Electronic Communications of the EASST
Volume 17 (2009)

Workshops der
Wissenschaftlichen Konferenz
Kommunikation in Verteilten Systemen 2009
(WowKiVS 2009)

Using WSDM and Web Service Ping for QoS based Web Service Selection

Thomas Schoene, Friedbert Kaspar, Christoph Reich

12 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Using WSDM and Web Service Ping for QoS based Web Service Selection

Thomas Schoene[1], **Friedbert Kaspar**[1], **Christoph Reich**[1]

Hochschule Furtwangen University[1]

Abstract: By using the standard Web Service Distributed Management (WSDM) [OAS08c] and Web Service Ping, we introduce a lightweight solution to the Web Service QoS problem. The Management of Web Services (MOWS) part of WSDM is used to publish Web Service's QoS parameters. Management using Web Services (MUWS), the second part of WSDM, is used to monitor IT resources' QoS. Examples are server's QoS, application server's QoS and network's QoS. Web Service Ping can be used as a simple diagnostic tool for Web Service's latency and Web Service's availability across organizational boundaries. Therefore, we propose to introduce a standardized Web Service Ping operation into all Web Services. All QoS data retrieved by using MOWS, MUWS and Web Service Ping, can be used for Web Service selection. We introduced a new Web Service selection architecture, the Delegation Web Service as selector. Compared to Web Service Broker as selector, consumer as selector and QoS enhanced UDDI as selector, the Delegation Web Service as selector offers a better solution for implementing Web Service load balancing and can increase the security of and for Web Services.

Keywords: SOA, Web Services, QoS, Web Service Ping, WSDM, MUWS, MOWS, Web Service Routing, Web Service selection, Delegation Web Service as selector

1 Introduction

The growing demand to increase the flexibility of the IT infrastructure to support rapidly evolving business needs, has led to a rising interest in Service-Oriented Architectures (SOA). Web Services are the dominant implementation technology for SOAs. More than 80 accepted Web Service standards exist [inn08]. They support a wide range of functional and nonfunctional requirements. To describe, monitor and manage Quality of Service (QoS) parameters, numerous approaches exist [Lee08, MRD08, GNCW03, TGRS04, BSR$^+$07b]. Our approach is to use the OASIS WSDM (Web Service Distributed Management) standard [OAS08c] and Web Service Ping (see Section 5) to solve the Web Service QoS problem. WSDM uses WSRF [OAS08d] and WSN [OAS08b]. WSRF is used to model and access stateful resources. A stateful resource may be a Web Service or any other IT resource. WSN is used for notifications and events. WSDM includes Management using Web Services (MUWS) and Management of Web Services (MOWS). Web Service Ping can be used to measure the RTT (Round Trip Time) of a Web Service and to determine the availability of a Web Service.
The article is organized as follows. First a short overview of related work is given (see Section 2). Then we discuss QoS parameters and how to choose appropriate measurement points for

these QoS parameters (Section 3). In Section 4, the capability of MOWS and MUWS to monitor QoS is discussed. Then we explain the idea of using Web Service Ping as an easy Web Service diagnostic tool by adding a standardized Web Service Ping operation to all Web Services (see Section 5). Implementation related considerations, of how Web Service frameworks do and can support WSDM and Web Service Ping follow in Section 6. Three architectures for Web Service selection have been identified and a new architecture for Web Service selection will be introduced (see Section 7). The new Delegation Web Service as selector approach will be introduced and compared to the three other architectures.

2 Related Work

Service Level Agreements (SLA) between consumers and Web Services, define a QoS agreement between consumers and Web Services. SLA can also be used to specify actions, taken in case of SLA violations. Several approaches to solve the QoS problem using SLA exist. [BSR+07b, BGR05] introduces WSQoSX, a framework for monitoring Web Service's QoS and ensuring a high SLA (Service Level Agreements) compliance by dynamic Web Service reselection in case of SLA violations. [SK08] introduces a decentralized SLM (Service Level Management) approach for management of SOAs using SLA. [GNCW03] introduces QUEST which uses SLA to dynamically compose Web Services, finding the best composition in means of QoS. In our approach, the QoS parameter value when requested by the consumer, can be understand as Service Level Agreement. If real SLA is required, our approach can be extended by including standards such as WSLA [IBM08]. [TGRS04] introduces WS QoS (Web Service QoS), a framework for QoS based selection and monitoring of Web Services. It advocates the use of a Web Service Broker for QoS based Web Service selection. [DA08] introduces a way of measuring Web Service provider's QoS and advocates the use of the Web Service Broker as selector architecture. [Lee08] introduces WSQMS (Web Service quality Management System) to measure QoS of Web Services. It further introduces WSQDL (Web Service Quality Description Language). It advocates the use of the QoS enhanced UDDI for Web Service selection. [JBF07] introduces a concept where consumers publish and retrieve QoS experience via a reputation mechanism. [NK06] concentrates on solving QoS violations in composed Web Services, using rediscovery and reselection. In comparison to all these works which introduce new frameworks, we do not introduce a new framework. This approach is more lightweight, yet Web Services have to comply with the MOWS (Management of Web Services) standard and include a standardized Web Service Ping operation. [Ran03] advocates the QoS enhanced UDDI architecture and discusses which QoS Parameters can be applied to Web Services. [HKKK06] introduces a Web Service Delegation model for providing safety and privacy. This work shows how a Delegation Web Service increases the security of and for Web Services.

3 QoS Parameters for Web Services

There are many QoS parameters which can be applied to Web Services [Ran03]. Different notions of quality exist, ranging from basic metrics like response time or availability to very sophisticated features like security or atomic transaction. As noted in [Lud03], some QoS parameters (like performance) can be measured at different points.

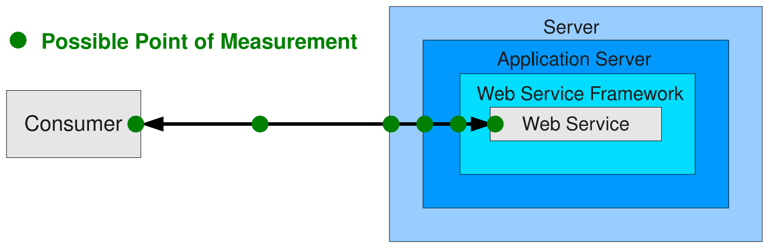

Figure 1: Points of Measurements

Depending on the QoS parameter, there can be up to six measurement points: at the Web Service, at the Web Service framework, at the application server, at the server, on the network (for example at an intermediate SOAP node or at a Delegation Web Service) or at the consumer. This yields multiple inconsistent measurement values. Different parties (for example, the consumer and the Web Service provider) can add an offset or a value computed from other QoS parameters, to control each other's measurement values. In this paper we focus on the measurement point at the Web Service and the measurement point at the consumer. Below you can find examples for these two measurement points. The MOWS (Management of Web Services) parameter LastResponseTime is an example of a QoS parameter measured at the Web Service. The service time measured using the Web Service Ping operation is an example of a QoS parameter measured at the consumer. For measuring QoS on the network, the Web Service Delegation Routing architecture as described below or SOAP nodes which add a time stamp to every SOAP message that passes them can be used. For measuring at the application server, the application server has to be extended (e.g. using the Aspect-Oriented Programming (AOP) paradigm). For measuring at the Web Service framework, the Web Service framework has to be extended (e.g. using the Aspect-oriented programming (AOP) paradigm). For measuring at the server, a proxy program which logs incoming and outgoing SOAP messages has to be used.

4 WSDM for monitoring QoS

WSDM (Web Service Distributed Management) is divided into the two parts: MOWS (Management of Web Services) and MUWS (Management using Web Services) [OAS08c]. MOWS (Management of Web Services) [OAS08c] was created for managing Web Service endpoints. Values from its WSRF properties part, that can be used as QoS parameters or for computing QoS parameters are: NumberOfRequests, NumberOfFailedRequests, NumberOfSuccessfulRequests, ServiceTime, MaxResponseTime and LastResponseTime. LastResponseTime can be used as a QoS parameter directly. ServiceTime divided by NumberOfRequests will compute the AverageResponseTime. Unfortunately, MOWS has few QoS parameters, but since the MOWS standard is extensible, it is possible to add more QoS parameters to the WSRF properties part of MOWS if required. There needs to be a discussion on which QoS parameters to add and how to model them. QoS parameters which are computed using already existing properties can also be added to relieve consumers. MUWS (Management using Web Services) [OAS08c] can be used to manage and monitor IT resources and their QoS parameters. MUWS can be used to make cross-layer management data available within the Web Service layer. Cross-layer data can be used for early detection of bad QoS or QoS violations [BSR+07a]. MUWS can function as an interface to SNMP (Simple Network Management Protocol) to monitor the CPU load of a server hosting Web Services.

5 Web Service Ping for QoS

Ping is a well known simple tool to check the availability of a resource under a particular IP address, the round trip time (RTT) of an IP based request-reply message and to create well defined loads to test IP networks. Ping represents several interesting features which may serve as an inspiration for a simple diagnostic tool for Web Services. To check the availability of a Web Service by a Web Service Ping operation without causing side effects may be useful. Measuring the RTT of a simple Web Service Ping operation, can give valuable information for choosing a suitable instance of a needed Web Service. A Ping concept for Web Services, may support the management of Web Services across organizational boundaries. The idea of a Web Service Ping has been used for performance measurements by [Jec08]. We want to develop these ideas further: We propose to introduce this nonfunctional feature in all Web Service interfaces to offer a simple diagnostic tool, which works across organizational boundaries if standardized. As a basis for this purpose, we propose a declaration of the Web Service Ping in WSDL [Sch08]. Clearly this part of the interface should be generated automatically by setting an appropriate flag of a WSDLtoTargetLanguage generator or by a suitable annotation in the code, which is intended to be wrapped into the Web Service interface. We investigated the language binding to Java with Apache Muse [Fou08] by developing a WS-Ping prototype.
Our proposition of the XML Schema for a Web Service Ping operation is as follows. The XML Schema file as well as WSDL samples can be found under [Sch08].

```
<xsd:element name="WSPing">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="ConsumerRequestTime" type="xsd:dateTime" />
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
<xsd:element name="WSPingResponse" type="xsd:dateTime" />
```

Listing 1: Web Service Ping Operation

Our suggestion includes the following: An element of the XML Schema type dateTime as a request parameter and an element of the XML Schema type dateTime in the response message. It is envisioned to measure three times: the time at the consumer when it sends the request (ConsumerRequestTime), the time at the Web Service (WebServiceTime) and the time at the consumer when it receives the response(ConsumerResponseReceivedTime). The consumer can subtract ConsumerRequestTime from ConsumerResponseReceivedTime to get the service time of the Web Service Ping operation. This service time can be used, for example, for Web Service selection as explained below. The ConsumerRequestTime and the Web ServiceTime can be logged by the Web Service. The consumer can log the WebServiceTime, ConsumerRequestTime and ConsumerResponseReceivedTime. This log information can be used for statistics, detection of QoS shifts and detection of manipulations.

6 Web Service frameworks and QoS

There are different Web Service frameworks, but unfortunately, there is no Web Service framework which implements MOWS or Web Service Ping. Apache Muse [Fou08] implements parts of MUWS and therewith parts of WSRF and WSN. Therefore, it is comparatively easy to implement MOWS conform Web Services with Apache Muse. We advocate the implementation of WSDM and Web Service Ping in all major Web Service frameworks. This would give all consumers, Web Service search engines and Web Service selectors the chance to compare Web Services based on their QoS. There are two ways how MOWS and Web Service Ping can be implemented within Web Service frameworks. It can be implemented within the actual Web Service frameworks deployment as central instance, managing MOWS parameters and Web Service Ping requests for all deployed Web Services. This would move the measurement point of MOWS parameters from the Web Service to the Web Service framework. The second approach is to put the MOWS and Web Service Ping functionalities into the Web Service. This means the WSDL-toTargetLanguage generator should add standard Web Service Ping and MOWS functionalities to the created class. If the code first approach is used, a special class containing MOWS and Web Service Ping functionalities needs to be extended or suitable annotations in the code have to be made. The first approach has the advantage that all already existing Web Services within the Web Service framework won't have to be changed. The second approach has the advantage that the standard implementation of the MOWS functionalities can be overwritten easily.

7 QoS based Web Service selection architectures

Four Web Service selection architectures have been identified. The first and easiest one is the consumer as selector architecture. Then there are two approaches which have been introduced in multiple papers: the QoS based Web Service Broker as selector[TGRS04, DA08] and the QoS enhanced UDDI as selector[Ran03, Lee08]. Then we have a new approach, the QoS based Delegation Web Service as selector. The architectures are illustrated based on a very easy selection example. This example includes two Web Services, where the first Web Service has a better QoS and will be chosen. In the illustrated sample, all MOWS functionalities are within the Web Services (inband). They could as well be outband (additional MOWS providing Web Services). Web Services can be discovered using Web Service discovery. Possibilities to discover Web Services include: UDDI (Universal Description Discovery & Integration) [OAS08a], Service Domain Inventories [Erl08], Enterprise Web Service Inventories [Erl08], Service Groups [OAS08d], relationships between Web Services, Web Service Aggregation and Web Service Announcements. The Web Service discovery process is illustrated by a simple arrow within the figures. The sample also includes MUWS Web Services that are used to retrieve IT resources' QoS parameters important to the Web Services' QoS such as the Web Services' server's CPU load. The architectures advantages and disadvantages will be discussed, making it easier to choose one based on requirements and design decisions.

7.1 Consumer as selector

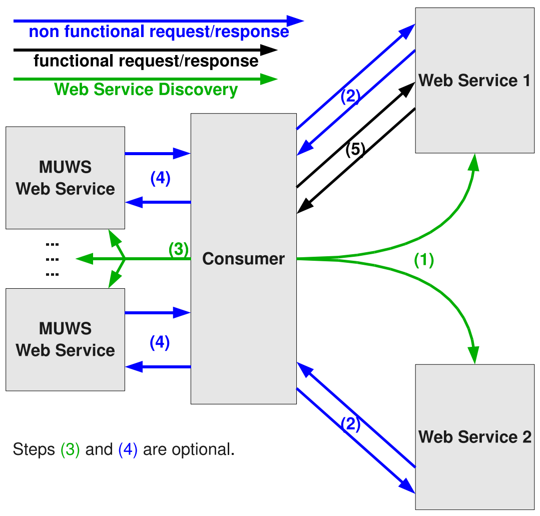

Figure 2: Consumer as selector

The first approach is to put the selection mechanism into the consumer. The consumer can be the actual consumer or a Web Service using further Web Services.

Advantages of consumer as selector:

- Does not introduce single point of failure.
- Does not introduce bottleneck.
- Does not introduce further central Web Services (decreases the total number of Services and the complexity of the architecture).
- Consumer controls selection and discovery (does not depend on third parties interest).
- Consumer can control stated QoS values.

Disadvantages of consumer as selector:

- Increases the complexity of the consumer.
- If new selection algorithms should be introduced or old selection algorithms should be updated, all consumers have to be updated.
- Consumer gains inside knowledge of the Web Services and the Web Services' network structure.
- Every consumer makes its own decision, therefore, this architecture cannot be used for Web Service load balancing.
- No centralized disqualification of unreliable Web Service providers.

7.2 Web Service Broker as selector

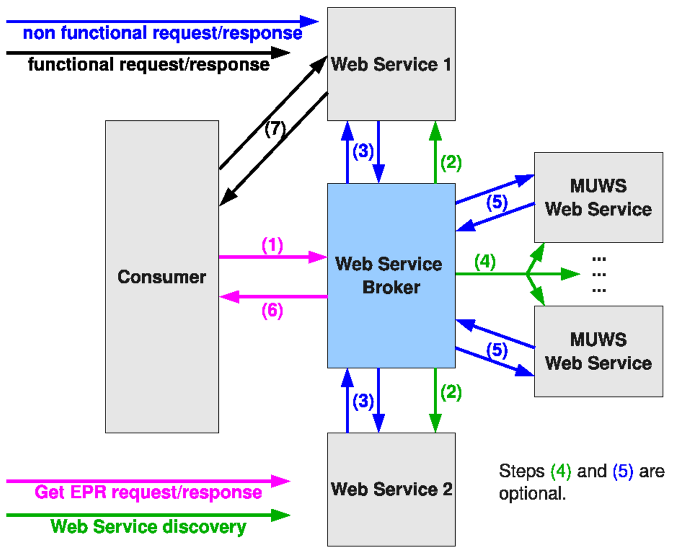

Figure 3: Web Service Broker as selector

Another approach is to put the selection functionality into a Web Service Broker, as discussed in [TGRS04, DA08]. The consumer will send a request (containing the wished functionality, special selection preferences and QoS requirements) to the Web Service Broker. The Web Service Broker will respond with the address of the best fitting Web Service. The consumer can access the Web Service directly, using the provided address.

Advantages of Broker as selector:

- Consumers are eased and their complexity is reduced.
- Web Service Broker can reuse selection decisions for multiple consumers.
- Central introduction of new selection algorithms and update of selection algorithms.
- Central disqualification of unreliable Web Services and Web Service providers is possible.
- Can be used for Web Service load balancing to a certain degree.

Disadvantages of Broker as selector:

- Introduces Single point of failure.
- Consumer still gains knowledge on Web Services.
- Consumer loses control over selection decisions and Web Service discovery.
- Web Service Broker can become a bottleneck.

7.3 QoS enhanced UDDI as selector

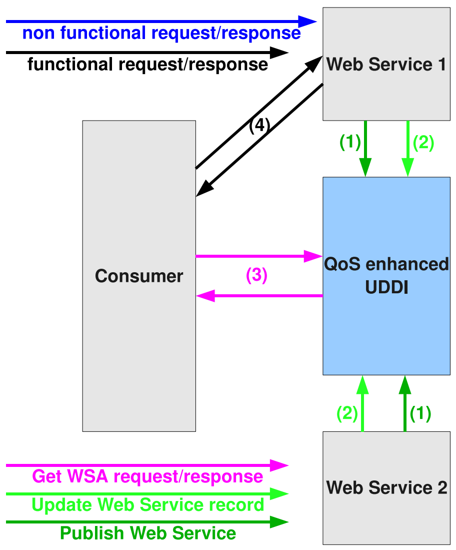

Figure 4: QoS enhanced UDDI as selector

Another approach is the QoS enhanced UDDI registry as discussed in [Ran03, Lee08]. It is similar to the Web Service Broker architecture, but the Web Service Broker is replaced with a QoS enhanced UDDI registry. Web Services have to publish themselves to the UDDI. Web Services need to provide their QoS parameters, when publishing themselves. Web Services also have to update their QoS parameters. The consumer can request a Web Service sending wished functionality, QoS requirements and QoS preferences to the QoS enhanced UDDI. The response will be the address of the best fitting Web Service. The consumer can control whether the Web Service really keeps its QoS promises and publish the results to the UDDI.

Advantages of QoS enhanced UDDI as selector:

- Central introduction of new selection algorithms and update of selection algorithms.
- Consumer are eased and their complexity is reduced.
- Existing UDDI registries can be used, by extending them with QoS.
- QoS enhanced UDDI can reuse selection decisions for multiple consumers.
- Central disqualification of unreliable Web Services and Web Service providers is possible.
- Can be used for Web Service load balancing to a certain degree.

Disadvantages of QoS enhanced UDDI as selector:

- Consumer loses control to a certain degree.

- MUWS won't be used, as keeping track of all the MUWS Web Services including their QoS parameters and their assignment to Web Services would overload the UDDI.
- UDDI needs frequent updates.
- UDDI can contain non-existing Web Services as well as wrong and inconsistent QoS values.
- UDDI registries can be manipulated by consumers and Web Service providers.
- Consumer still gains knowledge on Web Services.

7.4 Delegation Web Service as selector

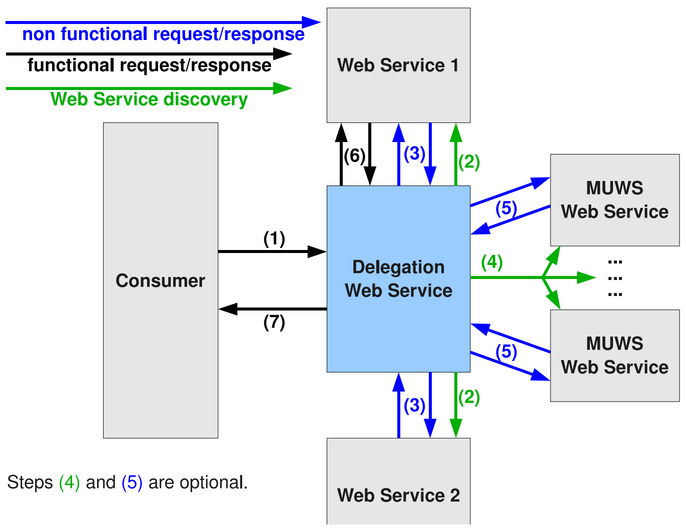

Figure 5: Delegation Web Service as selector

A new approach is the Delegation Web Service as selector architecture. There should be one Delegation Web Service for each Web Service type, as this simplifies the architecture as well as the Delegation Web Service's interface. However, one Delegation Web Service can also be used for multiple Web Service types. The Delegation Web Service does not implement any functional parts. It delegates functional request to corresponding Web Services. The Delegation Web Service will consume functional as well as nonfunctional parts from used Web Services. Nonfunctional parts will include MOWS and Web Service Ping, as well as MUWS from multiple MUWS Web Services. The Delegation Web Service will decide which Web Service it delegates to. The decision is based on nonfunctional QoS parameters, consumer QoS requirements and consumer preferences.

Advantages of Delegation Web Service as selector:

- Web Services and network structures are hidden from consumers.
- Delegation Web Service can reuse selection decisions for multiple consumers.
- Delegation Web Service is predestinated to implement Web Service load balancing.
- Central introduction of new selection algorithms and update of selection algorithms.
- Can be combined with the Web Service Broker as selector approach (Delegation Web Service can delegate the selection decision to the Web Service Broker).
- Request parameters can be checked, wrong request can be discovered and a SOAP Fault can be send directly. This increases Web Services' security.

Disadvantages of Delegation Web Service as selector:

- Introduces single point of failure.
- Delegation Web Service can become a bottleneck, as it processes all functional requests.
- Consumer totally loses control over selection decision and nearly every possible controlling.

8 Conclusion

QoS based Web Service selection is an important research area. Web Service selection can increase performance, reduce costs and positively influence important QoS parameters. A uniform, consistent, lightweight and standardized solution for mining QoS parameters should be aspired. This paper showed how to use MOWS, MUWS and Web Service Ping for this task. As there is a wide range of QoS parameters, MOWS has to be extended with further QoS parameters. The decision should always be based on QoS parameters important to the specific consumer . Web Service Ping should be implemented as a simple and uniform Web Service Ping operation. Different Web Service selection architectures have been discussed. The newly presented Delegation Web Service as selector architecture, hides the Web Services and is predestinated to implement Web Service load balancing. The architectures consumer as selector, Web Service Broker as selector and Delegation Web Service as selector have been implemented according to the illustrations using Apache Muse [Fou08]. All samples can be found under [Sch08]. Further and upcoming work includes the implementation of the Web Service selection samples in other Web Service frameworks, implementation of bigger samples, measuring and comparing the performance of different Web Service selector architectures in multiple scenarios and the investigation of different selection algorithms.

Acknowledgements: Special thanks to Annabelle Lee and Nicolas Repp for great support and reviews.

Bibliography

[BGR05] R. Berbner, T. Grollius, N. Repp. An approach for the Management of Service-oriented Architecture (SoA) based Application Systems. *Enterprise Modelling and Information Systems Architectures*, Oct. 2005.

[BSR$^+$07a] R. Berbner, M. Spahn, N. Repp, O. Heckmann, R. Steinmetz. Dynamic Replanning of Web Service Workflows. *Digital EcoSystems and Technologies Conference.*, pp. 211–216, Feb. 2007.

[BSR$^+$07b] R. Berbner, M. Spahn, N. Repp, O. Heckmann, R. Steinmetz. *Service-Oriented Computing ICSOC 2007*. Section WSQoSX A QoS Architecture for Web Service Workflows, pp. 623–624. Springer, Berlin / Heidelberg, Aug. 2007.

[DA08] D. A. DMello, V. S. Ananthanarayana. Quality Driven Web Service Selection and Ranking. In *Proceedings of the Fifth International Conference on Information Technology: New Generations*. Volume 5, pp. 1175–1176. Chicago, Apr. 2008.

[Erl08] T. Erl. SOA Patterns. Technical report, Oct. 2008.
http://www.soapatterns.org/

[Fou08] A. S. Foundation. Apache Muse. Technical report, Oct. 2008.
http://ws.apache.org/muse/

[GNCW03] X. Gu, K. Nahrstedt, R. N. Chang, C. Ward. QoS-Assured Service Composition in Managed Service Overlay Networks. In *International Conference on Distributed Computing Systems*. Volume 23, pp. 194–201. May 2003.

[HKKK06] H. S. Hwang, H. J. Ko, K. I. Kim, U. M. Kim. Agent-Based Delegation Model for the Secure Web Service in Ubiquitous Computing Environments. In *Proceedings of the 2006 International Conference on Hybrid Information Technology*. Volume 1, pp. 51–57. Nov. 2006.

[IBM08] IBM. Web Service Level Agreements (WSLA) Project. Oct. 2008.
http://www.research.ibm.com/wsla/

[inn08] innoQ. Web Service Standards Overview. Oct. 2008.
http://www.innoq.com/soa/ws-standards/poster/

[JBF07] R. Jurca, W. Binder, B. Faltings. Reliable and Inexpensive QoS Monitoring in Service Markets. *ERCIM NEWS*, 2007.

[Jec08] M. Jeckle. Ping for SOAP – SOA Ping. Oct. 2008.
http://www.jeckle.de/freeStuff/soaping/index.html

[Lee08] Y. Lee. Quality Context Composition for Management of SOA Quality. *2008 IEEE International Workshop on Semantic Computing and Applications*, pp. 117–122, Sept. 2008.

[Lud03] H. Ludwig. Web services QoS: external SLAs and internal policies or: how do we deliver what we promise? *Fourth International Conference on Web Information Systems Engineering Workshops.* 4:115–120, Dec. 2003.

[MRD08] O. Moser, F. Rosenberg, S. Dustdar. Non-Intrusive Monitoring and Service Adaptation for WS-BPEL. In *Proceeding of the 17th international conference on World Wide Web*. Pp. 815–824. Beijing, Apr. 2008.

[NK06] X. T. Nguyen, R. Kowalczyk. An agent based QoS conflict mediation framework for Web Services Compositions. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*. Pp. 522–528. Dec. 2006.

[OAS08a] OASIS. Universal Description, Discovery and Integration (UDDI). Oct. 2008. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec

[OAS08b] OASIS. Web Service Notification (WSN). Oct. 2008. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

[OAS08c] OASIS. Web Services Distributed Management (WSDM). Oct. 2008. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

[OAS08d] OASIS. Web Services Resource Framework (WSRF). Oct. 2008. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

[Ran03] S. Ran. A Model for Web Services Discovery With QoS. *ACM SIGecom Exchanges* 4(1):1–10, 2003.

[Sch08] T. Schoene. Download material. Oct. 2008. http://www.kuruk.de/index.php?id=42&L=0

[SK08] M. Schmid, R. Kroeger. *Distributed Applications and Interoperable Systems*. Section Decentralised QoS-Management in Service Oriented Architectures, pp. 44–57. Springer, Berlin / Heidelberg, May 2008.

[TGRS04] M. Tian, A. Gramm, H. Ritter, J. Schiller. Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*. Pp. 152–158. Sept. 2004.