

Semantics in QoS for Web Services: A Survey

Vuong Xuan TRAN¹ and Hidekazu TSUJI²

¹*Graduate School of Science and Technology, Tokai University, Japan, txvuong@yahoo.com*

²*School of Information Technology & Electronics, Tokai Univ., Japan, htsuji@keyaki.cc.u-tokai.ac.jp*

Quality of service (QoS) plays an important role in Web service discovery, especially in automatic Web service composition. Depending on the kinds of applications, different organizations and companies may adopt different QoS models for expressing quality of provided Web services. This leads to the issue of interoperability between and among service participants which join in service cooperation. Semantic technology has been applied in recent works aiming at providing an efficient and compatible method for describing and reasoning about QoS of different systems. In this paper, we present an overview of prominent research works related to developing QoS ontologies and discuss their advantages and shortcomings. Basing on such analysis, we indicate open issues that need further investigation in ongoing works aiming at developing and using QoS ontologies.

1. Introduction

Web service technology offers a promising approach for building distributed business processes and applications which can be accessible via the Internet. When individual Web services are not able to meet complex requirements they can be combined to create new value added composite services for those requirements. One of the fundamental operations related to Web service usage is the discovery of Web services. Basically, standards like WSDL can support service providers describe their services' functionality through standard interfaces and advertise them in some UDDI registries. When a service request is issued, available Web service descriptions and the service request description are matched together in order to find the services that can provide expected functionality (the matching step).

However, Web service functional descriptions are not sufficient for service discovery process. There are several reasons for this fact. *Firstly*, a key advantage of Web service technology is to enable Web services to be dynamically and automatically discovered and selected at runtime. In this case, an automatic mechanism is needed to support a system determine the best services to be chosen. *Secondly*, as more and more Web services are created by many providers and vendors, there is often the case where a number of Web services can satisfy functional requirements of a service request. Those reasons lead to the issue of ranking and selecting of the best Web services for a request among a list of candidate Web services which can provide similar functionality for the request.

Mainly, quality of service (QoS) information is used for computing the quality degree of candidate Web services. Such QoS information can be performance (in terms of response time, latency...), availability, accessibility, security, etc. [17]. These QoS information have substantial impacts on user's expectation and experience of using a Web service. Therefore they can be used as a main factor to distinguish quality of Web services. A Web service with highest ranking value will be selected. This ranking step is often performed after the above matching step.

Because different service providers and requesters may use different languages [1, 2, 3, 4] and models [5, 6, 7, 8, 9] for QoS advertisements and requirements, it is necessary to find a way for a system understanding different QoS concepts in QoS descriptions. Besides that, different domains and applications may require different QoS properties; therefore we need a more efficient and flexible method to express QoS information. Semantic technology, especially ontology, can be used for semantic interoperability of QoS. In this paper, we review existing approaches that have been introduced recently in order to develop QoS ontologies. We will discuss achievements and shortcomings of those approaches and then present open issues that have not been considered.

The remainder of this paper is structured as follows. In section 2, we review main QoS ontologies and also discuss about advantages and shortcomings of those approaches. Section 3 gives an analysis summary and comparison of existing QoS ontologies in order to indicate open issues that need further investigation in ongoing works. We then conclude the paper in section 4.

2. Existing Approaches for QoS Ontologies

OWL-Q [10] is an upper level ontology that extends OWL-S for describing QoS of Web services. It can be used by both service providers and service requesters. The ontology is separated into multiple facets which can be developed and extended independently. Each of these facets concentrates on a particular part of a QoS Web service description. They are summarized below.

The *Connecting Facet* supports linking OWL-Q ontology with OWL-S ontology. It also provides high level concepts and vocabulary for defining QoS advertisements and requirements. In this facet, a QoS attribute can refer to any service elements such as inputs, effects, conditions, etc. Each QoS attribute can be static or dynamic and it can be measured by one or more static or dynamic metrics respectively.

The *Basic Facet* associates a service profile with several QoS offers (given by providers) or with only one QoS request (given by requesters). This facet includes concepts for specifying QoS specification like security and transaction protocols, cost and associated currency, as well as validity period.

The *QoS Metric Facet* describes classes and properties used for defining QoS metrics. It contains relationships between a QoS metric and 1) a QoS attribute, 2) parties who provide metric values, 3) domains that the metric belongs to, 4) types of metrics (static or dynamic, simple or complex), and 5) other metrics (independent or related).

The *Function, Measurement Directive and Schedule Facets*: the Measurement Directive facet is used for measuring simple metrics while the Function and Schedule facets are used for computing and measuring complex and/or dynamic metrics.

The *Unit Facet* describes the unit of a QoS metric. It also contains concepts that are used to convert values of equivalent metrics having different units.

The *QoSValueType Facet* describes the value types, such as string, number, range, list, boolean, etc. that a QoS metric can take.

The OWL-Q ontology provides a detailed specification for modeling QoS information, especially those related to metrics, units, value types and relationships among QoS properties. The modular approach makes this ontology flexible to add more necessary support ontologies for an applied application, such as time or currency ontologies. However, it has weak support for the usage of the ontologies, for example QoS priority, quality level, QoS mandatory. It also does not include concrete definitions of common QoS properties.

The **QoS ontology language** in [11] focused on the development of a QoS ontology and vocabulary for specifying Web service QoS information. Their QoS ontology contains necessary classes for describing QoS

attributes: *QoSParameter* representing QoS attributes; *Metric* defining the way a QoS attribute is assigned with a value (including metric type, value, conversion formula, statistical functions); *QoSImpact* expressing how a QoS attribute contributes to the service quality; *Type* specifying the specific QoS category that a QoS attribute belongs to; *Nature* indicating the static or dynamic nature of a QoS attribute; *Aggregated* indicating how a QoS attribute is composed of other attributes; *Node* identifying the network node (service providers or service clients) that may impact on the value of a QoS attribute; *Relationship* representing the way a QoS attribute is correlated with other attributes. Basing on this generic QoS ontology model, the QoS ontology vocabulary is developed for important and common QoS attributes, including performance, scalability, availability, accessibility, accuracy, capacity, cost, configuration, integrity, reliability, security. Some of these QoS attributes may include subclasses for their sub-attributes, e.g. security including confidentiality, auditability, authentication, authorization, data encryption, and non repudiation.

Although providing a quite complete list of important QoS attributes, this ontology language has weak support for defining metrics and value types as well as the usage of QoS model by providers and requesters. It facilitates only for unit conversion but not between QoS metrics.

The **onQoS** ontology [12] is developed for specifying QoS requirements by service consumers and for publishing QoS advertisements by service providers. The ontology is composed of three extensible complementary ontology layers: upper, middle and low.

The upper ontology provides concepts for formulating and for answering QoS information, including the following concepts: *QoSParameter* is for QoS properties; *QoSMetric* is for type of measurement which relates to a *QoSParameter*; *MeasurementProcess* is the process by which numbers or symbols are assigned to QoS parameters according to defined rules; *Scale* defines categories or datasets in which the QoS parameters can be placed based on a particular QoS property; and *ScaleValue* is an assignment of a number or a symbol to a QoS parameter. A *QoSMetric* is also defined as a service parameter that can be included in a service profile in order to connect a QoS definition with service description.

The middle ontology extends and refines concepts in the upper ontology to specify domain-independent QoS parameters, QoS metrics, and QoS scales. *Firstly*, QoS parameters defined in this ontology are similar to those in [11], except that the model includes *WSQoS*, a measurement for the whole QoS of a Web service, which is calculated by a particular measurement process. *Secondly*, *QoSMetrics* are classified according to the used scale (nominal, ordinal, ratio...), to the number of measured QoS parameters, or to the exploited information

in the measurement process. Those categories are Elementary Metric, Derived Metric, Internal Metric, External Metric, Nominal Metric, Ordinal Metric, Numeric Metric, Ratio Metric, and Interval Metric. *Thirdly*, *QoS* *Metric* *Functions* are functions used to aggregate elementary, derived, internal, or external QoS metrics in order to define a derived metric. *QoS* *Metric* *Functions* are divided into *NominalMetricFunction*, *NumericMetricFunction*, and *OrdinalMetricFunction* and they can be *BinaryFunctions* (with two arguments) or *UnaryFunctions* (with one argument).

The low ontology defines concepts, properties, and constraints for QoS information of a specific domain. For example, *QoS* *NetJitter*, *QoS* *NetLatency*, *QoS* *NetRTT* are specific concepts for QoS of services in network domain.

This approach gives a well-defined specification for metrics, value types, metric transformation, as well as description of a set of common QoS properties. However it does not mention about many other characteristics of QoS properties, such as unit, impact direction, dynamism, QoS relationships, etc. The support for the usage of QoS is also very limited, for example, no quality level, QoS priority, QoS mandatory, etc.

The **QoS-MO** [13] is an upper level ontology that contains concepts for defining QoS characteristics, constraints, and levels of Web services described in OWL-S. When a QoS description of a Web service is created, an ontology is created, imports QoS-MO ontology, and then refines its concepts and instantiates necessary individuals. The QoS-MO includes the following classes. *Firstly*, for QoS characteristics, *QoSCharacteristic* is the main class for defining quantifiable characteristics of services which can be general or domain specific; *QoS* *Category* is used to group related QoS characteristics; *QoS* *Dimension* is for modeling the measurement of QoS characteristics; *QoS* *Context* allows the definition of quality expression that combine multiple QoS characteristics; *QoS* *Dimension* *Mapping* provides mechanisms for mapping *QoS* *Dimensions* from one to another; *QoS* *Value* represents the value of a *QoS* *Dimension*. *Secondly*, for QoS constraints, there are three types of QoS constraints that are *QoS* *Offered*, *QoS* *Required*, and *QoS* *Contract*. The *QoS* *Offered* can be defined by both service providers and clients for expressing the service quality that the provider will provide for its clients and the constraints that the client must guarantee when invoking the service. It is similar for the concept *QoS* *Required*. The *QoS* *Contract* connecting *QoS* *Offered* and *QoS* *Required* elements represents the agreed service quality between a service provider and a service requester. *Thirdly*, QoS levels specify different modes of QoS usage that a service can support. A *QoS* *Constraint* is associated with a *QoS* *Level* and it can be changed to other levels by *QoS* *Transitions* which specify essential actions for the transition.

There are several advantages of the QoS-MO in the support of the usage of QoS information. It allows for multiple quality levels that a requester can select according to its demands. There are interdependent requirements of QoS between providers and requests. Therefore, a provider can specify its constraints on a requester's system in order to enable good QoS communication between them when the service is executed. Drawbacks of this approach are weak supports for units, value types, QoS transformation, QoS priority, QoS mandatory, etc.

For the **WSMO-QoS** in [14], the authors used the WSMO [18] model and features to describe a QoS model, specific quality metrics, value attributes, and their corresponding measurements. An upper ontology called WSMO-QoS was developed to provide a means for specifying detailed quality aspects about Web services in the WSMO framework. A new class *QoS* which is defined as a subclass of the *nonFunctionalProperties* class in WSMO can be attached to the class *webService* or the class *Goal* [18]. The *QoS* class includes attributes for specifying details of a QoS property, they are *hasMetricName* (string), *hasValueType* (linguistic, numeric, boolean...), *hasMetricValue* (corresponding value which has value type specified in *hasValueType*), *hasMeasurementUnit* (Unit which includes conversion functions for different measurement units), *hasValueDefinition* (logical expression for computing QoS value), *isDynamic* (boolean), *isOptional* (boolean), *hasTendency* (low/small, high/large, given, for representing the value tendency from the user's perspective), *isGroup* (specifying that the property is defined by a group of other properties or not), *hasWeight* (representing the importance level of the property).

This ontology has no definitions of concrete QoS properties. The proposed ontology also gives a few supports for the usage of QoS information, except for QoS priority, mandatory, and QoS grouping.

In [15], the authors developed **QoSOnt** ontology which is based on existing QoS taxonomies and models. The *QoSOnt* ontology includes several ontologies which are organized into three layers: the base layer, the attribute layer, and the domain-specific layer.

The base layer includes a base QoS ontology consisting generic QoS concepts and unit ontologies, e.g. time ontology. The base QoS ontology represents a minimal set of generic QoS concepts. The central concept is *QoS* *Attribute* which are specialized to *UnmeasurableAttribute* and *MeasurableAttribute*. A measurable attribute may have one or more associated metrics. The *PhysicalQuantity* concept represents the value of a QoS attribute. In case of measurable attribute, the *PhysicalQuantity* may have one or more associated units. The values with different units can be converted to

each other due to the *ConversionRate* concept which specifies the conversion rules between two units.

The attribute layer contains ontologies defining particular QoS attributes and their metrics. Two ontologies which are dependability ontology and performance ontology are specified by defining a set of common QoS attributes introduced in [17].

The domain-specific layer links the lower layers to specific types of systems, e.g. network systems or Web service systems. For Web service system, this layer provides concepts for connecting QoS concepts in lower layers with OWL-S service profiles.

This ontology has similar approach to [10, 12] but does not support for specifying a QoS profile from a set of QoS characteristics and no support for QoS relationships. The conversion mechanism is used for units of QoS metrics, but not for mapping different QoS parameters. The usage support aspect of the ontology is also very limited.

In [16], the authors designed a Web service domain-specific QoS ontology, the **DAML-QoS** ontology, which was developed in order to supplement DAML-S ontology for QoS specification. It includes three layers: the QoS profile layer designed for matchmaking purpose, the QoS property definition layer for elaborating the property's domain and range constraints, and the metric layer for definition and measurement of QoS metrics.

In the QoS profile layer, *QoSProfile* is defined as a common super class for QoS matchmaking concepts including *ProviderQoS*, *InquiryQoS*, and *TemplateQoS*. *ProviderQoS* is the advertisement ontology published by the service provider. *InquiryQoS* is the service requester's inquiry ontology for QoS matchmaking. *TemplateQoS* is for QoS templates that can be reused among users.

In the QoS property layer, QoS property definition contains name, domain, and range of the property. The domains of QoS properties are subclasses of *QoSProfile* which can be *QoS*Core, *QoS*Input, *QoS*Output, *QoS*Precondition, and *QoS*Effect. The *QoS*Core stands for normal QoS properties. *QoS*Input and *QoS*Output are for QoS properties that are applied for inputs and outputs if those QoS properties are different. *QoS*Precondition is for external QoS conditions that need to be satisfied by the service requester in order to achieve the promised QoS levels specified by the service provider. *QoS*Effect is QoS level that results from the execution of the service. The ranges of QoS properties are QoS metric classes which are defined in the QoS metric layer (below).

In the QoS metric layer, the *Metric* class is a common super class of all QoS metrics. Each metric class has properties that indicate its name, value, and unit. There are two types of metrics: atomic metrics that are directly measured and complex metrics that are composed of other atomic and complex metrics. A complex metric has

relationship with the *Function* class representing how to compute the value of that metric.

Although supporting for quality levels and usage roles of providers and requesters, the proposed ontology is quite limited, such as no definition for impact direction, dynamism, QoS priority, and QoS mandatory. Concrete definitions of common QoS properties are also absent.

3. Analysis Summary and Comparison

The comparison of existing QoS ontologies is based on a number of requirements which are basically categorized into two fundamental questions: how to model QoS properties and how to support the usage of QoS properties in order to evaluate and rank Web services. The degree that a QoS ontology meets a criterion is evaluated by *none*, *weak support*, *support*, and *strong support*. The summary of analysis and comparison among reviewed QoS ontologies in section 2 is depicted in Table 1 (for the first question) and Table 2 (for the second question).

How to model QoS properties?

- *Modular and Flexibility*: Allow for extending and adapting QoS ontology for different domains and applications. Some approaches like OWL-Q, onQoS, QoSOnt provide good support for this requirement.
- *Metric*: Characteristics of a metric can be simple or complex, static or dynamic as well as relationships with other metrics like independence or correlation.
- *Unit*: Allow for various types of units, their equivalence and synonyms. Custom unit ontologies should be easily added to an existing QoS ontology.
- *Value Type*: A QoS ontology should include various data type definitions for specifying values of QoS metrics, for example string, numeric, boolean, set, list, etc. Those value types can be categorized as in [12].
- *Impact Direction*: This property is an important factor for evaluating QoS metrics and their values. A QoS property can have one of five impact directions: negative, positive, close, exact, and none.
- *QoS Property Relationships*: Specifying relationships between QoS properties such as independence or correlation (inversion, opposite, parallel) [10].
- *QoS Transformation*: Different participants may use and understand different metrics for the same or similar QoS factors. Therefore, a QoS ontology should support for converting and transforming QoS metrics as well as values and units of related metrics.
- *QoS Dynamism*: A QoS property can be specified once (static property) or requires periodically updating its measurable value (dynamic property).
- *Valid Period*: In fact, the value of a QoS property is not fixed all the time. Thus, we need specify its valid period so that other parties can correctly evaluate it.

How to support the usage of QoS properties?

- *Effect Level*: A service includes several elements such as port types, operations of a port type, parameters of an operation, etc. A QoS property should be attached concretely to a specific element of the service.
- *Quality Level*: Quality levels specify different usage modes so that a requester can choose the most suitable quality levels for its demands.
- *Roles*: Besides providers and requesters, other participants like certificate authorities should also be supported in a process of measurement and evaluation of QoS information.
- *Constraints*: This requirement relates to the question of how to specify a constraint on a QoS property. Almost existing approaches assume simple operators like $>$, $<$, $=$, $>=$, $=<$ for expressing QoS constraints. However other operators related to value types of string, set, list, etc. should be supported.
- *QoS Interdependence*: Not just a service requester requires QoS from a service provider but a service provider can also specify its QoS demands that a requester must guarantee in order to get expected QoS from executing provider service [13].
- *Concrete QoS*: A QoS ontology should contain a minimum set of common and domain-independent QoS properties such as those in [11, 12].
- *QoS Priority*: This requirement is essential for a service requester because QoS properties often have different important levels by different service requesters.
- *QoS Value Comparison*: Not all QoS properties have same mechanism for comparing their values. For example, numeric based QoS properties are compared differently from string based ones.
- *QoS Mandatory*: This requirement allows a requester specifies which QoS properties are strongly required while others may be optional.
- *QoS Grouping*: Allow for grouping QoS properties that share similar characteristics or impact in order to facilitate the evaluation and computing of the whole QoS value of a Web service.

Table 1. Summary analysis with the criteria of *how to model QoS properties*

	Modular / Flexibility	Metric	Unit	Value Type	Impact Direction	QoS Property Relationship	QoS Transformation	Dynamism	Valid Period
OWL-Q	Support	Support	Support	Support	None	Strong Support	Support	Support	Support
QoS Onto Language	Weak Support	Weak Support	Support	Weak Support	Support	Support	Weak Support	Support	None
onQoS	Strong Support	Support	None	Strong Support	None	None	Support	None	None
QoS-MO	Weak Support	Support	Weak Support	Weak Support	Support	Weak Support	Weak Support	None	None
WSMO-QoS	None	Weak Support	Support	Support	Strong Support	None	None	Support	None
QoSOnt	Support	Weak Support	Support	Weak Support	None	None	Weak Support	None	None
DAML-QoS	Weak Support	Support	Support	Weak Support	None	Weak Support	Support	None	None

Table 2. Summary analysis with the criteria of *how to support the usage of QoS properties*

	Effect Level	Quality Level	Roles	Constraints	Interdependence	Concrete QoS	QoS Priority	QoS Value Comparison	QoS Mandatory	QoS Grouping
OWL-Q	Support	None	Weak Support	Weak Support	None	None	None	None	None	Weak Support
QoS Onto Language	Weak Support	None	None	Weak Support	None	Weak Support	None	None	None	Support
onQoS	None	None	None	Support	None	Support	Support	None	None	None
QoS-MO	None	Support	Support	Weak Support	Support	Weak Support	None	None	None	Weak Support
WSMO-QoS	Weak Support	None	None	None	None	Weak Support	Support	None	Weak Support	Support
QoSOnt	None	None	None	None	None	Weak Support	None	None	None	None
DAML-QoS	Weak Support	None	Weak Support	None	Support	None	None	None	None	None

4. Conclusions

QoS plays an important role in Web service selection in order to evaluate and rank candidate Web services that are able to provide expected functionality. Similarly to the problem of Web service description; there may be various QoS models which are adopted by different service providers and service requestors for describing QoS information. It is necessary to match different QoS concepts such as QoS properties, metrics, and units which are specified in a QoS advertisement and a QoS requirement. Semantic technology is a promising method to enhance interoperability between different QoS models and languages. A number of QoS ontologies have been developed recently. This paper reviews those approaches, analyze their advantages and shortcomings as well as indicate open issues that need further investigation in ongoing works aiming at developing semantic QoS models and related algorithms for matching and ranking Web services based on QoS.

References

- [1] S. Frolund and J. Koisten. *QML: A Language for Quality of Service Specification*, Hewlett-Packard, 1998. <http://www.hpl.hp.com/techreports/98/HPL-98-10.html>
- [2] A. Dan et al. *Web Service Level Agreement (WSLA) Specification*, <http://www.research.ibm.com/wsla/>, 2002.
- [3] A. Sahai, A. Durante, and V. Machiraju. *Towards Automated SLA Management for Web Services*, HP, 2002, www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf.
- [4] V. Tomic, B. Pagurek, and K. Patel. *WSOL: Web Service Offerings Language*. In the book of *Web Services, E-Business, and the Semantic Web*, LNCS, Springer-Verlag, pp. 57-67, 2003.
- [5] S. P. Ran. *A Model for Web Service Discovery with QoS*. ACM SIGecom Exchanges, Vol. 4, No. 1, ACM Press, pp. 1-10, 2003.
- [6] M. Matinlassi and E. Niemela. *The Impact of Maintainability on Component-based Software Systems*. In Proc. of the 29th Euromicro Conference (EUROMICRO.03), Turkey, 2003.
- [7] B. Sabata, S. Chatterjee, M. Davis, J. J. Sydir, and T. F. Lawrence. *Taxonomy for QoS Specifications*. In Proc. of the 3rd International Workshop on Object-Oriented Real-Time Dependable Systems, USA, 1997.
- [8] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. In Kluwer Academic Publishing, 2000.
- [9] S. W. Choi, J. S. Her, and S. D. Kim. *QoS Metrics for Evaluating Services from the Perspective of Service Providers*. In Proc. of the IEEE International Conference on e-Business Engineering, ACM, pp. 622-625, 2007.
- [10] K. Kritikos and D. Plexousakis. *Semantic QoS Metric Matching*. In Proc. of the European Conference on Web Services (ECWS2006), IEEE Computer Society, pp. 265-274, 2006.
- [11] I. V. Papaioannou, D. T. Tsesmetzis, I. G. Roussaki, and M. E. Anagnostou. *A QoS Ontology Language for Web Services*. In Proc. of the 20th International Conference on Advanced Information Networking and Applications (AINA2006), IEEE Computer Society, pp. 18-25, 2006.
- [12] E. Giallonardo and E. Zimeo. *More Semantics in QoS Matching*. In Proc. of the IEEE Intl. Conference on Service Oriented Computing and Applications, IEEE Computer Society, pp. 163-171, 2007.
- [13] G. F. Tondello and F. Siqueira. *The QoS-MO Ontology for Semantic QoS Modeling*. In Proc. of the ACM Symposium on Applied Computing, ACM Press, pp. 2336-2340, 2008.
- [14] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. *A QoS-Aware Selection Model for Semantic Web Services*. In Proc. of the 4th International Conference Service Oriented Computing – ICSOC 2006, LNCS, Springer Verlag, Volume 4294, pp. 390-401, 2006.
- [15] G. Dobson, R. Lock, and I. Sommerville. *QoSOnt: a QoS Ontology for Service-Centric Systems*. In Proc. of the 2005 Euromicro SEAA, 2005.
- [16] C. Zhou, L. Chia, and B. Lee. *DAML-QoS Ontology for Web Services*. In Proc. of the International Conference on Web Services (ICWS04), 2004.
- [17] J. C. Laprie, B. Randell, and C. Landwehr. *Basic Concepts and Taxonomy of Dependable and Secure Computing*. In IEEE Transactions on Dependable & Secure Computing. Vol. 1, No. 1, pp. 11-33, 2004.
- [18] D. Roman, H. Lausen, and U. Keller. *Web Service Modeling Ontology (WSMO)*, <http://www.wsmo.org/2004/d2/v1.0/20040920/>, 2004.