JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IMPLEMENTATION METHOD OF ENTERPRISE SERVICE BUS SUPPORTING RESTFUL WEB SERVICES COMPOSITION

Kwang-Myong Jo [1 *], Nam-Hyok Jo [2], Song-Sun Jo [3],
Department of Information Science,
"**Kim Il Sung** University", Pyongyang, DPR of Korea

e-mail: edu25jkm@163.com, gfstk@star-com.net.kp

Abstract: *Web service composition is becoming one of the most appropriate methods in implementing enterprise application integration. In the paper, we propose a method to make the enterprise service bus(ESB) support heterogeneous web service composition (This includes the RESTful web service as well as the SOAP web service). In the paper, we analyse the characteristics of RESTful web services in aspects of service composition and present a business process execution algorithm and RESTful web service call algorithm based on the formulation of business process supporting SOAP and RESTful web service composition. Basing ourselves on the previous approach to extend the BPEL execution engine and invoke RESTful web services directly, and adding to it different content-types of RESTful web services and the way to map the data transferred to the path parameters to XML-based BPEL, we formulate the business process and newly extend the processing algorithm of the business process execution engine. At the end, we apply the suggested method for constructing the intelligent hospital information system and evaluate the effectiveness of web service composition.*

Keywords: RESTful web service, Enterprise Service Bus, Web Services composition, Business Process Execution Algorithm

I. INTRODUCTION

The researches to construct the complex intelligent hospital information system (this system includes sub-systems such as EMRS (Electronic Medical Record System), CPOE (computerized provider of entry), NIS (Nursing Information System), LIS (Laboratory Information System), PIS (Pharmacy Information system), and HRPS (Hospital Resource Planning System is in process [5, P. 459-528]. The development of the information system is being carried out based on ESB, which supports the implementation of the service-oriented architecture, as it requires a constant update of an enterprise system.

As the existing ESB, which we are going to apply to the intelligent medical service system, only supports SOAP-based web service composition, it can't easily be applied to the integration of the intelligent medical service system, where the majority of the web services are RESTful. Therefore, the existing ESB should be expanded to support RESTful web service composition as well as traditional SOAP web services.

This paper suggests a service composition method for SOAP and RESTful web services from the viewpoint of constructing an intelligent medical service system.

II. METHODOLOGY FOR RESEARCH

Service providers have recently started using RESTful services to make service provision and consumption easy, and it is possible to support different forms of information integration of application systems by implementing the composition of RESTful services and SOAP-based web services [2, pp. 851-866; 3, pp. 1-4; 4, pp. 142-159].

Business process involves the flow and use of information and resources as a set of activities that define the steps required to achieve the corresponding business goal.

In order to enable an ESB to synthesise not only SOAP web services but also RESTful Web services to define and execute business processes, it is necessary to define a dedicated activity node for the RESTful web service call and update the BPEL engine to configure and execute business processes using it.

First, the components for registering RESTful web services include URI, method, path parameter type, input content-type, input parameters, output content-type and output parameters. In the case of SOAP web services, only the URL that reflects the location information of the service and WSDL that reflects all the information of the service should be registered, but in the case of RESTful

web services, the URI that reflects the location information of the service, the standard HTTP methods provided by the service, the path parameter type information, the input and output parameters, and its content-types should be registered.

Second, the main content-type used in RESTful web services is JSON, and in addition to using XML and Form content-type. The business process, composed of SOAP Web services, represented the input and output data of SOAP Web services based on XML, and the data processing was performed using XPath and XSLT. However, as JSON is used as the main content-type of RESTful web services, it is necessary to implement the functionality of interactive data transformations between XML and JSON to perform the I/O process related to service calls within the business process.

Third, the RESTful web service implements a service method based on stateless operations (CRUD: Create, Read, Update, and Delete) and thus the service execution mode proceeds in a synchronous manner. As the service call of the RESTful web service takes place in a synchronous manner, unlike the SOAP Web service, if the service call fails, the service call must return the service call result to the call side of the business process rather than stop the execution of the business process.

The formulation of a business process that supports RESTful Web service composition is of great importance in updating BPEL and the BPEL execution engine that runs it, without adversely affecting the core function and performance indicators of the existing ESB.

As analysed in previous work, among Web service composition methods based on SOAP and RESTful web services, the method of directly invoking RESTful services by extending the BPEL of the corresponding ESB has the advantage that it is a relatively simple method that can fully implement the diversity of REST web services by adding information that can represent RESTful web services to the existing BPEL language and adding rules that can interpret and execute it to the BPEL execution engine[1, P. 110-119, 2, P. 851-866].

However, the previous approach to directly invoke RESTful services by extending the BPEL execution engine did not propose a way to map different content-types (JSON, XML, and FormData) of RESTful web services to XML-based BPEL in the context of service composition, the most core of the functions of the ESB. It also did not discuss how to handle data transferred to path parameters in RESTful web services. If the previous approach is applied, it is impossible to use data transmission by the inherent route parameters of RESTful web services or different content-types, but it makes it possible to perform service composition only for XML content-type on a fixed path.

Hence, basing ourselves on the previous approach to extend the BPEL execution engine and invoke RESTful web services directly, and adding to it different content-types of RESTful web services and the way to map the data transferred to the path parameters to XML-based BPEL, we formulate the business process and newly extend the processing algorithm of the business process execution engine.

[**Definition 1**] The business process is defined as a set of activities, a set of transitions, a set of variables, a set of SOAP web service call interfaces, and a set of RESTful web service interfaces as follows.

$$(1)\ BP = \{Acts, Trs, Ds, W, R\}$$

- ✓ $Acts$: A set of $Activitys$ that form a business process
- ✓ Trs: A set of $Transitions$ that connect between Activities
- ✓ Ds: A set of $DataField$ to implement the processing logic of a business process
- ✓ W: A set of SOAP Web service call Interfaces ($WsInterface : WI$)
- ✓ R: A set of RESTful web service call interfaces ($RESTful\ Interface : RI$)

The services used in the business process must be registered in W and R, respectively, as service call interfaces before the business process is defined.

[**Definition 2**] An activity node is defined as a set of names, identifiers, types, attributes, association types, and position as follows.

(2) $Activity = \{name, id, type, ActivityAttr, joinType, pos\}$

- ✓ $name, id$: The name and identifier of the activity node.
- ✓ $type$: is an element of Types (set of types of nodes) and is defined as follows. $Types = \{START, END, ROUTE, INVOKE, RECEIVE, TRANSFORM, REST\}$
- ✓ $ActivityAttr$: As a set of node attributes according to the type of activity, the following are:

  $ActivityAttr = f(t \in Types) = \{si | i \in N, s \in AttrS\}$

  $AttrS$ is an entire set of node attributes as follows.

  $AttrS = \{ActLogic, Pre\ Trans, AfterTrans, ErrorAct, CorSet\}$

  The meaning of the function f indicates that the node attributes change according to the activity node type, and the $ActivityAttr \subseteq AttrS$ relation holds true.
- ✓ $joinType$: It is an element representing the joint type of activity node and is defined as follows.

  $Enum(AND, XOR)$
- ✓ pos: In the edit state, the positions of the top left and bottom right corner points of the activity node are shown as follows.

  $pos = (leftX, leftY, rightX, rightY)$

[**Definition 3**] The RESTful web service call interface ($REST\ Interface$; RI) is defined as a set of service ids, service name, service description, URI, service method, parameter type, input content-type, input parameters, output content-type, and output parameters, as follows.

(3) $RI = \{id, name, des, URI, Method, PramType, InputMediaType, InputParam, OutputMediaType, OutputParam\}$

- ✓ URI: The call address of the web service
- ✓ $id, name$: The id and name of the RESTful web service
- ✓ des: RESTful web service description
- ✓ $Method$: A set of methods (opi) of RESTful web service
  $Method = \{op1, op2, ..., opn\}, n \in N$
- ✓ $ParamType$: The parameter type of RESTful web service
  $ParamType = \{QueryParam, PathParam, MatrixParam, HeaderParam\}$
- ✓ $InputMediatype, OutputMediatype$: The input and output content-type of RESTful web service.

$InputMediatype, OutputMediatype = \{json, xml, form\}$

- ✓ $InputParam, OutputParam$: Input and the output parameter name of RESTful web service.

The formulation of the business process that supports SOAP and RESTful web service composition functions generates a relevant data structure for the definition and implementation of the business process.

Based on the formulation of the business process supporting SOAP and RESTful web services composition, we propose the main algorithm of the Business Process Execution Language (BPEL) Engine and the service call method of the RESTful web service activity. The main algorithm of the business process execution engine that supports service composition of SOAP and RESTful web services is as follows.

Step 1: Creates the relevant business process. In this step, we initialise the business process instance and then move to step 8.

Step 2: Sets the activity node instance. In this step, we check whether there is an activity node instance to be executed and set up.

Step 3: Check the time exceeding.

Step 4: Perform a pre-transformation for the web service call. In this step, if the transformation before the web service call is valid, perform a pre-transformation.

Step 5: Invoke the relevant web service. A detailed method of RESTful web service call is described in the following section of the execution algorithm of the RESTful web service activity.

Step 6: Perform transformation after web service call. In this step, we perform post-transformation if the transformation after the web service call is valid.

Step 7: If an error occurs, the error condition is checked. In this step, the error condition is evaluated, and the corresponding processing is performed.

Step 8: Check the transition condition. In this step, we determine the transition condition and then the activity instances to be executed.

Step 9: Do the final check. In this step, we check whether there is an END activity node.

Step 10: Complete the execution of the business process. In this step, we set the process instance to completion and finish.

Based on the main algorithm of the business process execution engine, update the BPEL engine of the ESB.

III. RESULTS AND DISCUSSION

We have evaluated the accuracy of the definition and implementation of business processes based on the registration of RESTful web services. To do this, we have registered more than 200 RESTful services provided by more than 30 different business systems developed in Java, PHP, and C++ languages and analysed their feature details (Table 1).

Table 1. Details of registered RESTful web services.

Index name	Sub index	Count(%)
MethodType	GET	66
	POST	31.6
	PUT	1.2
	DELETE	1.2
ParameterType	PathParam	4.2
	QueryParam	89.2
	HeaderParam, MatrixParam	6.6

Input data type	JSON	75.2
	Form	24.8
Output data type	JSON	95.6
	XML	3.2
	Form	1.2

As shown in the table, we can see that the function of registration of RESTful web services provided by the ESB system accurately reflects all details of RESTful. The business process call performance of an ESB system supporting SOAP and RESTful web service composition is as follows (Table 2).

Table 2. Performance indicators of the process call

Case	Process call count	Success rate(%)	Average time(ms)	Maximum time(s)	Maximum data size(Mbyte)	CPU Usage(%)	Memory Usage(%)	Network traffic(Mbps)
1	100,000	99.81	150	1.23	8.5	18	65	0.8
2	250,000	99.83	180	1.25	8.5	23	68	1.2
3	500,000	99.85	200	1.23	8.5	45	70	1.42
4	750,000	99.89	200	1.24	8.5	50	72	2.0
5	1,000,000	99.98	200	1.23	8.5	48	72	2.02

As shown in the table, we can see that the average time of the business process is 200ms, which fully satisfies the characteristics of a RESTful web service that makes synchronous calls. It also shows that the consumption of system resources is very stable at certain limits.

IV. CONCLUSION

This paper introduces the methods and their algorithms to extend the ESB that provides only SOAP-based web service composition so as to support RESTful web service composition.

In this paper, our study is focused on building a business process execution engine and updating BPEL that can directly invoke a RESTful web service. Future studies will include proposing methods for enhancing the load distribution and exception throughput of business process execution.

REFERENCES

[1] Haupt F, Fischer M, Karastoyanova D, Leymann F, Vukojevic-Haupt K. 2014. Service composition for REST. In: Proceedings of the 18th International Enterprise Distributed Object Computing Conference (EDOC). September: P. 110-119

[2] Pautasso C. 2009. RESTful Web service composition with BPEL for REST. Data Knowl Eng 2009; 68(9): P. 851-866

[3] Peng Y, Ma S, Lee J. 2009. REST2SOAP: a framework to integrate SOAP services and RESTful services. In: Proceedings of the international conference on service-oriented computing and applications (SOCA). 2009. P. 1-4

[4] Pautasso C. 2009. Composing RESTful services with JOpera. In: Bergel A, Fabry J, editors, Software composition. Lecture notes in computer science, 2009. Vol: 5634: P. 142-159

[5] 李兰娟. 2015. “新型智能医院”, 科学出版社 医药卫生出版分社. P.