REST2SOAP: a Framework to Integrate SOAP Services and RESTful Services

Yu-Yen Peng¹, Shang-Pin Ma², and Jonathan Lee³
Department of Computer Science and Information Engineering, National Central University, Taiwan¹, ³
Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan²
965202007@cc.nchu.edu.tw¹, albert@mail.ntou.nchu.edu.tw², yjlee@selab.csie.ncu.edu.tw³

Abstract—Nowadays, the mainstream of Web 2.0 website services is in the REST style called RESTful web service. RESTful services have been widely accepted by the public because of the usability and simplicity. Meanwhile, web service technologies realize service-oriented architecture (SOA) successfully and are exploited in both industry and academia. Notably, service composition that can aggregate existing services into a new one delivers the most benefits of SOA through BPEL standard. However, BPEL cannot support RESTful services and integrate both Web 2.0 and SOA-based resources in a composite service directly. It means that it is costly or time-consuming to utilize RESTful services in an SOA-based application. Therefore, this paper presents a framework, namely REST2SOAP, to integrate SOAP services and RESTful services. REST2SOAP leverages WADL specification, and can wrap RESTful services into SOAP services semi-automatically. Using REST2SOAP, developers can create a BPEL service that combines SOAP, RESTful services and user interfaces simultaneously.

Keywords—Web 2.0; REST; RESTful service; SOA; SOAP service; service composition; BPEL

I. INTRODUCTION

Web 2.0 [12, 14] recent years ignited innovative and successful web applications such as Blogger, Flickr and YouTube. Although the term of Web 2.0 is regarded as a concept of the development of user-centric web [16], most Web 2.0 websites provide their own web services that are realized in REST fashion. REST [6] (REpresentational State Transfer) is a style of software architecture for distributing hypermedia systems such as the World Wide Web. REST defines a set of architectural principles [6, 15] by which one can design web services that focus on a system’s resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. The principles of REST include: (1) application state and functionality are abstracted into resources, (2) every resource is uniquely addressable using a universal syntax for use in hypermedia links, (3) all resources share a uniform interface for the transfer of state between a client and a resource, and (4) every interaction with a resource is stateless. Services following the principles of REST are often referred to “RESTful”, or RESTful services. Today the RESTful services have been widely accepted by the public because its design principles are simple and easy to follow.

As for SOA domain, web service technologies, such as SOAP [3] and WSDL [5], realize service-oriented architecture (SOA) successfully and are adopted in both industry and academia. Notably, web service technologies are best exploited by composing web services [9]. Composing multiple web services is more beneficial to users than a single service [11]. The standard of service composition is Business Process Execution Language (BPEL) [1, 13]. BPEL is a language used for composition, orchestration and coordination of WSDL-based web services that are usually implemented using SOAP. BPEL also provides a rich vocabulary for expressing the behavior of business processes.

Composition in Web 2.0, often called “Mashup” [4, 10, 17], is a web page or website that combines information and services from multiple sources on the Web. Although mashup notion can bring innovative composite applications that offers value added services for users, it is hard to gain widespread acceptance by enterprises because there is no standard process or specification to embody mashup. Since there is a certain degree of similarity between SOA and Web 2.0, a mashup application can be treated as a composition of loosely-coupled and reusable services, and BPEL can be the standard way to connect and manage these services. However, nowadays BPEL cannot integrate and invoke RESTful services currently due to there are a lot of differences, such as styles of operations, data types supported and representation media, between a RESTful service and a WSDL-based web service. It means that it is difficult or time-consuming to utilize RESTful services in a BPEL-based composite service. Therefore, this paper presents a framework, namely REST2SOAP, to integrate RESTful services into BPEL by transforming a RESTful service into SOAP services semi-automatically.

The rest of this paper is organized as follows: the REST2SOAP framework is described in Section II, and the conclusion of this work and future research directions are presented in Section III.

II. REST2SOAP FRAMEWORK

To realize the aforementioned integration of RESTful web services and SOAP web services, a REST2SOAP framework is introduced in this section.
A. Framework Overview

This framework (see Figure 1) is designed for two major actors: mashup service assembler and user.

- **Mashup service assembler**: mashup service assembler can use the framework to transform the RESTful services into SOAP services for further composition, or to deploy a composite service through uploading a BPEL document.
- **User**: user can use the composite service provided by the mashup service assembler.

In the framework, four kinds of documents are involved:

1. **WADL document**: the Web Application Description Language (WADL) [8] is an XML-based file format that provides a machine-readable description of HTTP-based web applications. These applications are typically RESTful web services. The proposed framework applies WADL to convert RESTful services into SOAP services.
2. **WSDL document**: WSDL is used to describe the interface of an atomic service in a composite service.
3. **XUL document**: XUL (XML User Interface Language) [2] is an XML user interface markup language that relies on multiple existing web standards and technologies, including CSS, JavaScript and DOM. XUL provides a portable definition for common widgets, i.e. elements of a graphical user interface (GUI). XUL is to make a BPEL service equipped with user interfaces in this framework.
4. **BPEL document**: it is to describe the flow of the composite service and can be deployed in the BPEL engine to produce an executable composite service for users.

The principle task of REST2SOAP framework is to convert RESTful services into SOAP services, so mashup service assembler has to provide the description document of RESTful services — WADL. The framework will convert the RESTful service described by a WADL document into a SOAP service so that the mashup service assembler can integrate it with other SOAP services to create a new composite service, namely BPEL service.

B. System Architecture

To realize BPEL-based service composition, the system is built on a BPEL engine as shown in Figure 2. Mashup service assembler can access the primary module “RESTful Service Publisher” to convert RESTful services into SOAP services, and deploy mashup services via this module. Regarding the inconvenience and complexity while developing the WADL document of RESTful service, the system integrates the “WADL Designer” [19] to assist mashup service assembler in developing WADL document. The RESTful service publisher module in system is as a façade [7] to interface other modules and control the whole REST2SOAP process, so mashup service assembler does not need to worry about how to manipulate other modules to accomplish the converting process. All the mashup service assembler has to do is uploading the WADL document or BPEL document.

RESTful service publisher module utilizes three main modules: WADL Convertor, Proxy Generator and REST2SOAP Deployment Assistant. Functionalities of each module are described in detail as follows.

1. **WADL Convertor**

   WADL convertor module is designed to deal with validating WADL document and generating WADL content tree. Validating WADL document is the first step in the REST2SOAP process, and only validated WADL can be accepted in the following steps. The root element of WADL is <application> representing a RESTful service. The <resource> element is the sub-element of <application> to organize resources provided by the RESTful service since the most important feature of REST architecture is the concept of resource abstraction. In addition, every resource posses one or more methods for users to perform different functions, and these methods are usually HTTP methods, such as GET, POST, DELETE and PUT.
The proposed approach to perform the validation of WADL document is by comparing with a standard content tree. This standard WADL content tree is built according to the specification before parsing the WADL document. When this module traverses the WADL document against the standard content tree, it will check all elements and assure each element contains legal children elements and attributes. After validating WADL document, WADL convertor module will establish a specific content tree. This content tree can be treated as an internal representation of the WADL document in the system. Another aim of WADL content tree is to be a basis for generating proxies.

2. Proxy Generator

Proxy generator is a module which generates the proxy of corresponding WADL content tree. A proxy in the system is a function-oriented class. In addition, proxy generator module also leverages WADL2Java [15], a tool of generating stub code, to generate executable stub code of corresponding RESTful service. Proxy generator module relies on the stub code and WADL content tree to generate the proxy of a RESTful service.

Establishing a SOAP service needs a service interface and a WSDL document. The operation names of the service interface are usually meaningful, and the parameters of an operation are typed explicitly. However, the generated stub from WADL2Java currently lacks meaningful operation names. Therefore, the system has to generate a proxy with a well-defined service interface, and use the proxy to obtain the WSDL.

To achieve above goals, proxy generator module has to establish a function list first. Proxy generator module should produce operations of the proxy class based on the function list which is extracted from the WADL content tree. Besides, proxy generator should follow a mapping rule to convert data type in WADL into Java data type when infusing the function list into the proxy class. Consequently, a mapper component is designed to assist proxy generator module in dealing with these issues.

The mapper is responsible for two major mapping tasks: proxy function name mapping and Java data type mapping.

a.) Proxy function name mapping

The most important feature of the design principles of REST architecture is abstracting the functionality of services into resources, and every resource has to be addressable, that is, every resource owns its URL. Furthermore, each resource can be accessed via HTTP methods. As a result, the mapper component combines nested resource paths and HTTP method to construct the function name. The mapping rule of function name is shown as “(1).”

Function name = [HTTP method]_{Resource Combination}  (1)

For example, Delicious service provides two functions: retrieve tag and rename tag. The service has been described as a WADL document as shown in Figure 3. The first function is “get_GetTags()” which means this function uses “GET” HTTP method, and its purpose is to retrieve tags. The second function is “post_RenameTags” which means this function uses “POST” HTTP method, and its purpose is to replace old tags with new ones. According to the mapping rule, every function in a proxy class is easily understood by its name, and meaningful function names are very helpful when converting into WSDL document. Moreover, users can modify or augment these operation names of the proxy if they have other concerns.

b.) Java data type mapping

Each <method> element contains input/output information, and <param> element represents the input variable. The types of the input variables are XML-based, either XML simple types or user-defined complex types. When proxy generator establishes operations of a proxy, these XML-typed input variables will be the parameters of the operation in the proxy and should be converted into Java types. Hence, proxy generator requires a mapping rule to transform a XML schema type into a Java type. Thus, the mapping table is adopted to transform XML-typed parameters into Java types as shown in table I.

<table>
<thead>
<tr>
<th>XML Type</th>
<th>Java Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsd:boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>xsd:byte</td>
<td>byte</td>
</tr>
<tr>
<td>xsd:date</td>
<td>java.util.Calendar</td>
</tr>
<tr>
<td>xsd:decimal</td>
<td>java.math.BigDecimal</td>
</tr>
<tr>
<td>xsd:double</td>
<td>double</td>
</tr>
<tr>
<td>xsd:float</td>
<td>float</td>
</tr>
<tr>
<td>xsd:hexBinary</td>
<td>byte[]</td>
</tr>
<tr>
<td>xsd:int</td>
<td>int</td>
</tr>
<tr>
<td>xsd:integer</td>
<td>java.math.BigInteger</td>
</tr>
<tr>
<td>xsd:long</td>
<td>long</td>
</tr>
<tr>
<td>xsd:QName</td>
<td>javax.xml.namespace.QName</td>
</tr>
<tr>
<td>xsd:short</td>
<td>short</td>
</tr>
<tr>
<td>xsd:string</td>
<td>java.lang.String</td>
</tr>
<tr>
<td>xsd:anyURI</td>
<td>java.lang.String</td>
</tr>
</tbody>
</table>

For example, “add a bookmark” in Delicious service contains a URI parameter, and according to the mapping
rule this parameter will be converted into Java String type (See Figure 4).

![Diagram of data type mapping]

**Figure 4** Java data type mapping

3. REST2SOAP Deployment Assistant

REST2SOAP deployment assistant module uses Java2WSDL [18] tool to generate the WSDL document of the proxy. Then this module parses the proxy class to generate deploy description file preparing for deployment, and to deploy the proxy to AXIS and to produce a new SOAP service.

III. CONCLUSION AND FUTURE WORK

This paper has presented a framework to integrate SOAP services and RESTful services through a REST2SOAP framework. Mashup service assemblers can convert a RESTful service into a SOAP service so that this RESTful service can be integrated with other SOAP services to construct a mashup application embedding a composite service for end users. Another significant strength of the proposed approach is reducing integration workload. Since wrapping RESTful services into SOAP services is almost an automatic task, mashup service assemblers do not need to build adapter services or glue codes to link RESTful services within a BPEL-based composite service.

As for the future work, one of the advantages of RESTful services is the various media types, such as Image, JSON, or Atom. Since REST2SOAP can handle string data type and XML message currently, supporting other media types for RESTful services is the next improvement of this framework. In addition, to fully exploit the potential of Web 2.0, a widget framework to build mashup applications is another future direction in order to enrich user experiences.

ACKNOWLEDGMENTS

This research was partially sponsored by National Science Council under the grant NSC 97-2631-H-008-001, and partially sponsored by Ministry of Economic Affairs in Taiwan under the grant MOEA/WRA0980008.

REFERENCES


[18] Java2WSDL. Available at http://ws.apache.org/axis2/ja/japect-guid.html#Java2WSDLBuildingWSDLFromJava