More Semantics in QoS Matching

Ester Giallonardo
University of Sannio
Department of Engineering
Benevento, 82100 Italy
ester.giallonardo@unisannio.it

Eugenio Zimeo
University of Sannio
Research Centre on Software Technology
Benevento, 82100 Italy
eugenio.zimeo@unisannio.it

Abstract

The evolution of the Web towards a global computing environment is promoting new research efforts aimed at the formal characterization of Web Services QoS. Reasoning on QoS is a key to improve matching process during the discovery of desired services and a step towards the transformation of applications in collections of loosely coupled services virtually connected by semantic similarities. The paper presents the onQoS ontology, an openly available OWL ontology for QoS, and evaluates it in a QoS-aware matching environment. The ontology can be used to express functions of QoS metrics useful to improve the recall tied to the matching of a template request with target Web Services. To this end, the ontology introduces the concept of derivation in the matching process. This gives the possibility of matching a QoS template with published Web Services by deriving different QoS parameters when a one-to-one matching fails. The proposed matching algorithm utilizes a reasoner that exploits the ontology to avoid apparent mismatches. An experimental evaluation shows that exploiting QoS knowledge significantly improves matching recall without deteriorating precision.

1. Introduction

In competitive marketplaces, consumers chose products, services, or resources based on a balance of functional proprieties, quality and cost. With the growing popularity of Web Services, and the advances in global computing research, the same situation progressively appears in the electronic market, where there will be a variety of functionally equivalent Web Services that differ for Quality of Service (QoS) proprieties. Our experience, matured in the context of a project that aims at using SOA and Web Services technology for managing automotive supply chains [1], confirms the importance of QoS in the discovery process. In this scenario, it is unlikely that anyone chooses a supplier (a Web Service) without considering response time, availability, cost, security, standards compliance or quality of the products delivered, such as output correctness compared with the declared semantics of the related operation (CAD artifacts in the automotive domain).
Starting from these considerations, we aim at defining a discovery engine able to search for the desired services in the network on the basis of functional semantics and QoS properties, taking into account the knowledge of the specific domain of the services. In fact in each specific and real context there are different and appropriate QoS characteristics that we have to consider in the selection of a Web Service.
QoS is defined in ISO 8402 as: "The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs". QoS describes the capabilities of a product or service to meet the requirements of consumers. Menascé [2] defines QoS as "a combination of several qualities or properties of a service". It is a set of non-functional attributes that may influence the quality of the service provided by a resource and consequently represent key components of a Web Service Agreement [27].
Quality of Web Services is emerging as a key aspect, as underlined by many recent research works that state the great importance of QoS properties [2, 3, 5-16, 22-24, 27] for using specific resources in a Service Oriented Architecture (SOA).
SOAP, WSDL, and UDDI, represent the basic technologies for Web Services. They encode data syntax and structure, but do not consider semantics and QoS properties, since these technologies do not describe the meaning of the information to be interchanged and do not propose a QoS standard or recommendation. Since matching algorithms depend mostly on the kind of information they exploit, a

IEEE International Conference on Service-Oriented Computing and Applications(SOCA'07)
0-7695-2861-9/07 $20.00 © 2007 IEEE

representation for the service descriptions, expressive enough to represent the knowledge of interest, is needed. The QoS syntactic descriptions are not adequate, since services are developed independently from different organizations that may specify the service parameters using different terminologies, different measurement processes, scales and measure units. Semantics of QoS became a necessity if we think that QoS measures are required and observed by Web Services users: Service Providers, Service Consumers and Services Monitors. These actors are not only human beings but for example also programs that send requests for services to web service providers, or software agents that monitor QoS parameters.

To enable automatic QoS based service discovery, we propose a formalized, machine understandable specification (QoS ontology), named *onQoS* (*όνQoS,* i.e. QoS entity, **ὄν** in the early Greek, part. of εἶναι: to be). It is able to support the reasoning power required to navigate the QoS terminology correctly and efficiently and it is able to formalize QoS knowledge, i.e. QoS parameters and their relationships. We utilize the ontology as the means to resolve terminology mismatches between the vocabularies, misunderstandings at message level, unsuccessful couplings between measurement processes, scales and units of measurement and as the means to derive knowledge utilizing the domain-dependent functions that can be specified between QoS metrics. The paper shows the powerful of the proposed ontology by presenting a matching algorithm that extends an existing one [10] with reasoning on the ontology.

The rest of the paper is organized as follows. In Section 2, we examine existing ontology specifications and propose a semantic space as a reference model to classify existing research efforts in QoS matching and the one proposed in this paper. Section 3 presents the *onQoS* ontology and motivates the choices for each entity we consider. In Section 4, we introduce the *Derivation* direction and present a matching algorithm that exploits the functional relationships among the QoS parameters. Section 5 reports the experimental results. Section 6 presents the conclusions and future works.

2. Related Works

The development of a well defined and shared QoS ontology is a lengthy and complex process. There are many good efforts to defining it. The DAML-QoS Ontology [3] is available on line, but the adopted approach is flawed. The authors, in fact, utilize incorrectly the OWL cardinality construct to impose bounds on QoS parameter values. This solution allows them to apply easily a variant of Paolucci et al. degrees [4] in the matching process so ensuring more specific and precise results. However, the cardinality ranges is over nonnegative integers and this implies that the QoS measurements have to be positive integers.

Most of the researchers in this domain share the methodology of designing ontology specifications in a modular way. So, the researchers focus on different aspects. For example, [5, 6] concentrate on the definition of the QoS vocabulary. These works capture a set of numerous and multifaceted quality attributes; however most of them simply describe attributes through name-value pairs [10] and default unit measures. We argue that an expressive semantic model is useful to enrich QoS service descriptions for enhancing the matching process. References [8, 9, 11, 27] identify the importance of QoS measurements, even if this aspect is not systematically tackled. For example, in QoSOnt ontology [8, 11] a metric is initially defined as "one way of measuring a specific QoS attribute", and then the authors state that "it must result in a numerical value". In this way, they do not take into account the most used and requested QoS client-side metrics, those based on nominal and ordinal scales. For example, the performance can be specified by means of QoS-grade indicators or keywords (i.e. using the simple values of the following set {"low", "medium", "high", "excellent"}) or by means of numbers.

An ontology definition should ensure different levels of accuracy. In fact, even though for discovery it is desirable to have a higher accuracy (e.g. interval or ratio), human readability and ease of interpretation may suggest the use of less accurate scales (e.g. nominal or ordinal). Furthermore, in [8] the authors state that an attribute is un-measurable if "it cannot necessarily be measured from a given viewpoint" and explain the concept by giving the example of the adherence to the *Data Protection Act*. This attribute is instead measurable through the Boolean nominal metric. The specific scale values that we can associate to this attribute are {true, false}. Also in [9] the authors introduce un-measurable attributes and define them imprecisely as dimensionless parameters (Unit = null). We believe that the concept of un-measurable QoS parameters is not relevant for the scope of the ontology, i.e. for QoS discovery process. QoS parameters that are not measurable do not help us to make decision about their degree of match. In [27], instead, the authors introduce the *Metric* concept in order to specify the unit and data type used to measure the quality dimension. Zeng [10],

Tsesmetzis [9] and Dobson[8] underline that the decreasing value of some QoS parameters, like for example latency or cost, provides a betterment in quality. Not always a bigger value of a QoS parameter is better. So they take into account this in their models: Zeng introduces a vector, Tsesmetzis utilizes the ontological concept *QoSImpact,* while Dobson associates an acceptability direction to a metric. Although everyone considers fundamental for the ontology to contain links to OWL-S, only few do this effectively as in [6, 8].

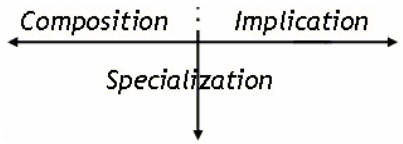

Figure 1: The three directions in the semantic matching

In order to set up a right process for the definition of a robust QoS ontology and to systematize the numerous proposals in this area, we identified the *Specialization*, the *Implication* and the *Composition* direction (*Figure 1)* to exploit the QoS knowledge in the matching process and we introduced a fourth one, as explained in Section 4.

In the *QoS Specialization* dimension, discovery is performed by exploiting generalization/specialization relations to compute the matching degree between QoS descriptions. Chen Zhou et al. [3] employ a different definition of the Paolucci et al. [4] matching degrees in their work. Dobson et al [11] use the "is a" relationship in the first phase of the matching process and only if the target is in exact match with the template they compare the QoS values. Cardoso [26] distinguishes four concepts based on the subsume relations: "equal", "specialization", "generalization", "not equals and not subsume". Then the matching is evaluated by analyzing how many common properties exist between the concepts and how many properties are different. In [27], instead, the semantic matchmaking exploits the *isofType* property of the proposed QoS ontology in order to check if two quality dimensions with different identifiers are semantically equivalent, i.e. if they belong to the same QoS dimension group.

The *QoS Implication (or Correlation)* dimension is typically linked to the domain. Taher et al. [12] introduce the QoS constraints in the first-order logic. Each QoS Constraint is an implication of the form $X \rightarrow Y$, where X is a premise and Y is a conclusion. X and Y are predicates applied on the QoS properties. The approach produces a better discovery effectiveness. Oldham et al. [13] utilize the rules, stored in a rules' storage or formulated in the user query, to correlate the QoS parameters. In [7] the authors introduce the composite requirement stating simply that if all sub-requirements are met then the composite one is necessarily satisfied.

The last dimension is the *QoS Composition (or Aggregation among Web Services)* one; QoS-based selection for service composition is an active area of research. This concerns to discover a set of Web Services whose composition satisfies a global QoS requirement. The works are generally classified in workflow or AI-planning based. Cardoso [22] discusses the aggregation of QoS proprieties in a workflow for some particular QoS parameters. Zeng et al [14] create value-added services by composing existing ones through linear programming techniques. On the contrary, in [16] Genetic Algorithms are used to determine (near) optimal solutions to the composition problem to handle attributes that are non-linearly composed.

Oldham et al. [13] use ontology and rules to improve matching. Contemporarily to their effort, we worked on similar objectives by coding a deep QoS knowledge directly in an ontology (called *onQoS*) so to capture widely shared domain rules as *Availability = MTTF / (MTTF + MTTR)*, without the necessity of introducing a storage for QoS rules. This way, we allow for semantic reasoning over the domain expressions of the ontology. We share the vision that considers rules as supplement domain knowledge to provide a high level of flexibility by stating customized user preferences. Rules can be used to consider the subjectivity of the latter two and implement a feature that allows for the specification of what the user prefers and what the user considers unsuitable.

3. The onQoS Ontology

onQoS is used for specifying the QoS advertisements (target *t*) by Service Providers and the QoS requirements (template *T*) by Service Consumers. The ontological concepts derive from the analysis of process measurement and from its employment in the specification of the QoS entity (ὄν in early Greek, part. of εἶναι: to be). We describe QoS by defining attributes that are useful and important to select the best service.

The quantitative approach is underlined by the chosen names for the main concepts of the ontology: *QoSParameter*, *QoSMetric*, *MeasurementProcess*, *Scale* and *ScaleValue*. *QoSParameter* is a (esp. measurable or quantifiable) QoS characteristic or feature. *QoSMetric* is a type of measurement which relates to a QoS parameter. *MeasurementProcess* is the process by

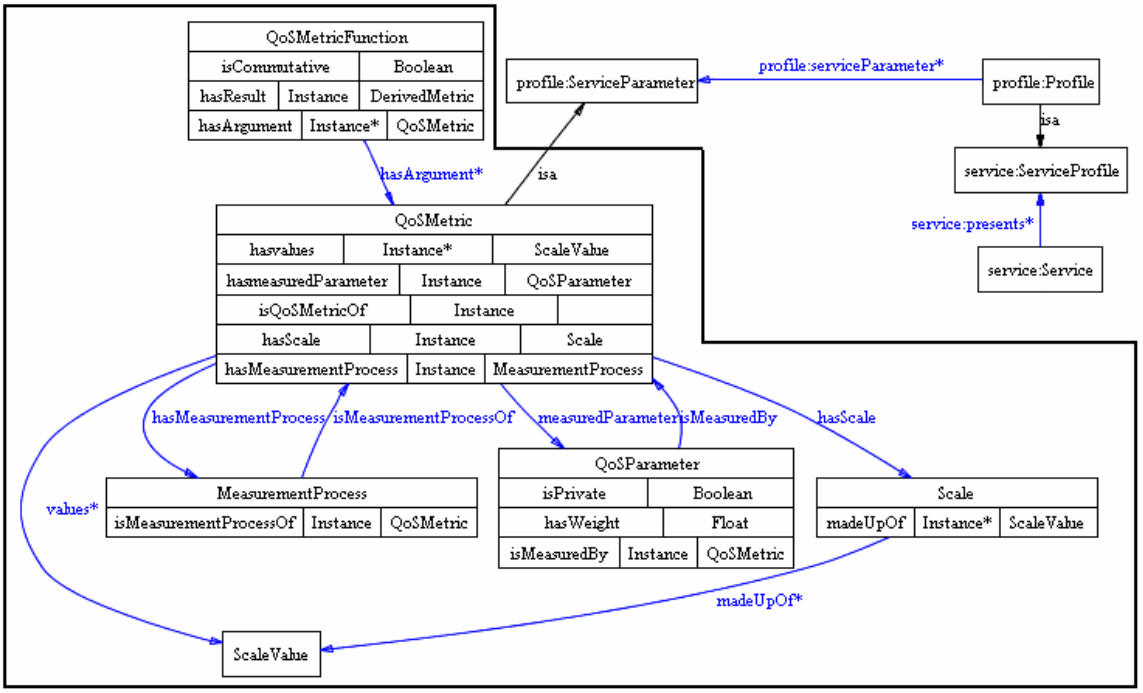

Figure 2: Ontological language

which numbers or symbols are assigned to QoS parameters according to clearly defined rules. For example, measurement processes of the QoS reputation parameter are investigated in [23, 24], where the authors propose models to compute the reputation values. *Scale* specifies the nature of the relationship between a set of values (*ScaleValue*); *ScaleValue* is a number or symbol that identifies a category in which the QoS parameters can be placed based on a particular attribute.

onQoS provides mechanisms that take into account:

- the relations (domain dependent or independent) among QoS parameters;
- the relations between QoS parameters and the context.

It was built in OWL [17] using Protegé [18] as editor and browser, whereas the figures shown in the paper were produced using the Protégé Ontoviz plugin [19]. To check the consistency of the ontology and to make inference we utilized the reasoner Pellet [20], while we used Jena as ontology query engine [21].

The ontology is composed of three extensible complementary layers: upper, middle and lower ontology.

3.1. Upper ontology

The upper ontology describes the QoS ontological language. It provides "the words" we need in order to provide the most appropriate information for formulating and for answering the QoS queries. Its structure is motivated by the need to provide a link to the OWL-S ontology and the means to specify QoS measurements. In fact (*Figure 2*), QoS policy is linked to service functionality through the *QoSMetric* concept.

The subject matter of the measurement process is the QoS of a Web Service operation. *QoSMetric* is a *ServiceParameter* of a *ServiceProfile*. The concept *QoSMetric* presents the properties *hasMeasurementProcess, hasMeasuredParameter, hasScale, hasValues* and *isQoSMetricOf* respectively limited to having a single value on the range of *MeasurementProcess, QoSParameter, Scale, ScaleValue* and multiple values on *QoSMetric*.

The concept *QoSParameter*, instead, presents the following data properties:

- *hasWeight* on the float range ([0,1]) specifying the user request preferences. A weight of 0 is equivalent to an unconstrained QoS attribute.
- *isPrivate* on Boolean range. This property is specified by the service provider to limit the visibility of a parameter value.

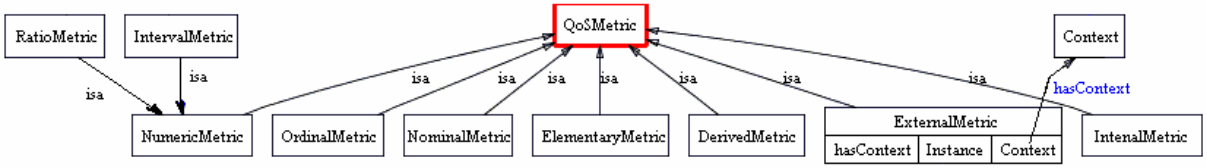

Figure 3: The QoS Metric middle ontology

3.2. Middle ontology

The middle ontology defines the standard vocabulary of the ontology, such as, for example QoS parameters, QoS metrics and QoS scales.

The QoS vocabulary is a set of domain-independent parameters according to the conceptualization of Maximilien et al. [6] and Ran [5]. QoS parameters can be evaluated from the perspective of providers and from the perspective of users. For service providers, a QoS parameter describes the provided QoS, for service consumers the required QoS, for service monitors the expected one.

The root of the taxonomy is the *QoSParameter* concept. It links the upper ontology with the middle ontology and it is a common super class for the following concepts:

- *Availability*: the likelihood that a service is available. It has two subclasses: MTTR (mean time to repair, meaning the average time for restoring a failed service) and Uptime (the duration for which the service has been operational continuously without failure).
- *Capacity*: a limit of concurrent requests for guaranteed performance.
- *Cost*: the amount of money for a single service execution.
- *Integrity*: a measure of the service's ability to prevent unauthorized access and preserve its data integrity.
- *Performance*: it has three sub concepts. Throughput (the number of requests served in a given time period), Response Time (the delay from the request to getting a response from the service) and Latency (the time between client request and the start of its response).
- *Reliability*: it is the guarantee that a service doesn't break within a given period of time in a way that the user (human or another service) notices the failure.
- *Robustness:* it is resilience to ill-formed input and incorrect invocation sequences.
- *Scalability:* it defines whether the service capacity can increase as needed.
- *Security*: it includes the existence and type of authentication mechanisms the service offers, confidentiality and data integrity of messages exchanged, non-repudiation of requests or messages, and resilience to denial of service attacks. It includes the following concepts: *Auditability* (the service maintains auditable logs); *Authentication* (the service either requires user authentication or accepts anonymous users); *Encryption* (the type and strength of encryption technology used for storage and messaging; it is also a sub concept of Supported Standard); and *Non Repudiation* (whether consumers can deny having used the service).
- *WSQoS* is a measure of the whole QoS of Web Service functionality, calculated by means of a particular measurement process. In our implementation we compute this value utilizing the model proposed in [10], but other measurement process can be used.

Figure 3 represents the QoS Metric Vocabulary. A QoS metric can be classified according on the used scale (Nominal, Ordinal, Ratio…), basing on the number of measured QoS parameters (one or more) or with regard to the exploited information in the measurement process. The properties *hasScale* and *hasValues* of *QoSMetric* (shown in Figure *2*) consent to define a type of metric, for example the *NominalMetric,* which has range only on values of a nominal scale. The identified concepts of the hierarchy are:

- *Elementary Metric*: it is characterized by the direct measure of a QoS parameter. A value in this metric can be evaluated as: $V_{el} = f_{el}(q)$, where q is a single QoS parameter and $f_{el}()$ the direct measurement process.
- *Derived Metric:* it is characterized by the indirect measure of a QoS parameter through the measures of correlated QoS parameters. A value in this metric can be evaluated as: $V_{de} = f_{de}(\boldsymbol{q})$, where $\boldsymbol{q}$ is a vector of QoS parameters and $f_{de}()$ the indirect measurement process.
- *Internal Metric*: it is characterized by a direct or indirect measure of one or more QoS parameters without other information. A value in this metric

can be evaluated as: $V_{in} = f_{in}(\boldsymbol{q})$, where $\boldsymbol{q}$ is a vector of QoS parameters and $f_{in}()$ a direct or indirect measurement process. This metric can be either derived or elementary according to the cardinality of $\boldsymbol{q}$.

- *External Metric*: it is characterized by an indirect measure of one or more QoS parameters that depends on context attributes. A value in this metric can be evaluated as: $V_{ex} = f_{ex}(\boldsymbol{C}, \boldsymbol{q})$, where $\boldsymbol{q}$ is the vector of QoS metrics, C_{is} the context pertaining to the parameters $\boldsymbol{q}$, and $f_{ex}()$ is the indirect measurement process. In service advertisements, C specifies the conditions the provider needs to ensure the QoS features, i.e. the environment in which the QoS measures are to be evaluated. On the other hand, in service queries, it describes the actual conditions or constraints that need to be taken into account in the matching process. This metric can be derived or elementary according to the cardinality of $\boldsymbol{q}$.
- *Nominal Metric:* it has range on values of a nominal scale. Symbols, generally in the form of strings, are associated to the QoS parameters.
- *Ordinal Metric:* it has range on values of an ordinal scale.
- *Numeric Metric:* in the measurement process only numbers (not symbols) can be assigned to the QoS parameters.
- *Ratio Metric:* it has range on values of a ratio scale.
- *Interval Metric:* it has range on values of an interval scale.

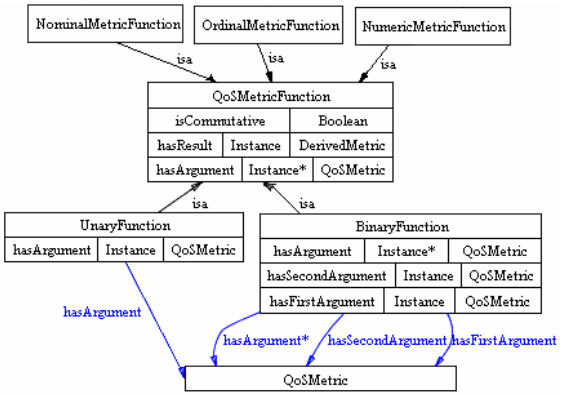

Figure 4: The QoS Metric Function Middle Ontology

Figure 4 reports the middle ontology of the Metric Functions. *QoSMetricFunction* permits to define functions of elementary, derived, external or internal QoS metrics in order to specify a derived metric. The arguments of *QoSMetricFunction* are *QoSMetrics*. The resulting metric is a *DerivedMetric*. The Boolean property *isCommutative* permits to specify the commutability of the arguments. The functions are classified basing on the number of their arguments. The concepts *UnaryFunction* and *BinaryFunction* describe respectively a function with arity one and two. We specialize the *QoSMetricFunction* in *NominalMetricFunction, NumericMetricFunction, OrdinalMetricFunction,* which enable to construct QoS expressions of the corresponding metrics. These expressions can be used for describing subjective interpretations of a QoS measurement.

3.3. Low ontology

The low ontology defines the concepts, the proprieties, and the constraints of a specific domain.

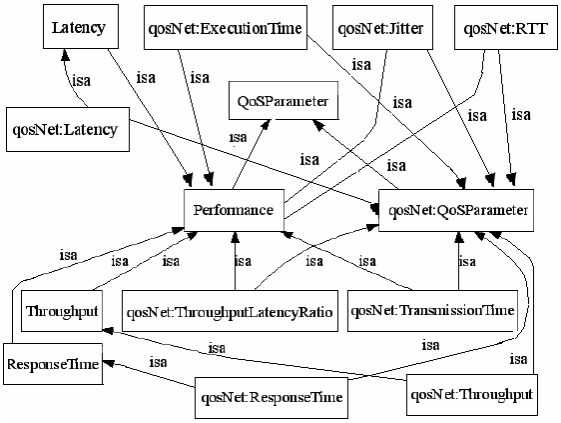

Figure 5: QoS Network Ontology

Typically, a low ontology contains the definitions of the concepts to be used in a real application. *Figure 5* shows a QoS Network Ontology that is reported in this paper to progressively explain the concepts introduced. We implemented also a QoS Automotive Ontology, not reported here due to limited space, but you can see it in [1]. At this level, an individual of a desiderate *QoSMetricFunction*, specifying the properties of the component metrics and the properties of the resulting metric, can be instantiated. This way, a domain-dependent QoS knowledge can be directly specified in the ontology.

4. A matching algorithm based on QoS ontology

In some domains, relationships among parameters are well-known. For example, the network latency (L) presents shared definitions that can be exploited in order to increase the recall of the matching process.

The common way ($L = RTT + t_e + t_t$) of measuring network latency takes into account the following contributors: round trip delay, execution and transmission times. As humans immediately derive the formula $RTT = L - t_e - t_t$ and are able to reason on it and obtain the right pairs in matching, so this type of reasoning can be implemented in the matching algorithm by making inference on the QoS knowledge. Therefore a pair of descriptions apparently mismatched (template T and target t_1) can result as the optimal solution in the services space. In fact if the *RTT* request value of the service template T is *"≤ 10 seconds"* and the latency L of the service target t_1 is *"≤ 10 seconds"*, while the specified values in the target t_2 are "*RTT=11s, t_e=8s, t_t=2s"*. The couple (T, t_1) is the best match and can be obtained by interpreting the semantic rule between the parameters.

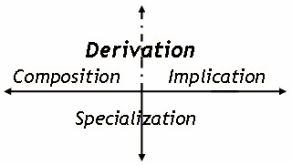

Figure 6: The four directions in the semantic matching

As consequence of the previous consideration, we introduce a forth dimension in the process of locating a semantic Web Service with regard to the above identified space of analysis: *QoS Derivation*. It refers to the process of combining several QoS parameters of a specified Web service into a single one and to the process of obtaining one quality dimension through reasoning on QoS formalized knowledge. Based on this dimension, we defined a matching algorithm (*Figure 7)* that extends on the one proposed in [10] by adding to it semantic QoS descriptions. The valuated score takes into account the weights that the users attach to each particular parameter. So services with higher scores reflect a higher similarity with the input description specified by the user.

The input of the matching algorithm is the QoS semantic descriptions and optionally the required level of confidence, i.e. the claimed degree of match (exact, plug in, subsume, intersection). In the QoS query descriptions the users can specify through the weights the importance of each parameter in the matching process. A weight describes how the user prefers a particular characteristic rather than another. The algorithm includes two phases: atomic matching (1:1) and aggregated matching (1:N). The outcome of the first phase is the *atomicMappingList* with the following information: required QoS metric, advertisement QoS metric, and matching score. We obtained these data utilizing the Pellet reasoner on the ontological descriptions. So we are able to understand the equivalence between measures expressed with different units. The second step recovers the *nonMappedListof* template metrics that are not in atomic matching and, for each QoS template metric of the *nonMappedList*, it searches a mapping with more QoS target metrics exploiting the QoS knowledge of the ontology.

```
qosMatch(T, t) {
 effectiveConfidenceLevel=0;
 mappingList=empty;
 reasoner.check(T);
 reasoner.check(t);
 atomicMappingList= atomicMatch(T,t);
 nonMappedList= findNonMappedMetrics(atomicMappingList);
 aggregateMappingList = aggregateMatch(nonMappedList, t);
 mappingList=atomicMappingList.append(aggregateMappingList);
 score=weightedAverage(mappingList);
 effectiveConfidenceLevel=getNumberOfSatisfiedMetric(T,t)/
                                    getMetricNumber(T);
return;}
```

Figure 7: Matching algorithm

5. Experimental results

To describe a QoS specification, we created an XML file, written in OWL, in which we instanced the individuals of the concepts of the *onQoS* ontology. The service QoS advertisements were stored in a registry. When the discovery function receives a request from a user, it runs the matching algorithm. Since matching is a particular instance of information retrieval problem, *precision* and *recall* [28] can be used to evaluate the algorithm.

Figure 8 shows eight QoS descriptions that we used both as template and targets, in every combination. In particular, we manually compared each description with each other one (64 comparisons in total) and observed the results reported in *Figure 9*.

	QoS requirements		QoS requirements
D1	*Authentication* *Authorization* *Cost <= 100€* *EncStand: RSA, PKI, OpenPGP, Triple-DES* *ExecutionTime <= 0.5 ms* *FaulRate <= 50%* *Jitter <= 0.3 ms* *NetThroughput >= 200 kbps* *RTT <= 14ms* *Scalability >= 78%* *TransmissionTime <= 7 ms* *UpTime >= 90%*	D3	*Cost <= 140 €* *EncStand: RSA, PKI* *Jitter <= 0.3 ms* *NetLatency <= 24.9 ms* *Privacy* *UpTime >= 65%*
		D4	*Authentication* *Authorization* *NetLatency <= 26 ms* *ThrLatRatio >=3.2 Mbps/s*
D2	*Cost <= 121 €* *EncStand: RSA, PKI, OpenPGP* *ExecutionTime <= 0.6 ms* *FaulRate <= 50%* *Jitter <= 1.5 ms* *Privacy* *RTT <= 17ms* *Scalability >= 43%* *ThrLatRatio >= 3.5Mbps/s* *UpTime >= 86%*	D5	*Authentication* *Authorization* *EncStand: RSA, PKI, OpenPGP* *ExecutionTime <= 0.8 ms* *Jitter <= 2.6 ms* *NetThroughput >= 10 kbps* *RTT <= 5 ms* *TransmissionTime <= 6 ms* *UpTime >= 65%*
		D6	*NetLatency <= 22 ms*
		D7	*Authentication* *Authorization*
		D8	*RTT <= 25ms*

Figure 8: QoS descriptions

During the automatic matching, the algorithm exploited the knowledge of the expression $L = RTT + t_e + t_t$ stored in the Network Ontology and the concepts of *Authentication* and *Authorization* specified by the Service Consumer.

	D1	D2	D3	D4	D5	D6	D7	D8
D1	match							
D2		match						
D3	match	match	match		match			
D4	match	match		match				
D5					match			
D6								
D7	match	match	match	match	match		match	
D8	match	match	match		match	match		match

Figure 9: Manual results of the matching process

The first rule is formulated through an instance of the *SimpleRatioSum* concept (the sum function on values of a ratio scale) that returns a simple ratio metric on the QoS parameter latency. The arguments of the function are the instances of the *SimpleRatioMetric* concept on the following three QoS parameters: RTT, execution and transmission times.

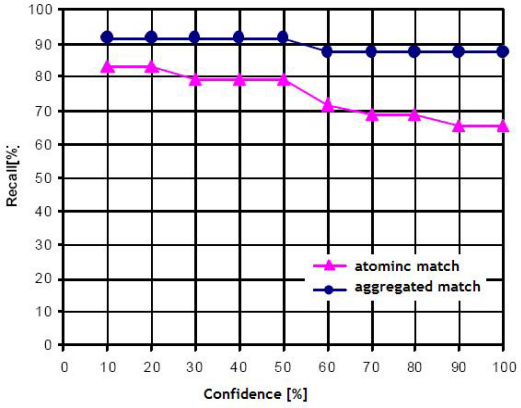

Figure 10: Recall

In D3 *(Figure 8)* the user requires the privacy requirement and states that the privacy is ensured if *Authorization* and *Authentication* are guaranteed. A different Service Consumer can specify another more appropriate combination of privacy factors among Authorization, Authentication and Accounting. So in D3, the Service Consumer demands privacy and specifies the privacy protection it needs utilizing an instance of the *MetricFunction* concept on the QoS parameter *PrivacyPreservation* of the Network Ontology *(Figure 5)*.

We evaluated our ontology and algorithm by computing recall and precision for each level of confidence, where 0% of confidence indicates the maximum level of flexibility of the matching algorithm (minimum level of robustness).

The results (*Figure* 10) show clearly that the recall is improved by using the proposed algorithm for each level of confidence given in input whereas the precision is not significantly decreased (*Figure* 11).

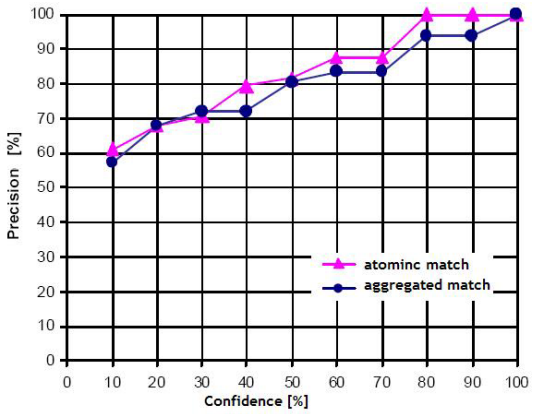

Figure 11: Precision

We can conclude that the proposed ontology and the introduction of functions to process derived QoS parameters significantly improve matching recall without degrading precision.

6. Conclusions and future works

In this work, we addressed QoS semantics to improve the recall of the matching process by exploiting domain knowledge. Our QoS specification approach uses metrics to understand, describe, and control the QoS in the matching process. The QoS upper ontology provides a generic QoS schema representation that captures the essential information needed in the web services discovery, i.e. the measurable ones. The middle ontology defines the basic concept of the QoS domain through the specification of the generic quality dimensions. The low ontology, instead, contains the domain nuances of a specific application domain.

By exploiting the derivation dimension in the matching process, we significantly influence service selection as demonstrated by the results of the experimental analysis. In the future, several open issues will be investigated. We plan to adopt SWRL, a combination of OWL and RuleML, in order to implement implications in our matching algorithm through the definition of axioms and we will continue to work on the ontology in order to detail other aspects in order to improve expressivity.

Acknowledgements

The work presented in this paper is partially supported by LOCOSP and ArtDeco projects funded

by Italian Ministry of University and Scientific Research, ex MIUR.
We also thank Giuseppe Damiano and Angelo Furno for their important support during the implementation phase of the matchmaking framework.

7. References

[1] LOCOSP, http://plone.rcost.unisannio.it/locosp.

[2] D. A. Menascé, "QoS issues in web services", In IEEE Internet Computing, IEEE Press, pages 72-75, November-December 2002.

[3] C. Zhou, L.-T. Chia, B.-S. Lee, "DAML-QoS Ontology for Web Services", Proceeding of the International Conference on Web Services 2004 (ICWS 2004), pages 472-479, San Diego, California, USA, July 2004.

[4] M. Paolucci, T. Kawamura, T. R. Payne, K. P. Sycara, "Semantic Matching of Web Services Capabilities", in Proceedings of the first International Semantic Web Conference (ISWC2002), 2002, pp. 333-347.

[5] S.Ran, "A Model for Web Services Discovery with QoS", SIGEcom Exchanges, vol. 4, n°1, pages 1-10, 2004.

[6] E. M. Maximilien, M. P. Singh, "A Framework and Ontology for Dynamic Web Services Selection", IEEE Internet Computing 8(5), pp. 84-93, 2004.

[7] HM. Kim, A. Sengupta and J. Evermann, "MOQ: Web Services Ontologies for QOS and General Quality Evaluations", Proceedings of the Thirteenth European Conference on Information Systems, Regensburg, Germany, May 2005.

[8] G. Dobson, R. Lock, "QoSOnt: an Ontology for QoS in Service-Centric Systems", UK e-Science AHM, 2005.

[9] D. T. Tsesmetzis, I. Russaki, I. V. Papaioannou, Miltiades E. Anagnostou, "A QoS Ontology Language for Web Services", 101-106, AINA (1) 2006.

[10] Y. Liu, A. H. H. Ngu, L. Zeng, "QoS computation and policing in dynamic web service selection", WWW (Alternate Track Papers & Posters), pp. 66-73, 2004.

[11] G. Dobson, R. Lock, I. Sommerville, "Quality of Service Requirements Specification Using an Ontology", SOCCER Workshop, Requirements Engineering 2005.

[12] L. Taher, R. Basha, H. El Khatib, "QoS Information & Computation (QoS-IC) Framework for QoS-Based Discovery of Web Services", UPGRADE, The European Journal for the Informatics Professional, Vol. VI, Issue No. 4, August 2005.

[13] N. Oldham, K. Verma, A. Sheth, F. Hakimpour, "Semantic WS-Agreement Partner Selection", 15th International World Wide Web Conference (WWW 2006), Edinburgh, Scotland, UK, May 2006.

[14] L. Zeng, B. Benatallah, Anne H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, "QoS-Aware Middleware for Web Services Composition", IEEE Trans. Software Eng. 30(5), pp. 311-327, 2004.

[15] X. Gu and K. Nahrstedt, "A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids," Proc. 11th IEEE Int'l Symp. High Performance Distributed Computing (HPDC), pp. 73-82, July 2002.

[16] G. Canfora, M. Di Penta, R. Esposito, M. L. Villani, "An Approach for QoS-aware Service Composition based on Genetic Algorithms", in Proceedings of the Genetic and Computation Conference, June 2005, Whashington, DC, ACM Press.

[17] OWL-S, "An OWL-based Web service ontology", http://www.daml.org/services/owl-s/1.0/

[18] Protegé ontology editor, release version 3.1.1, http://protege.stanford.edu

[19] The OntoViz Tab - Visualizing Protégé - 2000 Ontologies, http://protege.stanford.edu/plugins/ontoviz/ontoviz.html

[20] Pellet Reasoner, 1.3, April 17, 2006, http://www.mindswap.org/2003/pellet/

[21] Jena 2.4, http://jena.sourceforge.net

[22] J. Cardoso, "Quality of Service and Semantic Composition of Workflows", PhD thesis, Department of Computer Science, University of Georgia, Athens, GA (USA), 2002.

[23] S. Majithia, A. S. Ali, O. F. Rana, D. W. Walker, "Reputation-Based Semantic Service Discovery," 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04), pp. 297-302, 2004.

[24] R. Wishart, R. Robinson, J. Indulska, A. Jsang, "SuperstringRep: Reputation-enhanced Service Discovery", In Proceed. of the Australasian Computer Science Conference 2005.

[25] H. Do, S. Melnik, and E. Rahm, "Comparison of schema matching evaluations", in Proceedings of the 2nd Int. Workshop on Web Databases (German Informatics Society), 2002.

[26] J. Cardoso, "Discovering Semantic Web services with and without a Common Ontology Commitment", in Proceedings of the 3rd International Workshop on Semantic and Dynamic Web Processes, pp. 183-190, September 18-22, 2006, Chicago, USA.

[27] S.Bleul, T.Weise, Kurt Geihs, "An Ontology for Quality-Aware Service Discovery", special issue on "Engineering Design and Composition of Service-Oriented Applications", Computer Systems Science Engineering, 2006.