WSDL files. This tool enables providers or advanced users to annotate WSDL files with context, so it is possible to compose them with context-aware mediator Web services. We also developed a mediator Web service, that reads context annotation from WSDL files and converts data received from its source context to a target context. Our implementation performs at-runtime context mediation, enabling meaningful execution of composition. In the example of this paper, not only the “price” concepts match, but data is transformed at-runtime, to comply with the different scale factors, heterogeneous date formats (that allow getting up-to-date conversion rates between currencies), and different VAT rates (that also are not always included in the price), described in the context ontology.

[4] Sample rules available at http://www710.univ-lyon1.fr/∼mmrissa/conversion.drl.


Wsdl Annotation Editor

File Help

Location of the description file [✓ Get input] Read file

/home/mmrissa/dev/files/HotelBooking2.wsdl

Service	Port	Operation	Part
HotelBookingService	HotelBooking	estimatePrice	roomType

Extension	Semantic Context
Context	{http://www.domain.org}price {http://www.context.org}VATIncluded

Change Save file Quit

Fig. 4. Screenshot of the WSDL extension editor

Our current composition example is hosted in an Apache Tomcat container (http://tomcat.apache.org/). We also use Jena 2 (http://jena.sourceforge.net/) API and a Drools (http://www.drools.org/) rule engine, to access and manipulate OWL ontologies, infer modifier values and perform data conversion. Our prototype includes domain and context ontologies designed with Protégé (http://protege.stanford.edu/) for describing the “price” concept and context[5].

5 Related Work

This section presents different initiatives that relate to the semantic and mediation aspects of Web services, and to previous work on context description. These related works helped us build ideas, and backed our approach as they are important references of the domain.

Firstly, Semantic Web services constitute an active domain of research. Most approaches rely on ontologies to express the semantics of a domain, however, inserting semantics into Web Services involves using description languages like OWL-S [4], or extending syntactic standards with semantic features (WSDL-S) [6]. OWL-S is a subset of the OWL (ex-DAML) ontology language. It is a general ontology for building semantic Web services, and it was designed to be coupled with standard description formats like WSDL. Inspired from OWL-S, several research projects have been developed, such as ODESWS [13] that models Web services using problem-solving methods. From the DERI laboratory, WSMO [5] is a formal language and ontology that describes varied aspects of semantic Web services. It supports the development and description of semantic Web services and enables mediation as a service, so that it allows maximal decoupling between component Web services. With WSDL-S, Miller et al. annotate WSDL with several extensions related to operations and messages [6]. These extensions refer to concepts of domain models to specify semantics of messages, but also preconditions and effects of operations.

[5] Available at http://www710.univ-lyon1.fr/∼mmrissa/price.owl and http://www710.univ-lyon1.fr/∼mmrissa/PriceContext.owl.

Secondly, mediation between Web services is a hot topic and receives a lot of attention from the research community. Many mediation approaches rely on the concept of mediator for solving data heterogeneities between participants of an interaction. Cabral and Domingue [14] provide a broker-based mediation framework for composing semantic Web services. Their approach follows WSMO conceptual framework [5] that recommends strongly decoupled, service-based mediation. Williams et al. [15] use agents to perform semantic mediation between input and output parameters of Web services by encapsulating the composition into an agent, that controls the developpement of the operation. Spencer et al. [16] present a rule-based approach to semantically match outputs and inputs of Web services. An inference engine analyzes OWL-S descriptions and generates multiple data transformation rules using a description-logic reasoning system.

Thirdly, the use of context has been studied in several domains, in order to improve the adaptability of software applications to different views on information [17]. Some approaches provides formalisms for context representation. In the domain of database interoperability, the Context Interchange approach, firstly introduced by Sciore et al. [11], is based on the notion of semantic value. It has proved to be a highly scalable, extensible and adaptable approach to semantic reconciliation of data. Goh [18] and Firat [19] presented implementations and extensions to this approach. Then, Bornhövd [9] adapted this model to the description of semi-structured data.

While mediation and semantic description of Web services in a composition are very active research fields, to the best of our knowledge, none of these works actually consider the use of explicit context description to solve semantic heterogeneities of data in Web services composition.

6 Conclusion

In this paper, we presented an approach to support the semantic mediation of data exchanged between Web services engaged in a composition. To this end, we first developed a model that leverages data to the level of semantic object, then annotated WSDL descriptions with semantic metadata for capturing contextual information, and finally proposed a context- and rule-based mediation mechanism for Web services composition.

Our future work revolves around different aspects. We envision to automatically integrate mediator Web services into the composition at-runtime, to alleviate the task of composition designers. Also, further study of ontology-based solutions for describing multiples context representations is required. Lastly, we plan to consider the possibility for successful context-based mediation as a criteria of the selection step to improve the selection of Web services.

References

1. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., Winer, D.: Simple object access protocol (SOAP) 1.1. Technical report, The World Wide Web Consortium (W3C) (2000)

2. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1, W3C Note. Technical report, The World Wide Web Consortium (W3C) (2001)
3. UDDI: Universal Description, Discovery, and Integration of Business for the Web. (2001) URL: http://www.uddi.org.
4. Martin, D.L., Paolucci, M., McIlraith, S.A., Burstein, M.H., McDermott, D.V., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.P.: Bringing Semantics to Web Services: The OWL-S Approach. In Cardoso, J., Sheth, A.P., eds.: SWSWPC. Volume 3387 of Lecture Notes in Computer Science., Springer (2004) 26–42
5. Arroyo, S., Stollberg, M.: WSMO Primer. WSMO Deliverable D3.1, DERI Working Draft. Technical report, WSMO (2004) http://www.wsmo.org/2004/d3/d3.1/.
6. Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K.: 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.
7. Gruber, T.: What is an ontology? http://www-ksl.stanford.edu/kst/what-is-an-ontology.html (2000)
8. W3C: XML Schema Part 2: Datatypes Second Edition. Technical report, W3C (2004) http://www.w3.org/TR/xmlschema-2/.
9. Bornhövd, C.: Semantic metadata for the integration of web-based data for electronic commerce. In: Int'l Workshop on E-Commerce and Web-based Information Systems (WECWIS), Santa Clara, CA. (1999) 137–145
10. Schreiber, G., Dean, M.: Owl web ontology language reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/ (2004)
11. Sciore, E., Siegel, M., Rosenthal, A.: Using semantic values to facilitate interoperability among heterogeneous information systems. ACM Trans. Database Syst. **19**(2) (1994) 254–290
12. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. **33**(4) (2004) 65–70
13. Corcho, Ó., Gómez-Pérez, A., Fernández-López, M., Lama, M.: ODE-SWS: A Semantic Web Service Development Environment. In Cruz, I.F., Kashyap, V., Decker, S., Eckstein, R., eds.: SWDB. (2003) 203–216
14. Cabral, L., Domingue, J.: Mediation of Semantic Web Services in IRS-III. In: First International Workshop on Mediation in Semantic Web Services (MEDIATE 2005), Amsterdam, The Netherlands. (December 12th 2005)
15. Williams, A.B., Padmanabhan, A., Blake, M.B.: Experimentation with local consensus ontologies with implications for automated service composition. IEEE Trans. Knowl. Data Eng. **17**(7) (2005) 969–981
16. Spencer, B., Liu, S.: Inferring data transformation rules to integrate semantic web services. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: International Semantic Web Conference. Volume 3298 of Lecture Notes in Computer Science., Springer (2004) 456–470
17. Kwan, M.M., Balasubramanian, P.: Knowledgescope: managing knowledge in context. Decis. Support Syst. **35**(4) (2003) 467–486
18. Goh, C.H., Bressan, S., Madnick, S.E., Siegel, M.: Context interchange: New features and formalisms for the intelligent integration of information. ACM Trans. Inf. Syst. **17**(3) (1999) 270–293
19. Firat, A.: Information Integration Using Contextual Knowledge and Ontology Merging. PhD thesis, Massachusetts Institute of Technology, Sloan School of Management (2003)