Semantic QoS Metric Matching

Kyriakos Kritikos and Dimitris Plexousakis
Foundation for Research and Technology-Hellas (FORTH)
Institute of Computer Science (ICS)
Information Systems Laboratory
P.O. Box 1385, GR-711 10, Heraklion, Crete, Greece
kritikos@ics.forth.gr and dp@ics.forth.gr

Abstract

As the Web Service paradigm gains popularity for its promise to transform the way business is conducted, the number of deployed Web Services grows with a fast rate. While sophisticated semantic discovery mechanisms have been invented to overcome the UDDI's syntactic discovery solution in order to provide more recallable and precise results, the amount of functionally equivalent Web Services returned is still large. The solution to this problem is the description of the QoS non-functional aspect of Web Services. QoS encompasses the performance of Web Services and can be used as a discriminator factor for refining Web Service advertisement result lists. However, most scientific efforts presented so far are purely syntactic and are not capturing all aspects of QoS-based Web Service description leading to imprecise syntactic discovery mechanisms. This paper presents a novel, rich and extensible ontology-based approach for describing QoS of Web Services that complements OWL-S. It is shown that, by using this approach and by introducing the concept of semantic QoS metric matching, QoS-based syntactic matchmaking and selection algorithms are transformed to semantic ones leading to better results.

1. Introduction

Web Services (WSs) are modular, self-describing, loosely-coupled, platform and programming language-agnostic software applications that can be advertised, located and used across the Internet using a set of standards such as *SOAP*, *WSDL* and *UDDI*. They encapsulate application functionality and information resources, and make them available through standard programmatic interfaces. *SOAs* (Service Oriented Architectures) promise to enable the creation of business applications from independently developed and deployed services. A key advantage of SOAs is that they enable services to be dynamically selected and integrated at runtime, thus enabling system flexibility and adaptability – vital autonomic attributes for modern business needs.

However, current techniques only partially address the SOA vision. These techniques rely on static descriptions of service interfaces and other general non-functional service attributes for publishing and finding WSs. This situation creates two problems. Firstly, syntactic discovery efforts return results with low precision and recall. Secondly, no means are provided in order to select among multiple services that appear to perform the same function.

The first problem is solved by combining *Semantic Web* and WS technologies. Ontologies are used, which provide meaning to concepts and relationships between them and thus lead to semantic WS descriptions and discovery algorithms. These enhanced discovery algorithms equipped with Semantic Web technologies provide more precise and recallable results. One recent result of the joint Semantic Web and WS efforts is *OWL-S* [7], which is a W3C member submission.

The second problem can be solved by taking into account a big subset of all possible non-functional properties of WSs, collectively referred to as *QoS* (Quality of Service). QoS does not only encompass network characteristics like bandwidth, latency, or jitter but it is used in a broader end-to-end sense. It encompasses any characteristic of the service host, the service implementation, the intervening network and the client system that may affect the quality of the service delivered. Therefore it has a substantial impact on users' expectations from a service and can be used as a discriminating factor among functionally equivalent WS advertisements. Thus WS descriptions must be enhanced with QoS descriptions. Additionally, WS discovery algorithms should perform QoS-based filtering (matchmaking) and ranking (selection) on WS advertisements in order to produce fewer ranked results.

A *QoS offering* (or demand) of a WS is a set of con-

straints on some QoS metrics. These QoS metrics quantify QoS attributes/characteristics. Actually, current modeling efforts of QoS offers or demands only differ in the expressiveness of these constraints. However, when it comes to QoS attributes/metrics modeling, these efforts fail. The first reason is because the QoS attribute/metric definition is a syntactic one. As a result, QoS metrics like "average availability" may have different meanings to the parties that describe them or use them. Another reason is that the QoS metrics model is not rich enough, not incorporating the definition of measurement and currency units, and measurement methods. This deficiency results in similar QoS metrics that are produced differently or use different units leading to problems in QoS-based WS discovery. The last reason of failure is that the QoS metrics model is not extensible to include newly invented metrics while taking care not to change the underlying computation (matchmaking and selection) model.

Based on these deficiencies in QoS-based WS description, the most prominent QoS-based discovery algorithms fail to perform accurate semantic QoS metric matchmaking and thus produce results with low recall and precision. Therefore, it is clear that there is a need for the semantic QoS-based description and discovery of WSs. This work intends to address this need. First, we propose an upper ontology for QoS-based WS Description, called *OWL-Q*, which extends OWL-S. This ontology describes the possible parts of QoS metrics and constraints. It is an ontological description designed into several facets that can be easily extended and enriched. Based on this upper ontology, we also propose the development of a mid-level ontology that will define all domain independent QoS metrics and will form the basis for the definition of new QoS metrics on other, low-level ontologies. Second, in case where QoS-based advertisements and requests refer to different concepts/metrics of the same or different ontologies, we propose a *semantic QoS metric matching algorithm* that infers the similarity of two different metrics. For semantic QoS-based discovery, we propose to extend syntactic QoS-based matchmaking and selection algorithms by incorporating the aforementioned semantic matching process, leading to semantic QoS-based discovery algorithms.

The rest of this paper is organized as follows. Section "Related Work" reviews the state-of-the-art in QoS-based WS description and discovery. Section "QoS-based Web Service Description" describes the main requirements and principles for QoS-based WS description and unfolds our semantic approach for achieving it. Section "QoS-based Web Service Discovery" presents our approach for semantic discovery of WSs based on QoS. Last, "Discussion and Future Work" summarizes the paper and draws directions for further research.

2. Related Work

The *Web Service Description Language* (*WSDL*) and *UDDI* WS standards are *syntactical* approaches that do not express the QoS aspects of WS Description. While *OWL-S* is a standard *semantic* approach for WS Description, it does not describe QoS offers or demands as it only contains an attribute used for rating a WS.

Tosic, Esfandiari et. al. [9] argue that for the specification of constraints for QoS metrics/attributes, five ontologies must be developed from which the most important (the top one) is the metrics ontology. They describe the structure and involved elements in four out of the five ontologies but they did not develop any ontology. In addition, the requirements specified are incomplete as each of the four aspects of QoS description needs further analysis. Ran [6] proposes an extension to UDDI that represents description of QoS information about a particular WS. However, there is no actual description of the contents of this extension apart from its structure. Moreover, this extension relies on the UDDI (model), so it can be used only for syntactic matchmaking of offers and demands. Maximilien and Singh [4] present an architecture and a conceptual model of WS reputation (QoS) (which encloses a QoS attributes model). However, concepts like QoS constraints and QoS offers and demands are not modeled. Furthermore, the QoS metrics model is not rich enough. Tosic, Pagurek et. al. [10] present the *Web Service Offerings Language* (*WSOL*) and propose that a WS must offer different classes of service in order to satisfy a greater amount and type of customers and in order to deal successfully with situations where there is a variation in QoS due to network problems or mobility reasons. Their work comes with the following shortcomings: (a) no separation and integration of constraint dimensions; (b) no specification of a QoS demand; (c) metrics ontologies not developed. *WSLA* (*Web Service Level Agreement*) [2] is a XML language used for the specification of Service Level Agreements (SLAs). It represents a purely syntactic approach that can not be used during the WS Discovery process and that is not accompanied by a complete framework. Tian et. al. [8] analyze what must be enclosed into the QoS information for a WS request or advertisement with the help of a QoS ontology. However, not only there is no complete and accurate description of QoS constraints, but also metrics ontologies are only referenced. Oldham et. al. [5] offer a semantic framework for the definition and matching of *WS-Agreements*. However, within this framework, unary QoS metric constraints can only be expressed while QoS metric matching is enforced by manual incorporation of rules.

Zhou et. al. [12] extend the DAML-S WS description language so as to include a QoS specification ontology. In addition, they propose a novel QoS matchmaking algorithm, which is based on the concept of *QoS profile com-*

patibility. The deficiencies of this research effort are the following: (a) The metrics model is not rich enough; (b) QoS metrics have the set $\mathbb{N}^+$ as their range; (c) DL reasoners are slow and do not support the most complex mathematical expressions.

Martín-Díaz et. al. [3] use a symmetric QoS model expressing mathematical constraints for QoS metrics. However, *semantics is missing* leading to syntactic matchmaking and selection algorithms. Before matchmaking, a QoS specification is transformed to a *Constraint Satisfaction Problem* (*CSP*) [11] which is checked for *consistency*. Matchmaking is performed according to the concept of *conformance* (if every solution of offer is a solution of demand). Concerning WS Selection, the (QoS) score of a WS advertisement is expressed as a *Constraint Satisfaction Optimization Problem* (*CSOP*), where from all solutions of the CSP of an offer we try to find the one that minimizes the weighted sum of the weight of each metric multiplied with its utility assessment value. Unfortunately, CS(O)Ps can have non-polynomial solutions when there are non-linear expressions at QoS constraints.

3. QoS-based Web Service Description

3.1. Requirements for QoS-based Web Service Description

After reviewing related work in QoS-based WS Description, we have come up with the following requirements that must be satisfied by a QoS-based WS description language:

- *Devise an extensible and formal semantic QoS model* in order to: a) have terms with exact meaning in QoS-based WS specifications; b) exploit formal methods to be used in conformance checking of QoS-based WS specifications and in QoS-based WS discovery algorithms.
- *Comply with standards* in order to: a) use widely available tools; b) be widely acceptable and become also a standard.
- *Support the syntactical separation of QoS-based and functional parts of service specification* for: a) specification of different QoS offerings for different implementations of the same interface; b) deactivation, reactivation, dynamic creation or deletion of service offerings independently of the interface specification; c) reuse of QoS offers by different services.
- *Support refinement of QoS specifications and their constructs*.
- *Allow both provider and requester QoS specification*: providers and requesters should be given the capability to specify their QoS constraints/requirements with the same expressiveness and in a symmetric way.
- *Allow fine-grained QoS specification* i.e., QoS specs for the whole WS and its parts.
- *Devise an extensible and formal QoS metrics model*, which must at least specify: a) the value set of the metric; b) the metric's domain of knowledge; c) the relationship of the metric with other metrics; d) the association of the metric with a unit, a measured QoS property and a measurement function/directive; e) a (functional or ontological) description of the derivation of a complex service's metric value from the corresponding QoS metric's values of the component services.
- *Devise a corresponding extensible and formal QoS attributes, units, functions and measurement directives model*.
- *Allow classes of service specification*: Class of Service means the discrete variation of the complete service and QoS provided by one WS. The benefits of classes of service specification are the following: a) for providers: 1) different QoS provision for different classes of consumers; 2) achieve maximal gain with optimal utilization of resources; b) for consumers: can better select the service and QoS they need and are willing to pay for, while minimizing their price/performance ratio.

3.2. OWL-Q

Based on the requirements of QoS-based WS Description we have set in the previous subsection, we have developed an OWL-S extension (the requirement *syntactical separation* is satisfied as our ontology can be developed independently from OWL-S) for QoS-based WS description of both requests and offers. We have extended OWL-S ontological description for two reasons: to comply with Semantic WS description standards (*standards compliance*) and to use the OWL ontology formalism (*extensible and formal semantic QoS model*). OWL is one of the most expressive ontology languages and it is a W3C standard.

Our ontology is separated into several facets. Each facet can be developed and extended independently of the other (*syntactical separation and refinement of QoS specifications*). Each facet concentrates on a particular part of our QoS WS description. A document describing a QoS WS advertisement or request should reference all the facets of our ontology. In the sequel, we present and analyze all parts of the OWL-Q ontology.

Connecting Facet As can be seen in Fig. 1, the Connecting Facet connects OWL-S with OWL-Q and provides the high-level concepts that are appropriate for defining QoS

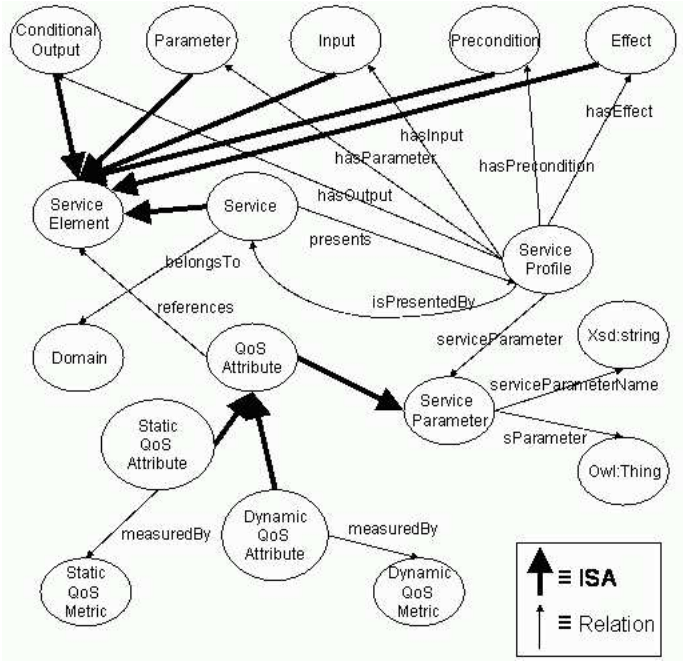

Figure 1. Connecting Facet of OWL-Q.

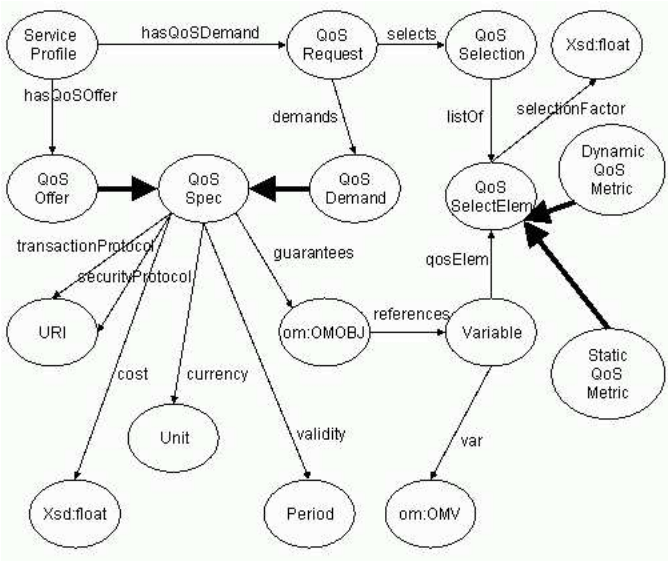

Figure 2. Basic Facet of OWL-Q.

advertisements and demands. For the connection of the two ontological descriptions, the *QoSAttribute* class is a subclass of OWL-S *ServiceParameter* and references a *ServiceElement*. Subclasses of the latter class are *ConditionalOutput*, *Parameter*, *Input*, *Precondition*, *Effect*, and *Service*. That is a *QoSAttribute* can reference any *ServiceElement* of a service's functional description (*fine-grained QoS specification*). Finally, a *QoSAttribute* can be static or dynamic (it changes with time) and is measured by one or more static or dynamic *QoSMetric*s respectively.

Basic Facet A *ServiceProfile* is associated with many *QoSOffers* (*classes of service*) or with only one *QoSRequest* (*both provider and requester QoS specification*). A *QoSRequest* is separated into a *QoSDemand* class and a *QoSSelection* class. The latter class is the actual incarnation of a list of <*QoSMetric*, *selectionFactor*> elements useful for the WS selection process. The *QoSSpec* class represents the actual QoS description of a WS. It describes the security and transaction protocols used, the cost of using the service and the associated currency for the cost, the validity period of the offer or demand and an arbitrary OpenMath expression (*om:OMOBJ*). This expression represents what is or must be guaranteed and contains variables which are associated to QoS Metrics.

QoS Metric Facet The Metric Facet describes all the appropriate classes and properties used for a proper formal definition of a QoS metric (*QoS metric model*). This metric facet is actually an upper ontology representing any abstract QoS metric. A specific QoS metric can be created

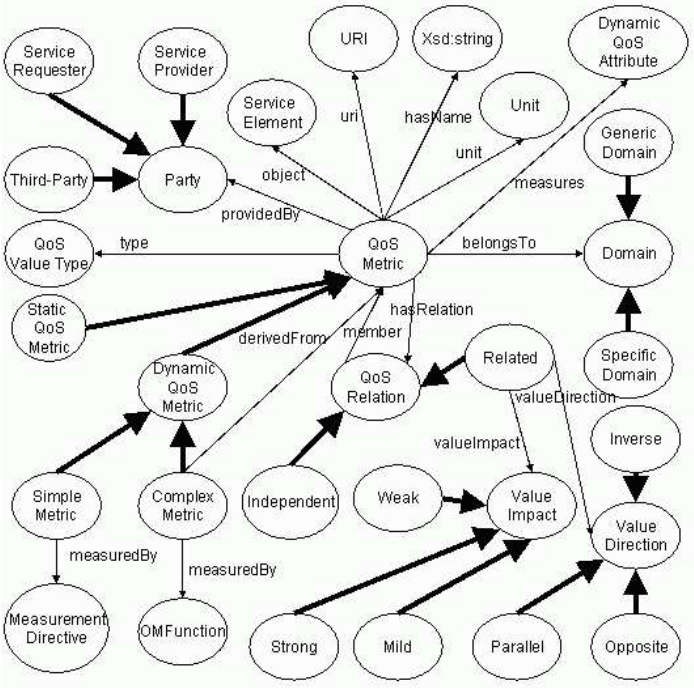

Figure 3. Metric Facet of OWL-Q.

by refining the *QoSMetric* class. Many specific QoS metrics (especially the general ones) can be part of a midlevel ontology created for QoS metric reuse. We prefer specialization to instantiation because it allows for a quicker reasoning process. We plan to develop a mid-level ontology defining cross-domain QoS metrics and a low-level ontology for defining QoS metrics for particular domains.

The *QoSMetric* is one of the most important classes of OWL-Q representing a QoS metric. The values of a QoS metric are provided by a service provider or a requester or a third-party. A QoS metric belongs to a *Domain* of knowledge, which is separated into one *Generic Domain* and several *Specific Domain*s. It has only one name. It measures a *QoSProperty* on a specific *ServiceElement*. The value type of a *QoSMetric* is an instance of the *QoSValueType* class (analyzed in a separate facet) while the unit of the value is an instance of the *Unit* class. A *QoSMetric* is separated into static and dynamic metrics. A *StaticQoSMetric* is computed only once to produce a value for a *StaticQoSAttribute*. A *DynamicQoSMetric* is computed repeatedly according to a *Schedule* to produce values of a *DynamicQoSAttribute* that change over time. It can be a simple QoS metric *measuredBy* a *MeasurementDirective* or a complex one. *ComplexMetric*s are derived from other metrics with the help of a *OMFunction* (analyzed later). Last but not least a *QoSMetric* is related to other metrics according to two types of *Relationship*s: *Independent* and *Related*. When two metrics are related, we can specify the direction of their values or the impact of one's value to the other's value.

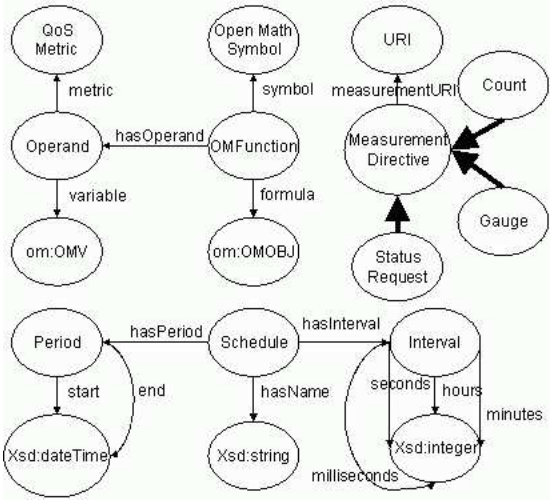

Figure 4. Function, Measurement Directive and Schedule Facets of OWL-Q.

Function, Measurement Directive and Schedule Facets The Function Facet describes all the appropriate concepts and properties for the proper definition of metric functions. The *OMFunction* class is the basic concept that represents a QoS Metric Function. A Metric Function is either expressed with an arbitrary OpenMath Formula (expressed in XML via the OpenMath XML Schema om:) or with a known OpenMath function (subclass of the *OpenMathSymbol* class of the MONET [1] Ontology). A Metric Function takes as input objects of the *Operand* class, which associate a *QoSMetric* with an OpenMath variable (*om:OMV*) that is used inside the function's formula.

The Measurement Directive Facet describes the concept of measurement directive which is used for the measurement of simple metrics. A *MeasurementDirective* is composed of a *URI* that describes where and how to get a value of a resource's property. A *Schedule* is used to compute the frequency of the computation of a Complex Metric's value. It has a specific name and is defined either by a starting and ending *Period* of *xsd:dateTime* type or by a Time *Interval* that is expressed in specific time units.

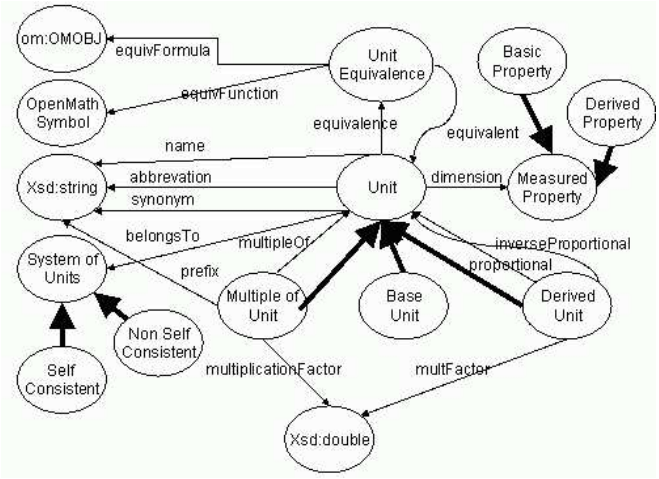

Figure 5. Unit Facet of OWL-Q.

Unit Facet The Unit Facet formally describes the unit of a QoS metric. A *Unit* has one name, several abbreviations and synonyms (even in different languages). A *Unit* belongs to a *System of Units* and is associated with the same *QoSProperty* as the one that is measured by the QoS metric of the unit. A *Unit* is separated into *BaseUnit*s, *MultiplesOfUnit*s and *DerivedUnit*s. The *BaseUnit* class expresses units which are used most of the times in measurements. A *MultipleOfUnit* is associated with a *BaseUnit* and converted to it by a constant (*multiplicationFactor*). It has a name composed of the name of its *BaseUnit* and a prefix. A *DerivedUnit* is proportional to some *Unit*s and inverse proportional to other *Unit*s. It also has a *multFactor* that is used to express its mathematical definition in relation to the other (inverse) proportional units. An unit is equivalent to another unit and can be converted to it with the help of the *Equivalence* class. This class correlates the equivalent metrics and defines the OpenMath functions or formulas that are used to convert the values of the one unit to the other. A midlevel ontology of units must be developed in order to have a semantic description of specific base units and their alternatives or multiples. This midlevel ontology will be very helpful in converting between values of metrics where these metrics are equivalent but use different units.

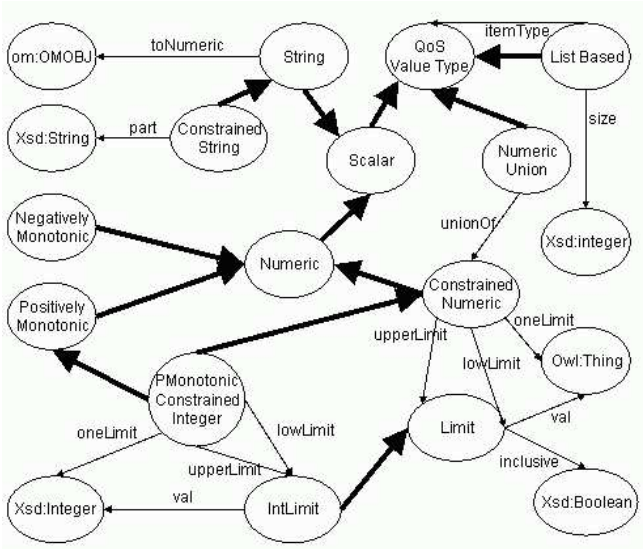

Figure 6. QoS Value Type Facet of OWL-Q.

QoSValueType Facet The *QoSValueType* ontology describes the types of values a QoS metric can take. The *QoSValueType*s can be *Scalar* or *NumericUnion*, or *List-based* types. *Scalar* value types are simple value types that can be *Numeric* or *String*. *NumericType*s can be *Positively-Monotonic* or *NegativelyMonotonic* in order to show the direction of values of the associated metric. *ConstrainedNumeric* value types represent *Numeric* value types that have (upper, low or one) limits (e.g. the Integers set [2,5] or the Integer value $\{2\}$). A *String* value type may have a *toNumeric* OpenMath formula that will be used to compute the numeric counterpart of its values. A *ConstrainedString* represents a finite closed String list which has specific String values as *part*s. The *NumericUnion* class represents value types that are expressed as unions of *Numeric* value types (e.g. $[1, 2] \cup \{4\} \cup [9, 11]$). The *List-Based* class represents list value types that have a specific size and whose elements are of a specific *QoSValueType*.

4. QoS-based Web Service Discovery

4.1. Semantic QoS Metric Matching

All current QoS-based WS discovery efforts try to infer if an advertisement constraint involving a particular QoS metric is stricter than a request constraint involving the same QoS metric. However, equivalence of QoS metrics is inferred by the syntactic comparison of their names. As a result, the QoS-based WS matchmaking process fails as it produces results with low precision and recall.

The solution to the above problem is to semantically define and compare two QoS metrics so as to infer that they are the same. For the semantic comparison of two QoS metrics, we have devised a *semantic QoS metric matching algorithm*. This algorithm makes the following assumptions:

- Both the requester and service provider use the OWL-Q semantic language for describing QoS metrics. However, they may reference different QoS metrics from different OWL-Q ontology instances.
- The description of a QoS metric references the unit of measurement, the QoS property that is measured, the type of values that this metric takes as a result of the measurement, and the service object that is measured. In addition, the metric has a direction of values (positive, negative) which indicates if a greater value reflects greater quality and is derived from the value type of the metric.
- QoS metrics are classified into two disjoint categories: Resource metrics and Composite metrics. Resource metrics are retrieved directly from the managed resources residing in the service provider's tier, such as routers, servers, implemented applications. Typical examples of resource metrics are counters and gauges. For every resource metric, a Measurement Directive is specified, which contains the command and other context information needed to retrieve the metric from the managed resource instrumentation. Composite metrics are created by combining several other (composite or resource) metrics according to a specific algorithm, such as averaging one or more metrics over a specific amount of time, or by breaking them down to specific criteria (top 10%, minimum, maximum values of a time series). This is usually done within the service provider's domain but can be outsourced to a third-party measurement service as well. We assume that Composite metrics are specified by means of a Function (a formula describing the input metrics and the arithmetic operations to aggregate them). We further assume that this Function is specified in the OpenMath XML language. OpenMath is an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. Translations (called phrase-books) to the internal (math object) representations of popular mathematical engines like Mathematica and Maple have already been developed for computing the result of OpenMath expressions. In addition, an OWL ontology for the semantic representation of OpenMath objects has been developed in the context of the MONET [1] project. This ontology is used for representing common functions which take only one argument as input and are used for unit transformations or composite metric value computations.

- The requester and service provider use three common ontologies: unit, QoS property and QoS ValueType. The unit ontology semantically describes the unit used for the measurement prescribed by a QoS metric. The QoS property is an ontology for the semantic description of domain-independent (like throughput, availability and response time) and domain-dependent (e.g. "flexibility of reservation changes" in the travel domain) QoS properties. Finally, the third ontology is used for the semantic description of simple (integer, real, string) and complex QoS types (real number in [0,1], binary integer, list, time series). It is easy to come up and agree with the content of these three ontologies as they usually contain terms/concepts with an easily understood meaning for people/users. Thus they will be common and available to every requester and service provider. If this is not the case, then matching algorithms for units, QoS properties and QoS ValueTypes must be developed and used.

The input to the described algorithm is the following:

- Three ontologies O_1, O_2, O_3. O_1 contains U concepts, each representing a unit. O_2 contains P concepts, each representing a QoS Property. O_3 contains V concepts, each representing a QoS Value Type.
- An ontology instance of a subpart of the MONET ontology describing functions which are subclasses of the $OpenMathSymbol$ concept and the OpenMath's XML Schema om: for describing arbitrary OpenMath formulas.
- Two different ontological instances D_1 and D_2 of OWL-Q which include the description of two different QoS Metrics M_1 and M_2 respectively. In general, if we have a QoS Metric concept M described by an ontology instance D, then:
  - $M.unit \in U$, where "unit" is a relation associating metrics and units,
  - $M.measures \in P$, where "measures" is a relation associating metrics and QoS Properties,
  - $M.type \in V$, where "type" is a relation associating metric and value types,
  - $M.object \in ServiceElement = \{Service, Operation, Activity, Flow, Endpoint, Port, Parameter, Input, ConditionalOutput, \ldots\}$, where "object" is a relation associating metrics and service elements,
  - If $M.type$ instanceOf $Number \wedge PositivelyMonotonic \subseteq V$ then valueDirection(M) = "positive" else if $M.type$ instanceOf $Number \wedge NegativelyMonotonic \subseteq V$ then valueDirection(M) = "negative" else $M.toNumeric$ instanceOf $om : OMOBJ$. It is stated that if the metric takes numeric values, then it has a positive or negative value direction. Otherwise, a function must be defined that computes the numeric counterpart of every value of the metric.
  - If M subclassOf $CompositeMetric$ then $M.measuredBy$ subclassOf $OMFunction$ and $M.derivedFrom$ subclassOf $Metric$. It is stated that if a metric is composite and not resource, then its values are computed by a function and is derived from other metrics.

The algorithm considers three different cases depending on the nature of the two compared metrics M_1 and M_2 (i.e. if they are Resource metrics or Composite):

[**First Case**] If M_1 subclassOf $ResourceMetric$ and M_2 subclassOf $ResourceMetric$, then

$$match\,(M_1, M_2) \equiv t \wedge M_1.object = M_2.object \wedge \\ \wedge M_1.measures = M_2.measures$$

$t = uvmatch\,(M_1.unit, M_2.unit, M_1.type, M_2.type)$. The matching of units and types is not going to be analyzed due to space limitation of this paper. We just note that when the same units are used, the types must be the same. Otherwise, if the units are compatible (one is multiple or equivalent to the other), then the types must be compatible (i.e. if they are constrained, then the transformation of the one type's limits must lead to values equal to the limits of the second). We assume that Resource metrics have always Numeric values, so there is no meaning to include a comparison of the "toNumeric" math objects. To provide a simple match example, consider two Resource metrics: "DownTimeMetric" and "UpTimeMetric". Assume that they both try to measure the "Availability" QoS property of a service, they take values from the Integers set and they use the same unit (of time). However, the "DownTimeMetric" has negative value direction while the "UpTimeMetric" has positive value direction. Thus, they are not matched.

[**Second Case**] If M_1 subclassOf $ResourceMetric$ and M_2 subclassOf $CompositeMetric$, then

$$rcmmatch\,(M_1, M_2) \equiv match\,(M_1, M_2) \wedge \\ \wedge M_2.derivedFrom \cap CompositeMetric = \oslash \\ \wedge M_2.derivedFrom! = M1$$

In other words, a Resource and a Composite metric can be matched if the Composite metric is only derived from Resource metrics that are different from the compared Resource metric and the conditions for resource-to-resource

metric match are met. This is the case where we have on one hand a Resource metric that is obtained in the form of a high-level reading from systems with advanced instrumentation and on the other hand the same but Composite metric that is derived from a Resource metric obtained in the form of low-level reading from systems with basic instrumentation. For example, rather than deriving DownTime as a Composite metric of time series of a system's Status (i.e. Resource metric), some systems offer this reading directly. To formally demonstrate the above example, the following hold:

$$M_1.unit = \text{“sec”} \wedge M_1.measures = Availability$$
$$\wedge\, M_1.type = NMonotInteger \wedge M_1.object = Service$$
$$\wedge\, M_2.unit = \text{“sec”} \wedge M_2.measures = Availability$$
$$\wedge\, M_2.type = NMonotInteger \wedge M_2.object = Service$$
$$\wedge\, M_2.derivedFrom = M_3 \wedge M_3.type = Percentage$$

It is easy to derive that $!rcmmatch(M_1, M_3)$ and $rcmmatch(M_1, M_2)$.

[**Third Case**] If M_1 subclassOf $CompositeMetric$ and M_2 subclassOf $CompositeMetric$, then we have the hardest case of all. Let us assume that: $M_1.derivedFrom = \{R_1, R_2, \ldots, R_n, C_1, C_2, \ldots, C_m\}$, where R_i subclassOf $ResourceMetric\ \forall i \in \{1, \ldots, n\}$ and C_j subclassOf $CompositeMetric\ \forall j \in \{1, \ldots, m\}$. Similarly, we assume that: $M_2.derivedFrom = \{R'_1, R'_2, \ldots, R'_{n'}, C'_1, C'_2, \ldots, C'_{m'}\}$, where R'_i subclassOf $ResourceMetric\ \forall i \in \{1, \ldots, n'\}$ and C'_j subclassOf $CompositeMetric\ \forall j \in \{1, \ldots, m'\}$. Initially, the algorithm assumes that both of the metrics are Resource and executes a resource-to-resource metric match. If this match fails, then we have a failed match. Otherwise, the algorithm proceeds in order to match every Resource metric R_i of M_1 with every resource metric R'_i of M_2. If there is a match, then it substitutes every occurrence of these matched metrics in the $M_1.measuredBy$ and $M_2.measuredBy$ mathematical expressions with a new variable $R_{i,j}$, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n'\}$. Similarly, the algorithm attempts to match every Composite Metric C_j of M_1 with every Composite Metric C'_j of M_2. If there is a match, then it substitutes every occurrence of these matched metrics in the $M_1.measuredBy$ and $M_2.measuredBy$ mathematical expressions with a new variable $C_{i,j}$, where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, m'\}$. If there are unmatched resource or composite metrics of M_1 and composite or resource metrics of M_2 respectively, then the algorithm attempts to match them and assign them a new variable name in the corresponding mathematical expressions of M_1 and M_2. After all of this groundwork, we have a match if the following hold:

$$ccmmatch(M_1, M_2) \equiv match(M_1, M_2) \wedge$$
$$\wedge\, equivalent(M_1.measuredBy, M_2.measuredBy)$$

In other words, the two Composite metrics are matched if the conditions for resource-to-resource metric match are met and they have equivalent functions for computing the derivation of the metrics' values. Two functions (i.e. mathematical expressions) are equivalent if for every instance of their variables, they take the same value. Equivalency of mathematical expressions is generally undecidable but the most powerful mathematical engines like Mathematica and Maple successfully deal with many hard cases.

Let us now examine a complete example of two QoS metrics that try to measure the QoS Property of Availability. For the first QoS metric $A1$, the following hold:

$$A1.unit = \oslash \wedge A1.type = PMonotonicPercentage$$
$$\wedge\, A1.measures = Availability \wedge A1.object = Service$$
$$\wedge\, A1.derivedFrom = \{D1, U1\}$$
$$\wedge\, A1.computedBy = \text{“}1 - D1/(U1 + D1)\text{”}$$
$$\wedge\, U1.unit = \text{“sec”} \wedge U1.type = PMonotInteger$$
$$\wedge\, U1.object = Service \wedge U1.measures = Availability$$
$$\wedge\, D1.unit = \text{“sec”} \wedge D1.type = NMonotInteger$$
$$\wedge\, D1.object = Service \wedge D1.measures = Availability$$

For the second QoS metric $A2$, the following hold:

$$A2.unit = \oslash \wedge A2.type = PMonotonicPercentage$$
$$\wedge\, A2.measures = Availability \wedge A2.object = Service$$
$$\wedge\, A2.derivedFrom = \{D2, U2\}$$
$$\wedge\, A2.computedBy = \text{“}U2/(U2 + D2)\text{”}$$
$$\wedge\, U2.unit = \text{“sec”} \wedge U2.type = PMonotInteger$$
$$\wedge\, U2.object = Service \wedge U2.measures = Availability$$
$$\wedge\, U2.derivedFrom = S2 \wedge D2.derivedFrom = S2$$
$$\wedge\, D2.unit = \text{“sec”} \wedge D2.type = NMonotInteger$$
$$\wedge\, D2.object = Service \wedge D2.measures = Availability$$
$$\wedge\, S2.unit = \oslash \wedge S2.type = \{0, 1\}$$
$$\wedge\, S2.object = Service \wedge S2.measures = Availability$$

Initially, the algorithm will try to match the two metrics, regarding them both as Resource. They match, so it continues. Then, it attempts to match resource-to-resource and composite-to-composite ancestral metrics. There a no such pairs, so it then tries to match resource-to-composite metrics. It infers that: $rcmmatch(D1, D2)$ and $rcmmatch(U1, U2)$. So it replaces $D1$ and $D2$ with D, $U1$ and $U2$ with U in the mathematical expressions of both $A1$ and $A2$. In other words, it now holds that: $A1.measuredBy = \text{“}1 - D/(U + D)\text{”}$ and $A2.measuredBy = \text{“}U/(U + D)\text{”}$. Next, the algorithm attempts to infer the equivalence of the "measuredBy" expressions of the two metrics. Indeed, it is obvious that: $equivalent(\text{“}1 - D/(U + D)\text{”}, \text{“}U/(U + D)\text{”})$, so the algorithm finally infers that: $ccmmatch(A1, A2)$.

4.2. Extending QoS-based Web Service Matchmaking Algorithm

One of the most prominent QoS-based WS discovery algorithm [3] expresses each QoS-based WS description as a CSP. Then it separates the QoS-based advertisements into two categories: the ones that satisfy completely the QoS-based request and the others that do not satisfy the request. However, this algorithm presents two major drawbacks:

1. It performs syntactic metric matchmaking producing false negative and false positive results.
2. The algorithm does not treat two cases at all: a) when the advertisement uses QoS metrics that are not used by the request; b) the opposite.

In the sequel, we present a QoS-based WS matchmaking algorithm that exploits the OWL-Q ontology model and the "semantic QoS metric matching" algorithm in order to extend the aforementioned algorithm. This algorithm takes as input the QoS offers of all the WS advertisements and the QoS demand of the request and returns four lists of WS advertisements. It is composed of three main sequential steps:

1. OWL-Q advertisements and OWL-Q request are transformed to CSP problems via XSLT as OWL-Q description files are expressed in XML. However, this transformation must be performed carefully according to the following two directives:
   (a) Only metrics which are semantically equal should correspond to the same CSP variable. That is by using the requester's QoS description, we check every provider's QoS description as follows: for each QoS metric X of the requester, we try to find a provider's QoS metric Y that is semantically equivalent. If Y is eventually found, then Y is set to X i.e. the two metrics are assigned to the same CSP variable.
   (b) If two equal metrics do not use the same units, then we consider the request's metric unit as the default and a unit transformation procedure is performed as follows: If X is the metrics CSP variable and f(x) = x/100 (for example) is the unit transformation function (i.e. a formula), then for every appearance of metric X at the CSP of the provider we set X to X/100.
2. We solve all advertisement CSPs and the CSP of the request with a known and efficient CSP engine.
3. We constrain the solution space of the advertisements and the request only to the metrics that are common. Then, for every solution of the CSP of an advertisement, we check if it is contained in the solution space of the CSP of the request. If this is not the case, the advertisement is considered as a *failed match*. Otherwise, if the advertisement has more variables or fewer solutions than the request it is considered as a *super match*. Otherwise, if it has the same set of variables and solutions, it is considered as an *exact match*. Otherwise, if it has the same or smaller set of solutions and fewer variables, it is considered as a *partial match*.

The algorithm produces four types of results: *super matches*, *exact matches*, *partial matches* and *failed matches* with decreasing order of significance.

4.3. Extending QoS-based Web Service Selection Algorithm

In some cases, there will be several QoS offers which will be returned from a QoS-based WS matchmaking algorithm. Therefore, a QoS-based selection mechanism is required to discover the best-fit WS with respect to its QoS offer. Based on the OWL-Q ontology, a QoS-based WS selection algorithm performs the following steps:

1. Step 1 is the same as the one at the previously analyzed QoS-based matchmaking algorithm. However, we now try to match every QoS metric appearing at the "QoS Selection" section of the demand with every QoS metric of an offer.
2. Based on the CSP c_ω of an offer ω, we compute the minimum and maximum utility assessment of the offer as follows:
   (a) The minimum utility assessment U_{min_ω} of an offer ω is derived from the following formula:

$$U_{min_\omega} = \min_{subject\_to\_c_\omega} \sum_{\substack{m \in D_{metrics} \\ \cap \omega_{metrics}}} U(m) \cdot W_D(m)$$

   , where m is a metric that in not only contained in the metric list $D_{metrics}$ of the "QoS Selection" part of the demand D but is also contained in the metric list $\omega_{metrics}$ of the offer ω, $U(m) = \frac{m - m_{min}}{m_{max} - m_{min}}$ is a utility function that assigns a unique assessment value to every value a metric m can take and takes values from the set [0, 1] of real numbers, m_{max} and m_{min} are respectively the maximum and minimum values a metric m can take by definition, and $W_D(m)$ is the weight of the metric m presented at the demand D. What is done here is that the CSP c_ω of an offer ω is is transformed to a CSOP, which consists of a standard CSP and of an optimization function that maps every solution to a numerical value. ¿From all the solutions of the CSP c_ω, we

choose the one that minimizes the optimization function, i.e. the formula of the minimum utility assessment U_{min_ω}.

(b) The maximum utility assessment U_{max_ω} of an offer ω based on a demand D is given by the following formula:

$$U_{max_\omega} = \max_{subject\_to\_c_\omega} \sum_{\substack{m \in D_{metrics} \\ \cap \omega_{metrics}}} U(m) \cdot W_D(m)$$

3. The overall utility assessment U_ω of an offer ω based on a demand D is given by the following formula: $U_\omega = a \cdot U_{min_\omega} + b \cdot U_{max_\omega}$, where $0 \leq a, b < 1$ and $a + b = 1$. This formula states that the overall utility assessment (score) of an offer depends on percentage a of the minimum utility assessment of the offer and on percentage b of the maximum utility assessment of the offer. Although most QoS-based WS selection algorithms are interested in the worst score that can be given by an offer, we believe that this is not enough; = we have to pick among the same "worst-score" offers the one that gives the best of the best scores.

This algorithm takes advantage of the "metric matching" concept previously introduced and extends the QoS-based WS selection algorithm presented in [3] by considering the maximum utility assessment of an offer.

5. Conclusion

This work addresses the need for semantic QoS-based WS description and discovery by proposing an appropriate semantic framework. For QoS-based WS description, a semantically rich, formal and extensible QoS model (as an extension of OWL-S) is proposed that is based on ontologies and captures every aspect of QoS description. This ontology-based QoS model is designed into several facets that can be easily extended and enriched.

For QoS-based WS discovery, a semantic QoS metric matching algorithm has been developed that can be used to enhance QoS-based WS matchmaking and selection algorithms in order to produce results with better precision and recall. Besides this extension, some flaws of these algorithms have been addressed. What remains to be done is the empirical evaluation of such extended algorithms.

As future work, we propose three extensions. The first is the separation of QoS constraints into hard and soft in cases where the QoS-based WS discovery process only produces *failed match* results. In this way, the results are first filtered by the hard (compulsory) constraints and then ordered by the number of soft (optional) constraints they satisfy. The second is the description of the context of both the WS and the WS requester. We believe that a Context-aware WS discovery process will be more accurate and customizable as the tasks of request and input completion, output adaptation and added-value composition of service-offerings become possible. The last extension is the implementation of several tools that will assist providers and requesters in providing less input and in fulfilling their goals.

References

[1] O. Caprotti, M. Dewar, and D. Turi. Mathematical service matching using description logic and owl. In A. Asperti, G. Bancerek, and A. Trybulec, editors, *Mathematical Knowledge Management (MKM)*, volume 3119 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2004.

[2] A. Keller and H. Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. Technical Report RC22456 (W0205-171), IBM, 2002.

[3] O. Martín-Díaz, A. R. Cortés, D. Benavides, A. Durán, and M. Toro. A quality-aware approach to web services procurement. In B. Benatallah and M.-C. Shan, editors, *Technologies for E-Services (TES)*, volume 2819 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2003.

[4] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[5] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic ws-agreement partner selection. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2006. ACM Press.

[6] S. Ran. A model for web services discovery with qos. *SIGecom Exch.*, 4(1):1–10, 2003.

[7] K. Sycara et al. *OWL-S 1.0 Release*. OWL-S Coalition, http://www.daml.org/services/owl-s/1.0/, 2003.

[8] M. Tian, A. Gramm, M. Nabulsi, H. Ritter, J. Schiller, and T. Voigt. Qos integration in web services. Gesellschaft fur Informatik DWS 2003, Doktorandenworkshop Technologien und Anwendungen von XML (Ph.D. students workshop Technologies and Applications of XML), October 2003.

[9] V. Tosic, B. Esfandiari, B. Pagurek, and K. Patel. On requirements for ontologies in management of web services. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 237–247, London, UK, 2002. Springer-Verlag.

[10] V. Tosic, B. Pagurek, and K. Patel. Wsol - a language for the formal specification of classes of service for web services. In L.-J. Zhang, editor, *ICWS*, pages 375–381. CSREA Press, 2003.

[11] P. Van Hentenryck and V. Saraswat. Strategic directions in constraint programming. *ACM Computing Surveys*, 28(4):701–726, 1996.

[12] C. Zhou, L.-T. Chia, and B.-S. Lee. Daml-qos ontology for web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 472, Washington, DC, USA, 2004. IEEE Computer Society.