Context and Semantic Composition of Web Services

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

[1]Claude Bernard Lyon 1 University, Lyon, France - *firstname.lastname@liris.cnrs.fr*
[2]Zayed University, Dubai, U.A.E - *zakaria.maamar@zu.ac.ae*

Abstract. Composition of Web services is a cornerstone step in the development of interoperable systems. However, Web services still face data-heterogeneity challenges, although several attempts of using semantics. In addition, the context in which Web services evolve is still somehow "ignored", hampering their adaptability to changes in composition situations. In this paper, we argue how context permits to determine the semantics of interfaces that Web services expose to third parties. We show the need for a context- and semantic-based approach for Web services composition.
Keywords. Web service, semantics, context, composition, mediation.

1 Introduction

Web services are now accepted as standards for bridging heterogeneous applications. Web services possess the capability of deploying high-level processes referred to as composite Web services. Composition addresses the situation in which a user request cannot be satisfied by a single Web service, whereas a composite Web service consisting of a combination of Web services (either simple or composite) could satisfy this request [2]. A composition is always associated with a specification that describes amongst others the list of component Web services, their execution chronology, and the corrective strategies in case of exceptions.

Current approaches only achieve Web services composition at the level of message interactions [6]. The standard Web services protocol stack (SOAP, WSDL, UDDI) was not initially built with meeting the requirements of semantic exchange. Composition needs, too, to be conducted at the semantic level. Ignoring or poorly assessing semantics are obstacles to composition since Web services have to be initially checked whether they can effectively work together [8].

Despite multiple attempts, the smooth automation of Web services semantic reconciliation remains a challenge. First, description techniques of the semantic functionalities of Web services are still in their "infancy" stage, despite the tremendous growth in semantic description languages like OWL-S [7]. Second, the context in which Web services evolve is to a certain extent "ignored", which hampers their adaptability to changes in composition situations.

In this paper we aim at *investigating how context drives the semantic reconciliation of Web services*. The rationale of this reconciliation is backed by

Maamar et al., who argue that Web services composition is subject to satisfying two conditions [5]. The first condition is that Web services must agree on the meaning of the exchanged data. And the second condition is that semantic-data conflicts must be automatically resolved using the information that context caters. In this paper we focus on data heterogeneities that arise when Web services from different origins take part in a composition. We propose a language for enhancing Web services descriptions with data semantics. This language is integrated into a *semantic-mediation architecture* that solves information heterogeneities using semantic values and Web services' context.

The role of context in data conflicts of type value is not properly handled in other semantic composition systems. Our proposed solution is built upon Sciore et al.'s work who suggest using the concept of *semantic value* to solve such conflicts [11]. A semantic value is a basic value (i.e., type instance such as like "5" is an "integer" instance), which has a semantic information for facilitating its contextual understanding, e.g., "5(currency=euro)" describes "5" units of currency of type euros. This semantic information will be anchored to the *semantic context* of the Web service.

Section 2 overviews related work on context awareness and semantics of Web services. Section 3 motivates the need for semantics in Web service description and details how WSDL is extended into $\mathcal{SVE}$-WSDL (standing for $\mathcal{S}$emantic-$\mathcal{V}$alue $\mathcal{E}$nhanced-WSDL). Section 4 explains the semantic-based mediation architecture for Web services engaged in composition. The implementation of this architecture is also given in this section. Finally, Section 5 draws our conclusions and overviews different aspects of future work.

2 Related work

2.1 Semantics and Web services

Semantic Web services constitute an active domain of research. There are several ways of inserting semantics into Web Services. One way consists of using description languages like OWL-S [7], and another way consists of extending syntactic standards like WSDL with semantic features [10, 13].

The first way consists of developing languages to semantically describe Web services: OWL-S and WSMO (Web Service Modeling Ontology) [1] currently are the leading languages towards semantic description of Web services. OWL-S is based on the OWL description language and was intended to be combined with syntactic description languages like WSDL. WSMO uses F-Logic to describe the features of Web services. Based on OWL-S, several research projects have been developed for example ODESWS [3] and METEOR-S (LSDIS) [10]. In these systems, ontologies are used to annotate Web services [12], which is useful during discovery, selection, and composition.

Martin et al. introduced the OWL-S language to describe a Web service along a profile, a model, and a grounding [7]. The service profile answers "what does the service do?". The service model answers "how does the service work?".

And the service grounding answers "how to access the service?". Spencer et al. propose a rule-based approach to semantically match outputs and inputs of Web services [14]. An inference engine analyzes OWL-S descriptions and generates multiple data transformation rules using a description-logic reasoning system.

The second way of inserting semantics into Web Services enriches WSDL with the semantics of a domain expressed in an ontology The WSDL-S language [10] from the LSDIS laboratory, extends the WSDL format by adding an "LSDIS-Concept" element to the "part" tag of the WSDL input/output message. This additional element does not itself contain the semantic information, but it rather refers to an element described in an OWL ontology. With the help of the extensibility support of WSDL, files can be extended with semantic information [13].

2.2 Context and Web services

In the area of Web services, context has been recently investigated in many research projects. The main objective of these projects is to facilitate the development and deployment of context-aware and adaptable Web services. Standard Web services descriptions are augmented with context information (e.g., location, time, user profile) and new frameworks are developed to support this. The approach proposed in [9] is intended to provide an enhancement of WSDL language with context aware features. The proposed Context-based Web Service Description Language (CWSDL) adds to WSDL a new part called Context Function which is used to select the best Web service. This function represents the sensitivity of the Web service to the context. Another interesting approach was proposed in [4] to deal with context in Web services. The approach consists of two parts: a context infrastructure and a context type set. The context infrastructure allows context information to be transmitted as a SOAP header-block within the SOAP messages.

3 Semantics and context for Web services composition

3.1 A motivating example

The following simple yet-realistic example discusses why semantic mediation is a key step for Web services composition. Let us assume planning a trip to Japan. The airport of our destination city offers free-of-charge access to the Internet. Promotion fliers distributed to passengers contain some recommended hotels with their HTTP addresses. A hotel, which has attractive rates, provides a Web service interface[1] for booking price estimation (implementation technology transparent to passengers). To check if this hotel is affordable for an European passenger, the following composition of Web services occurs: *HotelBooking* for estimate charges in Yens based on a number of booking nights, and *PersonalEuroBanking* for payment management.

[1] A WSDL interface is the set of functions with their input and output parameters.

This example highlights the importance of semantic reconciliation between both Web services. The semantics of their interfaces needs to be explicitly described for a free-of-conflict composition. Indeed each Web service manages a different currency, so there are two heterogeneous semantic contexts in this example: the first Web service in a "Japanese" semantic context delivers figures in Yens, whereas the second one in an "European" semantic context expects figures in Euros. The composition is not sensitive if the data from the first Web service is directly submitted to the second one without any extra-processing. This results in managing figures (e.g., 100 Yens, 250 Euros) without a real understanding of their appropriateness. This calls for adapting data between services. In this example, currency from *HotelBooking* service needs to be converted so that it matches the currency used by *PersonalEuroBanking* service.

3.2 The core idea of the proposed approach

In a simple composition (Figure 1-(a)), Web services' interfaces are described using WSDL. During Web services interactions, any data conflict of type value is resolved in ad-hoc way at the level of the receiver Web-service. For example, upon reception of a value V in Yens from WS_1, WS_2 explicitly converts it into V' in local currency.

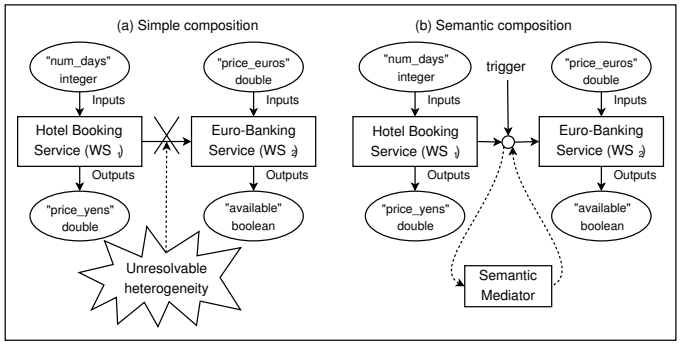

Fig. 1. Simple *vs.* semantic composition

To conduct semantic composition that includes the context of the exchanged data (Figure 1-(b)), the context that defines these data's role must be provided. This context is any metadata that explicitly describes the meaning of the data to be exchanged between Web services. When Web services' inputs and outputs are described using metadata, they can automatically be transformed from one context to another one during Web services execution. By automatically, we mean that conversion function is not embedded into the body of any Web service. This function is loosely-attached to Web services (i.e., independent), and becomes, thus, an active component that intervenes during Web services composition and execution. A possible solution to achieve a semantic composition of Web services is built upon the semantic-value concept.

3.3 Semantic value concept

Sciore et al. introduce the notion of semantic value to describe a value (i.e., type instance) with additional semantic information (its context) [11]. Formally, a semantic value A is defined as follows: $A = a(p_1 = a_1, ...p_i = a_i, ..., p_n = a_n)$ where a is a simple value, p_i a property, and a_i a simple or a semantic value.

While a simple value is simply defined as an instance of a type (e.g., 5 as integer), a semantic value associates a value with a context in which its interpretation happens. For example, 5(context=(currency, euro)) is a semantic value that defines 5 as a currency in euros. Recursive descriptions is also possible since the value of a property can also have its own context, e.g. 5(currency, euro (context=(scalefactor,1000))) is a recursive description of the semantic value 5.

It is important to note that the values "5" and "50" can be considered as different with each other if considered as simple values. However, they can be considered as equivalent if they are interpreted as semantic values with different contexts, e.g., 5(currency, euro) = 50(currency, yen) when 1 euro worths 10 yens.

Then, the main purpose of enabling semantic mediation between Web services consists of associating context to web services and using conversion functions to convert a semantic value from the context of a Web service to another. The semantic value 5(currency, euro) can be converted into another semantic value 50(currency, yen). Depending on the conversion to adopt, a conversion function can be lossless or lossy (like for compression/decompression), total or non-total (for example city to country is not reversible), and order preserving or not (if a $<$ b then cvt(a) $>$ cvt(b) where cvt is a function).

3.4 From WSDL to $\mathcal{SVE}$-WSDL

To develop an enriched semantic description of a Web service, we investigated how the representation of some WSDL elements could be extended. Each representation would be associated with some specific information that depends for example on the ontology that the Web service binds to. To add more semantics to Web services' descriptions, we proposed $\mathcal{SVE}$-WSDL as an extension of WSDL. This extension adds the concept of semantic value to a WSDL description.

To keep the paper self-contained, we overview the main structure of a WSDL document: `<type>` element defines data types exchanged in messages. `<message>` element describes the data elements of an operation. Each message consists of one or more parts. Each part has a `<name>` element, a `<type>` element, and may consist of additional `<element>` elements. These three constructs define the names and types of the parameters that are used by the Web service in the message. For platform independence requirement, WSDL uses XML syntax to define data types. Some message parts may be of type complex, and may contain a structure of several XML Schema elements. `<portType>` element provides an abstract description of the Web service. It defines the operations (using one or more `<operation>` elements) that can be performed over a Web service, and the messages involved. Each operation may contain one `<input>`, one `<output>`, and one `<fault>` sections. Each section contains a `<message>` element that specifies

what the Web service receives (input message) and sends when its execution fails (fault message) or succeeds (output message). Finally, `<binding>` element defines the message format and protocol details for each port. Several protocols may be used such as SOAP over HTTP.

Our proposal for $\mathcal{SVE}$-WSDL is to bind a semantic value description to the inputs and outputs of a Web service. The extension of the WSDL metamodel is partially shown in Figure 2. It is now enhanced with context meta-class associated with part meta-class. Input and output elements contain a single message,

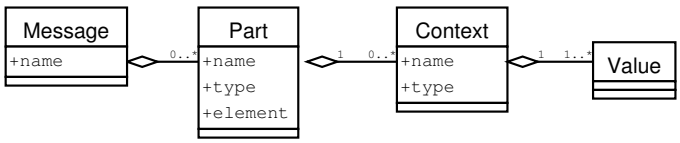

Fig. 2. Proposed extension of the WSDL metamodel

that consists of one or several parts. In fact, we aim at extending each part of a message with an additional element to be referred to as "context". Compared to the concept of semantic values of Sciore et al. [11], `<part>` tag describes the simple value and `<context>` structure represents the semantics associated with this value. As a result, we define a structure to describe the context to be associated with the values described in `<part>` elements of a Web service's messages. This structure consists of: (i) a `<name>` element that labels the semantic information; (ii) a `<type>` element that indicates the type of the semantic information; (iii) a `<value>` element that contains the information itself; and an optional `<context>` element providing additional context information to the `<value>` element, thus enabling recursive descriptions. Listing 1.1 illustrates a part of a $\mathcal{SVE}$-WSDL description of a Web service.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HotelService" tns="http://example.HotelService/"...>
    <message name="HotelServiceEndpoint-Booking">
        <part name="Num-Days" type="xsd:integer">
            <context name="time" type="xsd:string" value="time-unit">
                <context name="duration" type="xsd:string" value="day"/>
            </context>
        </part>
    </message>
    <message name="HotelServiceEndpoint-BookingResponse">
        <part name="price" type="xsd:int">
            <context name="currency" type="xsd:string" value="yen"/>
            <context name="scalefactor" type="xsd:string" value="1"/>
        </part>
    </message>
    <portType name="HotelServiceEndpoint">
        <operation name="Booking" parameterOrder="Num-Days">
            <input message="tns:HotelServiceEndpoint-Booking"/>
            <output message="tns:HotelServiceEndpoint-BookingResponse"/>
        </operation>
    </portType>
...
</definitions>
```

Listing 1.1. $\mathcal{SVE}$-WSDL description of a Web service

4 Semantic-based mediation for Web services

4.1 Components of the architecture

Figure 3 presents the architecture for context- and semantic-based mediation of Web services composition. It consists of six components:

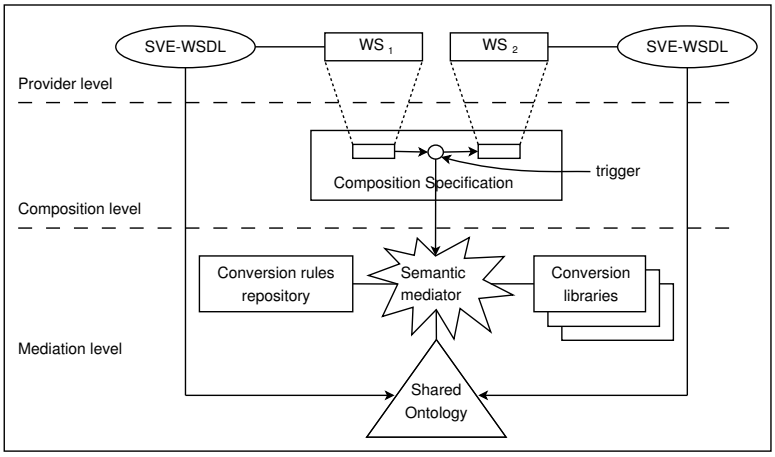

Fig. 3. $\mathcal{SVE}$-WSDL mediation architecture

$\mathcal{SVE}$-WSDL annotations. They extend WSDL descriptions of Web services by using semantics. Web services can now "understand" the values they exchange. Interesting to note that Web services still exchange simple values during the execution, so that the interactions with Web services during invocation remain unchanged. These values are more understandable since their semantics in terms of meta-data can be extracted from $\mathcal{SVE}$-WSDL descriptions. The Web service's provider does not have to undertake any changes, except of adhering their WSDL description to the $\mathcal{SVE}$-WSDL format.

Value triggers. The data flow during composition needs to be intercepted at some points between component Web services. To keep Figure 3 clear, triggers are not represented. A trigger intervenes between Web services, extracts data from SOAP messages, and sends a conversion request to the semantic mediator. Then the trigger waits for an answer from the mediator and, depending on the answer it receives (or a timeout), either forwards converted data to the next Web service(s), or throws an exception. Still remains the question about where to physically place the triggers. For scalability reasons, we recommended that triggers should be along with the specification execution program, most of the times on the client side. An example of implementation is to use the BPELJ language for specifying the composition, and to integrate the triggers under the form of code snippets that are located between composed Web services. Indeed,

as shown (Figure 1-(b)), the trigger intercepts the values exchanged during composition and passes them on to the mediator where data heterogeneities remain unresolved in a simple composition. Thus, triggers do not have to know that they are part of an architecture that deals with semantic values. This separation between the actual mediation process and the composition specification allows a better scalability and adaptability of the architecture to future modifications.

Semantic mediator. It works in close collaboration with the triggers by receiving their outputs. The semantic mediator is a service (either local service or Web service). It acts as a listener that gets a semantic value from a source context and returns, if possible, the conversion of this value to a target context. To this end, the semantic mediator is supported in its work with a conversion rules repository while searching for the appropriate conversion functions. The mediator delivers the resulting basic value to the trigger that sent out the request, or returns an exception in case of non semantic comparability (i.e. the conversion is not meaningful). The semantic comparability of two semantic values is detailed in Sciore's work [11].

Conversion rules repository. It contains the mappings between terms used in various semantic contexts and lists as well the conversion opportunities between these terms. It is expected that the repository can refer to several conversion libraries including remote ones. New conversion libraries can be added upon request, i.e., each time a composition requires new conversion functions for a new application domain.

Conversion libraries. Conversion functions are stored in libraries and permit converting semantic values from one context into another.

Application domain ontologies. The mediator refers back to these ontologies to check whether or not it is possible to clearly identify a semantic value. We do not discuss here the mechanisms that could help the mediator identify a term in an ontology, as we focus on the enhancement of the process of mediation with semantic values. This point is to be considered in future work. For the moment, let us consider that the identification of the terms is performed by a simple pattern-matching technique. For instance, in the domain of currencies, providers may differently name the same concept (devise, currency, moneyUnit, etc.) and its instances (USD, Dollar, USDollars, etc.). The ontologies help the mediator find out what similarities and inconsistencies are between semantic value descriptions.

4.2 Architecture operation through the prototype

A prototype that supports the proposed architecture is fully operational. We use Java^{TM} NetBeans environment. Figure 4 is the GUI to read/write *context* annotations from/to WSDL files.

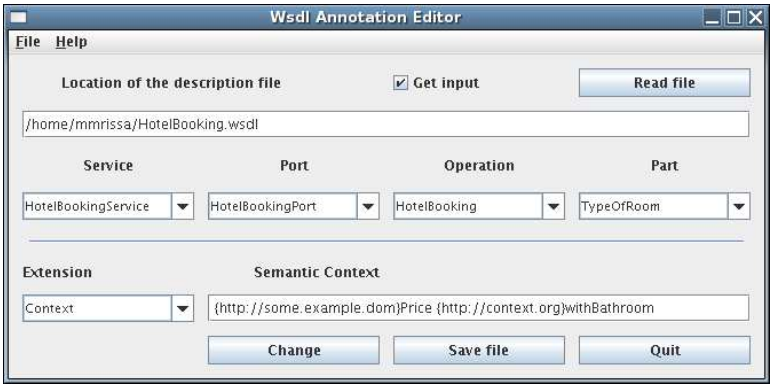

Fig. 4. Screenshot of the WSDL extension editor

As part of our implementation efforts, we developed the context-aware mediation architecture for Web services. This consists of reading context annotation from WSDL files and converting data received from a source context to a target context. This mediation is part of the example given in Section 3.1. We deployed the "HotelBooking", "Mediation" and "PersonalEuroBanking" Web services on an Apache Axis server and composed them using BPEL. The implementation shows that dynamic and accurate interpretation of context enable a meaningful composition to be performed. With the context-aware mediation Web service and annotated WSDL files, not only the "price" semantic types match, but data is transformed at the execution stage, 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) that form the *context* of data.

5 Conclusion

In this paper we presented an approach that supports semantic interoperability among individual Web services engaged in composition. To this end we enhanced WSDL descriptions of Web services with semantic metadata for capturing contextual information. We also proposed an architecture that exploits these ontological annotations for adapting data on the fly when they are transferred from Web service to another constitutive one. The heart of our architecture is the context mediator; which adapts exchanged data so as to be compliant with the receiver's requirements.

Our future work revolves around different aspects. First, the use of context should be considered, so that applications can become aware of their environment. Second, different local ontologies can be used when providers develop Web services, so there is a need to generate contextual mappings between ontologies in an automatic way.

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. J. Bézivin, S. Hammoudi, D. Lopes, and F. Jouault. Applying MDA Approach for Web Service Platform. In *Proceedings of The IEEE Enterprise Distributed Object Computing Conference (EDOC'2004)*, Monterey, California, 2004.
3. Ó. Corcho, A. Gómez-Pérez, M. Fernández-López, and M. Lama. ODE-SWS: A Semantic Web Service Development Environment. In *Proceedings of The first International Workshop on Semantic Web and Databases (SWDB'2003)*, Berlin, Germany, 2003.
4. M. Keidl and A. Kemper. A framework for context-aware adaptable web services. In *Proceedings of The 9th International Conference on Extending Database Technology (EDBT'2004)*, Heraklion, Greece, 2004.
5. Z. Maamar, D. Benslimane, and N. C. Narendra. What Can Context do for Web Services? *Communications of the ACM*, 2006 (forthcoming).
6. Z. Maamar, N. C. Narendra, and S. Sattanathan. Towards an Ontology-based Approach for Specifying and Securing Web Services. *Journal of Information & Software Technology, Elsevier Science Publisher*, 2006 (forthcoming).
7. D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. abou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *Proceedings of The First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'2004)*, San Diego, California, USA, 2004.
8. B. Medjahed, A. Rezgui, A. Bouguettaya, and M. Ouzzani. Infrastructure for E-Government Web Services. *IEEE Internet Computing*, 7(1), 2003.
9. Soraya Kouadri Mostéfaoui and Béat Hirsbrunner. Towards a context-based service composition framework. In Liang-Jie Zhang, editor, *ICWS*, pages 42–45, Las Vegas, Nevada, USA, 2003.
10. P. Rajasekaran, J. A. Miller, K. Verma, and A. P. Sheth. Enhancing Web Services Description and Discovery to Facilitate Composition. In *Proceedings of The First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'2004)*, San Diego, California, USA, 2004.
11. E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2), 1994.
12. A. P. Sheth and C. Ramakrishnan. Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis. *IEEE Data Engineering Bulletin*, 26(4), 2003.
13. K. Sivashanmugam, K. Verma, A. P. Sheth, and J. A. Miller. Adding Semantics to Web Services Standards. In *Proceedings of The International Conference on Web Services (ICWS'2003)*, Las Vegas, Nevada, USA, 2003.
14. B. Spencer and S. Liu. Inferring Data Transformation Rules to Integrate Semantic Web Services. In *Proceedings of The International Semantic Web Conference (ISWC'2004)*, Hiroshima, Japan, 2004.