WSDL/WADL Conversion in MaaS to Ensure Interoperability in Heterogeneous Environment

https://doi.org/10.3991/ijes.v6i3.9490

Majda Elhozmari(✉), Ahmed Ettalbi
Mohammed V University, Rabat, Morocco
Elhozmari.majda@gmail.com

Abstract—Web services refer to a paradigm consisting on communication between applications interfaces. They are the main key to communicate between different environments, such as Cloud Computing based architectures, and constrained environment. SOAP web services are more used then REST web services in education and industrial environment; contrariwise, Media data in web applications used REST web services. The main aim of this paper is to make alive and complete a previously proposed middleware SaaS solution, to enable unattached WSDL and WADL file translation. The Maas architecture is strengthen by developing Converter component and mapping rules component used in our previous work.

Keywords—MaaS, Cloud Computing, Interoperability, Web service

1 Introduction

In our previous research papers [1,14,15] we have proposed a Middleware as a Service (MaaS) based architecture to enable interoperability in heterogeneous client and provider environment. The focus of our previous article [15] is to make our previous proposed architecture [1,14] more global and flexible for the client using JavaScript Object Notation (JSON) scheme. We added new converter to ensure conversion of request and response messages from Representational State Transfer (REST) to Simple Object Access Protocol (SOAP) and vice versa. Furthermore, we have set up this architecture as a set of components to enable our solution to be more flexible to maintain, update and to be a general architecture that provides solutions to interoperability problems in different levels. Also, this architecture respects the concept of modularity that separates tasks of the components.

In this paper, we explain in detail some part of our architecture such as WSDL/WADL converter that can support Web Services Description Language (WSDL) and Web Application Description Language (WADL) files illustrated as WSDL and WADL objects used by the Converter. We highlight how we make the conversion from WSDL to WADL using mapping rules, stored in a mapping rules storage component, and the method that we use to parse an Extensible Markup Language (XML) file to object.

The rest of this paper is organized as follows: Section two provides background of the technology used. Section three presents our motivation and related works. In Section four we present our proposed solution. We conclude this paper by presenting our further works.

2 Background

2.1 SOA and concept of service

Service Oriented Architecture (SOA) is one of the most important enterprises IT system architectures. More than technology or method, it is a convergence of several existing approach. SOA integrates different practices in an organized framework to permit to deliver to companies a self-describing and unattached platform for business functionality [17], strongly guided by the business. Indeed, too focused experiments on a single approach or guided by the technique did not provide satisfactory answers.

Caspersen and DiMare define SOA as an approach to design software that dissolves business applications into separate "services" which can be used independent of the applications of which they are a part and computing platforms on which they run [18].

British Computer Society's defines SOA as follow: "SOA is about the evolution of business processes, applications and services from today's legacy-ridden and silo-oriented systems to a world of federated businesses, accommodating rapid response to change, utilizing vast degrees of business automation" [16].

Services are the essential concept of SOA. There are not originally technical concepts. The idea of services was developed in the world of business. It consists of a well-defined function or feature. Also a standalone component is independent of any context or external service. It is divided into operations, which constitute specific actions that the service can achieve.

2.2 REST and WADL

REST [2] is an architectural style for distributed hypermedia systems, created by Fielding in 2000. He finds particular applications in the World Wide Web. Fielding's work highlights the benefits of this architectural style over other styles of Web application architectures. Among others:

The application is simpler to maintain, because it disconnects the client part of the server part. The hypermedia nature of the application provides access to the following states of the application by inspecting the current resource.

The absence of client state management on the server leads to greater independence between the client and the server. It also eliminates the need to maintain a permanent connection between the client and the server. The server can thus respond to a number of other clients' requests without saturating or blocking all of its communication ports. This becomes essential in a distributed system.

The absence of state management of the client on the server also allows a distribution of requests on several servers: a client session is not to maintain on a certain server (via a sticky session of a load balancer), or to propagate on all servers (with problems of freshness of session). It permits also special scalability and fault tolerance (one server can be added easily to increase processing capacity, or to replace another).

In a Web context: Using the HyperText Transfer Protocol (HTTP) allows taking advantage of its envelope and its headers. The use of Uniform Resource Identifier (URI) as a representative of a resource makes it possible to have a universal system for identifying the elements of the application. The self-descriptive nature of the message allows the setting up of cache servers: the necessary information on the freshness and the expiry of the resource are contained in the message itself.

The main disadvantage of REST is the need for the client to keep locally all the data necessary for the smooth running of a request, which leads to a higher network bandwidth consumption. Especially in the world of mobile devices, each round trip connection is consuming bandwidth. The latency of the connection also makes the interaction less fluid.

REST is used to realize SOA using Web services to enable communication between different machines. REST web services have many benefits, like using for every process instance a unique address and one generic listener interface for notifications for a specific client [3].

WADL [4] is an XML-based computer language that describes REST applications. Its primary purpose is thus to describe the services offered by an application on the internet. This specification competes with WSDL 2.0, which allows the description of SOAP web services. In addition, WADL is still very poorly supported by all existing frameworks which limit its use.

2.3 SOAP and WSDL

SOAP [3] is an object-oriented remote procedure call (RPC) protocol built on XML. It allows the transmission of messages between remote objects, which means that it allows an object to invoke object methods physically located on another server. The transfer is most often using the HTTP protocol, but can also be done by other protocols such as Simple Mail Transfer Protocol (SMTP).

The SOAP protocol consists of two parts:

- An envelope, containing information about the message itself to allow its routing and processing;
- A data model, defining the format of the message, that is to say the information to be transmitted.

SOAP was initially defined by Microsoft and IBM1, but has become a reference since a The World Wide Web Consortium (W3C) recommendation. The SOAP protocol uses metadata. It is no longer an acronym since version 1.2. Indeed, SOAP v1.2 has been rewritten according to XML info sets and no longer as serializations <? Xml

...?> as it was in v1.1. The concept of object (specified in Simple Object Access Protocol) thus becomes obsolete.

The Web Services Description Language is a language proposed by W3C [5] to describe Web Services. WSDL is a description of a web service in XML format. WSDL 1.1 was proposed in 2001 to W3C for standardization but is not approved by the W3C. Version 2.0 was approved on June 27, 2007 and is now an official W3C recommendation. The WSDL describes a public interface for accessing a web service, particularly in the context of SOA architectures.

2.4 Cloud Computing

Definition and Characteristics of Cloud Computing. National Institute of Standards and Technology (NIST) defines Cloud Computing as follow: "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [6]. Cloud model is divided on five principal characteristics, four deployment models and three service models [6].

The main essential characteristics of the Cloud Computing are:

On-demand self-service: Stored resources are accessible by user at all times directly; the service provider does not have to intervene to make the resources available. The implementation of the system is fully automated, and the user sets up and manages the configuration from a distance, by means of a command console.

Broad network access: The resources are accessible everywhere and on various platforms. These processing centers are usually connected directly to the Internet backbone for excellent connectivity.

Resource pooling: Resources are available to multiple users (multi-client model) depending on the request, and located in a specific location often unknown to users. Most of these centers have tens of thousands of servers and storage to enable fast load outs. It is often possible to choose a geographical area to put the data "near" users.

Rapid elasticity: Resources are adjusted quickly according to the demands and needs of users. The upload of a new instance of a server is done in a few minutes, shutdown and restart in a few seconds. Scripts can do all these operations automatically. These management mechanisms make it possible to fully benefit from usage billing by adapting computing power to instantaneous traffic.

Measured service: According to the type of services Cloud providers do supervision and control of the use of resources. There is usually no cost of commissioning (it is the user who performs the operations). Billing is calculated based on the duration and quantity of resources used. A stopped processing unit is not charged.

Deployment Models. The deployment models of Cloud Computing are:

Private Cloud: The private cloud is a cloud in which services and infrastructure are on a private network. It is a set of cloud computing resources used exclusively by a company or organization. The private cloud can be physically located in the local data center of the company. Some companies also pay service providers to host their private cloud.

Community Cloud: Community Cloud is the ability for multiple entities or members of organizations with the same needs to use a single cloud solution. In this case, the Cloud is shared within several structures (companies, subsidiaries, business combinations, business groups...). It can then be used for generic applications adapted to the concerns of the group, or to host specialized applications common to many companies who wish to federate their efforts. Brief, the Community Cloud allows multiple companies to share resources and access to the same data.

Public Cloud: Public clouds are offered by a cloud service provider, which provides computing resources, such as servers and storage, over the Internet. In a public cloud, cloud provider owns all hardware, software, and infrastructure. Users access these services and manage their account via a web browser.

Hybrid Cloud: The hybrid cloud brings public and private clouds together, linked by technology to share data and applications. By enabling data and applications to migrate from private cloud to the public cloud, the hybrid cloud provides to organizations a greater flexibility and more deployment options.

Limits of Cloud Computing. Cloud Computing has a lot of benefits, but also still some drawbacks such as:

Security and privacy: The challenges remain numerous for the suppliers. We need to give confidence in the security model and in the management tools that are proposed. Management tasks are performed indirectly through an interface since the user does not have direct control over the physical infrastructure. This sharing of responsibilities complicates security audits a bit.

Portability: Currently, there is no widely accepted standard for interoperability and portability between cloud service providers. Hence, users must take all necessary steps to avoid being trapped; consumers can find themselves in vendor lock-in situation [7].

Standardization: Cloud computing has grown considerably in a few years. This rapid growth, however, has led to the emergence of many incompatible achievements despite the efforts of standardization groups. Lack of standardization, particularly with regard to security or data protection, makes it difficult to compare offers. However, interoperability between offers and the portability of services become a very important aspect for users to maximize their return on investment by choosing the best offers while avoiding being locked up in a single provider. “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” [8].

3 Motivation and Related Work

Many Middleware Framework are created to solve interoperability problem in heterogeneous environment, such as Internet of thing (IoT) environment in [19].

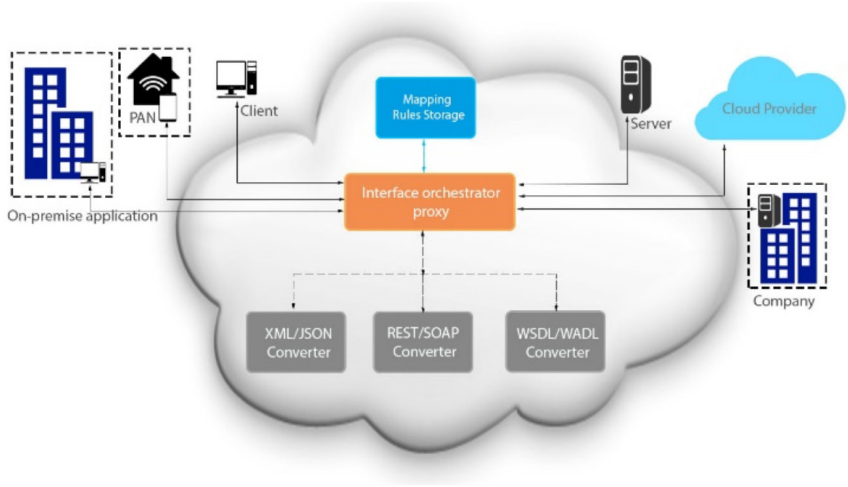

Fig. 1. Global Architecture of MAAS

In [1-15], we did a global study about interoperability issues between SOAP web service client and REST web service Cloud provider, clarifying the problem with a case study. We have proposed a Middleware as Service (MaaS) as shown in fig. 1, which is an extension to our previous work [1]. The objective is to enable the architecture to support not only the translation of WSDL and WADL files, but also the translation of the requests and responses between REST and SOAP messages. Besides, it implies the translation between XML and JSON scheme. Furthermore, the main objective of the structure of our architecture is to make it straightforward, by composing the architecture in several components, each component is responsible for a specific treatment, and outsourcing mapping rules to a Data as a Service (DaaS) to simplify update and evolution of system to tolerate other standards. To ensure security and semantic to our middleware, we combine Client interface and Provider interface from [1] in one Proxy interface orchestrator as shown in fig. 1.

REST2SOAP [9] is framework which interprets RESTful web service into SOAP web service semi-automatically. The generation of the WSDL file is been using JAVA2WSDL by converting RESTful service automatically to enable RESTful through Business Process Exception Language (BPEL) based on composite service. The main weakness of this solution is semi-automatic integration, and the target of this framework is just wrapping the RESTful web service into SOAP web service, not to make an equivalent SOAP file by mapping a RESTful file.

SOAP to RESTful HTTP mapping (STORHM) [10] is a protocol adapter which allows companies to converse with an existing SOAP web service client to RESTful HTTP web service server. Without influencing on existing clients, Wizard takes as an input WSDL file, WSDL schema and WADL (facultative), these files are used to bring out different services from WSDL file. To map SOAP message to RESTful HTTP format, users have to interact with Wizard form to enter information about WSDL files. The translation is from SOAP message service to RESTful HTTP service, in one sense.

DreamFactory [11] is a platform middleware REST Application Programming Interface (API). The function of this platform is to pack SOAP service into REST API and create REST endpoints. This conversion makes a request using JSON, calls the legacy SOAP service and then the SOAP response is converted back to JSON for the client application. To make translation, developers must interact with system and know what the mapping methods are talking about, and have the ability to use it in application.

Migration of SOAP-based services to RESTful services: "An approach generates automatically the configuration files needed to deploy the RESTful services and wrappers for accessing SOAP based services in the REST architecture" [12]. This is to allow the interaction between SOAP-based services and RESTful services; this approach convert dynamically messages between SOAP and REST web services. It also permits the translation from SOAP to REST protocol. It is based on provider side, so; the provider has to have RESTful and WSDL based services to send then to the right client, WSDL to SOAP client and RESTFul to client using REST web service.

MicroWSMO [13] supports dynamic substitution of REST and SOAP web services inner a service composition, even in presence of syntactic mismatches between service interfaces. It is based on a running prototype implemented in order to apply this approach to lightweight processes execution. The main limitation of this solution is that the translation is semi-automatic.

4 Proposed solution

To make alive our architecture, we start by creating the WSDL/WADL converter and the mapping rules storage.

4.1 WSDL/WADL Converter

As mentioned in [1-15], WSDL/WADL Converter is the main component of our architecture. It guarantees interoperability and standardization in heterogeneous middleware. It supports WADL object as illustrated in detail in fig. 2 and WSDL object as shown in fig. 3. These objects are generalized, and can support the maximum of WSDL and WADL files.

The converter has a WADL file as input that represents the response of the Cloud provider to a client request. Then, as a first step, the converter receives the parser from mapping rules storage, the goal here is to transform a WADL file that has the XML format into a WADL object. The WADL object architecture corresponds to a WADL file. Each WADL file contains 'application' and 'resources'. The service is described using a set of 'resource' elements. Each 'resource' contains elements of "prameters" to describe the "inputs" and "methods" that describe the "response" and "request" of a resource. Each answer has its "presentation", which describes the representation of the service response.

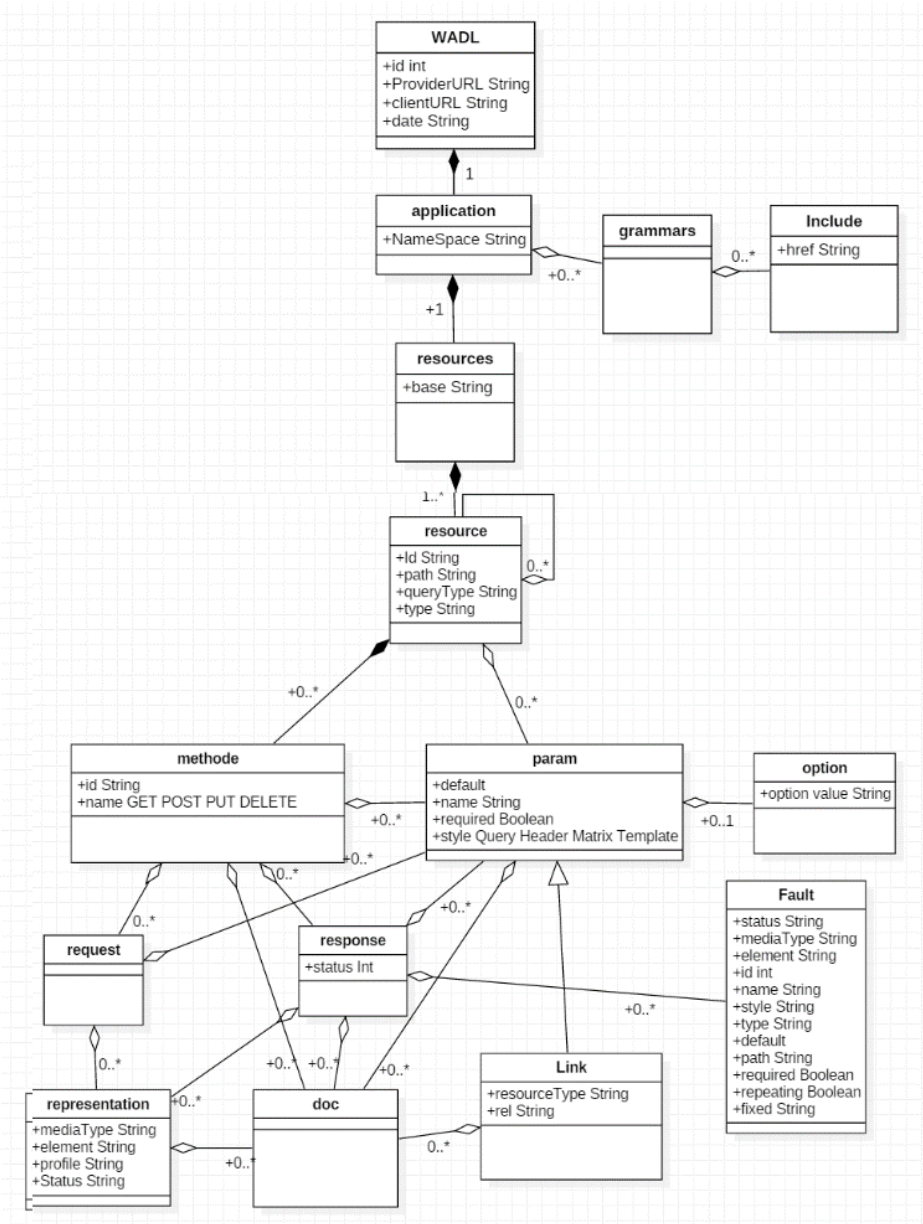

Fig. 2. WADL Object

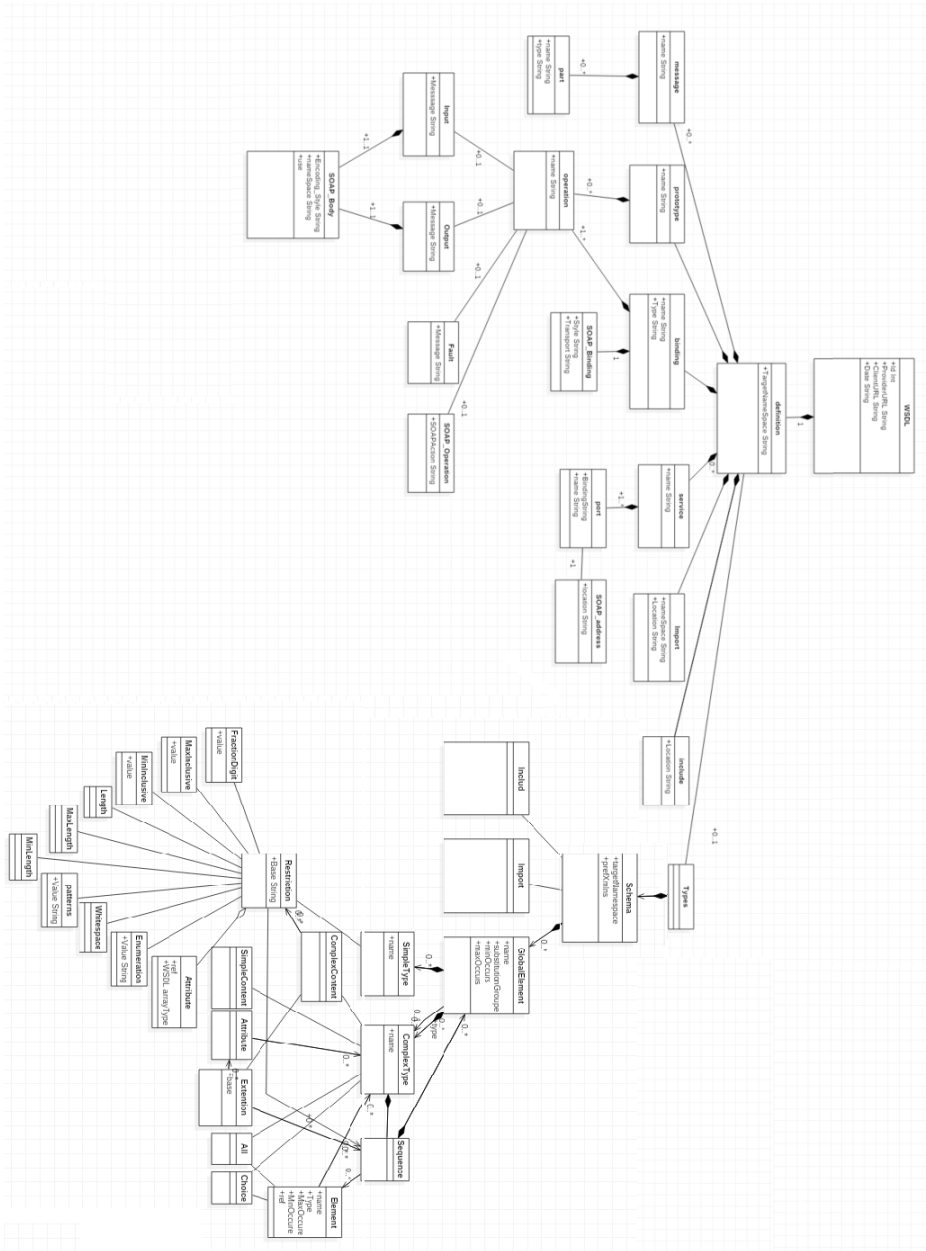

Fig. 3. WSDL Object

Next step is to convert WADL object to WSDL object respecting mapping rules coming from mapping rule Storage described in table 1. The first column of this table represents WADL elements, the equivalent WSDL elements are represented in the second column, and the third column represents mapping rules from WADL to

WSDL elements. The equivalence of "Operation" element of WSDL object is "Method" element from WADL object, so the name of the operation is the name of method concatenate with the path of it resource as follow:

(Name_operation = name_method+path_ressource).

In the case of "Service", "Port", "Binding", "Soap:binding", and "Soap:body" elements, the equivalents of these elements do not exist from a specific elements from WADL object, and there are obligatory to have a correct WSDL file to be understandable by SOAP client. For that, we have create mapping rules from a set of WADL elements, we arrived at this representation by studying a set of WSDL files. In fig 5 and fig 6, we present an example of conversion from WADL file to WSDL file. Fig 5 is the input of the conversion algorithm and fig 6 is the output.

Table 1. Mapping rules WADL object to WSDL object

WADL	WSDL	Mapping
Methode	Operation	Name_operation=name_method+path_ressource
Param	Part	Name_Part=name_param Type_part="String"(default)
Request/ Response	Message	Name_message=Name_operation+"Request" or "Response"
	Service	Name=Name of web service+"Service"
	Port	Binding name= web service name+"Port"
Resources	Soap:address	Web service URL= value of resource base attribute
	Binding	Name=webservice Name+ "Binding" type=Name of port type
	Soap:binding	Style=RPC Transport=http://schemas.xmlsoap.org/soap/http
	Soap:body	Encoding style and ilts namespace
	Input/output	Attribute message= name_method+"request" or "response"

Fig. 4 is a part of conversion algorithm, which enables conversion from WADL to WSDL object respecting mapping rules in table 1 and WADL and WSDL structures in fig 2 and fig 3. Each method of WADL object is an operation of a WSDL object, so we search from WADL object all methods for every resource. Input and output of operation are created from request and response elements of WADL object. Parameters of input and output are the parameters of request and response elements.

The conversion directed from WSDL file to WADL is complicated then WADL to WSDL because of the complexity of WSDL file. In WADL file there are four types of methods (GET, POST, PUT and DELETE). Contrariwise in WSDL file we should have a semantic algorithm to detect the type of methods from the meaning of operation name, for that we take equation (1) for 'Put' type and equation (2) for 'Post' type, equation (3) for 'Delete' type, and equation (4) for 'Get' type.

```
for i=1 to resourcesNbr do
  for j=1 to methodesNbr do
    %Create operation
    Operation_Name ← meth-
od.Name+resource.Path
    %Create input and output objets
    Input_Message← meth-
od_Name+resource_Path+ "Request"
    Out-
put_Message←method_Name+resource_Path+
"Response"
     %Create message and parameters of in-
put messages
    Message_Name← meth-
od_Name+resource_Path+ "Request"
    %Create parameters of input messages
    for r=1 to RequestLength do
      if Request not null
        for k=1 to ReqParamNbr do
          part_Name← ReqParam_Name
          part_Type← ReqParam_Type
        end for
      end if
    end for
    %Create message and parameters of out-
put messages
```

Fig. 4. Conversion algorithm from WADL to WSDL

Equations:

$$\forall Op \in Pu, M \leftarrow Put \quad (1)$$

$$\forall Op \in Po, M \leftarrow Post \quad (2)$$

$$\forall Op \in D, M \leftarrow Delete \quad (3)$$

$$\forall Op \in G, M \leftarrow get \quad (4)$$

Pu = {Put, submit, …, n}
Po = {Post, …, n}
D = {Delete, erase, …, n}
G = {Get, obtain, …, n}
Op= Operation_Name
M = Method

4.2 Mapping rule storage

Mapping rules storage contains all mapping rules used by the system. As an example, fig. 4 presents a WSDL mapping, which enables converter component to convert WSDL object to WSDL object.

Fig. 4 shows a part of algorithm of mapping rules. This part of mapping is stored in mapping rules storage component as mentioned in [15]. To insure this conversion, we used Java Architecture for XML Binding (JAXB) tool. This tool permits to developers to rig an XML document without knowing the way an XML document is handled as the case with Document Object Model (DOM) or Simple API for XML (SAX). The handling of XML document is accomplished by using objects generated from a Document Type Definition (DTD) for JAXB 1.0 and an XML schema of the document to be processed for JAXB 2.0.

```
<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. 5. Example of WADL file

```
<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. 6. Generated WSDL

5 Conclusion

In this paper, we have proposed some solutions used to make alive our MaaS architecture. The objective is to complete the architecture, and make it global and flexible to every client using REST or SOAP web service to communicate with any Cloud Provider or other System using SOAP or REST web service, by making translation between WSDL and WADL files. In addition, we make sure that the proposed MaaS tolerates maximum of WSDL and WADL file by WDSL and WADL object used in our converter. We gave an example of WSDL mapping algorithm to parse an XML file to an object and vice-versa.

In perspective, we will make alive our system in Cloud environment.

6 References

[1] M. Elhozmari, and A. Ettalbi, (2016) "Towards a Cloud Service Standardization to en-sure interoperability in heterogeneous Cloud based environment", International Jour-nal of Computer Science and Network Security, Vol.16, No. 7, pp. 60-70.

[2] R. T. Fielding, (2000) "Architectural styles and the design of network-based software architectures" Ph.D. dissertation, University of California, Irvine.

[3] M. Zur Muehlen, J. V. Nickerson, and K. D. Swenson, (2005) "Developing web ser-vices choreography standards, the case of rest vs. soap," Decision Support Systems, Vol. 40, No. 1, pp. 9–29. https://doi.org/10.1016/j.dss.2004.04.008

[4] H. Hadley, and J. Marc, (2006) "Web application description language (WADL)".

[5] R. Chinnici, J. Moreau, A. Ryman, IBM, and S. Weerawarana, (2003) "Web Services Description Language (WSDL) Version 1.2". World Wide Web Consortium (W3C), retrieved from: http://www.w3.org/TR/wsdl12/.

[6] M. Hogan, F. Liu, A. Sokol, and J. Tong, (2011) "Nist Cloud computing standards roadmap," National Institute of Standards and Technology Special Publication, vol. 35. https://doi.org/10.6028/NIST.SP.500-291v1

[7] J. Loeckx, and G. Ogonowski, "Cloud computing concept vaporeux ou relle innova-tion?" 2011. Retrieved from: https://www.smalsresearch.be/download/presentations/Cloud%20computing%20-%20Concept%20vaporeux%20ou%20reelle%20innovation.pdf.

[8] A. J. Sammes, and J. Rak, (2014) Computer Communications and Networks. Springer.

[9] Y. Y. Peng, S. P. Ma, and J. Lee, (2009) "Rest2soap: A framework to integrate soap services and restful services," in Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference. pp. 1–4. https://doi.org/10.1109/SOCA.2009.5410458

[10] S. Kennedy, R. Stewart, P. Jacob, and O. Molloy, (2011) "Storhm: a protocol adapter for mapping soap based web services to restful http format," Electronic Commerce Re-search, vol. 11, no. 3, pp. 245–269. https://doi.org/10.1007/s10660-011-9075-3

[11] https://www.dreamfactory.com/resources.

[12] B. Upadhyaya, Z. Ying, and X. Hua, (2011) "Migration of SOAP-based services to RESTful services". Web Systems Evolution (WSE), 13th IEEE International Sympo-sium, pp. 105-114. https://doi.org/10.1109/WSE.2011.6081828

[13] T. D. Giorgio, G. Ripa, and M. Zuccalà, (2010) "An approach to enable replacement of SOAP services and REST services in lightweight processes." International Conference on Web Engineering. Springer Berlin Heidelberg, pp. p. 338-346.

[14] M. Elhozmari, and A. Ettalbi, (2015) "Using cloud SaaS to ensure interoperability and standardization in heterogeneous Cloud based environment". International conference on Information and Communication Technologies (WICT), Marrakech, Morocco, pp. 29-34.
[15] M. Elhozmari, and A. Ettalbi, (2017) "Straightforward MaaS to Ensure Interoperabil-ity in heterogeneous Environment". 13th international conference on Information As-surance and Security (IAS), Marrakech, Morocco pp. 211-220. Springer, Cham.
[16] D. Barnes, (2007) "The service oriented architecture: more than just another TLA?". British Computer Society, UK.
[17] I. Cartwright, and E. Doernenburg, (2006) "Time to jump on the bandwagon" ITNow, Vol.48, I.3, British Computer Society, UK, May.
[18] J. Caspersen, and J. DiMare, (2006) "Making Connections: Using SOA to Enable collaboration in travel and transportation". IBM Global Business Services: http://www-935.ibm.com/services/us/gbs/bus/pdf/gbe03148-usen-01.pdf.
[19] S. M. Balakrishnan, and A.K. Sangaiah, (2015) "Aspect oriented middleware for Inter-net of things: a state-of-the art survey of service discovery approaches". International Journal of Intelligent engineering and Systems, 8(4), pp. 16-28.

7 Authors

Majda Elhozmari is PhD student at IMS Team, ADMIR Laboratory, ENSIAS, Rabat IT Center, Mohammed V University of Rabat, Morocco

Ahmed Ettalbi is Professor at Software Engineering Department of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat, Morocco. His main research interests Object Modeling with Viewpoints, Software Architecture, Cloud computing and Business Process Modelling architecture.

Article submitted 04 September 2018. Final acceptance 06 October 2018. Final version published as submitted by the authors.