http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (5): 786-790, 2009
ISSN 1812-5638

Composing Disparate Services and Data Dynamically Based on EBS

Shu-Qing Peng and De-Yun Chen
College of Computer Science and Technology,
Harbin University of Science and Technology, Harbin, 150080, China

Abstract: In this study, a new framework that disparate services and data are composed dynamically based on ESB is proposed, in which business process is analyzed into multiple work-data flows. Dynamic routing mechanism is applied to map abstract description of services and data to their respective providers. This new framework can implement dynamic composition of disparate services and data.

Key words: Dynamic composition, business process, work-data flows, dynamic routing, updating algorithm

INTRODUCTION

It is very important for enterprises to utilize disparate services and available data on web to integrate a system quickly and with little cost (Kuppuraju *et al*., 2007). This is dynamic composition of services and data. There are many research works about this problem. Fujii and Suda (2005) presented a semantic-based dynamic service composition system which provides a functionality of semantics-awareness, middleware and semantics-based service composition mechanism. Penta *et al.* (2006) described WS binder which is a framework for making dynamic binding of service compositions feasible according to functional and nonfunctional preferences and constraints. It supports three kinds of binding types including pre-execution binding, run-time binding and run-time rebinding. Laliwala proposes a semantic-based and rule-based dynamic composition method which generates business processes (Laliwala *et al*., 2006). The generation of business processes is based on event-condition-action rule, forward chain algorithms and backward chain algorithms. Chafle *et al.* (2006) proposed a solution for creating and executing web services, which can adapt to changes in the operating environment at the stages of composition, deployment and runtime.

ESB is often used as the infrastructure for service connection and message exchange in SOA (Schmidt *et al*., 2005). A key feature of ESB-enabled SOA is that the service definition is separated from the mechanism for locating and invoking services. Services and data can be deployed over intranet and via the internet to support business process implementation (Alonso and Casati, 2005). Each service, often described in WSDL, has an interface and access method. Data often lie in databases on network and data access will occur in different computers.

Disparate services and data are loosely connected to each other, which makes flexible application development possible. At the same time, ESB is also a standard integration platform that supports messaging, web services, data transformation and intelligent routing to reliably connect and coordinate the interaction of diverse applications (Jean-Louis, 2006). ESB mainly consists of four elements. Message Oriented Middleware (MOM), web services, intelligent routing based on content and XML data transformation. The core functionality of intelligent routing is supporting distributed applications and disparate services to communicate with each other using different protocols such as HTTP, SOAP, JMS, MQ, SMTP, FTP and etc., (Min *et al*., 2005). But current routing techniques in ESB can only support static routing table definition (Xiao-Ying and Ji-Hui, 2007). This routing mechanism uses the uris of service providers and datasources to describe routing paths. The routing path is fixed once the ESB is initialized and can not be changed.

In this study, a new framework is presented, in which business process is analyzed into multiple work-data flows which includes activities implemented dynamically by disparate services and data. The weight computation method for evaluating the degree that a business process is analyzed into activities is proposed here. Dynamic routing mechanism is used here.

DYNAMIC COMPOSITION FRAMEWORK

A business process always consists of many activities and some data may be used in its implementation. After these activities have been

Corresponding Author: Shu-Qing Peng, College of Computer Science and Technology,
Harbin University of Science and Technology, Harbin 150080, China Tel/Fax: 86-0451-53623726

implemented on data in some sequence, the business process is over. These activities can be implemented by provided services on EBS and these service providers may be located at different computers on network. In order to utilize data and services on network to carry out these activities, ESB is used to connect requesters and providers here. A new framework to compose services and data dynamically is described in Fig. 1.

There are three parts in this framework including business process layer, enterprise service bus layer and service-data implementation layer. Business process layer is responsible for analyzing the business into activities and data. Enterprise service bus layer takes charge of integrating disparate services and distributed data. There are lots of providers which publish datasources and services in service-data implementation layer.

Business process layer: In business process layer, the analyzer analyzes the business process into work-data flows which consists of activities and data based on ASDT (Abstract Service Description Table) and ADDT (Abstract Data Description Table). At the same time, these activities and data are specified. The reason of considering services and data access separately is that data access can be implemented by expanded sql commands in constant time, but the size of one service module is different from others. This is convenient and accurate to separate business process into activities.

Logically, several work-data flows can be analyzed from one business process. This is because you can consider it at different points of view. May be you separate a business process into 9 activities, while another person analyzes it into 6 activities. But when you implement these activities on data, you can all finish the business process.

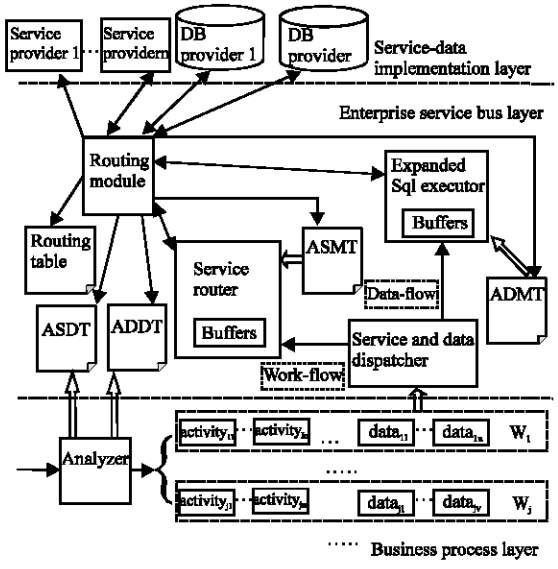

Fig. 1: Dynamic composition in business process

A weight is given for every work-data flow to reflect the rationality of analyzing. For example, a business process can be separated into $activity_{11}$, …, $activity_{1n}$, $data_{11}$, …, $data_{1u}$ and weight w_i is given to it. At the same time, $activity_{j1}$, …, $activity_{jm}$, $data_{j1}$, …, $data_{jv}$ are gotten and its weight is set to w_j. The analysis process is based on the following principles:

- A business process should be separated into several steps according to its logical work sequence. Every step is seen as an activity. For every activity, if we can not find a service in ASDT to implement it, the activity should be analyzed
- The analyzer will look up ASDT and finds fewer and more creditable services to compose the business process, although it can not conform to its work sequence logically. But such analysis will bring more efficiency and decrease the risk

The weight reflects the degree that work-data flow can be implemented. Maybe some service providers can not be accessed after the analysis or the provided data source is not stable. So, we should score work-data flows and rank them in descending order according to their evaluation scores. The top one will be firstly tried. If it can not work, the second will be used. The weight is computed according to analysis steps, the granularity of service module, attributes of service and etc. For $activity_{j1}$, …, $activity_{jm}$, $data_{j1}$, …, $data_{jv}$, its weight W is computed as shown in formula (Eq. 1):

$$W = \sum_{i=j1}^{jm}\left(c(activity_i)+\frac{1}{g(activity_i)}\right) \quad (1)$$

If $activity_i$ confirms to work sequence logically, c ($activity_i$) will be set to 0.1. Otherwise, its value is set to 0.2. It means that the cost will be doubled if the analysis can not be accorded with logical work sequence. g ($activity_i$) means the granularity of service module which will be used to implement $activity_i$.

The g ($activity_i$) is computed based on the module size of service which is described in ASDT.

Formula (Eq. 1) means that if the analysis confirms to logical work sequence more and the number of used service modules is smaller, the work-data flow should be considered firstly.

Enterprise service bus layer: There are mainly four important components including routing module, service router, expanded sql executor and service and data dispatcher in enterprise service bus layer. ASDT (Abstract Service Description Table), ADDT (Abstract Data Description Table), ASMT (Abstract Service Mapping Table), ADMT (Abstract Data Mapping Table) and routing table are used to provide information for routing of service and data here. Every entry in ASDT (Abstract Service Description Table) describes the interface of service on ESB, which is the specification of the service such as service name, function, module size and etc. (Table 1).

Every entry in ADDT (Abstract Data Description Table) describes the interface of datasources on ESB, which is the specification of data views in databases. For example, data view, data source, table and the description of data view (Table 2).

The specification is not the actual implemented module, but an interface description.

ASMT is Abstract Service Mapping Table, in which every entry describes a mapping from abstract service description to uri of one service provider or uris of several service providers. Note that a service can be implemented by different providers. It is shown in Table 3, ADMT is Abstract Data Mapping Table, in which every entry describes a mapping from abstract description of data to uri of one data source or uris of several datasources. Note that an array of data may be located in different databases (Table 4).

Routing table is central to the routing module and hence they must have compact representations and support fast lookup and update operations. The cardinality of the map in routing table equals to the cardinality of the address space. The size of routing table can be significantly reduced by combining addresses into equivalence classes based on their mapped values. The compaction principle is that if addresses x_1 and x_2 map to the same nexthop address set H, then the routing table need only store the mapping $x \rightarrow H$ instead of the two mappings $x_1 \rightarrow H$ and $x_2 \rightarrow H$.

After some time, service provider may find that there are some problems in its module. It must update its module or give new function, so these changes should be notified to service requesters. Service provider will send a message to routing module. The message at least consists of service_change_specification. When new database is published, data views in some database are modified and a database is deleted from ESB, DB provider will send a message to routing module. The message at least consists of data_change_specification. Then routing module will modify ASDT, ADDT, ASMT, ADMT and routing table, which makes abstract description of services and data consistent with corresponding service provider and DB provider. The modifying algorithm is shown in Fig. 2.

Table 1: Abstract service description table

Abstract service	Function	Size
…	…	…
AS_i	Compute_interest	10 k
…	…	…

Table 2: Abstract data description table

Data view	Data source	Table	Viewdes
…	…	…	…
$Data\_view_i$	$Data\_Base_i$	$Table_i$	Des_i
…	…	…	…

Table 3: Abstract service mapping table

Abstract service	Uri of service provider	Description
…	…	…
AS_i	202.114.201.808	…
…	…	…

Table 4: Abstract data mapping table

Abstract data	Uri of DB provider	Description
…	…	…
$Data\_view_i$	202.112.119.808	…
…	…	…

Modifying_algorithm(service_change_specification, data_change_specification, impl_part)
(1) Routing module sends a message that ASDT, ADDT, ASMT, ADMT and routing table will be modified to service router and expanded sql executor. (After receiving the message, service router and expanded sql executor stop working.)
(2)If impl_part="service" then routing module modifies the corresponding entry in ASDT and ASMT based on service_change_specification. At the same time, routing table is updated according to uri_path in service_change_specification.
(3) If impl_part="data" then routing module modifies the corresponding entry in ADDT and ADMT based on data_change_specification. At the same time, routing table is updated according to uri_path in data _ change_ specification.
(4) Routing module notifies service router and expanded sql executor that modifying process is over. (After receiving the message, the preceding work-data flow in buffers will be processed again.)

Fig. 2: Updating algorithm in routing module

After the analyzer analyzes the business process into work-data flows, service and data dispatcher will send the workflows to service router and send dataflows to expanded sql executor one by one. There are buffers in service router and expanded sql executor respectively, which are used to record workflows and dataflows. After modifying ASDT, ADDT, ASMT, ADMT and routing table, the cancelled workflow and dataflow should be gotten from buffers and processed again.

Service router gets the activity from its buffers and obtains its corresponding service providers according to ASMT. Maybe an abstract service can be implemented by different service providers. Service router checks ASMT and gets its uri which identifies the location of the service provider and its description information. Expanded sql executor gets the data from its buffers and obtains its corresponding DB providers based on ADMT. The processed result is <sql command, uri of database>.

Service-data implementation layer: There are lots of service providers which provides different functional services and databases in DB providers on EBS. Providers can publish new services and databases. Some old services and databases can be deleted from EBS. Service modules and databases can be updated at any moment. After that, providers should notify routing module. This will make transparent services and disparate data integrated.

ROUTING OF SERVICES AND EXECUTING OF EXPANDED SQL COMMANDS

After work-data flows are ranked in descending order, the top one will be tried firstly. Service router will change work flow into a list of uris of services providers and expanded sql executor will change data flow into a list of <sql command, uri of database>. Then routing module will send these requests of services and data accessing to corresponding providers according to routing table.

ROUTING OF SERVICES AND DATA

Routing of services and data is fulfilled with the sequence as shown in Fig. 3.

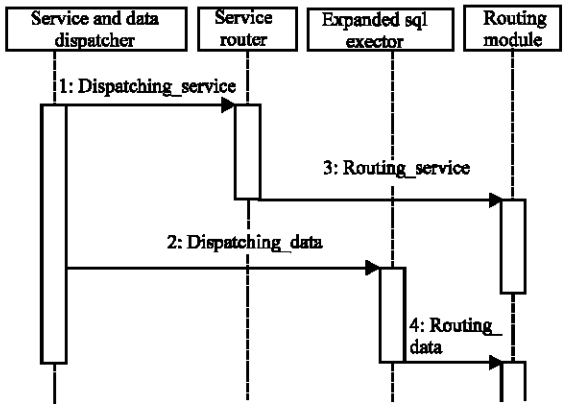

Fig. 3: Routing process of services and data

When service and data dispatcher invokes the dispatching_service method, service-flow will be transferred to service router component. Then service router will decide the best choice from service candidates and invoke the routing_service method. When service and data dispatcher invokes the dispatching_data method, data-flow will be transferred to service router component. Then expanded sql executor will decide the best choice from DB provider candidates and invoke the routing_data method.

CONCLUSION

In this study, a new framework is presented to integrate transparent services and disparate data on ESB, where a business process is separated into multiple work-data flows. The work-data flow is evaluated by weight for selecting most suitable services. The updating algorithm in routing module is called when provided services and databases are updated.

ACKNOWLEDGMENT

This study is supported by High and New Technology Industry Foundation of Harbin under Grant No. 2008FG2CG021.

REFERENCES

Alonso, G. and F. Casati, 2005. Web services and service-oriented architectures. Proceedings of International Conference on Data Engineering, Apr. 5-8, Academic Press, pp: 1147-1147.

Chafle, G., K. Dasgupta and A. Kumar, 2006. Adaptation in web service composition and execution. Proceedings of International Conference in Web Services, Sept. 21-23, Academic Press, pp: 549-557.

Fujii, K. and T. Suda, 2005. Semantics-based dynamic service composition. IEEE J. Selected Areas Commun., 23: 2361-2372.

Jean-Louis, M., 2006. Combining service-oriented architecture and event-driven architecture using an enterprise service bus. IBM Developworks.

Kuppuraju, S., A. Kumar and G.P. Kumari, 2007. Case study to verify the interoperability of a service oriented architecture stack. Proceedings of International Conference on Services Computing, July 9-13, IEEE, pp: 678-679.

Laliwala, Z., P. Majumdar and S. Chaudhary, 2006. Semantic and rules based event-driven dynamic web services composition for automation of business process. Proceedings of IEEE Services Computing Workshops, Sept. 11-13, IEEE, pp: 175-182.

Min, L., B. Goldshlager and L.J. Zhang, 2005. Designing and implementing Enterprise Service Bus (ESB) and SOA solutions. Proceedings of the International Conference on Services Computing, July 11-15, IBM Global Services, USA., pp: 14-14.

Penta, M., R. Esposito and M. Villani, 2006. WS Binder: A framework to enable dynamic binding of composite web services. Proceedings of the 2006 International Workshop in Service-Oriented Software Engineering, May 27-28, IEEE, pp: 74-80.

Schmidt, M.T., B. Hutchison and P. Lambros, 2005. The enterprise service bus: Making service oriented architecture real. IBM Syst. J., 44: 781-797.

Xiao-Ying, B. and X. Ji-Hui, 2007. DRESR: Dynamic routing in enterprise service bus. Proceedings of International Conference on E-Business Engineering, Oct. 24-26, IEEE, pp: 528-531.