GRAPHICAL REPRESENTATIONS OF MESSAGE EXCHANGES INTO WEB SERVICE-BASED APPLICATIONS

René Pegoraro
Computer Science Department, São Paulo State University at Bauru, UNESP, Brazil
Laboratoire d'Analyse et d'Architecture des Systèmes, LAAS-CNRS, Toulouse, France

João Maurício Rosário
Mechanical Design Department, University of Campinas, UNICAMP, Brazil

Khalil Drira
Laboratoire d'Analyse et d'Architecture des Systèmes, LAAS-CNRS, Toulouse, France

Keywords: Web services, SOAP, Web applications, Graphical representation.

Abstract: Web service-based application is an architectural style, where a collection of Web services communicate to each other to execute processes. With the popularity increase of Web service-based applications and since messages exchanged inside of this applications can be complex, we need tools to simplify the understanding of interrelationship among Web services. This work present a description of a graphical representation of Web service-based applications and the mechanisms inserted among Web service requesters and providers to catch information to represent an application. The major contribution of this paper is to discus and use HTTP and SOAP information to show a graphical representation similar to a UML sequence diagram of Web service-based applications.

1 INTRODUCTION

Web service-based application is a collection of Web services that communicate with each other to execute a process. Every Web service participates as a functional building block, loosely coupled, and reusable.

W3C (Booth et al., 2004) defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network. Some specifications have been developed to extend Web Services capabilities; among them WS-Addressing that defines XML elements to identify requester and provider endpoints.

We may implement Web service-based application by using orchestration, choreography and other approaches, including an invocation of Web services as service component parts of other Web service. Web services are a technology well established, we may publish, discover, and use them in a standard form, but interrelationship among Web services is not easy to verify or understand in a process execution, since many communications may take place in parallel.

The objective of this paper is to propose an automatic graphical representation of message exchanges into Web service-based applications, allowing developers and administrators visualize and understand the interrelationship among an application and its Web services. This graphical representation uses the information taken from packets HTTP and Simple Object Access Protocol (SOAP) messages used in most of Web service communications and the times involved in these communications.

As a proof of concept, we present a representation of a Web food shop prototype using BPEL (Web Services Business Process, 2007).

The organization of this paper is as follows: Section 2 presents some concepts of Web service and WS-Addressing; section 3 describes how the system gathers information; section 4 discusses the

Pegoraro R., Maurício Rosário J. and Drira K. (2008).
GRAPHICAL REPRESENTATIONS OF MESSAGE EXCHANGES INTO WEB SERVICE-BASED APPLICATIONS.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 367-370
DOI: 10.5220/0001720303670370

representation and shows an experimental result; and section 5 conclusion and future works.

2 WEB SERVICES

A Web service is set of software operations designed to support interoperable machine-to-machine interaction over a network. It uses XML standards allowing integration of applications through different standards of operating systems and programming languages.

Web services use Simple Object Access Protocol (SOAP) to exchange messages. SOAP messages consist of an envelope that defines a framework for describing what is in a message. The envelope groups an optional header and a mandatory body. The SOAP header can contain information about the message. The SOAP body contains the message.

There are varieties of specifications that have been developed or are currently being developed to extend Web Services capabilities. These specifications are normally referred to as WS-* and they may complement, compete, or supersede with each other. WS-Addressing (WS-A) specification is our choice to identify the endpoints that are exchanging SOAP messages. WS-A describes the ways to indicate requester and provider addresses inside of the header of a SOAP message.

2.1 WS-Addressing

WS-Addressing (WS-A) is a recommendation to incorporating message addressing information into SOAP messages. This separates information of Web services from message transportation protocols, allowing the endpoint identifications in SOAP level and making independent of the underlying transport. Besides, WS-A incorporates, into a SOAP envelope, reply destination to allow asynchronous operations and fault handler addressing information.

The WS-A 1.0 (Box, 2004) describes, among others, the following elements:

To – This optional element provides the value for the destination property.

From – This optional element provides the value for the source endpoint property.

ReplyTo – This optional element provides the value for the reply endpoint property.

Action – This required element conveys the value of the action property.

MessageID – This optional element conveys the message identify property.

RelatesTo – This optional element information item is used to link this message with another previous message.

2.2 HTTP Addressing

SOAP does not specify a transport protocol, but using SOAP carried in HTTP is the way most commonly used in transactions between Web services.

When a standard SOAP message is sent over HTTP, HTTP header consists at least of three lines: the start-line, *Host* and a specific header field *SOAPAction*. The start-line includes the method to be applied to the resource, the Web service address, and the HTTP version. The header field *Host* indicates the address of the provider and the SOAPAction the operation inside of the specified Web service that will be invoked. This interaction between SOAP and HTTP is described in (Box, 2000).

3 GATHERING INFORMATION FROM MESSAGES

The interactions inside a Web services-based application happen using HTTP and SOAP message exchanges. From these messages, we can take endpoint informations and the times involved in Web service communications.

To gather information from Web service messages, the communication links must be instrumented. As enumerated in Rud (2006), many ways of instrumentations may be used in Web service environments. We choose HTTP proxy server because it is a technology well established in network, easy to implement, multi-platforms, and possible to install in almost all Web application servers with few configurations. A standard proxy server can be placed in the client computer, in the provider computer, or at a specific location between the client and the provider. Our intention is to take endpoint informations and execution time measures. If we want to measure time values that does not include times spend in the network, it is better to install the proxy as close as possible to the provider computer; on the other hand, if we need to measure time values including network times, the better place is in the client computer or in its neighbourhood. Obviously, if we need to measure the times of the network and the services, we should install two proxies, one beside client and other beside provider.

However, this paper just concerns to Round-Trip Delay Time (RTT), consequently, we used just one proxy placed in a computer close to a client. The RTT is the total time used in a complete Web service operation call, starting from the moment that the client sends the request until the client receives the response.

When proxies are inserted into a Web service application, every message passes through them. Each proxy gathers the information from SOAP and HTTP with some relevance to graphical representation. This information is stored in a log. In the log we have the fields:

"SOURCE" – it is the message origin, it is taken from optional SOAP element “From”. If it is omitted in SOAP, the machine name or IP comes from communication socket and stored in log;

"DESTINATION" – it is the address and name of the Web service to invoke, optional SOAP element “To” provides this value. If it is omitted, the destination is taken from HTTP header start-line and field “Host”;

"ACTION" – it is operation name, it is taken from mandatory SOAP element “Action”. If the SOAP does not use WS-A extension, the operation is taken from HTTP header field “SOAPAction”;

"MSG_UUID" and "RELATES_TO_UUID" they are message identifications, from SOAP “MessageID” and “RelatesTo”, respectively;

"T1", "T2" – they are the measured times to execute the Web service. This identifies the start and the end times of the service. RTT can be calculated from the difference between "T1" and "T2".

4 EXPERIMENTAL RESULTS

As a proof of concept, we insert our architecture in the FoodShop Company prototype connections. We deploy the food shop in four virtual machines: one Shop, one Warehouse, and one Supplier. Another virtual machine was also used in the test environment to host the proxy. In order to keep things as simple as possible, we decided to use only one centralized proxy. Thereby, all the connections pass through this proxy, which can take information from the messages

4.1 Food Shop

The food shop prototype used has become the standard test bed in the frame of Ws-Diamond Project (IST-516933) and it involves characteristics, as asynchronous and synchronous invocations, compositions using BPEL, and simple Web services.

The FoodShopping example is concerned with a FoodShop Company that sells and delivers food. The company has an online SHOP and several WAREHOUSEs and SUPPLIERs located in different areas.

Figure 1 shows our graphical representation of the food shop execution.

4.2 Graphical Representation

Our graphical representation of the Web service application represents the execution times of every operation. Thereby the execution time start at the top of representation and ends at the bottom. The arrows represent message exchanges observed inside the Web service application and the rectangles are the Web service operation executions. Arrows with dark heads are the synchronous invocations, dashed arrows represents the return from synchronous invocations. Arrows with open heads mean asynchronous messages. To simplify the diagram, operation names are not directly represented, but the user can choose an operation execution to see its informations.

In a general sense, Web services are black boxes. We do not know what they do internally. Since, our representation use just messages, some internal behaviour is not possible to be represented. We represent the Web service executions as follow:

- Each big rounded white rectangle, which we call swim lane, represents a computer in the network.
- Each column with vertical names represents a Web service;
- Each small rectangle is an operation execution into a Web service column. Every operations are represented inside a swim lane;
- Gray small rectangles represent synchronous operations;
- White small rectangles represent asynchronous operations

Horizontal size of each operation rectangle represents the time of the operation execution.

We use all available information inside the SOAP/HTTP messages. However, if the origin is not specified (in “From” SOAP field), it is not possible to state exactly the Web service source of the message, then the only available information is the network IP of Web service source machine. Thereby, the arrow starts from the swim lane border

of origin host. However, it is just possible if the origin and Web service invoked are in different machines. If both are in the same machine, no arrow is drawn.

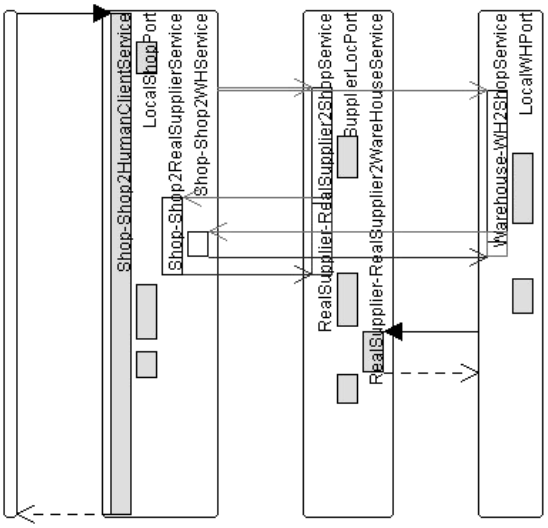

Figure 1: Food shop representation from our system.

We can observe in Figure 1 that there are some operations that seemingly execute alone, without a invoker to request them. The food shop example was constructed using BPEL, this kind of orchestration work like a Web service, but it can remain executing even after it sends the response message. Our representation just uses messages and cannot identify internal executions without message exchanges.

5 CONCLUSIONS

In this paper, we described a graphical representation of Web service-based applications. It is based on the information obtained from HTTP/SOAP messages.

We presented a methodology by using proxy servers to obtain information of message exchanges between Web services, and identify requester and provider endpoints participating in application.

The presented methodology directed the prototype implementation of a Java proxy server and a graphical tool that produces a graphical representation of the Web service interactions.

5.1 Futures Works

This visualization tool is a step toward to performance hot spots identification in Web service-based architecture. With the taken information by proxy is possible to orientate efforts improving business goals, making better QoS agreements, and achieving better customer experience.

Some Web service invocations may be anonymous to permit override in Web service operations. To complete the information of this kind of communications, we intend to take missed data from the WSDL.

ACKNOWLEDGEMENTS

This work was supported by CAPES – Brazilian Council of Research and LAAS-CNRS, France, through collaboration research project CAPES-COFECUB.

REFERENCES

Booth, D., Haas, H., McCabe, F., Newcomer , E., Champion, M., Ferris , C., Orchard, D. (Eds.) (2004, February 11). Web Services Architecture. *W3C Working Group Note.* Retrieved March 22, 2007, from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

Box, D., and Curbera, F. (Eds.) (2004, August 10). Web Services Addressing (WS-Addressing). Web Services Addressing (WS-Addressing), *W3C Member Submission*, Retrieved July 30, 2007, from http://www.w3.org/Submission/ws-addressing/.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S., Winer, D., (Eds.) (2000, May 08) Simple Object Access Protocol (SOAP) 1.1. *W3C Note,* Retrieved July 30, 2007, from http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

Gudgin, M., Hadley, M., Rogers, T. (Eds.) (2006, 9 May). Web Services Addressing 1.0 – Core. Retrieved March 25, 2007, from http://www.w3.org/TR/ws-addr-core/

Mahmoud, Q. H. (2005, April). *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI).* Retrieved November 21, 2007, from http://java.sun.com /developer/technicalArticles/WebServices/soa/.

Rud, D., Schmietendorf, A., Dumke, R. (2006). Performance Modeling of WS-BPEL-Based Web Service Compositions. *IEEE Services Computing Workshops (SCW'06)* 140-147.

Web Services Business Process Execution Language v2.0 (2007, April 11). Retrieved November 20, 2007, from http://docs.oasis-open.org/wsbpel/2.0/OS/ wsbpel-v2.0-OS.html.