Towards a Cloud Service Standardization to ensure interoperability in heterogeneous Cloud based environment

Majda Elhozmari and Ahmed Ettalbi

Models and Systems Engineering Team, SIME Laboratory
ENSIAS, Mohammed V University Rabat, Morocco

Summary
During the last years, because of the attractive advantages given by Cloud providers to consumers, many companies outsource their data or Information Technology (IT) system to the Cloud based technology. Many Internet protocols for service access standards exist and are used to ensure communication between user's browser and Web server, such as Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). The most of Cloud providers use REST as an interface Web service because of its advantages in the Web environment. Contrariwise the SOAP Web service is used in several areas such as education and industrial fields. These client areas had a limited choice of Cloud provider that can respond to their needs, because the number of Cloud providers using SOAP Web service is restricted and each service has its specific characteristics. As a solution, we propose a Cloud SaaS middleware based on two interfaces and an internal component. This Cloud enables communication between Cloud consumer using SOAP Web service and Cloud provider using REST Web service. This is to achieve benefits such as standardization and interoperability and minimizing lock-in of Cloud providers. This paper motivates Cloud SaaS architectures and presents the architecture of the Cloud service standardization.

Key words:
Cloud Computing; Heterogeneous Cloud; SOAP; REST; Interoperability.

1. Introduction

Traditional business applications have always been very complicated and expensive. Nowadays Cloud Computing technology is easier and quicker to integrate with enterprise applications. It can help to eliminate problems of managing hardware and software, and responds to customer needs with less cost.

The Cloud exists in several forms: private Cloud when the Cloud infrastructure is operated solely for an organization, community cloud when the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns, public Cloud when the Cloud infrastructure is made available to the general public or a large industry group. There is a classification of services that can be found in the Cloud: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). The most popular type of service is SaaS, which is to make available a Web application ready for use and can be used directly as soon as the payment has been confirmed. The popularity of these solutions comes from their very low cost of entry and rapid provision users. In addition, these solutions are generally less expensive than traditional applications and do not require technical personnel for maintenance because consumer is not forced to make updated.

Many Cloud Computing applications exist, and every Cloud provider has a specific architecture and Web service interface used to communicate with consumers. For example, Azure use REST as an interface style. In educational and industrial applications SOAP is most used as a Web service interface. Nowadays, the mainstream of website applications is based on REST Web services. In this heterogeneous environment, standardization was and still a big problem in the business-to-business and also in business-to-consumer segment, essentially in service access interfaces Protocol.

In this paper, we focus on interaction between Cloud providers using REST Web services and consumers using SOAP as a Web services technology. We propose a global solution to guarantee interoperability and standardization in heterogeneous Cloud based environments between Cloud providers and consumers. This solution aims maximizing the level of interoperability and portability. Finally, we presented a case study on which we apply our proposed approach.

The rest of this paper is organized as follows: Section two provides an overview of the technology used. Section three presents the problematic and the related work. In Section four, we present our proposed architecture and process to ensure communication between SOAP client and REST Cloud provider. In Section five, we outline a case study and we conclude this paper by presenting our further works.

Manuscript received July 5, 2016
Manuscript revised July 20, 2016

2. Overview of technology used

2.1 SOAP Protocol

SOAP is a messaging protocol based on XML (eXtended Markup Language) for the exchange of information in a decentralized environment. SOAP is commonly used to establish a communication channel between Web services. It defines a set of rules for structuring messages that can be used in simple way transmissions, but it is particularly useful for performing RPC (Remote Procedure Call) request-response dialogues. Using SOAP enables messaging protocols based on XML to exchange information between different applications. Therefore, it offers some benefits such as using unique address for every operation, increased privacy and complex operations can be hidden behind facade [2].

2.2 REST Architecture

REST [3] is not a protocol such as HTTP (Hypertext Transfer Protocol). This style of architecture is particularly well adapted to the World Wide Web but is not addictive. Constraints, as defined by Roy Fielding, can be applied to other application as HTTP protocols. This architectural style is not limited to performing application to a human user. It is increasingly used for the realization of SOA (Service Oriented Architecture) using Web services for communication between machines. REST brings benefits, like using unique address for every process instance and client can have one generic listener interface for notifications [2].

2.3 REST vs SOAP [4]

To understand the REST principle (figure 1), the Web server returns a representation of the resource. This resource places the client into a new state. The links in the response XML document point to the next state. The SOAP messages contain no hyperlinks but only data. The client cannot obtain information on what to do next from the SOAP message but must get this information into the client application in WSDL (Web Service Description Language) file (figure 2).

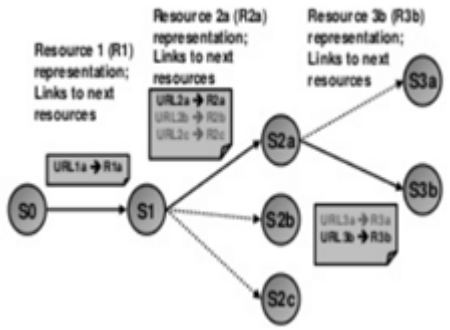

Fig. 1 Presentation of REST resources.

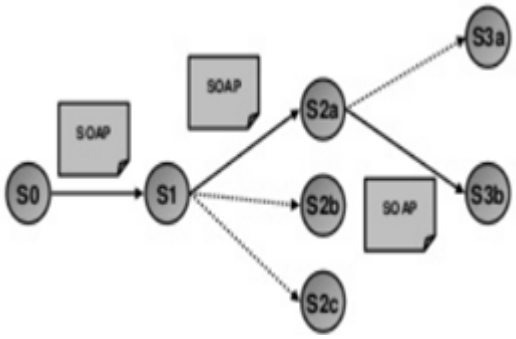

Fig. 2 Presentation of SOAP resources.

As shown in figure 3, REST uses different URLs (Uniform Resource Locator) to address resources. The Web server directly dispatches a request to a handler, and REST maps the type of access to HTTP methods. Contrariwise (figure 4), SOAP uses the same URL for all interactions, and then the SOAP server parses the SOAP message to determine which method to invoke.

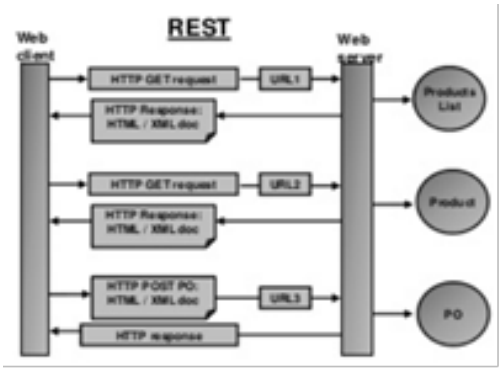

Fig. 3. REST architecture

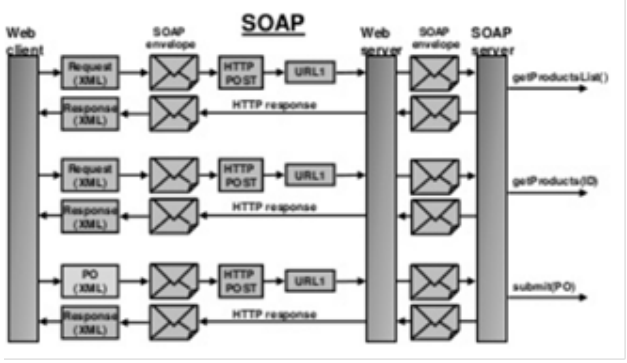

Fig. 4. SOAP architecture

2.4 Cloud Computing

NIST [5] defines Cloud Computing as "a model for allowing ubiquitous, convenient; on-demand network access to a shared pool of configurable computing that can be rapidly provisioned and released with minimal management effort or service provider interaction". According to NIST, Cloud Computing has five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service. Cloud Computing has a lot of advantages, but also still some pitfalls such as:

- Security and privacy: Public Cloud providers form an attractive target for hackers because of the number of users [6]

- Portability: Many customers are attracted by the public Cloud because of its attractive cost, but recover data may even be impossible. Consumers can find themselves in vendor lock-in situation [6]

- Standardization: Absence of standardization, essentially to relate different Cloud providers. Even if it is possible to provide service for diverse Cloud interfaces through a middleware, there are no rules that can be supervised by Cloud providers [7].

3. Problem presentation and related work

3.1 Heterogeneous protocols used in Cloud environments

Cloud Computing offers business users the chance to immediately implement services with usage-based billing that are tailored to their requirements, often without the need to consult with the IT department. However, using multiple Clouds which called "Sky Computing" [9] has a lot of benefits such as high availability, cost optimization, quality of service improvements, avoiding vendor lock-in, disaster recovery by decentralization of data. But there are some drawbacks like security, architecture complexity, data and service availability, and standards that are often not evaluated. Therefore, Cloud consumers find themselves at lock-in of the Cloud provider by many ways, some of them is service access standards.

Table 1: Comparison of BOX, AWS and AZURE [11]

Service feature	***Box***	AWS	Azure
Protocol supported	REST, SOAP	REST, SOAP	REST

However, there may be many Internet protocols for service access standards used to permit interoperability between users browser and Web server, such as REST and SOAP.

In the case where Web browsers are consuming SaaS applications, there are a number of standards that are used to achieve interoperability between users browser and Web server such as Internet Protocol (IP), Transmission Control Protocol (TCP), HTTP, XML, REST, SOAP, Really Simple Syndication (RSS) and JavaScript/JSON.

There are no standards Cloud-specific [5]. The specification documents focus on the description of a RESTful interface, but the standard separates the API design from the particular communication protocols to use. In particular, future developments will include SOAP and RPC implementations of the same interface [10].

3.2. Presentation of the problem

Multiple standards enable interoperability like REST and SOAP. Wherever data is acted on by several services, some of Cloud providers provide their own Web services in REST fashion, but SOAP is adopted in industry and academic. Table I shows Web service protocols used by some Cloud providers.

There is no Web standards that are specific to Cloud and these standards are used in many Web browser interfaces. Some of Cloud consumers and providers use SOAP, other ones use REST. Each service has its specific characteristics such as authentication and security requirements. Hence, Cloud providers attempt to lock consumers into proprietary interfaces. So consumers can find themselves at not expects inevitable vendor lock-in.

As indicated in figure 5, consumers C1 C2 and C3 want to outsource a part of their system application in Cloud Computing. So, consumers must choose the Cloud providers that have the same service access standard

interface. As shown in this figure, C1 and C2 with REST interface have two choices of Cloud providers with specific characteristics. C1 and C2 can choose CP1 or CP3, and C3 can choose CP2. The same thing with Cloud providers, the compatibility of service access interface between Cloud providers and consumers is mandatory.

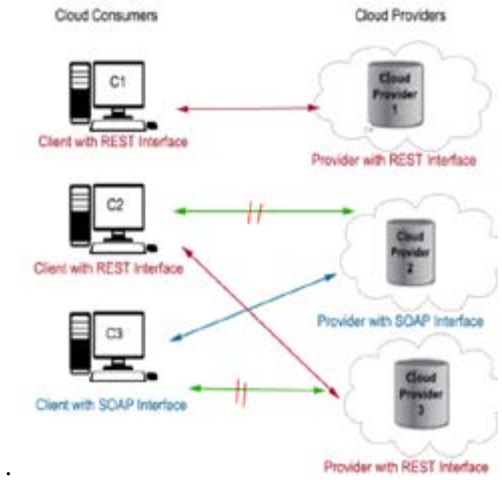

Fig. 5. Challenge in service access standards SOAP REST [12]

On the side of the consumers and Cloud providers, some questions are raised through programming system application that communicates with other environments. One of these questions is: which service access interface has to choose: REST or SOAP?

3.3 Problematic challenges

There are many challenges related to this heterogeneity of access interfaces such as:

- **Challenges on Interoperability Standard**

Interoperability is one of the important challenge axes that exist in IaaS, PaaS, and SaaS levels. Each one of these levels, which may joint in any particular product or Cloud services, presents special considerations, and as a result, the standards landscape will be unique and specific to each level. In this case, interoperability is a property of consumer system, whose interfaces are completely understood, to work with other Cloud providers, present or future, with restricted access.

Two systems in heterogeneous environment using different service access SOAP or REST cannot communicate with each other, because of that consumer can fall in interoperability challenge. When a provider pretends compliance with any other standard, he has to cite the specific version, publish implementation, and testing notes [5].

– Environment heterogeneity: Enterprises think that it is simple to add one Cloud service at a time and they do not expect the inevitable complexity of multiple Cloud providers for integrating their applications running on different Cloud platforms which end up with a traditional way of point-to-point integration approach. So, enterprises need to think about the complexity on integration of multiple applications on different Cloud platforms along with on premise applications and they also should think about service access technology used [13].

– Standardization: Is one of the most problems in the Cloud environments, many of the interfaces offered are unique to a particular provider. Also the risk of provider lock-in is raising [14]. Simon Wardley writes in [15], "The ability to switch between providers overcomes the largest concerns of using such service providers, the lack of second sourcing and the fear of vendor lock-in (and the subsequent weaknesses in strategic control and lack of pricing competition)".

– Complexity of multiple Clouds: The use of multiple Clouds providers in big systems increases the level of communication complexity between Cloud providers. Also, choosing the technology service access compatible with technology used increases the level of complexity. Some examples are given in [16] and [17].

- **Challenges on portability standard**

Cloud portability means that the consumer can move his own data from one Cloud system to another. One of portability constraints is the lack of standardization of Cloud services access protocols, which can be an obstacle to Cloud consumers to easily migrate to a new Cloud services provider when availability requirements are not same [5].

3.4 Related work

- **StoRHm**

StoRHm [1] is one of the protocol adapters which guarantee manual process transition from SOAP Web services to REST Web services, to enable existing SOAP clients to interact with REST services. The objective of StoRHm is to allow existing SOAP clients to converse with a REST Web service which tantamount to the original message SOAP Web service. StoRHm elements are highlighted with an overshadow background. A WSDL file, the WSDL schema and optionally a WADL (Web Application Description Language) file are a wizard inputs. These files are analyzed and present the user

information. Then the user matches the SOAP operations with the URIs (Uniform Resource Identifier); selects the HTTP verbs and the response Web services to be used. The output is the mapping file. StoRHm integrates in the client side a "SOAP-to-RESTful HTTP protocol adapter". The original SOAP request is converted to RESTful HTTP and leaded to the RESTful Web Service. The RESTful Web Service response goes back to the adapter and is mapped back to the SOAP so that the SOAP Web service client can understand it. Therefore, the SOAP client is abstracted from the mappings from the point of view of the SOAP client. This solution is used for a specific environment. It is semi-automatically because the user must to do a manual process transition from SOAP Web services to REST

- **REST2SOAP**

REST2SOAP [8] is a framework to integrate SOAP services and RESTful services. REST2SOAP converts RESTful services into SOAP services. The document of RESTful services description is provided by the "mashu service assembler" WADL. The framework will convert the RESTful using service description in WADL into a SOAP service in order to create a new composite service so that the "mashup service assembler" can integrate it with other SOAP services. However, REST2SOAP framework wraps RESTful services into SOAP services semi-a

4. Our proposed solution

4.1 General presentation of the architecture

Our proposal solution (Figure 6) aims to extend the work published in our previous publication [12]. It consists on adding a Cloud middleware to figure 5 between Cloud providers and Cloud consumers in heterogeneous Cloud environments. The objective is to remove the direct communication between interfaces. This approach guarantees interoperability and standardization in heterogeneous Cloud based environments. In this case, Cloud consumers using SOAP interface can communicate easily with Cloud providers using REST interface.

Our architecture makes easier the task of choosing a Cloud provider who'll satisfy the client needs because the client has not to select Cloud providers compatible with his environment. The architecture is essentially Cloud SaaS based on three principal components including a Client Interface Receptor (CIR), an Interne Converter (IC) and a Cloud Provider Interface Receptor (CPIR).

Communication between these components goes through five main steps (Figure 6):

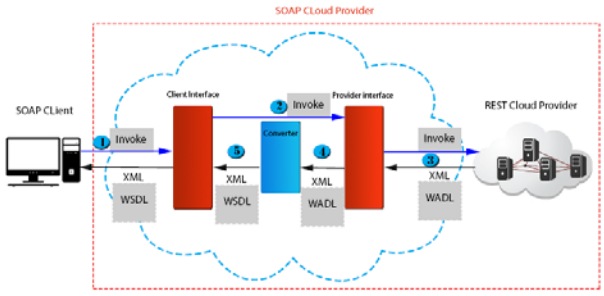

Fig. 6. Proposed solution of Cloud standardization by steps

Step 1: To establish connection with Cloud provider, client must have a WSDL file to find resources for that client and send an "Invoke" to the CIR.

Step 2: To know if the service interfaces client and provider are able to occur together automatically. In the first request sent by client, the CIR directs the architecture service interface SOAP used by client to the CPIR.

Step 3: The CPIR recovers the WADL file from the Cloud provider.

Step 4: The CPIR redirects the WADL file to the IC who converts the WADL file to the WSDL file.

Step 5: The output of the IC is a WSDL file, built from the WADL file, and sending to the CIR. This means that client service interface can understand the response directed to the CIR.

In this solution every component has a specific role which we explain in details in subsections below.

4.2 Client Interface Receptor

As mentioned in figure 6, an interface is related directly to the client. The objective is to process consumer request messages and responses. This interface allows the interaction with different services access standards REST and SOAP. It can deal and differentiate between SOAP and REST messages coming from Cloud consumers. This interface has an auto detect service interface protocol and can make a compatible environment of standards that help an easy communication between service access interfaces.

Also response coming back from the Cloud provider can be compatible with service interface technology used by consumer. The CIR component makes sure that service

interface protocol used by client is compatible with Web service used by client. The services standards used are SOAP and/or REST. In this case consumers can outsource their IT or applications without thinking about technology used by Cloud providers.

4.3 Cloud Provider Interface Receptor

Another interface is associated to the Cloud provider. Its objective is to manage provider response messages and requests. This Interface permits the interaction with different services access standards REST and SOAP by the side of Cloud provider. It can deal and differentiate between SOAP and REST messages coming from Cloud provider. This interface has an auto-detect service Interface Protocol, it can receive request from Converter or CIR and send the response or the WADL file to the Converter to make a compatible environment of standards and so helps an easy communication between service access interfaces.

4.4 REST/SOAP Converter

a) Presentation of the Converter

As shown in figure 6, SOAP/REST Converter is an intern component in our proposed solution. It ensures standardization and interoperability in heterogeneous Cloud based environments. Using this component, Cloud provider using REST interface can communicate easily with Cloud consumers using SOAP interface, by converting WADL file to WSDL file. As shown in figure 7, the Converter has a WADL file as input and a WSDL file as output.

To convert a WADL file to a WSDL file, the Converter follows the steps below:

- The Converter receives a WADL file as input, then the Parser parses WADL file and takes out the needful data to construct a WADL object. This WADL object is created by filling classes with necessary data.

- It instantiates a new WSDL object from WADL object by applying the rules that we have defined (Figure 9).

- Using the WSDL object, the Converter creates a new WSDL file corresponding to the WADL file.

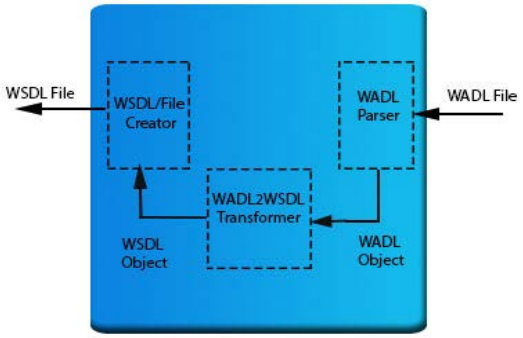

Fig.7: WADL/WSDL Converter

b) WADL File/Object Parser

The Converter has a WADL file as input which represents the response of the Cloud provider to a client request. Then, as a first step, the Converter parses this WADL file in order to build a WSDL object. The WADL object architecture corresponding to a WADL file is shown in figure 8. Every WADL file contains 'application' and 'resources'. The service is described by using a set of 'resource' elements. Every resource contains 'parameters' elements to describe the inputs and 'methods' which describe the 'response' and 'request' of a resource. Each response has its 'presentation', which describes the representation of the service's response.

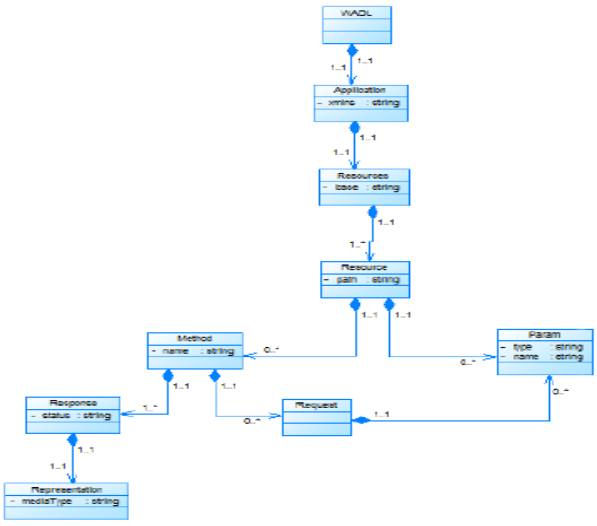

Fig.8: WADL Object

c) Object WADL/WSDL translation rules

After parsing the WADL file into a WADL object, the Converter has to translate this WADL object into a WSDL object. This translation is performed according the algorithm shown in figure 9 and goes through three steps:

- The Converter extracts resources from the XML file sent by the client using a resources extractor,

- Then, it defines the methods by a method extractor,

- At the end, it builds a REST URL.

A WSDL object as shown in figure 10 includes 4 sub items: 'message', 'portType', 'binding' and 'service'.

d) WSDL Object/File parser

Once the WSDL object is generated, it is transformed by the Converter to a WSDL file. This WSDL file represents the final output of the Converter and it is finally sent to the client.

4.5 Benefits of this solution

Our proposed solution can be used in any environment because it has an automatic translator of REST and SOAP Web service and it is an independent middleware. It doesn't have to be installed in client or Cloud provider sides. Some benefits of the whole process are presented below.

a) Benefits on Interoperability

Systems in heterogeneous environment using diverse interface service access SOAP or REST fall in interoperability challenge. Our solution gives some benefits of interoperability level as mentioned below.

```
Foreach(resource : wadl..getResourceList) {
foreach(method: wadl..getMethodList){
wsdl..OperationList.get(i).setName=method.getName + resource.getPath
wsdl..input.setMessage= wsdl..OperationList.get(i).getName +" request"
wsdl..output.setMessage= wsdl..OperationList.get(i).getName +"response"
j++
}
I++
}
Foreach(operation:wsdl..getOperationList) {
wsdl..messageReqList.get(i).setName= operation.getName +" request"
wsdl..messageRespList.get(i).setName= operation.getName +" response"
Foreach(param:wadl..getParamList){
wsdl..partReqList.get(j).setName=resource.param.getName
wsdl..partReqList.get(j).setType=resource.param.getType
}
wsdl..partResp.setName= wsdl..messageRespList.get(i).getName
wsdl..partResp.setType="string"

i++
}
wsdl..binding.setName=WSname + "Binding"
wsdl..binding.setType=wsdl..PortType.getName
wsdl..soapbinding.setStyle="rpc"
wsdl..soapbinding.setTransport=" http://schemas.xmlsoap.org/soap/http"

foreach(operation: wsdl..getOperationList){
wsdl..operationsoapbinding.setName= wadl..operation.getName
}
wsdl..soapbody.setEncodingStyle= "http://schemas.xmlsoap.org/soap/encoding"
wsdl..soapbody.setUse="encoded"
wsdl..soapbody.setNamespace=

wsdl..service.setName= WSname +"Service"
wsdl..port.setBinding=wsdl..binding.getName
wsdl..port.setPortName=WSname+"Port"
...
```

Fig.9: Algorithm of the WADL/WSDL translation

- **Standardization:** The big problem in Cloud provider side is standardization. Many of the interfaces offered are unique to a particular vendor, raising thus the risk of vendor lock-in. For that, our Cloud solution ensures communication between REST and SOAP interfaces by using a central component that can make a translation from SOAP to REST interfaces.

- **Environment heterogeneity:** Cloud consumers and providers can communicate even if they use different service access interface technologies. The Cloud solution simplifies deployment of existing applications and services without thinking about technologies used by Cloud consumers or providers.

- **Using multiple Clouds:** In case of using multiple Clouds environments, Cloud consumers can use heterogeneous Cloud environments. SOAP client can interact with REST Cloud provider by using Cloud service interface standardization.

- **User benefits:** Reduces development cost and time of new applications and services compatible with a specific Cloud provider.

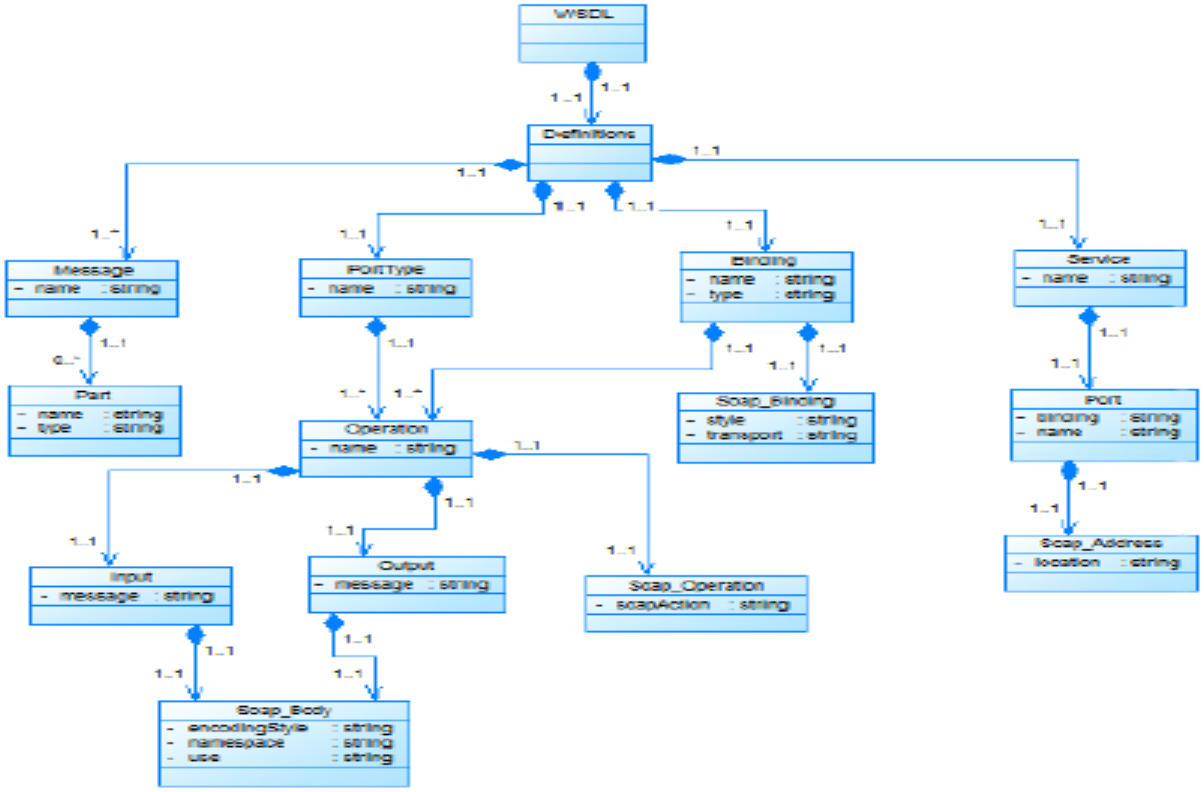

Fig.10: WSDL Object

- **Management of access user:** Management of users is also controlling by the Cloud service standardization and every access to the data by Cloud providers is controlled by Cloud system. Thus, the control of interaction between Cloud providers and consumers is not lost.

- **Availability of data:** In this case, data is available with both technology REST and SOAP thanks to Converter component, which can translate REST message to SOAP message adapted to Cloud consumer and provider.

b) Benefits on Portability

Cloud consumers who wish to migrate to another Cloud provider find themselves in not expected vendor lock-in. Our proposed solution brings some benefits at this level such as minimizing vendor lock-in and make maintenance easier.

- **Vendor lock-in:** Is a situation in which a consumer using a Cloud provider service cannot easily migrate to a competitors Cloud providers. Our solution minimizes the lock-in of Cloud providers to consumers by solving the problem of standardization of service access interface technology between SOAP and REST.

- **Outsourcing applications easily:** As Cloud systems are typically external components in a consumer organizations overall IT system, especially in the outsourced deployment models, the need to have seamless security integration calls for interoperable standard interfaces for authentication, authorization, and communication protections [5].

- **Easy Maintenance:** Maintainability is one of the main advantages of a centralized system. In our case, the proposed Cloud solution is located between Cloud provider and Cloud customer. So the maintenance task is not a customer or Cloud provider task.

- **Independent environment:** Contrary to other systems like StoRHm and REST2SOAP, our proposed solution brings an automatic conversion from REST to SOAP, without changing or adapting client and Cloud provider side. So, it is an independent and automatic middleware.

5 CASE STUDY

In this section, we present a case study which aims validate our proposed approach. The example [18] of WADL file is presented in figure 11. The main objective of this simple example is to represent a list of books. The resource would be http://example.com/api/books. The list

of books can be recovered by a 'GET' method, putting a new version by 'PUT' method and deleting the resource by 'DELETE' method using the same URI.

```
<application xmlns="http://wadl.dev.java.net/2009/02">
   <resources base="http://example.com/api">
      <resource path="books">
         <method name="GET"/>
         <resource path="{bookId}">
            <param required="true" style="template"
name="bookId"/>
            <method name="GET"/>
            <method name="DELETE"/>
            <resource path="reviews">
               <method name="GET">
                  <request>
                     <param name="page" required="false"
default="1" style="query"/>
                     <param name="size" required="false"
default="20" style="query"/>
                  </request>
               </method>
            </resource>
         </resource>
      </resource>
      <resource path="readers">
         <method name="GET"/>
      </resource>
   </resources>
</application>
```

Fig.11: WADL file of the example

The converter parses the WADL file and extracts the necessary data to build a WADL Object (Figure 12). Some of these data are the XML name space (xmlns) path for application classes, the path for resources and resource classes, names of methods, paths of request and the parameters of methods.

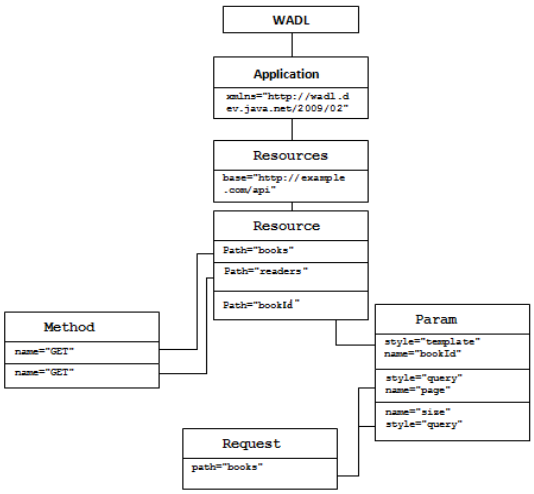

Fig.12: Simplified WADL Object of the example

The next step is to convert WADL objet to WSDL object presented in figure 13 respecting the transition rules. The Converter generates automatically the names of message. The message object represents the inputs and outputs of the operations indicated in the subject 'PortType'. So we'll start with this latter and filling the 'name' attribute the name of the Web service tracking by the 'PortType'. For each 'method' of each 'resource' of WADL, it creates an object 'operation' in the WSDL by completing the 'name' attribute with the name of the object 'method' of WADL follow the path of the resource. The attributes 'message' sub objects 'input' and 'output' filled by the name of 'method' follow 'request' and 'response' respectively.

The sub objects 'request' and 'response' of the object WADL are the portion 'message' of the WSDL object, so the attribute values will be taken to 'Param' to fill the attributes 'name', 'type' and sub items 'part' of WSDL.

The object WSDL 'binding' embodies the protocol and data format exchanged for 'PortType', so the Converter fills the 'name' attribute with the name of the Web service followed by 'Binding' and the attribute 'type' by the name of 'PortType'. The object 'soap: binding' has for attribute 'style' the value 'RPC' and for attribute 'Transport' the value 'http://schemas.xmlsoap.org/soap/http' indicates that this is the http protocol.
For each 'operation' specified in the 'PortType', it creates an object 'operation' which indicates the input and output to the encoding style and namespace through 'soap: body' objects.

The object 'service' takes the name of the Web service followed by 'service' as 'name' attribute. Sub object 'port' has two attributes: 'binding' which is the name of the binding and 'name' that has as value the name of the Web service follow 'Port'. Sub object 'soap: address' has as location attribute, which specifies the Web service URL in question, and we take its value from the attribute 'base' of the 'resources' object.

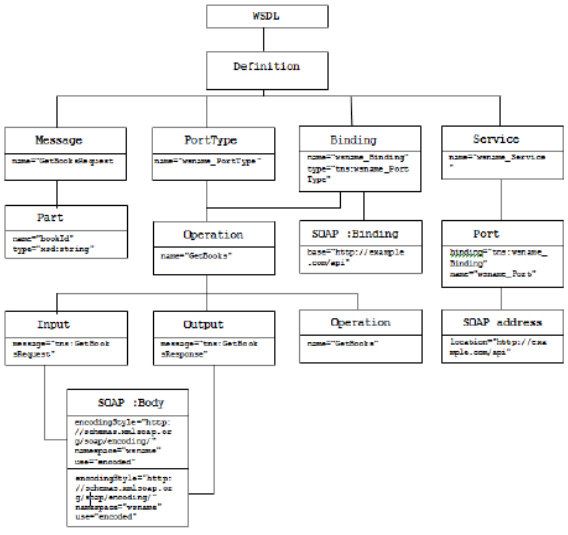

Fig.13: WSDL Object of the example

The next operation is to build the WSDL file from the WSDL object. This WSDL file is presented in figure 14.

```
<definitions name=" wsname"
   targetNamespace="ournamespace"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="ournamespace"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="GetBooksRequest">
      <part name="bookId" type="xsd:string"/>
   </message>

   <message name="GetBooksResponse">
      <part name="bookId" type="xsd:string"/>
   </message>

   <portType name="wsname_PortType">
      <operation name="GetBooks">
         <input message="tns:GetBooksRequest"/>
         <output message="tns:GetBooksResponse"/>
      </operation>
   </portType>

   <binding name="wsname_Binding" type="tns:wsname_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetBooks">
         <soap:operation soapAction="GetBooks"/>
         <input>
            <soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="wsname"
               use="encoded"/>
         </input>

         <output>
            <soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="wsname"
               use="encoded"/>
         </output>
      </operation>
   </binding>

   <service name="wsname_Service">
      <port binding="tns:wsname_Binding" name="wsname_Port">
         <soap:address
            location="http://example.com/api" />
      </port>
   </service>
</definitions>
```

Fig.14: WSDL File of the example

6. Conclusion and perspectives

In this paper, we have proposed an architecture that enables communication between SOAP client and REST Cloud provider. This solution is based on three components: a Client Interface Receptor (CIR), a Cloud Provider Interface Receptor (CPIR) and a Converter. The main objective of the Converter is to translate a WADL file into a WSDL file so that the communication between the client and the Cloud provider became easily and technology independent. Our proposed solution offers advantages such as minimizing vendor lock-in, availability of data and services and easy maintenance. This solution also guarantees portability and interoperability in heterogeneous Cloud environments. We have also presented a study case of our proposed solution that shows the necessary steps to follow for WADL/WSDL file translation.

In perspective, we will strengthen security of data and make implement our algorithm which ensures the WADL/WSDL translation. Another perspective is to extend this architecture to support JSON files.

References

[1] S. Kennedy, R. Stewart, P. Jacob, and O. Molloy, “Storhm: a protocol adapter for mapping soap based web services to restful http format,” Electronic Commerce Research, vol. 11, no. 3, pp. 245–269, 2011.

[2] M. Zur Muehlen, J. V. Nickerson, and K. D. Swenson, “Developing web services choreography standardsthe case of rest vs. soap,” Decision Support Systems, vol. 40, no. 1, pp. 9–29, 2005.

[3] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[4] P. R, “Rest,” 2015. [Online]. Available: http://indigoo.com/dox/wsmw/2WebServices/REST.pdf

[5] M. Hogan, F. Liu, A. Sokol, and J. Tong, “Nist Cloud computing standards roadmap,” NIST Special Publication, vol. 35, 2011.

[6] G. O. Johan Loeckx, “Cloud computing concept vaporeux ou relle innovation?” 2011. [Online]. Available: http://documentatie.smals.be

[7] A. Sammes, Computer Communications and Networks. Springer, 2014.

[8] Y.-Y. Peng, S.-P. Ma, and J. Lee, “Rest2soap: A framework to integrate soap services and restful services,” in Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on. IEEE, 2009, pp. 1–4.

[9] K.Keahey., “Argonne nat. lab., argonne, il,” http://ieeexplore.ieee.org, 2009.

[10] B. Di Martino, G. Cretella, and A. Esposito, Cloud Portability and Interoperability. Springer, 2015.

[11] S. Srinivasan, “Cloud Computing Basics”. Springer, 2014.

[12] M. Elhozmari and A. Ettalbi, “Using Cloud saas to ensure interoperability and standardization in heterogeneous Cloud based environment,,” the IEEE 5th World Congress on Information and Communication Technologies WICT, December, 14 – 16, 2015, Marrakesh, Morocco. Paper selected and published in Journal of Network and Innovative Computing (JNIC), Volume 4, pp. 029–036, 2016, www.mirlabs.net/jnic/index.html.
[13] N. V. S. Kolluru and N. Mantha, “Cloud integrationstrategy to connect applications to Cloud,” in India Conference (INDICON), 2013 Annual IEEE. IEEE, 2013, pp. 1–6.
[14] J. Kress, H. Normann, D. Schmiedel, G. Schmutz, B. Trops, C. Utschig- Utschig, and T. Winterberg, “Industrial soa soa blueprint: A toolbox for architects,” 2013.
[15] S. Wardley, “Cloud recap the Cloud today,” 2008. [Online]. Available: CloudToday(http://blog.gardeviance.org/2008/10/cloud-recap.html)
[16] M. Elhozmari and A. Ettalbi, “Using a software architecture based on a private central proxy Cloud to improve a health center system,” conference on Software Architecture CAL, Mai, 13 – 15, 2015, Hammamet, Tunisie. Paper selected and published in RNTI review, vol. RNTI-L-8, pp. 35-46. www.editions-rnti.fr
[17] M. ELhozmari and A. Ettalbi, “Using a central private Cloud to improve a complex system in multi-Cloud environement,” third World Conference on Complex Systems WCCS, November, 23 – 25, Marrakesh, Morocco 2015.
[18] T. Nurkiewicz, “Gentle introduction to wadl (in java),” 2012. [Online]. Available at: http://www.nurkiewicz.com/2012/01/gentle-introduction-towadl- in-java.html

Majda ELHOZMARI PhD student in the IMS (Models and Systems Engineering) Team of the SIME(Mobile and Embedded Information Systems) Laboratory AT the Higher National School for Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. Her research interests include Cloud Computing interoperability.

Ahmed ETTALBI Professor at Software Engineering Department and member of the IMS (Models and Systems Engineering) Team, SIME (Mobile and Embedded Information Systems) Laboratory of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat. His main research interests include Cloud Computing, Web-Services, Object Modeling with Viewpoints and Software Oriented Architectures.