REST and SOAP Aware for Web Service Composition

Ali Bentaleb[1†] and Ahmed Ettalbi[2††],

IMS Research Team, ADMIR Laboratory, ENSIAS, Rabat-IT Center
Mohammed V University in Rabat, Morocco

Summary
Web services composition is a prominent feature for systems using service oriented architecture. Nowadays, most web services are presented to be consumed either using Soap especially for legacy system, and Rest for modern architectures. In our paper, we deal about the challenge of composing these different ways of serving services and propose a formal model for automating the composition. Use case is presented to prove the theoretical approach of our work.

Key words:
Rest, Soap, service Composition.

1. Introduction

Since introduced by Fielding [1, 2], the modern web is an instance of the Representational State Transfer (REST). The REST principles are: (1) any information that can be named is a resource and the resource identifier (URI) is the unique pointer to this resource. (2) HTTP operations can be used to manipulate these resources. (3) REST components communicate by transferring a representation of the data in different formats as needed by the client. The REST architecture is becoming more dominant in nowadays networks due to its flexibility and easy adaptation with HTTP protocol. Moreover, SOAP (Simple Object Access Protocol) is becoming the legacy protocol used for interacting with old systems that rely heavily on object operations. In our paper, we address the concern of automating the service composition between REST and SOAP and also the quality of service when such composition is occurred. The rest of this paper is organized as follows: Sect. 2 provides background of the Technology used. Section 3 presents our motivation and related works. In Sect. 4 we present the problem and our proposed solution. We conclude this paper by presenting our further works.

2. Background

2.1 REST

REST is a web architectural style, the current web is an instance of this architectural style [1,2]. There are three main classes of architectural elements: data elements which represent elements containing data across the web, connecting elements which represent the API, interfaces and connectors, and the processing elements which are the servers and the components. When a link is selected, data must be sent from one place to another, three possible scenarios are presented: either to render the data in the engine and send the rendered image to the client which is the usual client server architecture; send data and engine to client and let it do the work (the mobile object style) which will overload the network and loses the hidden structure; or send data with metadata and let the client choose based on metadata, which will allow simple and scalable data transfer but it will lose hidden info and the sender and receiver must speak the same language.
REST allows the hybrid of these three possible scenarios: it consists to send data with metadata and limited to interfaces which will lower information detection, and the components communicate by transferring a representation of the data in different format as needed by the client.

2.2 SOAP

SOAP [4] is a messaging protocol based on XML (eXtended Markup Language) used to exchange information in a decentralized environment. SOAP provides a way to communicate between applications over HTTP. It is composed from two basic parts: the envelope which contains information about the message and its destination, and a data model with its format. The SOAP message exchange relies heavily on XML and it follows the following order: the server exposes its WSDL (Web service description language), then the client gets the WSDL to understand the data to be sent by which method, and finally the server receives requests and responds to them accordingly which means the heavy coupling between the server and the client.

2.3 Service Composition and Qos Service composition

The service composition is the action of composing services in order to get one service that will fulfill client need. Most services are designed to accomplish one task at a time while clients are more interested to have elaborated

Manuscript received November 5, 2019
Manuscript revised November 20, 2019

services that solve one of their needs in an overall view approach. The QoS (Quality of Service) service composition is the QoS of the composed service [8].

3. Related Work

The problem of service composition has received a lot of attention. The work in [3] is dealing with automating composition for REST and SOAP services based on ontology, however there is too much focus on REST with neglecting SOAP particularities.
In [11], the need for a description machine understandable language for composition is prominent; however the proposed languages like WADL, WSDL2.0 are influenced by the existing language and describe only input/output interactions, while REST is about resources and state transfer between resources.
In [4], a lot of effort has been done to compose services using existing WSDL, but since SOAP is operation centric and REST is resource centric, the approach has a lot of difficulties. The Mashups are data applications that combine data from different sources, e.g: database sources, resource set and combine it. Mashups are usually limited to data combining and thus not dealing with real REST and SOAP service composition.
In [6], the authors have been through multiple articles dealing with REST and service composition. They have talked about Mashups and they have divided them in three categories:
- Data-oriented Mashups: convert, transform and combine similar data elements represented by resources into one single resource;
- Process-oriented Mashups: integrate with existing business services new business services;
- Consumer-oriented Mashups: consumer can do its own customization (eg Yahoo pipes inspired from UNIX pipes).
They have also tackles service discovery and composition.
The problem in REST is that the Client has to find the desired web services prior to composing it with other services, for SOAP, the UDDI is playing this role, while in REST you need to look up for URLs. Researchers have proposed different approaches for finding services. These approaches are: Programmable web which collects and categorizes published services, SA-REST (Semantic annotations of web resources) which handles the semantic part of the REST composition, and MicroWSMO which is the micro format based on web service modeling ontology.
The article dealt also with Heterogeneous composition approaches i.e. combining SOAP with REST, the authors has proposed the extension of the BPEL (Business Process Execution Language) to enable composition of both SOAP and REST using the WSDL2.0 which is based on HTTP bindings, this approach will help to describe the RESTful services.
BPEL process can be published as RESTFul WS by exposing certain parts of its execution state using the REST primitives (PUT, POST, DELETE, and GET).
BPEL extensions map the resource abstraction and support the corresponding interaction mechanisms and invocation patterns: BPEL activity for each REST primitive to invoke RESTful services, and dynamic resource representation to expose the execution state of a BPEL process to clients, then the revision of BPEL constructs to fit with REST design principles.
The work in [12] presented the standardized interface for placing a service order with all the parameters needed by a group of professionals, the work focus on the REST API.
In[5], the authors tackle the security domain with QoS Aware for REST composition, they were focused about the decentralized and asynchronous aspect of REST.
In [7], the journal has gathered all the articles treating the web service composition in a conceptual way.
The work in [9, 10] establishes the guidelines and the basics of ontology and used it to describe the world in a very abstracted way, which gives the foundation of ontology science.

4. Problem Statement

4.1 Definitions for the new notions introduced in the problem

In order to better understand the problem, we will introduce concepts and their definition to ease the understanding of the situation.

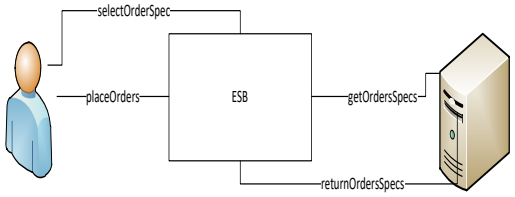

Fig. 1 request-response flow

- The client sends a request orders to a specified URL.
- The URL and the HTTP method are combined to identify which action to trigger.
- When a URL is triggered with the GET http Method, a request is submitted to the ESB (Enterprise service Bus).
- The ESB adds the necessary properties to the message, and then submits the message to the provider.

- The provider uses a technical catalog based on the specification sent for each order then returns the order, with its characteristic and the supported delivery: in our work, we are more focused on REST and SOAP.

Since modern clients are using REST architecture, we suppose that the client will be placing orders using REST and specifying the order with JSON (JavaScript Object Notation).

For the sake of simplicity, we suppose that the client has a functional catalog which associates the desired function with its mapping to an order expressed in JSON format. The set of desired functions will be mapped to a JSON order which contains the sum of all desired order items.

Table 1: order Properties and description.

Item	Description
order	One or many orders
orderId	Unique ID for the order
orderAttributes	A set of order attributes, each element has an attribute name and an attribute value
orderMethod	The functional method Name to be invoked for the order
httpAction	The action to be performed
orderMeta	A set of order MetaIds
orderMetaId	Unique Id for the order MetaID
orderMetaProperties	Properties with name and values for the QoS of the order

Table2: placeOrders in JSON

```
{"orders": [{
"order": {
"orderId": "100",
"orderAttribute": [{
"name": "Attribute1",
"value": "value1"
}, {
"name": "Attribute2",
"value": "value2"
}
]
},
"orderMethod": {
"methodId": "method1"
},
"HttpAction": "DELETE"

}, {
"order": {
"orderId": "102",
"orderAttribute": [{
"name": "Attribute12",
"value": "value12"
}, {
"name": "Attribute22",
"value": "value22"
}
]
},
"orderMethod": {
"methodId": "method12"
},
"HttpAction":"POST"
}
],
"id": "1001",
"ordersDate": "19/11/2019 22:22:00"
}
```

The list of orders is been sent to the ESB, each order is identified by an orderId and contains attributes and values. Attributes could be one to many attributes, we specify the orderMethod that is retrieved from the business catalog which match the functional need asked by the client, and we specify the HTTP action to use. The ESB processes the message and assign an orderId unique to the order and add the header information's for routing and retrieving back the response.

Table3: getOrderSpecs in JSON

```
{"orders": [{
"order": {
"orderId": "100",
"orderAttribute": [{
"name": "Attribute1",
"value": "value1"
}, {
"name": "Attribute2",
"value": "value2"
}
]
},
"orderMethod": {
"methodId": "method1"
},
"HttpAction": "DELETE",
"HttpID": 701,
"OrderMetaInfo": [{
"estimatedTime": 0.01,
" throughput ": 0.7,
"cost": 0.23,
"availability": 1,
"archi": "REST",
"orderMetaId":"8001"
},
"estimatedTime": 0.012,
"throughput": 0.76,
"cost": 0.23,
"availability": 1,
"archi": "SOAP",
"orderMetaId":"8002"
```

```
}], {
"order": {
"orderId": "102",
"orderAttribute": [{
"name": "Attribute12",
"value": "value12"
}, {
"name": "Attribute22",
"value": "value22"
}
]
},
"orderMethod": {
"methodId": "method12"
},
"HttpAction": "POST",
"HttpID": 702,
"OrderMetaInfo": {
"estimatedTime": 0.02,
"throughput ": 0.4,
"cost": 0.25,
"availability": 1,
"archi": "SOAP",
"orderMetaId":"8001"},
}
],
"id": "1001",
"ordersDate": "19/11/2019 22:23:00",
"ordersStatus": "finished"
}
```


Once the ESB submits the orders to the provider, the provider details for each order the set of orderItemMeta which contains the QoS of the service and a uniqueId for the orderItemMeta in case it is selected to launch the execution of the order. Moreover, the response returns the architecture supported to execute the order. For sake of simplicity, we are focused on SOAP and REST.

5. Advantage of the Approach

5.1 Benefits on Overall Architecture

The proposed architecture and approach will help the client be more involved in orderings and choosing what is more convenient for them.

The introduction of the ESB on the middle of the architecture will reduce the technical details for the client and help route, enrich and isolate the client side from the provider's side.

More security is provided by introducing the new layer that will separate the client side from the server side.

5.2 Standardization of the Exchange Messages

The ordering way is toward the standardization of the ordering which in such case, we need to have the set of the orders, the id of the orders and their characteristic as wished by the client. This need captures will then be translated lately to technical orders with more information.

5.3 Better Visualization of Orders Information

Our proposed solution is to respond in the same order and fulfill it with more technical details especially for the orderMetaInfo which will bring more information on quality of service and the way to deliver it in our case either by SOAP or REST. Then it will make the path to let the client decides in case the number of returned orderMetaInfo is reduced. In case the number is more significant then we can proceed with the optimization approaches previously demonstrated [8].

6. Conclusion and Perspective Works

In this work we highlight the advantages of RESTful Web Services and propose an approach to work with both SOAP and Restful web services. The approach also emphasizes the return of quality of service for each service introduced in the composition. In this way we can refer to the previous work [8] done on optimizing the composition of web service with minimal costs and better overall performance. The approach proposes a standardized interface in which a client can perform orders, then our ESB with the consultancy of the providers will return for each order a set of orderMetaAttributes which they will all fulfill the same functional tasks but with different QoS. We can then help the customer choose to optimize the convenient combination of the orders or he can have his own choice of composing the orders as preferred. As a perspective, we want to validate the approach with fewer hypotheses and using the ontology as a mechanism for better automating the composition.

References

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

[2] R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. In ICSE, pages 407–416, 2000.

[3] H. Zhao. Towards Automated RESTful Web Service Composition. 2009 IEEE International Conference on Web Services

[4] World Wide Web Consortium (W3C) Working Draft 3: Web Services Description Language (WSDL) Version 1.2 (2003). http://www.w3.org/TR/wsdl12/

[5] C. Supelveda. QoS aware descriptions for RESTful service composition: security Domain. Springer Science+Business Media New York 2014.
[6] M. Garriga, C. Mateos. RESTful service composition at a glance: A survey. Journal of Network and Computer Applications, 2015.
[7] https://www.sciencedirect.com/topics/computer-science/service-composition.
[8] A.BENTALEB, A.ETTALBI. Toward Cloud SaaS for web service composition optimization based on genetic algorithm.CloudTech 2016, Marrakech, Morocco.
[9] https://www.w3.org/Submission/2004/SUBM-SWRL-20040521
[10] J. McCarthy. Situations, actions and causal laws. Technical report, AI Laboratory, Stanford University, 1963.
[11] M. J. Hadley. Web application description language (wadl) specification. https://wadl.dev.java.net, 2006
[12] Service Ordering Management API REST Specification: https://www.tmforum.org/resources/specification/tmf641-service-ordering-api-rest-specification-r18-5-0/, 04/05/2019