Building Semantic Mashup

Abdelhamid Malki, Sidi Mohammed Benslimane

EEDIS Laboratory , University of Djilali Liabes , Sidi Bel Abbes, Algérie

abdelhamid.malki@gmail,Benslimane@univ-sba.dz

Abstract. Mashups allowed a significant advance in the automation of interactions between applications and Web resources. In particular, the combination of Web APIs is seen as a strength, which can meet the complex needs by combining the functionality and data from multiple services within a single Mashup application. Automating the process of building Mashup based mainly on the Semantics Web APIs facilitate to the developer their selection and matching. In this paper, we propose SAWADL (Semantic Annotation for Web Application Description Language), an extension of the WADL language that allows the semantization of the REST Web Service. We introduce a reference architecture with five layers representing the main functional blocks for annotating and combining web APIs, and therefore make the engineering process of Mashup applications more agile and more flexible.

Keywords: Semantic Mashup, Matching, API, SOAP, REST, SAWADL, SAWSDL.

1 Introduction

Dynamics, agility and efficiency are concepts of the future. The World Wide Web is undergoing an evolution from a static environment to a dynamic world in which mashups will play a central role. The Mashups are web applications developed by the combination of data, business logic, and/or user interfaces of web sources published and reused via APIs [8]. Thus, Mashups are designed to reduce the cost and development time of web applications.

Despite these advantages, engineering of Mashups applications requires the intervention of the developer which needs not only programming skills but also to understand the structure and semantics of APIs that wants to integrate. Currently, several tools Mashup (e.g. IBM WebSphere[1], Yahoo-pipes[2], etc.) are used by end-users (i.e. with less programming skills) to facilitate the building of Mashup applications. However, the intervention of the professional developer is required when the application Mashup is complex, thing that has prompted researchers to find effective solutions for creating Mashups, So that end users can build an application with a tool Mashup that guarantees the discovery, selection, and automatic or dynamic superposition of APIs

[1]http://www-01.ibm.com/software/webservers/

[2]http://pipes.yahoo.com/pipes/

based on the semantic approach, the so-called ‘‘Semantic Mashups’’. The semantic Mashups is a Mashup whose combined APIs are supported (or annotated) by a semantic layer that allows to select and compose them in an automatic way (unambiguous).

We propose in this work SAWADL, a novel language for the semantization of REST web services [1]. SAWADL uses WADL[3] description to enrich RESTful APIs with a semantic layer that allows the discovery and automatic superposition of APIs in order to automatically build Mashup applications. SAWADL is more flexible and adaptive with respect to other approaches of semantization such as SAWSDL [2] which is used to annotate the WSDL[4] description of SOAP web services with ontological concepts.

The rest of the paper is organized as follows. Section 2 presents briefly the semantic Mashup, and presents some related work for the semantization of REST web services. In Section 3, we introduce SAWADL, a semantic annotation language for REST APIs. Our approach to build Semantic Mashup is described in Section 4. Finally we conclude and give some perspectives in Section 5.

2 Related Works

Web services enable applications to call remote procedures and to exchange data by passing well-defined messages. This can easily be used for Mashup application as a way to orchestrate different web applications. For instance, *Amazon Web Service*[5] allows users to access most of the features of Amazon.com by using SOAP-based web services and REST-based web services. The semantic Mashup is Mashup whose combined APIs are annotated by a semantic layer that allows to select and compose them in an automatic way. In order to build an automatic Mashup, it is necessary to semanticize these APIs.

For SOAP-based Web services there are two types of semantization approaches. The first (service ontology) consists of developing a complete language that describes Web services and their semantics in a single block (e.g. OWL-S, WSMO, etc.). The second approach (semantic annotation) consists of annotating existing web services with semantic information. WSDL-S, SAWSDL used to manually annotate a WSDL description with elements referring to ontologies.

As SOAP-based Web services, semantic REST-based Web services can be classified in two approaches. The first approach consists of developing an ontology that describes the REST-based Web services and their semantics in a single block. The second approach consists of annotating existing languages with semantic information. In the following, we present different propositions for the second approach.

- ***SOOWL-S advertisements (a social-oriented version of OWL-S advertisements)***

The SOOWL-S advertisements [6] proposes an extension of the OWL-S ontology in order to semanticizes the different types of APIs (e.g. SOAP, REST, JS, RSS, etc.) used in the construction of Mashup applications.

SOOWL-S ontology annotates just the I/O parameters and non-functional properties of a Web service (using the service-Profile module of the OWL-S ontology).

[3] http://www.w3.org/Submission/wadl/
[4] http://www.w3.org/TR/wsdl
[5] http://aws.amazon.com/

Thus, SOOWL-S ontology allows searching and automatic selection of APIs, but not their combination owing to the absence of the extension of service-Model module of the ontology OWL-S.

- ***SA-REST(semantic annotation for REST)***
  According to J. Lathem [4], most of RESTful web services use HTML pages to describe to users what the service does and how to invoke it. However, HTML is designed to be human legible but not machine readable. In order to solve this problem, [4] have used the RDFa micro formats[10] which allows the integration of RDF triples above HTML description in order to add semantics to REST service and make it visible and interpretable by the machine.

- ***SWEET (Semantic Web Services Editing Tool)***
  Maleshkova et al [5] propose an integrated approach to formally describe the semantics of RESTful web services. The approach enables both the creation of machine-readable RESTful service descriptions using the hRESTS (HTML for RESTful Services) Microformat [3], and the addition of semantic annotations by the MicroWSMO Microformat[6], in order to better support discovering services, creating mashups, and invoking them.

Table 1. shows a comparison between the different approaches of semantics REST web services.

Table 1. Comparison between the different approaches of semantics REST Web services

	SOOWL-S	**SA-REST**	**SWEET**
Type of semantization	Service Ontology	Annotation	Annotation
Publication of services	+	-	+/-
Discovery of services	+	+	+
Combinaton of services	-	+	+
Annotated description	Absent, is a Service Ontology	HTML	HREST
Type of accepted ontology	Owl	All	All
Type of API semantized	SOAP, REST, RSS	REST	REST

3 SAWADL

In this section we propose an annotation language that allows the semantization of RESTful web services to strengthen the selection and superposition of these services in Mashups applications.
SAWADL, the extension of WADL language that we propose is part of those approaches that add semantic annotations above the service description while most approaches are based on a semantic annotation above a description based on HTML which gives less homogeneity between semanticized REST web services. SAWADL does not specify a language for representation of semantic models. Rather, it provides mechanisms for referencing ontological concepts defined in the external of WADL document.

[6] http://www.w3.org/TR/rdfa-syntax/

The methods of annotation in SAWADL are summarized in two mechanisms: `modelReference` and `SchemesMapping`. This is done by the attribute "`sawadl`" followed by the appropriate extension.

`ModelReference` attribute used to associate a WADL's component to a concept of a semantic model. The items annotated a REST web service described by WADL description are the methods (`<method id="method1" name="GET">`) and parameters of input / output (`<param name = "name" type="xsdtype"/>`) of the service. The semantic concept (ontological) associated to elements of WADL through the `modelReference` attribute is represented by zero or more URLs separated by spaces, which are references to ontological concepts.

The mechanism of `schemesMapping` is achieved through two other attributes `liftingSchemesMapping` and `loweringSchemesMapping`. These attributes are used to specify the mappings between semantic data and WADL elements. The mechanism of `schemaMapping` is very interesting to understand. In fact, we employ the `loweringSchemesMapping` attribute when an element annotated in the WADL description matches more than one ontological concepts, and the URIs of the `loweringSchemesMapping` attribute point to files containing SPARQL[7] queries and XSLT[8] transformations. While we use `liftingSchemesMapping` when several elements annotated in the WADL description represent a single ontological concept, and URIs can point to files that contain XQuery[9] queries or XSLT transformations.

3.1 Annotation of methods

SAWADL provides mechanisms to annotate methods in a WADL documents. To illustrate these mechanisms, we use a domain ontology of tourism `TravelOnto` (which we implemented in OWL) to annotate the `BookFlight` operation of Flight API. Although traditionally the inputs and outputs provide an intuitive semantics of an operation, a simple semantic annotation can be helpful. Thus we will annotate the `BookFlight` operation by associating through the `modelReference` attribute with a `BookFlight` concept in the `TravelOnto` ontology (Figure.1).

3.2 Annotation of Inputs/Outputs parameters

In SAWADL, the Input/Output parameters annotation is done in two different ways:

Internal Annotation. This annotation consists in associating each parameter input/output "`<param…>`"of a method to a concept in an ontology. This supposes that for each parameter input/output of a method there exist a corresponding concept in the ontology. For example, the input of the operation `BookFlight` is composed of name and age of the passenger, and the number and class of Flight. We suppose that for each attribute, there exists a concept that corresponds to it in the `TravelOnto` ontology. In the case where there is no match, the semantics of the input/output parameters is not specified. Figure 2 show an example of internal annotation.

[7] http://www.w3.org/TR/rdf-sparql-query/

[8] http://www.w3.org/TR/xslt

[9] http://www.w3.org/TR/xquery

```
1 <resources base="http://localhost/ApiRest/Flight"
2 <method id="BookFlight" name="GET"
3 sawadl:modelReference=TravelOnto#BookFlight">
4 <request>
5 <param name="la_Name" type="xsd:string"/>
6 <param name="fi_name" type="xsd:string"/>
7 <param name="Age" type="xsd:int"/>……
```

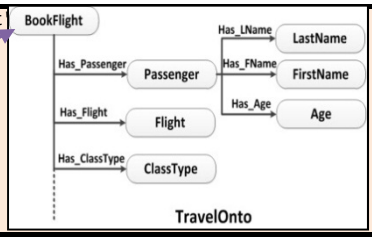

Fig. 1. Annotation of methods with SAWADL

```
1 <resources base=" http://localhost/ApiRest/Flight ">
2 <method id="BookFlight" name="GET">
3 <request>
4 <param name="la_name"  type="xsd:string" sawadl:modelReference=”TravelOnto#LastName"/>
5 <param name="fi_name" type="xsd:string" sawadl:modelReference="TravelOnto#FirstName"/>
6 <param name="Age"  type="xsd:int" sawadl:modelReference=”TravelOnto#Age"/>
7 <param name="NFlight" type="xsd:string" sawadl:modelReference=” TravelOnto#Flight" />
8 <param name="Class" type="xsd:String" sawadl:modelReference=”TravelOnto#ClassType"/>
9 </request>…
```

Fig. 2. Internal Annotation

External Annotation. In this case, the parameters are annotated globally via the tag ‘‘`<request>`’’, however, it must create a schémaMaping for specifying the transformation rules between the input’s/output’s parameters and the domain ontology.

As an illustration, we take the example of `credit card` defines in WADL and the OWL ontology `TravelOnto` (see Figure 3). In this ontology there is no individual correspondence for the two attributes `last_name` and `first_name`. However, the `Owner` concept of ontology is the merger of these two attributes. To establish the correspondence between the inputs of the `credit card` API and `CreditCard` concept, it must first associate using `sawadl:modleReference` and then define a transformation scheme using an XSL style sheet via the attribute `sawadl:liftingSchemaMapping` (see Figure 4).

```
1 <resources base="http://localhost/ApiRest/Billet">
2 <method id="buyBilletOp " name="GET">
3 <request sawadl:modelReference=TravelOnto#CreditCard"
4 sawadl:liftingSchemaMapping=http://localhost/Mapping.xsl/>
6 <param name="La_name"  type="xsd:string"/>
7 <param name="Fi_name" type="xsd:string/>
8 <param name="Number"     type="xsd:int"/>
9 <param name="Type"       type="xsd:string"/>
10<param name="ExpirDate"  type="xsd:date"/>
```

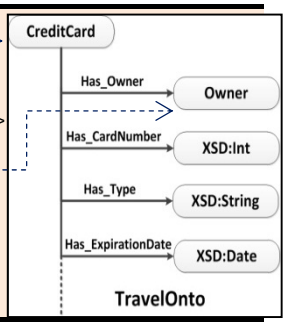

Fig. 3. External Annotation

```
<xsl:transform version="2.0"
   xmlns:Travel=http://localhost/ApiRest/Billet#  xmlns:TravelOnto="http://localhost/TavelOnto#>
   <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"/>
   <xsl:template match="/">
    <rdf:RDF>
     <TravelOnto:CreditCard>
      <hasOwner  rdf:resource="#Owner">
       <xsl:value-of select="concat(Travel:./param[@name='La_name'],Travel:./param[@name='fi_name'])"/>
      </hasOwner>
      <hasCardNumber rdf:datatype="xs:Int"><xsl:value-of select="Travel:./param[@name='Number']">
      </hasCardNumber>
      <hasType rdf:datatype="xs:string"> <xsl:value-of select="Travel:./param[@name='Type']">
      </hasType>
      <hasExpritionDate rdf:datatype="xs:Date"> <xsl:value-of select="Travel:./param[@name='ExpirDate']">
      </hasExpritiondate>
     </TravelOnto:CreditCard>
    </rdf:RDF>
   </xsl:template>
</xsl:transform>
```

Fig.4. XSL style sheet via the attribute sawadl: lifting Schema Mapping

4 Building semantic Mashup

The construction of automatic Mashups necessarily requires a semantic layer on top of APIs (web services). As the dynamic composition of standard web services, the semantic Mashup allows a more rapid development and transparent composition to the user. But unlike to that of traditional web services, the Mashups are composed of APIs of different nature which makes their combination process more difficult.

Figure 5 shows reference architecture for Semantic Mashup. This architecture consists of five layers. The layers represent the main functional blocks for automatic generation of Mashup. The ontology is used to enrich the engineering process by a semantic layer that allows him an automatic selection and a combination of APIs included in the Mashup application.

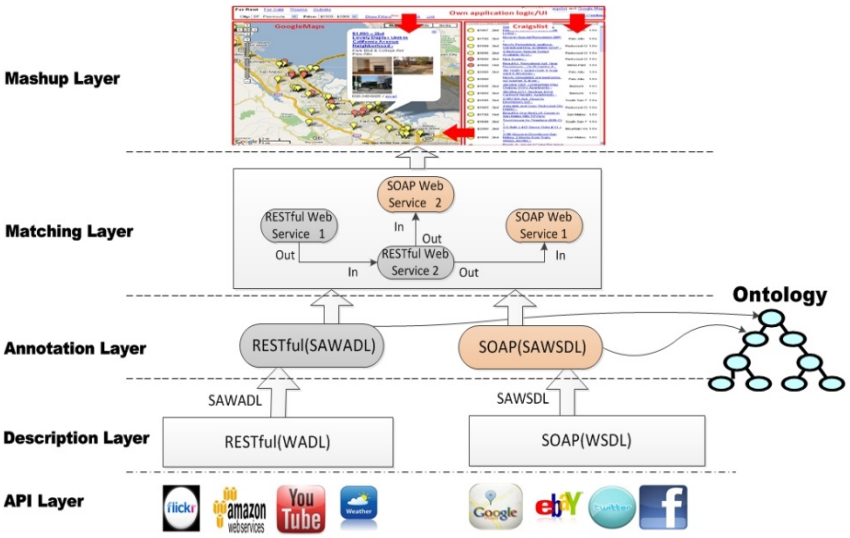

Fig. 5. Reference architecture of a Semantic Mashup .

4.1 API Layer:

At this level several types of APIs are concerned. In particular APIs based on SOAP and RESTful which are the most widely used in engineering applications Mashups.

4.2 Description Layer:

At this layer, WADL and WSDL languages are used respectively to describe SOAP and REST APIs.

4.3 Annotation Layer:

In addition to the SAWADL language that we propose in this paper, several languages of web services annotation are considered at this level. In particular, SAWSDL which is used to semanticize SOAP-based web services by annotating the input/output of WSDL file with ontological concepts. This layer will be used in the automatic construction of Mashups, by allowing discovery, selection and combination of unambiguous way of the various APIs.

4.4 Matching Layer:

The heterogeneities between different annotation languages are resolved at this layer. In the following, we propose four rules to match SAWSDL and SAWADL annotation languages.

Rules1. A method described by the tag "<method>" of a resource or a sub-resource "<resource>" of a SAWADL file corresponds to an operation described by the tag "<operation>" of a SAWSDL file.

Rules2. An input described by the tag "<param>" for a set of inputs "<request>" of a SAWADL file corresponds to an entry described in the web service's XML schema by the tag "<element> " of a "<complexType>" of an operation's Input described in SAWSDL file.

Rules3. An output described by the tag "<response>" of a SAWADL file corresponds to an output described in the web service's XML schema by the tag "<element>" of a "<complexType>" of an operation's output described in SAWSDL file.

Rules4. The "modelReference", "liftingSchemaMapping", "loweringSchemaMappin" attributes of a SAWADL file correspond to the "modelReference", "liftingSchemaMapping", "loweringSchemaMapping" attributes of a SAWSDL file.

Correspondences between APIs are established based on of semantic similarity [7] which allows calculating a distance between the ontological concepts of Input/Output. This distance will be compared with a predefined threshold in order to know if an API could be combined with another or not.

The matching score between a pair of matching services $\boldsymbol{S_{in}}$ and $\boldsymbol{S_{out}}$ is calculated using the following formula:

$$\boldsymbol{Match(S_{in}, S_{out}) = 2 * hi * \sum (1 - dist(i,j)) / (n_{in} + n_{out})}$$

Where $\boldsymbol{n_{in}}$ is the number of query attributes of the service $\boldsymbol{S_{in}}$ And $\boldsymbol{n_{out}}$ is the number of annotated attributes present in service $\boldsymbol{S_{out}}$, $\boldsymbol{hi}$ is the number of annotated attributes of services $\boldsymbol{S_{out}}$ that have been matched out of $\boldsymbol{n_{in}}$, and finally $\boldsymbol{dist(i,j)}$ is the ontological distance score between the $\boldsymbol{j}^{th}$ term in service $\boldsymbol{S_{out}}$ and a corresponding query term.

4.5 Mashup Layer

At this layer, an application mashup is really created based on the results obtained by matching layer. The Mashup layer integrates APIs that have a matching value greater than or equal to a threshold predefined by domain experts. The combination of APIs can be made using different technologies (e.g. Ajax, PHP, JSP, etc.).

5 Conclusion and perspective

The Mashups are web applications developed by combining data, business processes, and/or user interfaces of web sources published and reused via APIs. Thus, Mashups aimed at reducing the cost and development time of web applications. However in order to address the shortcomings of existing languages and protocols established by the IT community, we saw that the work related to engineering the Mashups applications are particularly oriented towards the semantic level.

The aim through the use of semantics is to enable machines to interpret the processed data and seize their significance in an automatic way in order to automate the selection and combination of APIs used to build the Mashup application.

Many languages and semantic annotations have been proposed for the semantic description of RESTful APIs. However, they did not give a great success and are not simple to implement. For example, SA-REST and SWEET approaches require an HTML web page that describes the API and that will be later transformed into a machine readable description to add semantic annotations. One thing that is not always true and that makes it more difficult especially if the REST API does not have a web page that describes it. In order to respond to these problems, that we conducted our research. Our work focuses on the semantics, and more particularly towards a proposal for an annotation language for semantic REST Web services. Our language SAWADL is one of the approaches that add semantic annotations on top of the service description. Unlike approaches that annotate on top of an HTML description, we use the WADL description which is used to describe syntactically REST web services. Semantization APIs is not sufficient to design and implement an automatic Mashup. Thus a process of matching is necessary to find correspondences between the different APIs, and to discover automatically the Mashable components followed the needs of users.

Finally, several perspectives can be considered in order to contribute more to the agility and flexibility of the semantic Mashup building. We cite as an examples:

- The Semantization of other Web APIs such as javascript or RSS / ATOM that represent Mashable components widely used in the development of Mashups. However, the absence of a structured and modular description of these APIs makes this task a big challenge.
- The use of ontological resource and service like OWL-S and WSMO.
- The use of the semantic approach in the construction of process-oriented enterprise Mashups that allows a user to automate her tasks.

REFERENCES

[1] R.Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD thesis, University of California, 2000.

[2] J.Kopecký, T.Vitvar, C.Bournez, J.Farrell: SAWSDL: Semantic Annotations for WSDL and XML Schema, IEEE Internet Computing, vol. 11, no. 6, pp. 60–70, November-December 2007.

[3] J.Kopecky , K.Gomadam, T.Vitvar: hRESTS: an HTML Microformat for Describing RESTful Web Services , Proceedings of the 2008 IEEE/WIC/ACM Inter-national Conference on Web Intelligence (WI-08), 2008.

[4] J.Lathem, K.Gomadam, P. Sheth; SA-REST and (S)mashups Adding Semantics to RESTful Services , Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), September 17-19, 2007, Irvine, California, USA. IEEE Computer Society 2007.

[5] M.Maleshkova, C.Pedrinaci, J.Domingue, Supporting the Creation of Semantic RESTful Service Descriptions, 2009, In: 8th International Semantic Web Conference (ISWC 2009), 25-29 Oct 2009, Washington D.C., USA.

[6] G.Meditskos, N. Bassiliades , A combinatory framework of Web 2.0 mashup tools, OWL-S and UDDI, Expert Systems with Applications, vol. 38, no. 6, pp. 6657–6668,June 2011.

[7]H.Ngu Anne, P.Carlson Michael, Z.Quan Sheng. Semantic-based Mashup of Composite Applications, IEEE Internet Computing, vol. 3, no. 1, pp. 2–15,January-March 2010.

[8] J.Yu, B.Benatallah, F.Casati, F.Daniel. Understanding Mashup Development and its Differences with Traditional Integration, , IEEE Internet Computing, vol. 12, no. 5, pp. 44–52,September-October 2008.