A Study of Service Composition with QoS Management

Casey K. Fung
Engineering and Information Technology
Boeing Phantom Works, USA
casey.k.fung@boeing.com

Patrick C. K. Hung
Faculty of Business and Information Technology
University of Ontario Institute of Technology, Canada
patrick.hung@uoit.ca

Guijun Wang
Engineering and Information Technology
Boeing Phantom Works, USA
guijun.wang@boeing.com

Richard C. Linger
Software Engineering Institute
Carnegie Mellon University, USA
rlinger@sei.cmu.edu

Gwendolyn H. Walton
Mathematics and Computer Science Dept.
Florida Southern College, USA
gwalton@flsouthern.edu

Abstract

Quality of Services (QoS) management in compositions of services requires careful consideration of QoS characteristics of the services and effective QoS management in their execution. A Web service is a software system that supports interoperable application-to-application interaction over the Internet. Web services are based on a set of XML standards such as Simple Object Access Protocol (SOAP). The interactions of SOAP messages between Web services form the theoretical model of SOAP Message Exchange Patterns (MEP). Web Services Business Process Execution Language (WSBPEL) defines an interoperable integration model that facilitates automated process integration in intra- and inter-corporate environments. A service-level agreement (SLA) is a formal contract between a Web services requestor and provider guaranteeing quantifiable issues at defined levels only through mutual concessions. Based on a prior research work on Message Detail Record (MDR), this paper further proposes a SOAP message tracking model for supporting QoS end-to-end management in the context of WSBPEL and SLA. This paper motivates the study of QoS management in a Web service composition framework with the evolution of a distributed toolkit in an industrial setting.

Keywords: OOA, CBA, SOA, Web services, SOAP, BPEL, MEP, QoS, MDR, SLA.

1. Introduction

Enterprise systems face many challenges, including seamless integration of systems, allowing data access from anywhere anytime, and providing services to customers and partners inside and outside the enterprise. In addition to functionality, quality considerations such as extensibility, flexibility, connectivity, and interoperability require that enterprise systems be easily accessible through published interfaces and easily enhanced in order to offer value-added services. One approach to meet these challenges is to consider a system to be a composition of a collection of services. Each service makes its functionality available through well-defined or standardized interfaces. The result of this approach is a Service-Oriented Architecture (SOA), in which services are fundamental elements that can be independently developed and evolved over time. Each service is a self-describing, composable, and open software component. A SOA consists of services, their compositions, interactions, and management [1, 5]. In particular, management of quality in SOA is challenging because quality identification, measurement, analysis, and traceability have to be dealt with in a heterogeneous environment and a scalable fashion.

Quality of Services (QoS) management in compositions of services [32], especially for distributed services, requires careful consideration of QoS characteristics of the services and effective QoS measurement in their execution. In the context of SOA, different service providers and requestors may have different QoS requirements in terms of performance, reliability, timeliness, and security. For example, some providers may have higher priority than others and may require their message deliveries to be guaranteed with correct ordering in faster response time. Similarly, some requestors may be more critical than others, and thus require shorter delays in receiving messages. To meet QoS requirements of many concurrent providers and requestors, QoS management must provide services such as performance, admission control, prediction, resource management, monitoring, and adaptation.

Current trends in information and communication technology (ICT) are accelerating widespread use of Web services in supporting SOA. A Web service is a software system designed to support interoperable application-to-application interactions over the Internet. Web services are based on a set of standards such as Universal Description, Discovery and Integration (UDDI) [27], Web Services Description Language (WSDL) [28], and Simple Object Access Protocol (SOAP) [13, 14]. The interactions of SOAP messages between Web services form the theoretical model of SOAP Message Exchange Patterns (MEP). The Web Services Business Process Execution Language (WSBPEL) is an OASIS-proposed standard for formal specification of business processes and interaction protocols [33], formerly known as BPEL4WS [18].

WSBPEL defines an interoperable integration model that facilitates automated process integration in intra- and inter-corporate environments. A service-level agreement (SLA) is a formal contract between a Web services requestor and provider guaranteeing quantifiable issues at defined levels only through mutual concessions [9]. The negotiation issues are described as SLA parameters, and the SLA parameters are based on the domain specific vocabularies. W3C also emphasizes the important role of SLA in the selection of QoS. According to some existing SLA specifications, one potential solution that can be applied in this model is Web Service Level Agreement (WSLA). Further, WSLA even provides an extensible mechanism to include domain specific vocabularies [29]. Thus the SLA of the receiver should be advertised before the sender examines whether the SOAP message could be disclosed to the receivers.

Based on a prior research work [11] on Message Detail Record (MDR), this paper further proposes a SOAP message tracking model for supporting QoS end-to-end management in the context of WSBPEL and SLA. This paper is organized as follows: With a travel reservation example, Section 2 elaborates on the composition study and the QoS management in SOA and introduces the conceptual model of SOAP MEP, WSBPEL and SLA for illustrating QoS end-to-end management. Section 3 illustrates the SOAP message tracking model for supporting QoS end-to-end management with a travel reservation example. This model is adopting MDR in SOAP message headers. Section 4 discusses conclusions and future research issues.

2. QoS End-to-End Management in SOAP Message Exchange Patterns

Many companies provide services on the Internet for supporting automated business-to-business (B2B) applications, e.g., a travel reservation process. A business process contains a set of activities that represent both business tasks and interactions between Web services. Because of standardization trends, the need to coordinate Web services for supporting business processes in loosely coupled B2B environments is becoming more critical. For illustration, a travel reservation example basically may involve three parties: a customer, a travel agency, and an airline. Referring to Figure 1, the travel reservation process is described as follows. The customer's application sends a reservation request in SOAP message to the travel reservation system. The travel agency receives a customer request to make a travel reservation. The request may include information defining origin, destination, and day and month of travel. The customer may also specify other requirements such as a preferred mileage club (e.g., Star Alliance or One World), a direct or indirect flight, and the maximum reservation price. The travel agency may specify QoS attributes that constrain the selection of the candidate airline, such as the response time of service and a lower bound for acceptable commissions to be provided to the agency by the airline.

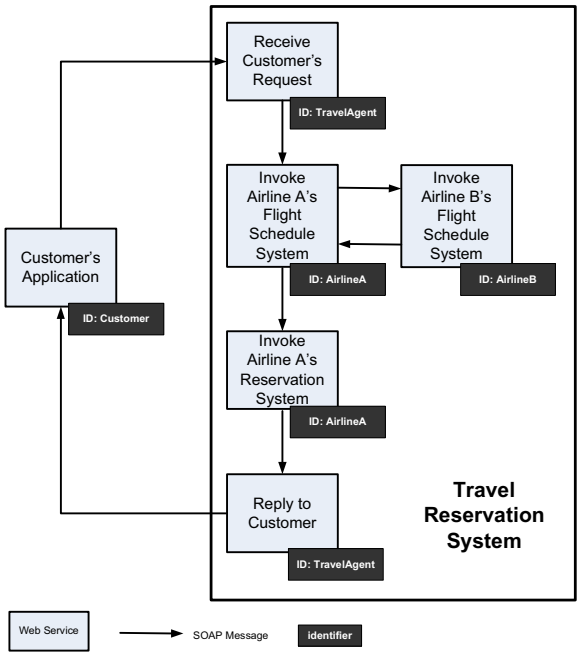

Figure 1. An Illustrative Travel Reservation Process

Referring to the publish/find/bind model in Web services [12], a SOAP message is fundamentally a one-way transmission between SOAP nodes, from a SOAP sender to a SOAP receiver. The SOAP message contains two SOAP-specific sub-elements within the overall envelope, namely a header and a body. As Web services become more prevalent, the nature of electronic transactions on Internet changes from simple browser-to-business clicks to an orchestrated flow of messages between cross-enterprise serviced. Consequently, more than one Web service could participate in the federated execution of a single transaction. In such a case, the problem of end-to-end management becomes very important. The problem of message tracking is to be able to track all the messages that are part of one such transaction through all the web services in which the transaction executes.

The SOAP message tracking helps in QoS end-to-end management of Web services because one can infer metrics for the message flow between Web services. QoS end-to-end management can be considered from different perspectives, depending on the layer of the

protocol stack being addressed. In the network layer, the resources of concern are bandwidth and traffic control. Thus, QoS attributes of interest include performance, reliability, and timeliness. In the middleware layer, communication, CPU and memory are the main concerns, so QoS attributes are, again, performance, reliability, timeliness, and security. In the application layer, resources are services and flows. Therefore, QoS attributes can be any optional features provided by a service.

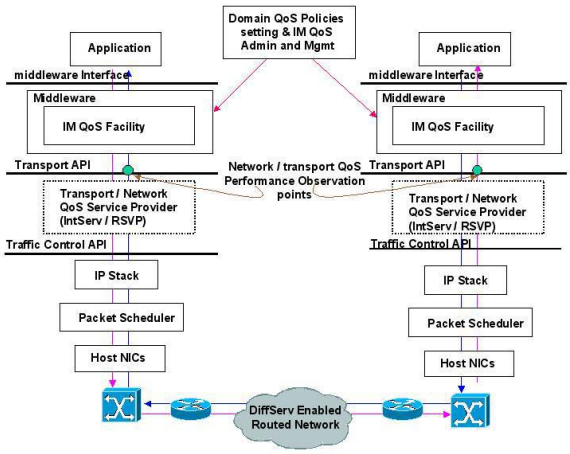

Figure 2. End-To-End QoS Management Framework

Figure 2 depicts an end-to-end QoS management framework. At the Transport API, the QoS enforcing service (e.g., AMS, intermediary) passes the protocol data unit and the transport/network service parameters enquiries to transport/network. QoS enforcement down to the network layer is enabled by the end-to-end Transport parameters (between the two observation points) such as transport-to-transport delay, TCP retransmission count in the last observation interval, etc. For example, if the Resource Reservation Setup Protocol (RSVP) is not implemented on the end-system, then depending on the QoS policy definition, either the Web service QoS or the Transport API will map the Web service QoS parameters to the Differentiated Services Codepoint (DSCP) field. If metrics based on historical data are being maintained, enforcing end-to-end QoS requirements can also be carried out in the service layer without enquiring into the network layer. (We will revisit this topic in Section 3 together with the tracking method.) Further, the execution of a single business transaction can involve multiple SOAP messages being exchanged between Web services. In general, headers are typically used to represent management attributes about messages. Web services requestors may need to track their requests and understand causes for failure of their requests. Web services providers that are using other Web services to provide composite services may like to understand the interactions between composite Web services [10]. This sort of information can improve the QoS in many Web services-based business applications. However, none of the requirements in the "Web Services Architecture Requirements" document [12] defines the importance of QoS in SOAP message flows. Thus, the major goal of this paper is to propose a SOAP message tracking model for supporting QoS end-to-end management and to present the importance of QoS in per-to-peer SOAP Message Exchange Patterns (MEP) [13]. Figure 3 depicts QoS concerns existing between the SOAP message sender and ultimate receiver. The information exchanges between the SOAP sender and receiver contain different QoS attributes. The sender may define its QoS requirements (e.g., response time) in the SOAP message to be sent to the receiver. To satisfy the sender's requirements, the receiver may have to enforce certain QoS.

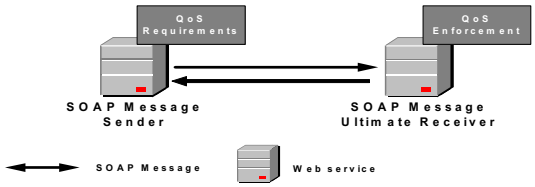

Figure 3. QoS Requirements and Enforcements between SOAP Message Sender and Ultimate Receiver

Referring to Figure 1, the "Customer's Application" Web service is the SOAP message sender and the "Travel Reservation System" can be considered to be a Web service. From the perspective of "Customer's Application," the "Travel Reservation System" is considered to be an ultimate message SOAP receiver. For illustration, the "Customer's Application" may specify the response time as "less than 60 seconds" as a QoS requirement to the "Travel Reservation System." Referring to the MEP, SOAP messages may be transmitted through the intermediaries before reaching the ultimate receiver. SOAP intermediaries (e.g., Internet service providers, proxies and others) should be privy to the exchange of SOAP messages between the sender and receiver. Thus, the sender may have to define the QoS requirements not only for the ultimate receiver, but also for those intermediaries. For example, the sender may request the intermediaries to forward the SOAP message immediately. Referring to the SOAP message framework [13], the forwarding intermediaries and active intermediaries between the sender and ultimate receiver are defined as follows:

- Forwarding Intermediaries: The SOAP header blocks in a SOAP message require that the SOAP message be forwarded to another SOAP node on behalf of the

SOAP sender. In this case, the processing SOAP node acts in the role of a SOAP forwarding intermediary.

- Active Intermediaries: Based on the forwarding SOAP intermediaries, active SOAP intermediaries can modify the inbound SOAP message and then send the modified outbound SOAP message. In most cases, the active SOAP intermediaries are performing security services, annotation services, and content manipulation services.

In principle, the active and forwarding intermediaries should not need to store the inbound SOAP messages. In this scenario, the intermediaries should enforce the QoS requirements from the sender. This QoS requirement can be usually expressed via SOAP headers. A SOAP header [14] is an extension mechanism that provides a way to pass control information in SOAP messages such as passing directives or contextual information related to the processing of the message. This allows a SOAP message to be extended in an application-specific manner that might be encountered in the path of a message from a SOAP sender to an ultimate receiver. SOAP headers always involve the participation of SOAP intermediaries along a path from an initial SOAP sender to an ultimate SOAP receiver in MEP. Referring to Figure 4, there are two possible paths for the response SOAP message from the ultimate receiver to the sender. Path 1 shows a direct response from the ultimate receiver to the sender, and Path 2 shows an indirect response from the ultimate receiver to the sender via an intermediary.

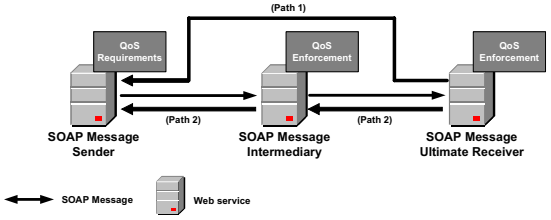

Figure 4. SOAP Message Intermediary

Referring to Figure 1, there is a SOAP message intermediary "Airline A" between the "Customer's Application" and travel agent. There may also be a QoS requirement such as the response time as "less than 40 seconds" enforced at the intermediary. In this scenario, the "Travel Reservation System" directly sends the response message to the "Customer's Application" and the customer may not know who are the intermediaries. Referring to Figure 5, the QoS requirements also have to be enforced in a delegation and propagation situation. In many cases, the ultimate receiver may delegate some activities to other Web services. This assignment process is also called delegation or propagation [11]. In this scenario, the ultimate receiver has to govern other receivers' QoS enforcement in accordance with the sender's and also its own QoS requirements. For example in the transmission of high bandwidth streaming video, content may require differential QoS protocol in the network layer, rather than the "best effort" protocols in use today. Referring to Figure 1, the "Airline A" Web service delegates the work to "Airline B" Web service for finding a flight schedule. Thus "Airline A" may define a QoS requirement to "Airline B" (such as the response time is "less than 20 seconds.")

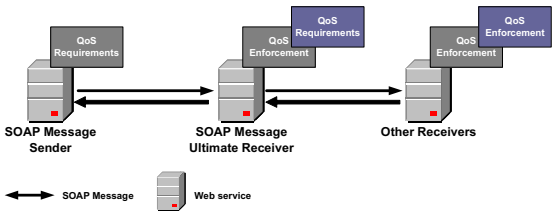

Figure 5. Propagation and Delegation Scenario

3. Towards A QoS End-to-End Management Model

Referring to Figure 1, the travel reservation system invokes Airline A's Web service to retrieve flight schedules. If Airline A can not take the customer's request, Airline A may invoke a partner (such as Airline B) to handle the request. When the results are returned from the Web service, the travel reservation system again invokes the Airline A's Web service to make a reservation. Once the travel reservation system receives a confirmation from the airline's Web service, the travel reservation system will reply to the customer with reservation information and will store relevant data on the transaction.

Assume the following for the travel reservation example: A user requests that the travel agency service provides a reservation with the following parameters: *direct flight, from Tampa, to New York/Laguardia airport, costing no more than $300*. The travel agent service provides an additional parameter to the Web service broker: *Response Time < 40 seconds*. From the perspective of the travel agency service, *ResponseTime* includes network latency as well as the amount of time the airline's Web service takes to process a request. Thus, the travel agency defines *ResponseTime* for a request to a specific airline Web service as the turn-around time for the request, as calculated by the travel agency service. (That is, *Response time* = elapsed time between issuance of the request to the service and receipt

of a valid response.) If the airline's service does not provide a valid response before the travel agency service's timeout threshold, *ResponseTime* for this request = some predefined constant value C, where C is greater than the travel agency service's timeout value. The simplified WSBPEL is shown in Figure 6. The technical details of each activity are as follows:

- *received*: It allows the business process to do a blocking wait for a matching message to arrive, e.g., TravelAgency.
- *invoke*: It allows the business process to invoke a one-way or request-response operation on a portType offered by a partner, e.g., airlineA, airlineB and QoSManager.
- *Reply*: It is used to send a response to a request previously accepted through a receive activity.

```
<process name="TravelReservationSystem"

targetNamespace="http://travelagencies.com/travelscheduleprocessing"
      xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/"
      xmlns:as="http://airlines.org/wsdl/AirLineWS"
      xmlns:qm="http://qos-management.org/wsdl/QoSManagerWS"
      xmlns:ta="http://travelagencies.org/wsdl/TravelAgencyWS"
      xmlns:bk="http://bank.org/wsdl/BankAgencyWS"
      suppressJoinFailure="yes">
  ...
  <flow>
    ...
    <receive partnerLink="TravelAgent"
          portType="ta:TravelAgencyWS_IF"
          operation="request"
          variable="request" createInstance="yes">
      <source linkName="Request-to-AirlineA"/>
    </receive>
    <invoke  partnerLink="AirlineA"
          portType="as:AirLineWS_IF"
          operation="getSchedulesRequest"
          inputVariable="request"
          outputVariable="responseA">
      <target linkName="Request-to-AirlineA"/>
      <source linkName="AirlineA-to-AirlineB"/>
      <target linkName="AirlineB-to-AirlineA"/>
      <source linkName="AirlineA-to-Reservation"/>
      <agreement location="http://airlines.org/wsla/AirLineA"/>
    </invoke>
    <invoke  partnerLink="AirlineB"
          portType="as:AirLineWS_IF"
          operation="getSchedulesRequest"
          inputVariable="request"
          outputVariable="responseB">
      <target linkName="AirlineA-to-AirlineB"/>
      <source linkName="AirlineB-to-AirlineA"/>
      <agreement location="http://airlines.org/wsla/AirLineB"/>
    </invoke>
    ...
    <invoke  partnerLink="AirlineA"
          portType="as:AirLineWS_IF"
          operation="makeReservation"
          inputVariable="request"
          outputVariable="confirm">
      <target linkName="AirlineA-to-Reservation"/>
      <source linkName="Reservation-to-Reply"/>
    </invoke>
    ...
    <reply partnerLink="TravelAgent"
          portType=" ta:TravelAgencyWS_IF "
          operation="replyCustomer" />
      <target linkName="Reservation-to-Reply"/>
    </reply>
    ...
  </flow>
</process>.
```

Figure 6. A Simplified WSBPEL Document for the Travel Reservation System

Each of the activities in a flow model must be executed by an appropriate Web service. In this scenario, the role of Web services broker is to assign an appropriate Web service for each activity based on the operations provided by the Web service and the requirements specified by the activity. This assignment process is called matchmaking. Each Web service will have a SLA for specifying the QoS attributes to the requestors in different contexts, e.g., response time. In many scenarios, a Web service may have more than one SLA for different requestors. For example, an airline Web service may provide higher QoS for its alliance members than other customers. Referring to Figure 6, the WSBPEL document of the "Travel Reservation System" must interact with the WSLA document of each Web service involved in the process. However, there is no such feature to define the WSLA in WSBPEL. Thus we propose a new assertion <agreement> in both "AirlineA" and "AirlineB" activities in the context of WSBPEL. This assertion describes the location of the WSLA document of the Web service that matched with each activity. For example, Figure 7 shows the WSLA document of "AirlineA" that defines the response time in 2 seconds.

```
<SLA xmlns="http://airlines.org/wsla"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.ibm.com/wsla
c:\Projects\WSLA\wsla.xsd"
      name="AirlineAServiceAgreement" >
    ...
    <ServiceDefinition name="AirlineAService">
      <Operation
xsi:type="wsla:WSDLSOAPOperationDescriptionType"
name="WSDLSOAPReserveSeat">
        ...
        <SLAParameter name="ResponseTime"
              type="float"
              unit="seconds">
          <Metric>responseTime</Metric>
        </SLAParameter>
        <Metric name="responseTime" type="double" unit="seconds">
          <Source>ms</Source>
          <Function xsi:type="wsla:NumberLessThanThreshold"
resultType="integer">
            <Metric>responseTimeHost</Metric>
```

```
            <Value>
              <LongScalar>2</LongScalar>
            </Value>
          </Function>
        </Metric>
      </Operation>
      ...
    </ServiceDefinition>
    ...
</SLA>
```

Figure 7. A Simplified WSLA Document for Airline A Reservation Web Service

Message Detail Record (MDR) is a collection of management attributes that are attached to every SOAP message flowing between Web services. Based on a prior research work [10] on MDR in SOAP message headers, this paper further proposes a SOAP message tracking model for supporting QoS end-to-end management/measurement. Based on an SLA, the sender may fill in some of the attributes in an MDR. The receiver may fill in others. The MDR structure consists of a unique identifier for the message, a time stamp when the message was sent, and a time stamp when the message was received. Referring to the travel reservation example, the QoS requirement in response time from the travel agency can be evaluated by the time stamps in the MDR structure. For example, the travel agent fills in the response time requirement (e.g., 60 ms) to be enforced at Airline A and sent time attributes of TravelAgent's MDR's before it is sent to the Airline A's Web service. Once the Airline A's Web service receives the Travel Agent's message, it will fill in the received time of TravelAgent's MDR and create a new AirlineA's MDR. In this scenario, Airline A will also propagate the SOAP message to Airline B's Web service with its own response time requirement (e.g., 40 ms) to be enforced at Airline B. Again, Airline B will also fill in the received time of AirlineA's MDR and create a new MDR, and so on. With this message tracking information collected on the header of a message, the ResponseTime between any two hosts in a transaction can be calculated.

When historical data of this kind is collected for a group of related services, prediction can be carried out based on formal analytical methods (e.g., Bayesian or causal network model). To calculate a projected QoS attribute value (e.g., ResponseTime) for a service for which the system state is unknowable requires a method that makes use of every source of available relevant evidence, including information obtained from vendors, personal judgment, accumulated history from past use of the service, and any knowledge of specific events (e.g., updates to a component of the service or changes in network configuration) that may invalidate the history. A promising approach to dynamic calculation of QoS attribute values involves application of Bayesian statistical methods to measurement history data [31]. Bayesian analysis starts with a suitable set of pre-defined beliefs that form the "prior distribution." As new data become available, they are combined with the prior distribution to obtain a new distribution that can be used to support analysis. When there is no evidence on which to base an initial prediction, the priors must be based on professional judgment. For example, for services of unknown quality that are candidates for business-critical use, one might use a prior distribution that represents the worst case until evidence is available that indicates that the service does in fact perform better than the worst case. Referring to the travel reservation example, the initial value of the projected *ResponseTime,* for which there is no history, can be based on the travel agency's best estimate or, in the absence of any information at all, can be set pessimistically as equal to the travel agency service's timeout value. *ResponseTime* can be automatically calculated and submitted to the history database by the travel agency service each time it issues a request to an airline service.

Figure 8 shows a simplified SOAP message that is received by Airline B. We propose to define the QoS requirements in the SOAP header. In this scenario, the response time is calculated for every outgoing SOAP message as the duration between the time when a message is sent out and the time when the response is received. In particular, the MDR of a new message is related to the MDR of the earlier message through a parent-child relationship called MDRForest that is supported with a set of mechanisms [10]. Based on the MDRForest, a collection of aggregate metrics such as average response times can be evaluated for further analysis in supporting different QoS scenarios. Message tracking could also be the basis for collecting other accounting or auditing information about policy and security enforcement.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Header
xmlns="http://registry.example.com/2003/soap-
header-p3p-extension.xsd" xmlns:env=
"http://www.w3.org/2003/05/soap-envelope"
id='header'>
<QoS>
  <QoS_id>TravelAgent-QoS-1111</QoS_id>
  <require_from>TravelAgent</require_from>
  <enforced_at>AirlineA</enforced_at>
  <response_time>40</response_time>
  <time_set>2004-08-12 12:00:23</time_set>
</QoS>
<MDR>
  <parent_mdr>None</parent_mdr>
  <message_id>TravelAgent-MDR-1111</message_id>
  <message_type>Flight-Schedule</message_type>
  <source>TravelAgent</source>
  <target>AirlineA</target>
  <time_sent>2004-08-12 12:00:23</time_sent>
```

```
  <time_received>
    2004-08-12 12:00:33
  </time_received>
<MDR>
<QoS>
  <QoS_id>AirlineA-QoS-1111</QoS_id>
  <require_from>AirlineA</require_from>
  <enforced_at>AirlineB</enforced_at>
  <response_time>20</response_time>
  <time_set>2004-08-12 12:00:35</time_set>
</QoS>
<MDR>
  <parent_mdr>TravelAgent-MDR-1111</parent_mdr>
  <message_id>AirlineA-MDR-1111</message_id>
  <message_type>Flight-Schedule</message_type>
  <response_time>40</response_time>
  <source>AirlineA</source>
  <target>AirlineB</target>
  <time_sent>2004-08-12 12:00:35</time_sent>
  <time_received>
    2004-08-12 12:00:38
  </time_received>
<MDR>
…
<env:Body Id="TravelAgent-Request-1111">
<getSchedules
xmlns="http://www.airlinecompany.com/services/Ai
rLineAWS">
  <origin>Tempa</origin>
  <destination>New York/Laguardia</destination>
  <day_of_travel>25</day_of_travel>
  <month_of_travel>September</month_of_travel>
  …
</getSchedules>
</env:Body>
</env:Header>
```

Figure 8. A Simplified SOAP Message Received at Airline B

4. Conclusions and Future Work

This paper motivates the study of QoS management in a Web service composition framework with the evolution of a distributed toolkit in an industrial setting. It presents the importance of QoS in SOAP MEP. One of the major processes in a loosely coupled Web services execution environment is matchmaking, that is, an appropriate Web service is assigned to satisfy a requestor's requirements with or without the assistance of a service locator [19]. Alternatively, matchmaking can also provide a ranked list of the n best candidates with respect to the requestor's requirements [20]. Usually, more than one Web services claim they have the same, or very similar, capabilities to accomplish a requestor's requirements. In many cases, the QoS may vary from Web service to Web service. The majority of Web services providers are not concerned about the level of QoS provided to their requestors [21]. However, there exist an increasing number of concerns about QoS with respect to maintaining their popularity and reputation [22]. Thus, it is obvious that the QoS perceived by the requestors is becoming a dominant factor for the future success of a Web service. In general, the principal QoS attributes of a Web service include a diverse set of service requirements such as the service availability, accessibility, performance, time, efficiency, reliability, scalability, dependability, regulatory, integrity and security [23, 24, 25]. In the future, one may expect that Web services providers should provide a list of trade-off alternatives between the Quality of Service (QoS) and the Cost of Service (CoS) to Web services providers, especially in one-off services. Such a list would allow Web services requestors to evaluate trade-off alternatives according to their specific requirements and constraints. CoS is measured by the resources required to procure the QoS such as capital, hardware or software. Thus, matchmaking can be based on binding support, historical performance, QoS, and CoS classifications [26].

As Web services become more popular and complex, the need for locating Web services with specific capabilities at the service locator become more and more important. We are currently working on the following technical research issues:

- *QoS Analysis*: The decomposition of quality requirements into measurable specification on features and characteristics of a service;
- *QoS Management Implementation*: We have been talking about QoS Management at the architecture or language level. To implement the QoS management in the QoS enforcing services is also a very challenging task;
- *QoS Ontology*: Development of QoS vocabularies and representation of their ontologies formally based on QoS analysis;
- *Federated Management*: Based on the concepts of SLAs as the tool to define relationships among loosely coupled services, the use of intermediaries to form a management network is promising and challenging;
- *Security and Privacy Access Control Policy*: Who can access, add or delete MDR in the SOAP header; and
- System States that can be monitored by this service level measurement method.

In the future, the QoS requirements should also be exchanged with the negotiation protocol between the SOAP message sender and receiver. One of the candidates of the negotiation protocol proposed is WS-Negotiation [15]. The advantages of using the protocol are that: (1) the SOAP message sender could send its privacy preference files to the ultimate receiver; and (2) as a result of the negotiation with the policies and sender's preference, an intelligent decision could be made for both the sender and the ultimate receiver for suiting a specific environment.

References

[1] G. Wang, and C. Fung, "Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems," Proceedings of the 37th Hawaii International Conference on System Sciences, January 5-8, 2004.

[2] I. Foster, C. Kesselman, and Steve Tuecke, "The Anatomy of the Grid," International Journal of Supercomputer Applications, 2001.

[3] IEEE STD 1471-2000, IEEE Recommended Practice for Architectural Description of Software Intensive Systems, 2000, http://standards.ieee.org/catalog/software4.html#1471-200

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Pattern: Elements of Reusable Object-Oriented Software," Addison-Wesley Press, 1995.

[5] M.P. Papazoglou, and D. Georgakopoulos, "Service-Oriented Computing," Communications of ACM, Oct. 2003.

[6] M. Uschold, P. Clark, F. Dickey, C. Fung, S. Smith, S. Uczekaj, M. Wilke, S. Bechhofer, and I. Horrocks, "A Semantic Infosphere," Proceedings of the 2003 International Semantic Web Conference (ISWC 2003), 2003, pp. 882-896.

[7] C. Szyperski, "Component Software: Beyond Object-Oriented Programming," Addison Wesley, 2002.

[8] C. Fung, S. Uczekaj, G. Wang, and S. Moody, "The Evolution of Composition Framework in a Distributed System Toolkit," Proceedings of 2004 IEEE International Conference on Web Services, pp. 600-604.

[9] A. Sahai, A. Durante, and V. Machiraju, "Towards Automated SLA Management for Web Service," HP Technical Report, 2002.

[10] A. Sahai, V. Machiraju, J. Ouyang, and K. Wurster, "Message tracking in SOAP-based Web services," Proceedings of the 2002 IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), 15-19 April 2002, pp. 33 - 47.

[11] IBM Corporation, "Security in a Web Services World: A Proposed Architecture and Roadmap," White Paper, Version 1.0, 2002. Online: www-106.ibm.com/developerworks/library/ws-secroad/

[12] C. Mohen, "Tutorial: Application Servers and Associated Technologies," ACM SIGMOD International Conference on Management of Data (SIGMOD'02), Madison, USA, June 2002.

[13] World Wide Web Consortium (W3C), "SOAP Version 1.2 Part 1: Messaging Framework," W3C Proposed Recommendation, 07 May 2003. Online: www.w3c.org/TR/2003/PR-soap12-part1-20030507/

[14] World Wide Web Consortium (W3C), "SOAP Version 1.2 Part 0: Primer," W3C Proposed Recommendation, 07 May 2003. Online: www.w3c.org/TR/2003/PR-soap12-part0-20030507/

[15] P. C. K. Hung, H. Li, and J. J. Jeng, "WS-Negotiation: An Overview of Research Issues," Proceedings of the IEEE Thirty-Seventh Hawaii International Conference on System Sciences (HICSS-37), CD-ROM, January 5-8, 2004.

[16] Workflow Management Coalition (WfMC). Online: www.wfmc.org

[17] F. Leymann, D. Roller, and M.-T. Schmidt, "Web services and business process management," IBM Systems Journal, vol. 41, no. 2, 2002, pp. 198-211.

[18] IBM Corporation, "Business Process Execution Language for Web Services (BPEL4WS)," Version 1.0, 2002.

[19] P. C. K. Hung, and G. S. Qiu, "Implementing Conflict of Interest Assertions for Web Services Matchmaking Process," Proceedings of the IEEE Conference on E-Commerce (CEC03), Newport Beach, California, USA, June 24-27, 2003.

[20] D. Veit, J. P. Muller, M. Schneider, and B. Fiehn, "Matchmaking for Autonomous Agents in Electronic Marketplaces," Proceedings of the fifth International Conference on Autonomous Agents, 2001, pp. 65-66.

[21] W. E. Youngdahl, and D. L. Kellogg, "Relationship between Service Customers' Quality Assurance Behaviors, Satisfaction, and Effort: A Cost of Quality Perspective," Journal of Operations Management, Volume 15, Issue 1, February 1997, pp. 19-32.

[22] J. Lee, J. Kim, and J. Y. Moon, "What Makes Internet Users Visit Cyber Stores Again? Key Design Factors for Customer Loyalty," Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems, 2000, pp. 305-312.

[23] M. Conti, and M. Kumar, "Quality of Service in Web Services," Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001, pp. 3550-3550.

[24] V. Firoiu, J. Y. Le Boudec, D. Towsley, and Z. L. Chang. "Theories and Models for Internet Quality of Service," Proceedings of the IEEE, Volume 90, Issue 9, September 2002, pp. 1565-1591.

[25] A. Mani, and A. Nagarajan, "Understanding quality of service for Web services: Improving the performance of your Web services," Online: http://www-106.ibm.com/developerworks/library/ws-quality.html

[26] IBM Corporation, "Web Services Conceptual Architecture (WSCA 1.0)," 2001.

[27] Universal Description, Discovery and Integration (UDDI), "UDDI Version 3.0," UDDI Specification Technical Committee Specification, 19 July 2002. Online: uddi.org/pubs/uddi-v3.00-published-20020719.htm

[28] World Wide Web Consortium (W3C), "Web Services Description Language (WSDL)," Version 1.2, W3C Working Draft, 9 July 2002. Online: www.w3.org/TR/2002/WD-wsdl12-20020709/

[29] IBM Corporation, "Web Service Level Agreement (WSLA) Language Specification," Version 1.0, 2003.

[30] R. Linger, M. Pleszkoch, G. Walton, and A. Hevner, "Flow-Service-Quality Engineering: Foundations for Network System Analysis and Development," CMU/SEI-2002-TN-01, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2002.

[31] C. Fung, P. Hung, R. Linger, and G. Walton, "Extending Business Process Execution Language for Web Services with Service Level Agreements Expressed in Computational Quality Attributes," Proceedings of the IEEE Thirty-Eighth Hawaii International Conference on System Sciences (HICSS-38), Big Island, Hawaii, USA, January 3-6, 2005.

[32] G. Wang, A. Chen, C. Wang, C. Fung, and S. Uczekaj, "Integrated Quality of Service (QoS) Management in Service-Oriented Enterprise Architecture," Proceedings of the 8th IEEE Enterprise Distributed Object Computing Conference, September 2004, pp. 21-32.

[33] OASIS, "Web Services Business Process Execution Language," Version 2.0. Online: http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm.