Towards a standard Restful WADL implementation of Multi-view web services

Anass Misbah† and Ahmed Ettalbi††,

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

Summary
Communication between heterogeneous systems is becoming more widespread throughout the multiple networks and platforms. One of the main technologies that are used nowadays to deal with applications portability and interoperability is Web services. In fact, this concept allows reaching a high level of standardization and normalization regarding both data structure and transfer protocols.
Web services do not include natively the end user constraints and needs. Therefore, multiple works have been carried out and proposed a variety of models and concepts to integrate the end user specifications since an early phase of the conception of Web services. One of them is the Multi-view Web services.
A previous implementation of Multi-view Web services has been proposed using the standard definition of WSDL. In this work we will carry on improving this implementation by proposing another standard definition which is WADL.
This alternative consists of a new implementation of Multi-view Web services using a Restful architecture in order to take advantage of all benefits that such a style of architecture can offer. One of the most important advantages of Restful architecture is the native support of HTTP methods, simplicity, improving performance, less data consuming and easiness of implementation.
Key words:
Multi-view; Web services; Standard; WSDL; WADL; Automatic Generation; Restful; SOAP.

1. Introduction

Working with Web services in all kinds of Information Systems has become a necessity; indeed, it brings many benefits in terms of flexibility and reusability. However, as Information Systems are becoming more and more open to the outside networks, and exchange with a lot of systems and actors, security issues are arising and that makes the Web-based applications more vulnerable and exposed to many threats.

These threats can be addressed by means of many IT security concepts and approaches. Most of these methods, deal with the users access rights in the implementation phase of the development process, which can reduce the efficiency and lead to security breach.

In our work we propose another way to address this issue, which is based on taking under consideration the end user constraints since an early phase of the application design. In Web services technologies, this concept can be achieved through the concept of Multi-view Web services.

Previous works [4] and [5], proposed an integration of the notion of Points of view with Web services and an implementation using WSDL-Us definition in order to build Multi-view Web services. The work of [1] brought an improvement of the previous implementation by introducing automatic transformation Meta rules and standard implementation based on WSDL native syntax which allowed setting up a standard compliant automatic approach to generate Multi-view Web services.

In this work, we will go further with the improvement of this approach, by proposing another alternative implementation. This new implementation consists of a Restful WADL definition of Multi-view Web services. In fact, Restful architecture helps reaching a high level of flexibility, simplicity, increased performance, native HTTP methods support and interoperability.

In section 2 we discuss briefly the technology overview, and then in section 3 we present the state of the art. Afterwards in section 4 we introduce our approach and illustrate the WADL implementation through a study case. Section 5 is dedicated to a focus on the main advantages of our approach. Eventually, in section 6 we conclude by giving a succinct summary of this work in addition to some perspectives.

2. Technology Overview

2.1 Web services

Web services can be defined as Web-based operations that can be invoked remotely using Internet protocols. This architecture is one of many representations of SOA (Service Oriented Architecture) [6] Design. The main

Manuscript received April 5, 2017
Manuscript revised April 20, 2017

concept is to consider each function doing a specific treatment and returning some parameters as a service which is defined by a contract (interface). The main advantages of this notion are the reusability, interoperability, flexibility and maintainability.

The applied protocols can be SOAP or Rest [7] depending on the context needs and constraints. In both cases, the exchanged and returned parameters are standardized by means of languages such as WSDL [8] and WADL [9].

Communication between entities is operated through Client-Server architecture (Fig. 1). This architecture is composed with 3 principal nodes:

- **Server** : The service provider which hosts the Web service operations
- **Client** : The requestor of the service
- **Directory** : A repository that contains all needed information of the Web service to be discovered and invoked (such as UDDI : Universal Description Discovery and Integration)

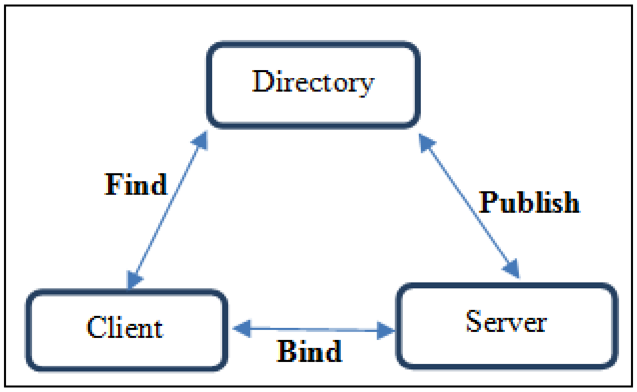

Fig. 1 Basic Web service architecture

2.2 SOAP messaging protocol

SOAP (Simple Object Access Protocol) [10] is a message transmission protocol, this XML-Based format is designed to be a one-way message exchange format with state-less sessions between the sender and the receiver. This protocol is platform independent since it can operate whatever the operating System, the technology or even the internet protocol used is. However, it is commonly used with HTTP transport layer.

The SOAP messaging Framework is a W3C recommendation, thus, the syntax of the message wrappers must follow a predefined rules and comply with standard templates.

As a description language, WSDL (Web Services Description Language) [8] is used to describe both abstract and concrete parts of SOAP-based Web services, this language is also standardized through a W3C recommendation, and therefore the used syntax must adhere to a specific format. The main WSDL directives (Table 1) are below.

Table 1: WSDL basic description elements

Service	Collection of ports
Port	Network address
Binding	Protocol + Data format, related to a PortType
PortType	A collection of operations (similar to interface in Java)
Operation	A web service action described by its messages (similar to method or function in Java)
Message	A set of data of a certain type according to a specific system (like W3)
Types	Definition of typed data using a given standard (such as XSD)

2.3 RestFul Web services architecture

Unlike SOAP which is based on SOA (Service Oriented Architecture) architectural design, Rest (Representational State Transfer) [7] is based on ROA (Resource Oriented Architecture) architectural design. This new paradigm consists of an architecture style that considers the existing Web applications resources.

Indeed, Rest did not come up with a new messaging protocol or wrapper format, it relays mainly on the existing HTTP URLs which are considered as a unique identifier of each resource. This state-less exchanging architecture brought flexibility, simplicity and reusability of the existing Web application operations.

Rest takes advantage of the native existing HTTP methods to perform the desired operations, each HTTP bind is secured using the header credentials (user name and password) of the browser session. The main HTTP used commands are the following:

- **Get** : Used to retrieve an object or a list of objects from the server
- **Post** : Creating one or multiple persistent objects upon the Web Server
- **Put** : Can be used to create objects but it is commonly used to update existing objects
- **Delete** : Used to delete existing objects

RestFul architecture is used generally with Web applications that require high level of performance and implementation simplicity such as Mobile Web applications and automated exchanges between systems. Nevertheless, it does not allow as much functionalities as SOAP in terms of transactions management.

The description language commonly used with Rest is called WADL (Web Application Description Language) [9], this language is a machine-aware description of HTTP-based exchanges between Web applications. It allows a semantic clarity and self description of the content exchanged as well as a standardization of the RestFul Web services definition.

WADL description can make use of either XML or JSON data definition. Many WADL elements can be used to describe the HTTP-based Web services, the most used are the following (Table 2).

Table 2: WADL basic description elements

Application	The root node of the description
Grammar	The container of the exchanged data formats
Include	Including by reference external data formats
Resources	A collection of resources, each resource define the Web service operation binding information and encapsulates one or more Method elements
Method	Define the used HTTP method as well as request and response formats, each method contains :Request and Response elements
Request	Specify the request parameters and encapsulates the definition of the exchanged Param elements
Param	Define the name, type, style and other attributes for each parameter
Response	Define all response possible status and representations, this element contains the Representation element
Representation	Indicates The mediaType of the response and encapsulates the response Element object
Element	Encapsulates the result set or the response objects

2.4 Comparison between SOAP and Rest

Generally speaking, Rest is more suitable for platforms exchanging lightweight messages and requiring high level of performance, in fact, SOAP allows more functionalities but requires more time to wrap and parse messages. Hence, Rest presents an interesting alternative depending on the context [13].

Bellow some characteristics of each paradigm:

- SOAP
  - Widely used especially in B2B enterprise Web applications
  - Require more tools and skills to be implemented and used
  - Can be used with all Internet transport protocols
  - Uses Post HTTP method and build a XML message format for each call, which required more bandwidth
  - Returns always an XML format
  - The client side is heavily dependent on the server, thus every change in the server side has an impact on the client side
- Rest
  - Relays on the existing Web applications infrastructure and do not need a lot of standardization requirements
  - Web services are simpler to develop and involve less effort
  - Makes use of HTTP or HTTPS existing methods and standardized URIs
  - Requires less bandwidth
  - Changes on server side can be done without impact on the client side
  - The Rest interfaces are more human friendly and allow more flexibility
  - Widely implemented in critical Banking Web applications

3. State of the Art

3.1 Multi-view Web services

The Multi-view concept consists of taking under consideration the end user access rights since an early phase of design and conception. A previous works, including [2] and [3] have dealt with this notion and proposed many definitions, implementations and application examples with UML (Unified Markup Language) models and diagrams. This concept allows many advantages such as redundancy elimination, enhancing security, access control efficiency and native integration with Oriented Modeling Languages.

Multi-view is a user oriented concept which is based on two main paradigms:

- **Views**: An operation that contains one or more tasks that can be done separately by at least one specific user
- **Points of view**: A set of unique views that can be done by one or more users

The integration of Multi-view concept with Web services was proposed in the work of [4]. Multi-view Web services can be designed through a process of decomposition; Every main Web service is decomposed into sub Web services (represent the Views) and then sub Web services are gathered in such manner that every user has a unique set of sub Web services (represents the user's Point of view).

The work of [5] proposed an implementation of Multi-view Web services which is based on an extension of the standard WSDL called WSDL-Us definition. This implementation made it possible to define Multi-view Web services, however, we noticed some limits of this implementation, below the main ones:

- Redundancy of elements definition
- Creation of a specific syntax and formats
- In the WSDL-Us definition sub Web services are considered as Web services rather than an operation
- There are no automatic generation rules
- As the WSDL-Us is not standard compliant, it cannot be translated and validated automatically.

3.2 Standard definition of Multi-view Web services

As an improvement of the work of [5], in our work of [1], we came up with another way to implement Multi-view Web services. This improvement relays mainly on WSDL standard definition. As a result we brought an enhancement in three main areas:

- **Generic approach**: Meta rules allowing automatic transformation of Web services to standard Multi-view Web services.
- **Standard compliant**: The used WSDL syntax to generate the output file is standard, with no new element.
- **Quality of the script**: The generated standard WSDL is clearer and can be integrated natively with existing Web services platforms and Frameworks.

4. Our Approach

4.1 Presentation of the approach

Both Rest and SOAP are transport-independent architectures, the security layer should be implemented through a specific concepts. In the work of [1] we proposed an implementation of the security layer by introducing the Multi-view concept with SOAP-based Web services using the WSDL standard definition.

Though, this implementation may not be suitable for all contexts, thus, we are enriching our work by proposing another alternative of Multi-view Web services implementation, which is based on the WADL standard definition.

As a first step, we will define a generic Meta rules for automatic transformation of any Web service model to Multi-view Web service WADL definition.

Bellow (Fig. 2) the proposed footsteps to follow for standard Multi-view WADL automatic generation:

1. Define the business objects in **Grammar** section of WADL document
    - This definition represents a description of objects that will be exchanged through sub Web services invocations, this objects might be defined or included.
2. Define a **Resources** block that represents the main Multi-view Web service and acts as a container for sub Web services
3. Define each sub Web service as a Resource and describe the related parameters (Path, Method name, Param, …)
    - For each Sub Web service (a Resource), define a **Param** called "User" which contains the list of allowed users encapsulated in **Option** elements
    - For each **Resource** define the possible responses in **Response status** section, then specify the result type using the **Representation** element

Fig. 2 Meta rules for automatic Multi-view WADL generation

4.2 Case study

As to illustrate our approach, we are considering the same study case of the work of [1], and then applying the Meta rules defined above (Fig. 2) in order to generate a RestFul-based Multi-view Web services WADL definition (Fig. 3). Below a reminder of the study case (Car Stock Management Web service):

We consider two points of view

- **Manager** : Allowed to Consult, Add, Modify or Delete a Car stock (View1 + View2)
- **Customer** : Allowed to Consult a Car Stock (View2)

Below the generated WADL (Fig. 3) after applying Meta rules of (Fig. 2)

```
<?xml version="1.0"?>
<application        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd"
    xmlns:tns="urn:CarStock:yn"
    xmlns:xsd=http://www.w3.org/2001/XMLSchema
    …

       <grammars>
              <include   href="CarStockService.xsd"/>
           <include   href="Error.xsd"/>
   </grammars>

   <resources base="http://CarStockUri/V1/">
       <resource path="consultCarStock">
           <method name="GET" id="consult">
              <request>
                  <param name="user" style="query">
                     <option value="Manager"/>
                     <option value="Customer"/>
                      </param>
                      // Others parameters to be set here …
                 </request>
                 <response status="200">
                     <representation      mediaType="application/xml"
                     element="yn:ResultSet"/>
                 </response>
          </method>
    </resource>
            // Others resources to be set here with appropriate user's
           definition…
        </resources>

</ application >
```

Fig. 3 The generated Multi-view WADL definition using the Meta rules

As a result of the new approach, Multi-view Web services might be implemented with SOAP-based WSDL definition as well as RestFul-based WADL definition.

5. Advantages of our Approach

By proposing another definition language of Multi-view Web services which is WADL, we gave an alternative to the previous definition language which was WSDL. Consequently, we made it possible to capitalize on the several advantages of RestFul-based Web services architecture. In addition, we put forward a generic Meta rules allowing the automatic transformation of standard Web service to Multi-view Web services with RestFul-based WADL definition.

As a result we can notice many advantages such as:

- The simplicity and flexibility of the generated Multi-view Web services
- The clarity and ease of use of the output WADL definition
- The defined Multi-view Web services require less effort of integration with existing Web applications
- The Client side will be less impacted by the Server side modifications
- The WADL generated is validated automatically and make use only of standard syntax, thus this generation can done easily by an automation program or tool
- RestFul Multi-view Web services require less effort of development and integration since they are using existing HTTP methods and exchanging much less data comparing to SOAP-based Multi-view Web services
- RestFul Multi-view Web services are more suitable for platforms demanding high level of performance such as mobile Web applications, Banking transactions systems, …
- RestFul-based Multi-view Web services will be able to inherit all Rest community contributions and techniques in terms of security patterns and models such as (HTTPS, Authentication credentials, tokens, multi-tenancy, …)

6. Conclusion and perspectives

In this work we highlight the advantages of RestFul Web services and we put ahead this architecture as an interesting alternative in particular contexts. Moreover, we continue enriching our previous research concerning Multi-view Web services through the integration of RestFul architecture with Multi-view Web services concept.

This integration consists of proposing and applying automatic transformation and generation Meta rules so as to create RestFul Multi-view Web services WADL implementation. Therefore we combined both advantages of Multi-view and RestFul Concepts.

Hence, our work brings more capabilities in terms of generating and implementing Multi-view Web services, and therefore enlarges and extends the application domains and contexts that can make use of this concept.

As a future work, we will carry on studying other study cases in order to confirm the sustainability of our work. Furthermore, we will continue improving the concept of defining and using Multi-view Web services by exploring other Web-based architectural designs and patterns.

References

[1] Misbah, A., Ettalbi, A.: Towards a standard WSDL implementation of Multiview web services. In: 5th International Conference on Multimedia Computing and Systems (ICMCS'16) – IEEE Conference, 29 September – 1 October (2016), Marrakech, Morocco.

[2] Kriouile, A.: VBOOM, Object-oriented analysis and design method by points of view, PhD thesis. Mohammed V University. Rabat, Morocco. (1995)

[3] Marcaillou, S.: Integration of the points of view concept in object modeling - The VBOOL Language. PhD thesis, the Paul-Sabatier University, Toulouse, France, (1995)

[4] Boukour, R., Ettalbi, A. and Nassar, M.: Multiview Web Service: The Integration of The Notion of View And Point of View in The Web Services. International Journal of Computer Science and Network Security (IJCSNS), vol. 14 no.2, pp. 31-36, February (2014)

[5] Boukour, R., Ettalbi, A. and Nassar, M.: Multiview web service: The description Multiview WSDL of Web Services. In: International Symposium on Signal, Image, Video and Communications (ISIVC'2014), November 19-21, (2014), Marrakech, Morocco.

[6] Papazoglou, M.P., Heuvel, W.: Service oriented architectures: Approaches, technologies and research issues, The VLDB Journal, Springer Verlag, (2007), pp.389-415.

[7] Alonso, G., Casati, F., Kuno, H.: Web Services: Concepts, Architectures and Applications, Springer Verlag, Heidelberg, Germany, (2004)

[8] W3C: Web Services Description Language (SDL) 1.1. December, (2006)

[9] Marc J. Hadley.: Web Application Description Language (WADL), Sun Microsystems Inc., February 2, (2009)

[10] W3C: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation, 27 April (2007)

[11] Kishor Wagh Dr., Ravindra Thool, Marathwada Mitra Mandals, Shri Guru Gobind Singhji: A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. Journal of Information Engineering and Applications ISSN 2224-5782 (print) ISSN 2225-0506 (online) vol. 2, No.5, (2012)

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

[13] Elhozmari, M., Ettalbi, A.: Towards a Cloud Service Standardization to ensure interoperability in heterogeneous Cloud based environment. IJCSNS International Journal of Computer Science and Network Security, vol. 16 No.7, pp. 60-70, July (2016)