The QoS-MO Ontology for Semantic QoS Modeling

Gustavo Fortes Tondello
Federal University of Santa Catarina
Informatics and Statistics Department
Florianópolis, SC – Brazil – 88040-900
fortes@inf.ufsc.br

Frank Siqueira
Federal University of Santa Catarina
Informatics and Statistics Department
Florianópolis, SC – Brazil – 88040-900
frank@inf.ufsc.br

ABSTRACT
This paper presents the QoS-MO ontology. This ontology enables the specification of QoS requirements for Semantic Web Services and can easily be combined with OWL-S in order to fully describe Web Services. The QoS specifications created using the QoS-MO ontology may be employed on the design and development of Web Services and on the publication and discovery of Web Services on the Semantic Web.

Categories and Subject Descriptors
D.2.12 [**Software Engineering**]: Interoperability – *Interface definition languages*.

General Terms
Documentation, Design, Standardization.

Keywords
Semantic Web Services, QoS, Quality of Services, OWL-S.

1. INTRODUCTION
The Semantic Web is defined by Berners-Lee, Hendler and Lassila as "an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [1].

One of the core technologies of the Semantic Web is Semantic Web Services. The well-defined semantics of these services makes them suitable for automatic publication, discovery, composition and execution.

In this paper we present the Quality of Service Modeling Ontology (QoS-MO). Its main goal is to allow semantic specification of QoS constraints of Semantic Web Services. Several existing QoS specification models were studied by the authors with the purpose of bringing the best characteristics presented by each one to the semantic web scenario and for addressing the flaws that exist in similar proposals.

Quality of Service (QoS) is the quality delivered by one service, expressed by means of non-functional characteristics with quantifiable parameters [6]. A QoS-MO specification may be used along the whole life cycle of the Web Service, from its design until its utilization. On the design phase, a QoS-MO specification may be converted to/from a UML specification that follows the OMG UML Profile. On the publication and execution phase, a QoS-MO specification may be integrated with the existing OWL-S description of the functional characteristics of the Web Service, extending the OWL-S standard [7] with well-defined QoS semantics.

The remainder of this paper is structured as follows. Section 2 presents related works that were taken as reference for the definition of QoS-MO. Section 3 presents the structure and specification of QoS-MO. Section 4 describes a Semantic Web Service search tool that uses the QoS information expressed with QoS-MO to find the Web Services. Section 5 presents the conclusions and suggestions for future works.

SAC'08, March 16-20, 2008, Fortaleza, Ceará, Brazil.

2. RELATED WORKS
This section presents the existing QoS models that were studied by the authors in order to design the QoS-MO ontology.

2.1 OMG QoS Metamodel
The OMG QoS framework [6] metamodel defines the abstract language for a modeling language that supports modeling general QoS concepts. It was designed as a metamodel to the definition of the QoS UML Profile. This metamodel is widely accepted in its area of research due to its adequacy for modeling QoS concepts, and is also adopted as the basic reference for the definition of the QoS-MO ontology. This approach has two main advantages: the first one is that a well studied standard that addresses all the requisites to a proper QoS profile specification is being adopted. The second advantage is that QoS-MO will have a tight correspondence with the OMG UML profile, thus facilitating the task of automatic conversion of a QoS-MO based QoS description to/from one based on the UML profile.

The OMG QoS metamodel defines three packages. The QoS Characteristics package contains the model elements for the description of QoS Characteristics, QoS Dimensions (different ways to quantify a characteristic) and QoS Contexts (quality constraints that combine several QoS Characteristics). The QoS Constraints package includes the modeling elements for the description of QoS contracts and constraints. The QoS Levels package includes the modeling elements for the specification of QoS modes and transitions.

The QoS UML Profile is the implementation of the QoS framework metamodel that extends UML 2.0 with QoS specification capabilities. The meta-classes of the metamodel are

defined as UML stereotypes. Using the OMG UML Profile, QoS Characteristics are UML Classes with the appropriate stereotypes and the relationships between UML elements may have QoS Contracts (annotations with the appropriate stereotype).

2.2 Existing QoS Specification Languages

Several existing QoS Specification languages have been analyzed, trying to take the best characteristics of each one and adapt them to be employed in the proposed ontology model.

The analyzed specification languages were: Quality of Service Modeling Language (QML) [3], Web Service Level Agreement (WSLA) Language [5], QoS Specification Language (QSL) [8] and QoS Description Language (QDL) [13]. Despite providing mechanisms for QoS specification, these languages lack the flexibility provided by an ontology for dealing with the semantic meaning of QoS constraints.

2.3 Existing QoS Modeling Ontologies

The QoSOnt ontology [2] defines a model for the specification of QoS for Semantic Web Services. It allows the association of a QoS specification with an OWL-S service profile. However, it does not provide any means to specify a QoS profile from a set of QoS characteristics, nor to reuse or extend a previous specification. QoSOnt has a mechanism that allows the conversion of different units of QoS metrics, but it has no specific mechanism to map different QoS parameters.

The DAML-QoS ontology [12] defines a model for the specification of QoS parameters and profiles. DAML-QoS allows the specification of a QoS profile as a set of QoS characteristics. Nevertheless, there is no possibility to extend an existing profile or create a new profile as a composition of others. It also allows the definition of complex metrics with a function that makes possible to calculate their values from the values of other metrics. The main flaw of DAML-QoS is that it uses the cardinality restrictions between a QoS profile and its QoS metrics to set the values of QoS characteristics, thus limiting their values to non-negative integer numbers.

The OWL-Q ontology [4] is an upper level ontology which extends OWL-S to describe the possible parts of QoS metrics and constraints. Using OWL-Q it is possible to define complete QoS specifications, and it also provides means to specify unit conversion and composite metrics. In addition, semantic matchmaking and selection algorithms that use the OWL-Q ontology have been developed by the authors.

2.4 QoS-enabled Semantic Web Services search engines

In [10] the authors describe a QoS-enabled Semantic Web Service discovery framework. Although they have briefly discussed a semantic description model for QoS of Web Services, the main focus of this work is on the discovery and ranking mechanisms of Web Services and on the description of a solution for dynamic assessment and management of QoS values based on user feedback. In section 4 we will also describe the design of a Semantic Web Services search tool in order to validate our proposal, but our main contribution is the QoS modeling ontology, not the search tool itself.

3. THE QoS-MO ONTOLOGY

The Quality of Service Modeling Ontology (QoS-MO) is an upper level ontology that contains elements for the description of QoS Characteristics and Constraints of Web Services described in OWL-S. To instantiate a QoS description of a particular Web Service, one must create a new ontology that will import the QoS-MO ontology, extend its classes as needed and create the required individuals. The functional and non-functional (QoS) specifications of a Web Service may be described on a single ontology or on two separated ones. Figure 1 depicts the hierarchy of ontologies used for QoS definition of a Web Service.

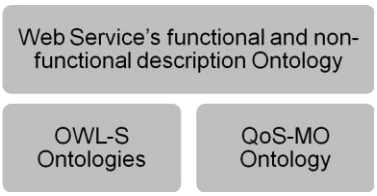

Figure 1. Ontologies for QoS description of a Web Service.

The *Protégé Ontology Editor and Knowledge Acquisition System* [9] was employed to build the QoS-MO ontology. In the future we intend to provide the developer with a tool to facilitate the creation of a new ontology to model Web Services using QoS-MO without modifying the QoS-MO ontology itself.

3.1 QoS Characteristics

The class **QoSCharacteristic** is the main class in the QoS-MO ontology for the definition of quantifiable characteristics of services. Characteristics may be general like latency, throughput, availability, reliability, security and accuracy, or domain specific ones. QoS characteristics may be grouped in categories using the class **QoSCategory**. Both classes can take advantage of extension/specialization: QoSCharacteristics may be reused and extended, while QoSCategories may form a hierarchical classification system. New characteristics and categories are created as subclasses of these ones.

The dimensions for the quantification of characteristics are modeled using class **QoSDimension**. One QoS Characteristic may have more than one way of measuring it, or it may require more than one dimension for its quantification. Examples of dimensions for latency are (from [6]): minimum latency, maximum latency and jitter. The property **direction** of the QoSDimension class specifies whether the values of an ordered dimension are increasing or decreasing. The property **unit** allows the specification of the measurement unit, and property **statisticalQualifier** identifies the type of the statistical qualifier (i.e., average, standard deviation, etc.) when the dimension represents a statistical value.

The class **QoSContext** allows the definition of quality expressions that combine multiple QoS Characteristics. A QoS Characteristic defines a QoS Context that references only itself; hence QoSCharacteristic is a subclass of QoSContext. A QoS Context can be composed of other contexts or characteristics.

Class QoSCharacteristic does not have a direct relationship with class QoSDimension. Instead, dimensions are created as subclasses of QoSDimension and a subclass of QoSCharacteristic will have new object properties which range will be a subclass of QoSDimension. This allows the definition of dimensions as properties of the characteristic class. This strategy is the same used on the QoS UML Profile. A QoSDimension may also be defined as a reference to a QoSCharacteristic, making possible the composition of characteristics within another one.

Figure 2 shows the modeling elements for QoS Characteristics.

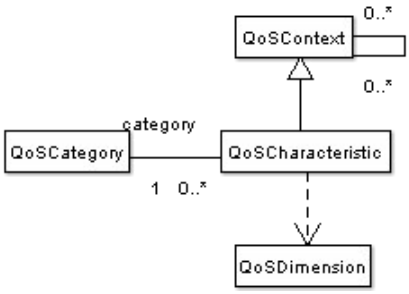

Figure 2. Modeling elements for QoS Characteristics.

Figure 3 shows an example of a QoS profile for the OWL-S example BravoAir Web Service [7]. The service has a QoS Context represented by the class **BravoAirQoS**, with two characteristics represented by classes **ResponseTime** and **Availability**. Each characteristic has some dimensions.

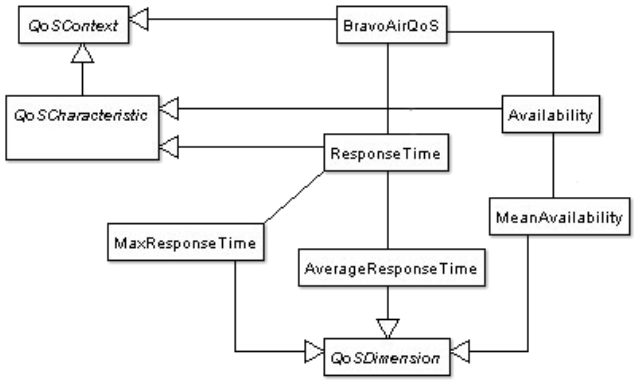

Figure 3. Example of a QoS profile definition.

3.2 Dimension Mapping

Mapping scripts between dimensions may be defined using class **QoSDimensionMapping**. This class has two properties, **sourceDimension** and **targetDimension**, to identify dimensions involved on the mapping. This allows search mechanisms or automatic QoS negotiation middleware to identify a dimension even if its value is not given for a particular characteristic, provided that there is a mapping script between it and any other dimensions of the same characteristic or even of other characteristics.

Class QoSDimensionMapping has a property **expression** that defines the mapping script. Mapping scripts are defined using logical and arithmetic operations and if/else conditional expressions. All the operations are written on the same syntax accepted by C, C++ and Java.

For example, there could be a QoSDimensionMapping individual with the targetDimension defined as the AverageResponseTime dimension, the sourceDimension defined as the set of AverageRequestTime, AverageExecutionTime and AverageReplyTime dimensions, and the expression defined as:

```
AverageResponseTime = AverageRequestTime +
AverageExecutionTime + AverageReplyTime;
```

3.3 Context Instantiation

A QoS Context and the other involved elements have to be instantiated to create a QoS definition for a particular Web Service, i.e., ontological individuals of each one of the classes must be created.

There could be a value property on the dimension classes to allow the setting of dimension values when instance individuals are created. However, it is better to define dimension values separated from the definition of the dimension itself. To accomplish this, there is another simple class named **QoSValue**. This class has only one property named **value**. An individual of class QoSDimension that must have a value must also be an individual of class QoSValue.

3.4 QoS Constraints

There are three types of QoS Constraints: **QoS Offered**, **QoS Required** and **QoS Contract**.

When a provider service defines a QoS Offered, it defines the quality of the service that it will be providing for its clients. When a client service defines a QoS Offered, it defines the constraints it will guarantee when invoking the provider service.

When a provider service defines a QoS Required, it defines the constraints that the client must achieve to get the expected quality. When a client service defines a QoS Required, it defines that the provider service must achieve some quality constraints. A QoS Required definition may also be created by a search mechanism to express the QoS constraints defined by the user and compare it with the QoS Offered definitions of the services being searched.

A QoS Contract constraint represents the final quality agreed between two services on an assembly. A QoS Contract may be statically calculated when all QoS characteristics involved have static values. But sometimes the contract depends on the resources available or characteristics defined dynamically. Even at this case, it is possible to statically identify the characteristics involved and some limit values based on offered and required QoS constraints of the services involved.

A QoS Constraint must reference the QoS Context or Contexts that provide the reference expressions and values associated to the constraint.

It is also possible to specify a **QoSCompoundConstraint** to define a global constraint decomposed into a set of subconstraints.

Figure 4 shows the modeling elements for QoS Constraints.

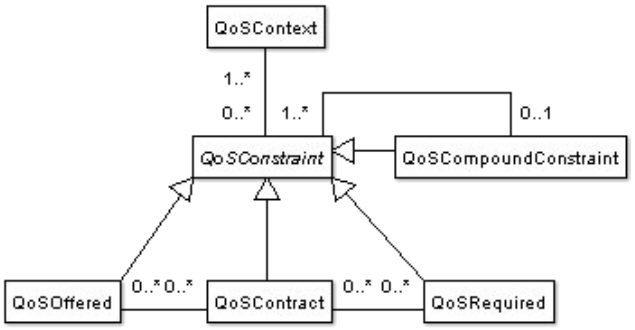

Figure 4. Modeling elements for QoS Constraints.

3.5 QoS Levels

QoS Levels represent the different modes of QoS that a service can support. The QoS Constraints associated with a QoS Level represent the constraints that the service must satisfy to operate on such level. It is also possible to represent a **QoSCompoundLevel** as a set of different QoS Levels.

When a service is no longer capable of operating at a given level, for example, when it is receiving more requests and its max response time constraint will have to change, a **QoS Transition** occurs. When this happens, probably the services involved will have to renegotiate their execution parameters and QoS Contracts. A QoS Transition definition may specify all the adaptation actions necessary for the transition.

Figure 5 shows the modeling elements for QoS Levels.

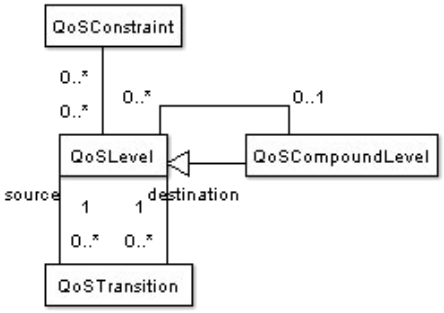

Figure 5. Modeling elements for QoS Levels.

3.6 OWL-S Extension

The QoS-MO Ontology may be used to extend OWL-S with QoS related constraints.

The QoS Constraints modeled with QoS-MO may refer to a Service as a whole or to a specific Process of a Service. Therefore, a QoS Constraint can have an association with an instance of either an OWL-S Profile or an OWL-S Process.

QoS Offered and QoS Required constraints will associate with only one Profile or Process. QoS Contracts will associate with two different Profiles or with two Processes from different Services that are involved on the contract.

Figure 6 shows the modeling elements for OWL-S extension.

Figure 7 puts everything together and depicts a complete example of an extension to the BravoAir Web Service profile [7] represented by the **Profile_BravoAir_ReservationAgent** individual. The QoS Offered constraint **BravoAir_Offered** is defined with one QoS Context **BravoAirQoS** that has one QoS Characteristic **BravoAir_ResponseTime** with one dimension.

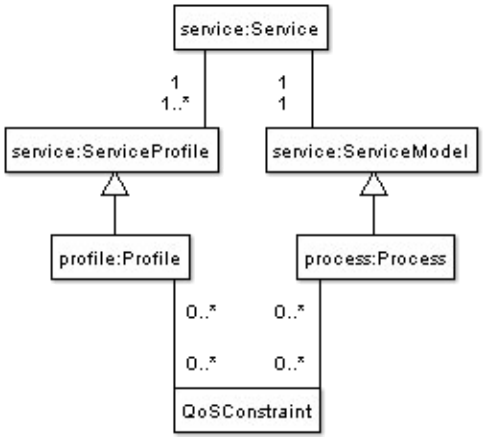

Figure 6. Modeling elements for OWL-S extension.

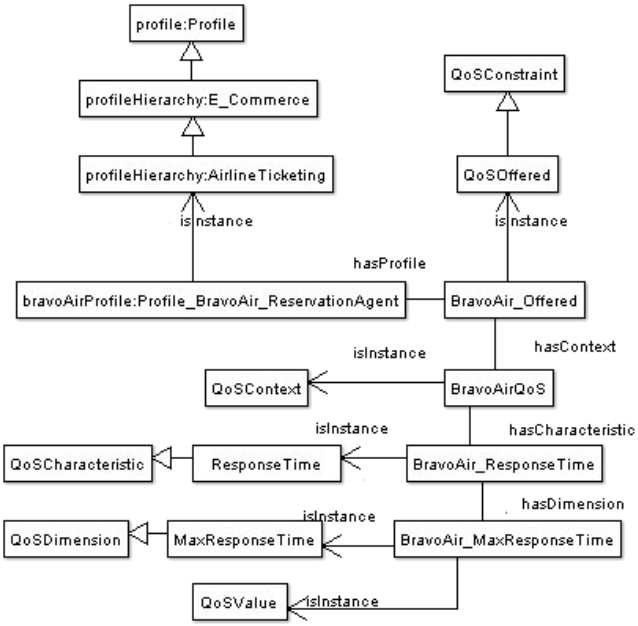

Figure 7. An example of extension of the BravoAir Web Service OWL-S profile with QoS constraints.

4. A QoS ENABLED WEB SERVICES SEARCH TOOL

As a proof of concept, we have designed a search tool that searches for Web Services modeled with OWL-S that fulfill some specific QoS requisites. This tool writes SPARQL [11] queries to search for QoS models that are within the specified constraints and return the selected Web Services.

Our first tests showed that it is possible to find and filter QoS specifications that fulfill certain QoS constraints using a SPARQL query. The following is the SPARQL query to find a

QoS Offered constraint that has any QoS Context with the ResponseTime characteristic with a MaxResponseTime dimension which value is lower than 10 seconds. Table 1 presents the results after execution of the query.

```
SELECT
  ?constraint ?context ?charact
  ?dimension ?value
WHERE {
  ?constraint rdf:type :QoSOffered.
  ?constraint :Context ?context.
  ?context :BasedOn ?charact.
  ?charact rdf:type :ResponseTime.
  ?charact ?predicate ?dimension.
  ?predicate rdfs:range :MaxResponseTime.
  ?dimension :value ?value.
  FILTER (?value < 10)
}
```

Table 1. Results after execution of SPARQL query

constraint	context	charact	dimension	value
BravoAir Offered	BravoAir QoS	BravoAir Response Time	BravoAir MaxResponse Time	5

The search tool consists on a Web interface where the user enters the QoS Constraints he/she wants to search; a component that converts these constraints to a SPARQL query and executes this query on the ontology that contains the QoS descriptions of the Web Services; and a component that retrieves all the Web Services matched by the query and returns them to the user.

The search results can be presented classified according to the level of quality presented by the Web Services: those with better quality will be presented first. When the constraint specified by the user has only one characteristic, it is easy to sort the returned services according to their quality level. When there is more than one characteristic on the search query, the user must specify which characteristic or characteristics he/she finds more important to be considered for sorting so that the tool will know how to present the results.

5. CONCLUSION

This paper presented the QoS-MO ontology for the description of QoS characteristics of Web Services on the Semantic Web.

The construction of the ontology was based on various existing QoS models, specially the OMG QoS Framework Metamodel. The QoS-MO ontology allows the extension of OWL-S specifications of Web Services with well-defined QoS Constraints. These constraints can be used by search tools for helping users to find Web Services that guarantee some level of quality besides providing the required functional aspects, or by the Web Services execution framework to guide the negotiation of QoS parameters when one client is going to invoke a service.

QoS specifications built with QoS-MO may also be converted to/from specifications built with the QoS UML Profile, allowing the integration of the QoS model in the design phase of the development life cycle of Web Services.

As a proof of concept, we have designed a search tool that is able to search within an ontology of Web Services descriptions made with QoS-MO for Web Services that meet certain quality constraints, and return the OWL-S and the QoS-MO descriptions of the matched services.

The use of this search tool will help to test the expressiveness of the proposed ontology and the performance of a search tool built upon execution of SPARQL queries over an ontology of Web Services functional and QoS descriptions.

6. REFERENCES

[1] Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, May 2001.

[2] Dobson, G., Lock, R. and Sommerville, I. QoSOnt: a QoS Ontology for Service-Centric Systems. *31st Euromicro Conference on Software Engineering and Advanced Applications* (Euromicro SEAA '05). Porto, Portugal, 2005.

[3] Frølund, S., Koistinen, J. *QML: A Language for Quality of Service Specification*, 1998. http://www.hpl.hp.com/techreports/98/HPL-98-10.html

[4] Kritikos, K., Plexousakis, D. Semantic QoS Metric Matching. *4th European Conference on Web Services* (ECOWS '06), December 2006, pp.265-274.

[5] Ludwig, H., Keller, A., Dan, A., King, R.-P., and Franck, R. *Web Service Level Agreement (WSLA) Language Specification*. January 2003. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[6] Object Management Group. *UML Profile for Modeling QoS and FT Characteristics and Mechanisms Specification, v1.0*. May 2006.

[7] OWL-S Coalition. *OWL-S 1.1 Release*. November 2004. http://www.daml.org/services/owl-s/1.1/

[8] Siqueira, F. Especificação de Requisitos de Qualidade de Serviço em Sistemas Abertos: A Linguagem QSL. *20th Brazilian Symposium on Computer Networks* (SBRC'2002). Búzios - RJ, Brazil, 2002.

[9] Stanford University. *The Protégé Ontology Editor and Knowledge Acquisition System*. 2007. http://protege.stanford.edu/

[10] Vu, L.-H., Hauswirth, M., Porto, F., and Aberer, K. A Search Engine for QoS-enabled Discovery of Semantic Web Services. *Special Issue of the International Journal on Business Process Integration and Management* (IJBPIM), Vol. 1, No. 4, 2006, pp.244–255.

[11] W3C. *SPARQL Query Language for RDF*. Candidate Recommendation, June 2007. http://www.w3.org/TR/rdf-sparql-query/

[12] Zhou, C., Chia, L. and Lee, B. DAML-QoS ontology for Web services. *IEEE International Conference on Web Services* (ICWS'04). San Diego, California, USA, 2004, pp.472-479.

[13] Zinky, J., Bakken, D. and Schantz, R. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Object Systems*, Vol. 3(1), January 1997.