Semantic-aware Service Quality Negotiation *

Marco Comuzzi[1], Kyriakos Kritikos[2], and Pierluigi Plebani[1]

[1] Politecnico di Milano – Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano (Italy)
{comuzzi, plebani}@elet.polimi.it
[2] Institute of Computer Science, FORTH
Heraklion, Crete (Greece)
kritikos@ics.forth.gr

Abstract. Goal of Web service discovery is twofold: (i) to find Web services able to perform the functionalities required by a user, and (ii) to select, among the found Web services, which are the ones able to work in the way the user wants. Focusing on the second step, usually, a matchmaking algorithm takes place to verify if the quality offered by the Web service provider contains the quality requested by the user. Since quality costs, a further step, i.e., the negotiation, should be performed to identify which is the quality level that must be ensured at run-time. In order to automate the negotiation process as much as possible, users and providers need to have a common understanding about the meaning of quality dimensions and negotiation strategies.
In this work, we join previous work on semantic-based quality definition model and negotiation process, to provide a framework enabling the automatic Web service negotiation. More specifically, OWL-Q, a semantic QoS-based WS description language, is extended with all the appropriate negotiation concepts and properties. Finally, by building on OWL-Q and rules, a small example is provided of how QoS negotiation can be assisted and get automated.

1 Introduction

According to the OASIS [3] definition [1] a "Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains". Thus, the ownership holds a key-role in realizing a SOA: for instance, who builds and makes a service available might be different from who consumes the service and they might not know each other in advance. As a consequence, at design-time, service providers try to identify which could be the requirements of a potential user and develop the service accordingly. On the other side, service consumers need to select the best service among a set of available ones considering both what the service does (functional perspective), and how the service works (non-functional perspective).

* The work has been supported by the EU S-Cube Network of Excellence

[3] Organization for the Advancement of Structured Information Standards - http://www.oasis-open.org

In this paper, we focus on the non-functional perspective and, in particular, on the quality of service (QoS hereafter) negotiation, i.e., the process that produces an agreement between a service consumer and a service provider with respect to a) the QoS a service must ensure during the its execution and b) the amount of money the consumer has to pay. In this scenario, the consumer and the provider might be aware of each other just before the service invocation takes place. Thus, the research direction is to make the negotiation process as much automatic as possible [2] but this goal is achievable only if both service providers and users agree on the same model for expressing QoS dimensions.

In a previous work [3], we introduced an approach where as soon as the users expectations are defined and providers clarify their capabilities, then the matchmaking and the agreement processes are automatically performed on-the-fly. The main limitation of this approach is about the assumption that both users and providers must share the same set of quality dimensions to describe the requirements and capabilities, respectively. Actually, in some application domains, different domain experts may not come to a complete agreement for a common (domain-dependent) QoS model. In this way, there may be two or more 'standard' QoS models for a specific application domain and users may choose one of them for expressing their QoS requirements/capabilities.

In this paper, we aim at overcoming this limitation by adopting and improving OWL-Q [4]: an extension of OWL-S for a rich, semantic and extensible QoS-based service description. There are a lot of reasons for using ontologies and rules to express quality models and to automate QoS matchmaking and negotiation. First of all, ontologies provide a formal, syntactic and semantic description model of concepts, properties and relationships between concepts. They give meaning to concepts like QoS dimensions, value types, offers and requests so that they are human-understandable and machine-interpretable while they provide the means for interoperability. Moreover, they are extensible as new concepts, properties or relationships can be added to an ontology. In addition, Semantic Web (SW) techniques can be used for reasoning about concepts or for resolving or mapping between ontologies. These techniques can lead to syntactic and semantic matching of ontological concepts and enforcement of class and property (e.g. type checking, cardinality) constraints. Therefore, by providing semantic description of concepts and by supporting reasoning mechanisms, ontologies cater for better discovery process with high precision and recall. Last but not least, ontologies can help specialized agents in performing very complex reasoning tasks like service discovery or mediation or negotiation. If QoS models and specifications are expressed with ontologies, then they could be aligned with each other so that the WS discovery and negotiation processes are not actually affected. This alignment process is realized with the additional use of rules that define mappings between concepts of different models and specifications.

The discussion about our proposal on semantic-aware quality negotiation will be tied to a running example: *SMS Monitoring*. This service is available to all the users that have a contract with a national mobile phone company. When a user sends the mobile phone number, the service returns how many SMSs have

been sent from that number starting from the beginning of the current month. In this case, we consider service *availability*, *response time*, and *coverage* (i.e., how many mobile phone company can be queried by the service) as the most relevant quality dimensions.

The work is structured as follows. In Section 2, we introduce which are the requirements of a quality model that need to be satisfied to support automatic negotiation. Section 3 briefly analyzes the main elements composing OWL-Q. In Section 4, we extend OWL-Q with new concepts specifically intended for supporting the automatic negotiation and we introduce the negotiation algorithm. Finally, Section 5 discusses related work and Section 6 concludes the paper outlining possible future research directions.

2 Quality model requirements

As discussed in Kritikos and Plexousakis [5] there is a set of requirements that a WS QoS description needs to satisfy to be really adopted. First of all, the quality can be defined both as a *requirement* and as a *capability*. Thus, it should be possible to specify both the QoS properties that clients require and the QoS properties that services provide. These two aspects should be specified in two separated documents that must be compared at service selection phase, to realize if the quality provided by a service satisfies the user's requirements.

Due to the domain dependance of the quality definition, a QoS model has to be *extensible*: it has to include both domain independent QoS dimensions, and domain specific QoS dimension, and new domain specific criteria should be added and used to evaluate QoS without changing the underlying computation (i.e. matchmaking and ranking) model. To make possible knowledge sharing and the comparison between capabilities and requirements, users and providers have to agree on the adopted syntax and semantics. About the syntax, the QoS model has to be *compliant* with already widely-accepted standards, e.g., WS-Policy. Concerning semantics, QoS concepts must be *formally* described in order to have terms/concepts with specific meaning for both requesters and providers.

To improve the flexibility of the QoS model, it should be *syntactically separated* from other parts of service specifications, such as interface definitions. On one hand, this improves reusability in describing several services with the same QoS or a service with different level of QoS. On the other hand, this allows the specification of *classes of service*: the discrete variation of the complete service and QoS provided by one WS.

Due to these initial requirements, QoS models are usually defined by a composition of several quality dimensions (a.k.a. quality parameters, or quality attributes). Each attribute is measured with the help of a *metric* that gives an objective way to state which are the possible and actual values for a given dimension. For each domain, the quality dimensions in that domain are important inputs to the overall QoS of a service. Some attributes are common across domains and some are specific to domains. More specifically, a *QoS dimension* should be defined at least by the following aspects:

- The value set for the metric (and its allowed value range) to know which are the admissible values for the dimension.
- The domains that this attribute belongs to.
- The weight of the metric relative to its domain and user preferences to rank the dimensions in order of importance.
- The characteristic of the function from metric values to overall QoS values to know how the quality varies with respect to a quality dimensions value variation.
- The temporal characteristic of the metric value.
- The description (mathematical or otherwise formal) of how a QoS metric value of a complex WS can be derived from the corresponding QoS metrics values of the individual WSs that constitute the complex one.
- A set of reference ontologies: e.g., ontology of measurement units, ontology of currency units, ontology of measured properties and ontology of measurement methods.

Focusing on the first point, i.e., the value set for the metric, it represents the starting point for the matchmaking phase. Indeed, in the user requirements document, this set expresses the value range in which a quality dimension has to vary during the service execution. For instance, the user requests that *availability_r* $\in$ [95%..99%] [4]. On the other side, in the provider capabilities document, this set expresses the value range in which the provider promises that the quality dimension varies (e.g., *availability_c* $\in$ [90%..99%]).

The intersection between these two sets is performed during the matchmaking phase to state if a non empty set of values exists that the provider supports and that satisfies the user, e.g., *availability_r* $\cap$ *availability_c* $\in$ [95%..99%]. If this happens for all the quality dimensions included in the requirements document, then the provider is able to support all the user requirements. It is worth noting that the matchmaking only gives a technical evaluation: it states that the user requirements can be fulfilled, but the actual values of quality dimension to be supported have to be defined also considering economical evaluation. For this reason we also consider the negotiation as a further step after the matchmaking. The goal of the negotiation is to identify, for each quality dimension, which are the values that maximize the user expectations having a specific budget. As a consequence, the QoS model has to include elements for evaluating the cost for supporting (provider perspective) or having (user perspective) a given quality dimension level. Since in this paper we also deal with negotiation, the requirements of a QoS model introduced in [5] and discussed above have to be updated accordingly.

From the provider perspective, QoS model needs to include the *cost model*: a function that calculates how much is the effort for the provider for offering a given value for a quality dimension. In this way, the provider calculates how much is the cost for providing a set of capabilities and, consequently, the provider decides

[4] Hereafter, we use the characters 'r' and 'c' as subscripts to indicate a quality dimension in the request or capabilities document, namely.

the *price* for the service. For instance, the cost is proportional to the availability value according to the following formula: $cost(availability) = availability \star 3\$$. Thus, assuming that the provider has a fixed revenue of 5$, the price for having an availability in the range [0.95%..0.97%] is $\in$ [7.85$..7.97$].

From the user perspective, QoS model needs to consider the user's *budget*: the amount of money the user is willing to pay for the service. During the negotiation the budget is strictly related to the weights that express the user preferences among the quality dimensions. Thus, the QoS model needs to consider the *Negotiation strategy*. Usually, the strategy assumes that the greater is the weight, the higher is the preference on that quality dimension. Thus, these weights indicate how the budget should be split among the quality dimensions. Assuming an overall budget of 20$, and availability and response time equally important (the related weights are both 0.5), then the user is willing to pay up to 10$ and, accordingly to the cost model expressed above, this can be feasible. On the contrary, assuming 7.9$ as the budget for availability, then the range of values for this quality dimension must be rearranged accordingly, i.e., [0.95%..0.966%].

It is worth noting that not all the quality dimensions can be negotiable. A user can ask that the range/set of values for a specific quality dimension must be entirely supported. For instance, for the user is mandatory that coverage includes the values of Orange and Verizon. So, the negotiation strategy is not allowed to modify this value set.

On the basis of the above quality model requirements, in this work we propose a framework for semantic-aware negotiation. As shown in Figure 1, we rely on OWL-Q for expressing user capabilities and provider requirements. An extension of OWL-Q, discussed in Section 4, allows the definition of negotiation strategies and cost models that will be used by the negotiation broker to generate the SLAs.

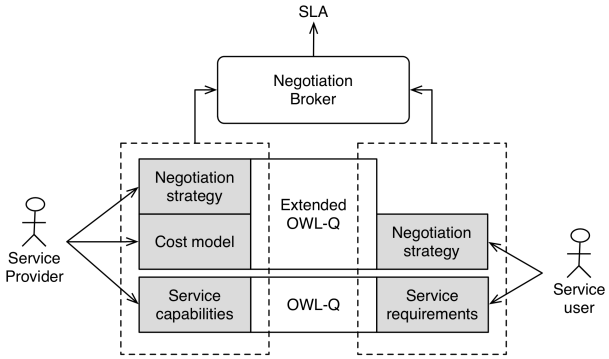

Fig. 1. Overall negotiation framework

3 OWL-Q

For a rich, semantic and extensible QoS-based WS description, OWL-Q has been introduced by Kritikos and Plexousakis in [4]. OWL-Q is an upper ontology that satisfies all the requirements introduced in the previous section. OWL-Q ontology complements OWL-S [6] and comprises of many sub-ontologies/facets. Each facet concentrates on a particular aspect of the QoS modeling and can be extended independently of the others. It is worth noting that some of the concepts here presented, that are also discussed in depth in [4], have been revised according to the quality model presented in [3] that put the basis for a model more oriented to the negotiation.

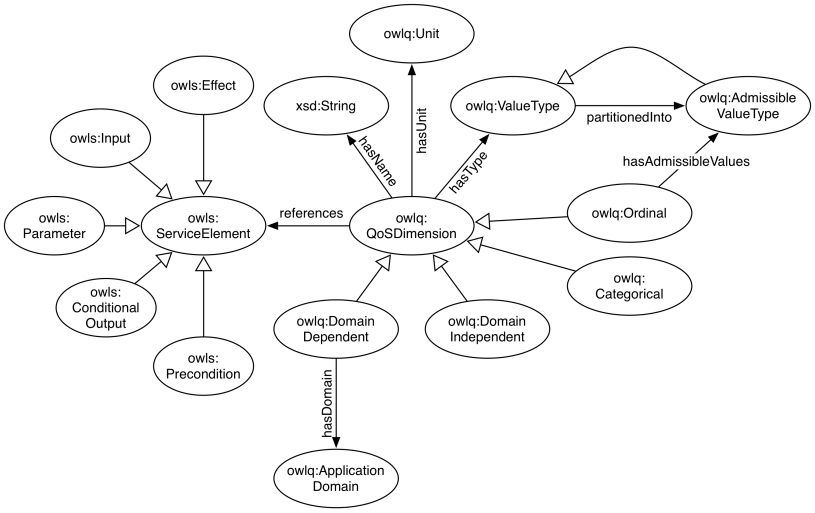

Fig. 2. Overview on OWL-Q

The main element in OWL-Q is the *QoSDimension* (see Figure 2) that can be attached to any *owls:ServiceElement* for expressing preferences or capabilities about such elements. Each dimension has a *Name* and can assume a set of values of a given *ValueType*. A dimension can be either *Categorical* or *Ordinal*. An example of a *Categorical* dimension is *coverage*, where its set of values are the list of mobile phone operators supported by the service, e.g., *Orange, Verizon, Cingular*. In case of *Ordinal* dimensions, the value type is a range of values in which a quality dimension varies. For instance, *availability* can vary from [0%..100%]. This set can be partitioned in several sub-ranges, i.e.,*AdmissibleValueType*s. This allows both users and requesters to express the values of requirements or of capabilities not as a single value but as a range of permitted values (e.g., [90%..94%];[95%..99%]), as discussed in the previous section. An additional classification is given by the dependency to the application domain. Some quality

dimensions are *Domain Independent*, so they can be useful regardless of the type of the service (e.g., *response time* and *availability*); on the contrary, some other quality dimensions are *Domain Dependent* and are related to an *Application domain* (e.g., *coverage*).

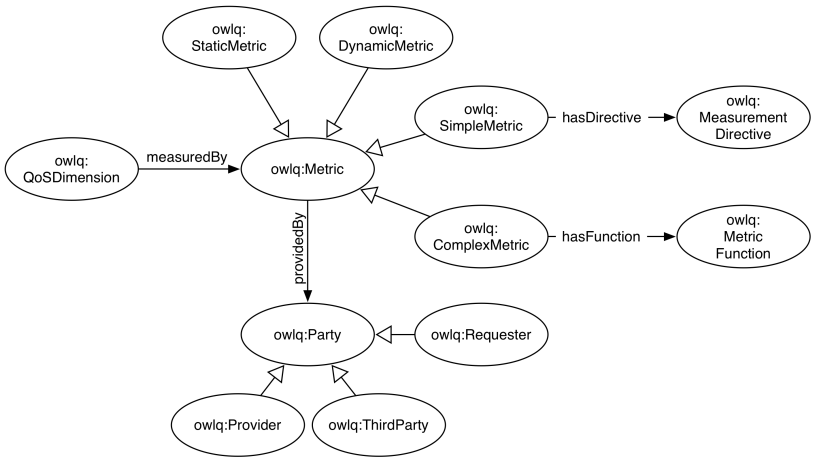

Fig. 3. OWL-Q Metric facet

A definition about who is in charge of measuring a dimension and in which way the measurement takes place is given by the *OWL-Q Metric facet* shown in Figure 3. In more detail, the values of a dimension is provided by a *Party* that is in charge of measuring it. This actor might be a *Provider*, a *Requester*, or a *Third-party*. There are simple QoS metrics *measuredBy* a *MeasurementDirective* or complex ones. *ComplexMetric*s are derived from other metrics with the help of a *MetricFunction*. Metrics can be positively or negatively monotonic. In this way, we know if one metric's value is better than another value. This is fundamental whenever we have to compare several quality definitions in order to state which is the best one. In addition, a *Metric* can be also classified in static and dynamic metrics: a *StaticQoSMetric* is computed only once according to a trigger, whereas a *DynamicQoSMetric* is computed repeatedly according to a schedule.

4 Semantic-aware negotiation

In the multiagent computing literature, a generic automated negotiation framework is usually defined by three elements [7]:

– Negotiation Object: it represents the features of what is under negotiation. In particular, the definition of the negotiation object concerns the identification of the issues under negotiation, their properties, and their admissible values;

– Negotiation Protocol: it identifies the rules that must be followed by the negotiation participants, defining the admissible states of a negotiation and behaviors that can be followed by participants. Classic negotiation protocols are the iterated bilateral negotiation protocol, in which two negotiators alternatively exchange offers, or price-only auction protocols, in which a third party, i.e., the auctioneer, collects offers from the negotiators and identifies the winner according to a pre-specified market clearing rule.
– Negotiators' decision model: it identifies the strategy of the participants to the negotiation framework. In particular, the decision model sets the rules followed by a negotiator to generate new offers and to decide whether to accept or refuse offers or to withdraw from the negotiation.

Our OWL-Q extensions focus on the Negotiation Object and on the Negotiators' decision models. We extend OWL-Q with a new set of concepts and properties that constitute *Quality-related* and *Negotiation-specific* extensions. The former extend the ontology in order to accommodate the Negotiation Object, i.e., the description of what can be negotiated. The latter introduce new concepts, such as the negotiator actor or service consumers' negotiation strategies, which are then used to define the negotiators' decision model. The extended OWL-Q is represented in Figure 4.

For what concerns the negotiation protocol, the extension of OWL-Q to describe various negotiation protocols is currently under development. In this paper, our approach is limited to QoS configuration algorithms for which the the negotiators' strategies are parameterized [3]. The WS QoS configuration protocols proposed in the paper can be implemented by a broker-based architecture for QoS negotiation, as shown in our framework in Figure 1.

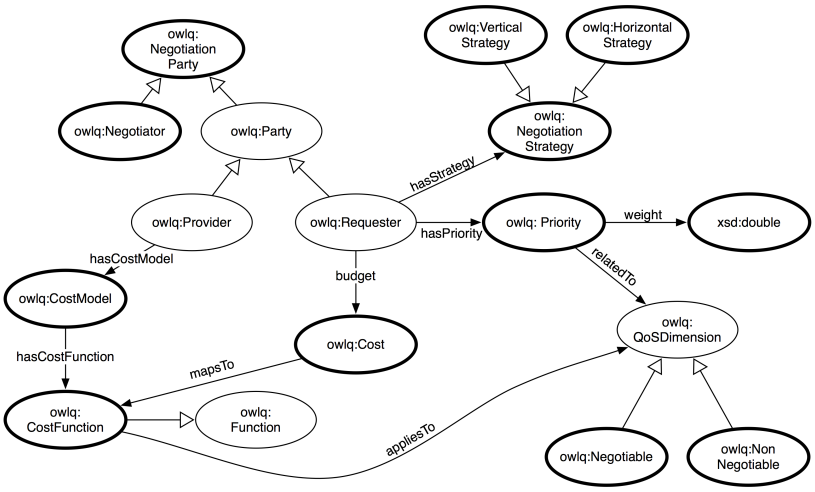

Fig. 4. OWL-Q Extensions

4.1 Quality-related extensions to OWL-Q

We introduce quality-related extensions to cope with two fundamental issues raised by the need to support QoS negotiation through an ontology:

- not all QoS dimensions, either technical or domain dependent, are negotiable. A QoS dimension is *Negotiable* when its value can be set by the service provider at runtime, i.e., when the service is invoked. *Non-negotiable* QoS dimensions are the ones for which the value cannot be set at runtime by the service provider. For instance, when a service is invoked, its reputation is fixed, regardless of the technique adopted for assessing reputation. Moreover, the service provider can always decide whether to allow or not the negotiation of a specific quality dimension. In our SMS service running example, *response time* can be considered as negotiable, since the response time value can be altered by the provider at execution time according to the customer requirements. However, in case the provider's provisioning infrastructure does not allow such an adaptation of response time, our ontology allows the provider to declare the response time as non negotiable;
- In order to automatically negotiate the QoS of a service [3], we also need to define a total ordering among admissible values identified for each QoS dimension. This ordering is established by the communities of domain experts that define the quality documents associated to a category of WSs.

4.2 Negotiation-specific extensions to OWL-Q

Negotiation-specific extensions concern the concepts and properties, besides QoS definition, required to establish a negotiation framework. In particular, we identify the following concepts.

- *Negotiator Actor*. Usually, the participants involved in WS QoS negotiation are the service provider and the service consumer. However, a more flexible approach should consider that the execution of a negotiation might be delegated to a trusted third party, such as, for instance, an ad-hoc agent explicitly designed to negotiate on the behalf of the service provider or the service customer. Introducing the negotiation actor also enables our framework to accommodate other negotiation protocols, such as, for instance, single-text mediated negotiations, which require the existence of a third party trusted by all the negotiation participants;
- *Negotiation Strategy* As previously introduced, our semantic-aware negotiation framework relies on the assumption of having parameterized negotiation strategies, i.e., negotiation strategies that are fully determined when a negotiator actor specifies the values of a set of parameters, such as the initial offer and the degree of concession over time to the counterpart.
- *Cost model* From the provider's point of view, the negotiation often relies on a cost model, i.e., parameterized functions that are used to evaluate the cost sustained by the service provider for giving a certain level of quality in

his service offers. Usually, WS QoS costs model are additive, that is, the cost of a service offer is given by the costs associated by the cost model to each single QoS value that appears in the offer.

4.3 Usage example of reasoning in negotiation

Derivation of new knowledge is actually the strongest point in using rules with ontologies. New knowledge may come with different forms like equivalence of quality dimensions, matchmaking a QoS offer with a demand, producing the price of a specific QoS level for a provider, etc. In fact, the whole discovery and negotiation algorithms may be written in the form of a modular set of rules so that only specific functions need to be really implemented in places where mathematical tools are required.

By using the application domain of SMS Monitoring we are now going to show a small example of how reasoning can support the negotiation process. Assume that a WS provider advertises that his WS has *availability* in the range of $[0.9, 0.99]$ (value type is $[0.0, 1.0]$), *response time* in the range of $[0.5, 2.0]$ seconds (value type is $(0.0, 2.0]$) and $coverage = \{Orange, Verizon, Cingular\}$. In addition, assume that cost model of the WS provider is defined by the formula:$price = cost_{avail} + cost_{resp} + cost_{cov}$ dollars, where $cost_{avail} = avail * 3$, $cost_{resp} = 3 * \frac{2-resp}{1.5}$ and $cost_{cov} = |coverage_{ad} \cap coverage_{req}|$. Further, assume that there is a WS requester that requests a WS having *availability* in $[95, 99]\%$ (value type is $[0, 100]$), *response time* in $[100, 1000]$ milliseconds (value type is $(0, 2000]$) and $coverage = \{Orange, Verizon\}$. Finally, assume that the WS requester has budget 7$.

Now consider that the WS's capabilities ad and the WS requester's requirements req have been submitted to a negotiation broker by using our proposed semantic QoS model. This broker uses the following rules (in abstract form) in order to infer if the QoS offer and demand are compatible for negotiation:

$$matches\,(ad, req) \Leftarrow \forall qdi_1 \in req\ \exists qdi_2 \in ad \text{ s.t } match\,(qdi_1, qdi_2) \tag{1}$$

$$match\,(qdi_1, qdi_2) \Leftarrow equiv\,(qdi_1, qdi_2) \wedge c\_values\,(qdi_1, qdi_2) \tag{2}$$

$$c\_values\,(qdi_1, qdi_2) \Leftarrow \exists v_1 \in qdi_1.values \wedge \exists v_2 \in dqi_2.values \text{ s.t } utf_{qdi_1.unit \mapsto qdi_2.unit}\,(v_1) = v_2 \tag{3}$$

$$compatible\,(ad, req) \Leftarrow matches\,(ad, req) \wedge req.Bugdet \geq MinCost\,(ad, req) \tag{4}$$

The first rule expresses that the QoS offer matches with the QoS demand when for each quality dimension (qd) of the demand there is a corresponding matching qd of the offer. The second rule expresses that two qds are matched if they are equivalent and they have a common value in their corresponding admissible values. Equivalence of qds is actually reduced to equivalency of their metrics. Kritikos and Plexousakis in [4] propose a semantic QoS metric matching algorithm in which two simple metrics are equivalent if they have compatible value types and units. In our example and considering this algorithm, it is easy to

see that the availability and response time of the ad and req specs are equivalent. More details of how this equivalence is inferred can be found in [4]. The third rule expresses that two qds have common values if there exists a value included in the admissible value type of the first qd that can be transformed to a value included in the admissible value type of the second qd by using a utility transformation function (utf). In our example, value 1000 of req's response time admissible value type is transformed to value 1.0 which is included in the admissible value type of ad's response time by using the utf: $utf_{ms \mapsto s}(x) = \frac{x}{1000}$. It is easy to see that this is also the case for the availability qd (the case of coverage qd is trivial). Thus, based on the first three rules, it is easy to infer that the QoS offer and demand of our example are matched.

The fourth and final rule infers that a QoS offer ad is compatible to a QoS demand req if they are matched and the requester's budget is less or equal to the minimum cost of WS. Rule $MinCost$ can be a user function that transforms a the offer and the demand to an optimization problem in order to find the minimum cost of the WS that respects the constraints of the demand. Based on our example, the smallest common value of availability is 0.95 and that of response time is 1.0. So the minimum cost of the WS will be: $2 + 3 * 0.95 + 3 * (2 - 1)/1.5 = 6.85$ which is less than the requester's budget. Thus, finally, the QoS offer and the demand are compatible as the fourth rule is satisfied.

5 Related work

The need for automated negotiation of quality SLAs or, more generally, contracts, is being addressed as one of the main driver for the adoption of service based systems in real-world scenarios [2].

As already remarked, we argue that giving formal semantics to the description of QoS and of the elements that compose the negotiation framework represents a tenet for modern service based systems. Semantic-based negotiation mechanisms and protocols have been often inspired by the agent community literature. In [8] a survey on approaches for multi-attribute negotiation in Artificial Intelligence is introduced.

Focusing on the SW community, Chiu et al. [9] discuss how ontology can be exploited for supporting negotiation. In particular, the authors highlight how shared and agreed ontologies provide common definitions of the terms to be used in the subsequent negotiation process. In addition, they propose a methodology to enhance the completeness of issues in requirements elicitation. Lamparter et al. [10] introduce a model for specifying policies for automatic negotiation of WSs. In this case, the starting point is the upper ontology DOLCE [11]. On this basis, this work proposes a policy ontology that also includes the user preferences specifications and an additional ontology for expressing contracts.

About the use of ontologies for specifying the agreement among parties, Oldham et al. [12] present reasoning methods for the components of an agreement which must be compatible for quality matches. This approach is based on WS-Agreement and takes advantage of SW methods to achieve rich and accurate

matches. With the same goal Uszok et al. [13] have developed KAoS policy ontology that allows for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex organizational structures.

Other approaches [14, 15] focus only on semantically describing the QoS capabilities and requirements of WSs for the purposes of WS discovery. These approaches differ in their expressivity, connection to a functional WS description language and their usage/support by a framework providing WS discovery capabilities.

onQoS-QL [16] and OWL-Q [4] are the most rich semantic languages for QoS-based WS description adopting all the requirements expressed in [5]. They are also supported by WS discovery frameworks. However, the main drawback of onQoS-QL is that its expressivity concerning metric functions, directives and QoS constraints is limited. In addition, the QoS profile of a WS contains only QoS metrics and not QoS constraints on these metrics.

Based on the above analysis, it is clear that each research approach, independently of its efficiencies and deficiencies, describes QoS for WSs focusing on supporting either WS discovery or WS negotiation. So in order to support the latter two processes, one has to follow two alternative directions: either use the best approach in each process and provide a mapping between them or create a new uniform approach by extending an existing one. This paper follows the second direction as this direction seems more promising and efficient and extends one of the best approaches in QoS-based WS description and discovery in order to further support WS negotiation.

6 Conclusion

Although the need for a semantic-aware description of the quality of WS is now recognized, most of the current work mainly focus on the definition of quality attributes. In this paper, we have proposed to go one step ahead discussing the need for a semantic-oriented negotiation in SOA. With this aim, starting from an existing QoS ontology, i.e., OWL-Q, we have identified which are the missing elements and we have proposed possible extensions that allow to deal with the negotiation process.

As our work is preliminary, it can be further extended in terms of concepts and mechanisms for reasoning on the quality attributes to assist and automate the algorithm for enacting the negotiation.

References

1. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference model for service oriented architecture 1.0. Technical report, OASIS (2006)
2. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the art and research challenges. IEEE Computer **11** (2007) 38–45

3. Cappiello, C., Comuzzi, M., Plebani, P.: On automated generation of web service level agreements. In Krogstie, J., Opdahl, A.L., Sindre, G., eds.: CAiSE. Volume 4495 of Lecture Notes in Computer Science., Springer (2007) 264–278
4. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: ECOWS '06: Proceedings of the 4th European Conference on Web Services. (2006) 265–274
5. Kritikos, K., Plexousakis, D.: Requirements for qos-based web service description and discovery. In: Proc. 31st Annual Int. Computer Software and Applications Conference (COMPSAC 2007). (2007) 467–472
6. Sycara, K., et al.: OWL-S 1.0 Release. OWL-S Coalition, http://www.daml.org/services/owl-s/1.0/. (2003)
7. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Int. Journal of Robotics and Autonomous Systems **24**(3-4) (1998) 159–182
8. Lai, G., Li, C., Sycara, K., Giampapa, J.A.: Literature review on multi-attribute negotiations. Technical Report CMU-RI-TR-04-66, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2004)
9. Chiu, D.K.W., Cheung, S.C., Hung, P.C.K., fung Leung, H.: Facilitating e-negotiation processes with semantic web technologies. In: Proc. 38th Annual Hawaii International Conference on System Sciences. (2005)
10. Lamparter, S., Luckner, S., Mutschler, S.: Formal specification of web service contracts for automated contracting and monitoring. In: Proc. 40th Annual Hawaii International Conference on System Sciences. (2007)
11. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: (Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology)
12. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic ws-agreement partner selection. In: WWW 2006. (2006) 697–706
13. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In: Proc. 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks. (2003)
14. Zhou, C., Chia, L.T., Lee, B.S.: Daml-qos ontology for web services. In: Proc. IEEE Int. Conf. on Web Services (ICWS'04). (2004) 472–479
15. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing **8**(5) (2004) 84–93
16. Giallonardo, E., Zimeo, E.: More semantics in qos matching. In: International Conference on Service-Oriented Computing and Applications. (2007) 163–171