Requirements for QoS-based Web Service Description and Discovery

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

Abstract

The goal of Service Oriented Architectures is to enable the creation of business applications through the automatic discovery and composition of independently developed and deployed (Web) services. Automatic discovery of Web Services (WSs) can be achieved by incorporating semantics into a richer WS description model (WSDM) and the use of Semantic Web (SW) technologies in the WS matchmaking and selection models. A sufficiently rich WSDM should encompass not only functional but also non-functional aspects like Quality of Service (QoS). QoS is a set of performance attributes that has a substantial impact on WS requesters' expectations. Thus, it can be used as a discriminating factor of functionally-equivalent WSs. The focus of this paper is twofold: to analyze the requirements of a semantically rich QoS-based WSDM and to provide SW and constrained-based mechanisms for enriching syntactic QoS-based WS Discovery (WSDi) algorithms. In addition, a roadmap of extending WS standard techniques for realizing semantic, functional and QoS-based WSDi is presented.

1. Introduction

WSs are modular, self-describing and loosely-coupled software applications that can be advertised, located and used across the Internet using a set of standards such as SOAP, WSDL and UDDI. They can be dynamically discovered and integrated at runtime in order to develop and deploy business applications. However, the standard WS techniques (such as WSDL and UDDI) fail to realize dynamic WSDi, as they rely on static descriptions of service interfaces and other non-functional service attributes for publishing and finding WSs. As a result, syntactic WSDi mechanisms return results with low precision and recall. In addition, no means are provided in order to select among multiple functionally equivalent WSs.

The key to dynamicity or automation of WSDi is the incorporation of semantics into a richer WSDM. Ontologies should be used in order to provide formal meaning to every term of the WSDM and in order to utilize the reasoning mechanisms of the SW. Reasoning provides discovery results with high precision and recall, thus leading to automation. A rich WSDM should incorporate both functional and non-functional aspects. Both of these aspects should unfold in great detail. In this way, more detailed WS specifications will result in more constrained WSDi result sets.

The functional part of WS description actually contains specifications of Inputs, Outputs, Preconditions and Effects. The non-functional part comprises the following: a subset of all possible WS non-functional properties is QoS. Usually, QoS is used as a synonym of performance. In our view, QoS is a broader term. QoS (for WS) is a set of WS properties that have an impact on the quality of the service offered by the WS, where quality is synonymous with meeting specifications. Each QoS property is measured by a QoS metric. A QoS metric describes in detail how the measurement is conducted, by whom, in what units and value types and when it is measured. In other words, a QoS metric is a realization of a QoS property. QoS WS specifications (advertisements or requests) are actually constraints over these QoS metrics.

While a lot of research has dealt with the semantic, functional WS description, little has been done for the non-functional part. Most of the research is concentrated on syntactic non-functional WS descriptions [6, 2, 3, 9, 5, 13, 15]. Additionally, the semantic research efforts lack a rich QoS-based WSDM [19]. The purpose of this paper is to analyze the requirements for a rich semantic QoS-based WSDM. In addition, semantic mapping algorithms and extensions to the most prominent WSDi algorithms [5, 19] are provided in order to equip WS requesters with semantic WSDi tools, which will produce results with high precision and recall. Last but not least, a roadmap is supplied that provides guidelines on how to extend the standard WS mechanisms in order to incorporate and use the semantic QoS-based WSDMs and discovery tools. The ultimate goal is to provide a

complete semantic framework for automating WSDi.

2. Requirements for QoS-based WS Description and OWL-Q

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 WSDM:

Extensible and formal semantic QoS model: In the presence of multiple WSs with overlapping or identical functionality, WS requesters need objective QoS criteria to distinguish WSs. However, it is not practical to come up with a standard QoS model that can be used for all WSs in all domains. This is because QoS is a broad concept that encompasses a number of non-functional properties such as privacy, reputation and usability. Moreover, when evaluating WS QoS, domain specific criteria must be taken into consideration. For example, in the domain of phone service provisioning, the penalty rate for early termination of a contract and compensation for non-service, offered in the service level agreement are important QoS criteria in that domain. Therefore, an extensible QoS model must be proposed that includes both the generic and domain specific criteria. In addition, new domain specific criteria should be added and used to evaluate QoS without changing the underlying computation (i.e. matchmaking and ranking) model. Last but not least, the semantics of QoS concepts must be described in order to have terms/concepts with specific meaning for both WS requesters and providers. In this way, QoS attributes like "application availability", which may have different meanings if not formally defined, will have a specific meaning in QoS description. The solution to the above problems is the use of ontologies. Ontologies provide a formal, syntactic, and semantic description model of concepts, properties and relationships between concepts. They give meaning to concepts 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 them. In addition, SW techniques can be used for reasoning about concepts or for mapping between ontologies. These techniques can lead to syntactic and semantic matching of ontological concepts and enforcement of class and property constraints (e.g. type checking, cardinality constraints, etc.). Therefore, by providing semantic description of concepts and by supporting reasoning mechanisms, ontologies cater for better WSDi with high precision and recall. Last but not least, ontologies can help specialized brokers in performing complex reasoning tasks like WSDi or mediation.

Standards compliance: It is important for the QoS-based WSDM to comply with already widely-accepted standards. In this way, it will be easily adopted by the research community. In addition, it will use all freely-available tools related to these standards for its development.

Syntactical separation of QoS-based and functional parts of service specification: QoS specifications should be syntactically separated from other parts of service specifications, such as interface definitions. This separation allows us to specify different QoS properties for different implementations of the same interface. Moreover, while functional constraints rarely change during runtime, QoS constraints can change during runtime. So the separation of service offerings from WSDL descriptions permits service offerings to be deactivated, reactivated, created, or deleted dynamically without any modification of the underlying WSDL file. Last, an offer could be referenced from multiple WSDL files and thus be reused for different services.

Support refinement of QoS specifications and their constructs: As previously described, syntactical separation provides reusability. But except from reusability, another form of extensibility is equally important. QoS specifications should not only be reused but also refined. This means that we can create a new WS QoS offering by referencing an older one and by adding constraints like refinement of an older QoS restriction or creation of a new one. In addition, templates of QoS offerings can be created and appropriately extended for every domain.

Allow both provider and requester QoS specification: It should be possible to specify both the QoS properties that clients require and the QoS properties that services provide. Moreover, these two aspects should be specified separately so that a client-server relationship has two QoS specifications: a specification that captures the client's requirements and a specification that captures service provisioning. This separation allows us to specify the QoS characteristics of a component, the QoS properties that it provides and requires, without specifying the interconnection of components. The separation is essential if we want to specify the QoS characteristics of components that are reused in many different contexts. Finally, QoS demands and offers should be specified in a symmetric way. Symmetric approaches usually achieve a greater deal of expressiveness to specify quality-of-service, since there is usually no restriction on the number of involved parameters or type of operators, so that non-linear or more complex expressions are allowed.

Allow fine-grained QoS specification: It should be possible to specify QoS properties/metrics at a fine-grained level. As an example, performance characteristics are commonly specified for individual operations. A QoS model must allow QoS specifications for interfaces, operations, attributes, operation parameters, and operation results. Generally speaking, any service object can have QoS attributes/metrics (e.g., elements defined in WSFL).

Extensible and formal QoS metrics model: For each domain, the attributes 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. Each attribute is measured with the help of a metric. Each attribute/metric has the following aspects:

- The value set for the metric (and its allowed value range).
- The domains that this attribute belongs to. For instance, is it a cross-domain attribute or an attribute specific for a domain?
- The weight of the metric relative to its domain and user preferences. This weight can also help in calculating the rank of a QoS offering.
- The characteristic of the function from metric values to overall QoS values. For instance, some attributes such as price are monotonic, at least in typical business scenarios.
- The temporal characteristic of the metric value. Matrics may have decaying values where the decay function can vary from exponential to a step function.
- There must be a description (mathematical or otherwise formal) of how a QoS metric's value of a complex WS can be derived from the corresponding QoS metrics' values of the individual WSs that constitute the complex one. For example, the execution time Tc of a complex WS C, which is defined as a sequence of two Web Services A and B, can be computed as the sum Ta + Tb of the execution times of the two individual Web Services. This description is essential for the automated estimation of the values of QoS metrics for a complex Web Service that is composed of other Web Services and individual operations. So this description is needed for automating the QoS analysis process, a prerequisite for a successful QoS-based WSDi. In addition, it helps automating the WS composition process and delaying individual WS selection as late as possible (i.e., at runtime).
- In [14], the authors argue for the need for several ontologies that would be used in the formal representation of QoS and other constraints. These ontologies include: ontology of measurement units, ontology of currency units, ontology of measured properties and ontology of measurement methods. So these ontologies must also be developed.

Allow classes of service specification: *Class of Service* [15] means the discrete variation of the complete service and QoS provided by one WS. It makes sense to discuss Class of Service at the level of WSs and not at the level of constraints or guarantees that are part of the overall service and QoS. Classes of service can differ in usage privileges, service priorities, response time guarantees, etc. The concept of classes of service also supports different capabilities, rights and needs of potential customers of the WS, including power and type of the devices they execute on. Furthermore, different classes of service may imply different utilization of the underlying hardware and software resources and, consequently, have different prices. Additionally, different classes of service can be used for different payment models. The issues of QoS and balancing of limited underlying resources are particularly motivating for having multiple classes of service for Web Services. If the underlying resources were unlimited, all consumers would always get the highest possible QoS. Unfortunately, this is not the case, so it is suitable to provide different QoS to different classes of consumer. Providers of Web Services want to achieve maximal monetary gain with optimal utilization of resources. Providing different classes of service and their balancing helps in achieving this goal because of the flexibility to accommodate several classes of consumer. On the other hand, consumers of such Web Services can better select service and QoS they need and are willing to pay for, while minimizing their price/performance ratio.

Based on the above requirements, we have developed an upper ontology for QoS-based WS Description, which is called OWL-Q [4]. This ontology describes in a syntactic and semantic way all possible parts of QoS metrics and of QoS constraints. It is an ontological description carefully designed into several facets that can easily be extended and enriched. This ontological description also complements OWL-S [12], a W3C submission for Semantic WS description, by subsuming OWL-S concepts or relating them to QoS concepts. Based on our upper ontology, we also propose the development of a mid-level ontology that will define all domain independent QoS metrics. Finally, based on the upper and mid-level ontology, new QoS metrics can be defined.

3. Requirements for QoS-based WS Discovery

The (QoS-based) WSDi process is decomposed into two sub-processes: matchmaking and selection. In the first subprocess, the (QoS-based) WS descriptions (advertisements and request) are matched and the outcome is a list of WS advertisements that completely satisfy the constraints of the request. In the second sub-process, this output list is sorted based on given weights of QoS metrics of the WS requester. Both of these sub-processes depend on the (QoS-based) WS descriptions.

Suppose, now, that a WS provider and requester provide QoS-based descriptions that include differently-defined metrics. That is their descriptions refer to metrics defined in different ontologies. It may also be the case that the units or value types used are different. This problem is solved by in-

troducing mapping algorithms that try to map or relate concepts of two or more ontologies. However, although there are general mapping algorithms, we believe that a specialized mapping algorithm must be used in the QoS metrics case because it will take advantage of the semantics of the QoS context. Thus, we have developed a QoS metric matching (QMM) algorithm [4], which is not evaluated yet.

The QMM/mapping algorithm is a tool that transforms syntactic QoS-based WS matchmaking algorithms to semantic ones as it enables them to semantically compare QoS metrics. One of the most prominent syntactic QoS-based WS matchmaking approaches [5] transforms a QoS WS description into a Constraint Satisfaction Problem (CSP) [16]. Before matchmaking, the CSP of QoS-based WS description is checked for *consistency* (if it has any solution). Matchmaking is performed according to the concept of *conformance* (if every solution of the offer is a solution to the demand). The algorithm returns two lists: a list of conformant advertisements and a list of non-conformant ones. Next, we show how to extend a QoS-based WS matchmaking algorithm like the above with the help of OWL-Q (or any other QoS-based WS description language) and the QMM algorithm by pursuing the following 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 equivalent 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 (from the provider's unit to the requester's) is performed.

2. We check all CSPs. If a CSP of an advertisement does not contain all the variables of the CSP of the request, it is considered as partial match.

3. We solve all advertisement CSPs and the CSP of the request with a known and efficient CSP engine.

4. Now for every solution of the CSP of the request, we check if it is contained in the solution space of the CSP of an advertisement. If this is not the case, the advertisement is considered as *fail match*. Otherwise, if the advertisement has more variables than the request it is considered as *super match*. If it has the same amount of variables, the advertisement is considered as *exact match*. *Partial match* advertisements must be handled in a different way: The solution space of both the CSP of the partial match advertisements and the CSP of the request is constrained only to the variables that are common. Then, it is checked if every partial solution of the CSP of the request is contained in the partial solution space of the CSP of an advertisement. If this not the case, then the advertisement is moved to the fail match category.

The above algorithm produces four types of results. *Super matches* are the best type as they include QoS advertisements that not only satisfy QoS request constraints but also provide for more QoS constraints. *Exact matches* are also preferable (but not as strong as the super matches) as they completely satisfy all the QoS request constraints. *Partial matches* are the next preferable as they contain those QoS advertisements that satisfy the QoS request only for the QoS metrics that appear in both types of specifications. *Fail matches* are useful as a result because sometimes all requester's QoS constraints are not satisfied although all requester's QoS metrics also appear at the provider's constraints. In this case, the requester can select the best failed result or perform a looser query.

In practice, it may be the case that the results of the QoS-based WS matchmaking sub-process always belong to the fail match category. The reason is obvious: not all of the QoS constraints of the WS requester could possibly be satisfied by any QoS offer. There is a solution to overcome this problem.

The requester's query should be refined. This can be done with the characterization of the constraints as hard and soft [10]. Hard constraints are obligatory and are dealt as before i.e. they must definitely be satisfied by QoS offers. Soft constraints are optional and can be used not only for better characterization of the matchmaking results but also for a first-time ordering of all the results. Now, the characterization of the matchmaking results is the following:

- *subsume matches* are the QoS offers that not only satisfy the requester's QoS hard and soft constraints but also impose more constraints;

- *perfect matches* are the QoS offers that satisfy all the requester's QoS hard and soft constraints;

- *partial matches* are the QoS offers that satisfy all the hard constraints and zero or more soft constraints of the requester;

- *fail matches* do not satisfy the hard and soft constraint of the requester.

The *subsume matches* are the best and can be ordered if they are applied to the selection sub-process. The *perfect matches* are the next best and they can also be ordered during the selection process. The *partial matches* are satisfactory as they satisfy all the hard constraints of the requester. This type of results can be ordered based on the number of soft constraints satisfied. If the situation is not clear enough, then the results are further ordered based on the selection sub-process. The purpose of the introduction of this category is clear. If we do not have subsumed or perfect matches, then we choose the QoS offer that satisfies all of the hard constraints and most of the soft constraints of the requester. The *fail matches* are not satisfactory at all. If we only have failed results, then either the user applies the procedure described in the previously described first way or he must further alter the characterization of his constraints (transform more hard constraints into soft ones).

Due to space limitations, we cannot show how to extend syntactic QoS-based WS selection algorithms. However, the extension procedure follows easily: we execute the first step of the above matchmaking extension algorithm. In this way, we guarantee that: a) all different metrics do not match with each other; b) all expressions are transformed appropriately if two matching metrics are found.

4. Roadmap

In this section, we comment on how to extend the current WS standard technology to enable it to perform semantic QoS-based WS description and discovery (WSDD). Additionally, there will be a discussion on what are the obligations of the parties involved in the WSDi Architecture (i.e., providers, requesters and registry).

First, two standard WS description languages are examined. The first one is WSDL. This language is just an interface description language specialized to specify how to call a WS. This language is not used for discovery purposes. However, in [11] an extension of WSDL called WSDL-S is proposed. Additional constructs and included terms referring to ontological concepts are added, independently of the ontology language used. Moreover, a semantic framework is proposed that maps WSDL-S to UDDI and enhances the facilities of UDDI registries to support semantic discovery queries. This framework is included in the METEOR-S framework [8]. The latter framework is a complete framework supporting semantic functional and non-functional discovery and composition of WSs. Unfortunately, the discovery process does not include semantic QMM algorithms and only deals with simple inequality constraints while the composition process takes into account the composed values of only three QoS attributes.

The second language under inspection is UDDI. This language is a syntactic structural language for WSDD. It does not include non-functional descriptions of WSs. In [9], an extension of UDDI is proposed (with additional elements and tModel concepts mapped to QoS attributes) that enables the QoS-based description and discovery of WSs. However, this extension is not semantic and only simple constraints are expressed and enforced.

In our view, to enable semantic QoS-based WS description, an ontology language like OWL-Q must be proposed, developed and supported, which will enable the use of the semantic facilities of reasoning and concept mapping. If this language is mapped to UDDI and appropriate semantic tools (UDDI mapper, QoS metric matcher, semantic discovery engine) are available, then UDDI registries will be extended to support QoS-based WSDD. For the functional WS description, two equivalent languages are available: OWL-S and WSMO [1], each one having a supported functional WSDi framework. OWL-S can be mapped to UDDI [7] and there is already a framework supporting this mapping. So, a semantic QoS-based WS description language like OWL-Q would be ideal for QoS-based WS description as it extends and complements OWL-S, which is mapped functionally to UDDI. Only the QoS-based mapping to UDDI (actually to the [9] UDDI extension) is pending and must be provided by our semantic QoS-based WS framework. To sum up, we believe that OWL-S, OWL-Q and a supporting semantic framework that maps these languages to UDDI (and its extension) is ideal for successful QoS-based WS description. If this framework is enriched with the semantic QMM and QoS-based WSDi algorithms, then semantic QoS-based WSDi can be realized.

Assuming that a complete functional and QoS-based WSDi framework is available, the new obligations of WS providers, requesters and registries are analyzed. WS providers must use simulation and other performance analysis tools in order to discover the QoS constraints of each class of service. Then they should use and feed their QoS offers to a tool that will produce OWL-Q (or other language) offers associated to the WS functional offer (of OWL-S or WSMO). This tool must be able to load and associate many ontologies (QoS attribute, metric, unit, measurement method). Moreover, they should publish these offers to registries where they have already published the associated functional descriptions of their WS implementations. A prerequisite of the above is the providers' frankness about their offerings.

WS requesters must be able to understand the needs of their applications and with the help of appropriate tools translate them to functional and QoS-based requests. They should comprehend that if they specify many hard constraints, then they may not get any discovery result or, on the other hand, if they are vague, then they will get too many

results. Thus they must be wise to express a specific number of constraints (hard or soft) and they should also provide weights and utility functions for every QoS metric that interests them. In the end, they should use appropriate tools to produce OWL-Q files and send them as queries to WS registries.

WS registries, if enriched semantically with a complete semantic framework, must provide an API for the semantic advertisement and enquiry of the functional and QoS-based parts of WSs. They should also implement APIs for the management of QoS and functional offers (activation, deactivation, creation, deletion). They should carefully associate functional with corresponding QoS offers (of the same WS) so as to avoid cases where a deletion of a functional offer does not cause the deletion of the corresponding QoS offers. Moreover, a leasing mechanism must be available that will deactivate QoS offers when they expire and delete them after a specific time period has passed without being renewed/reactivated. Last but not least, WS registries should interact with WS reputation and requester-feedback registries in order to update their WS reputation and appropriate QoS offer metric values entries.

5. Conclusion

In this paper, an analysis of the requirements for semantic QoS-based WSDD were provided. Additionally, a proposal of how to extend the current standard WS technology to incorporate and use an appropriate semantic QoS-based WS framework was supplied. Last but not least, the responsibilities of the participating entities in the WSDi architecture were analyzed in sight of the incorporation of the referred QoS-based framework.

We're currently completing the specification of OWL-Q that has already been presented in [4] and the implementation of our semantic QMM algorithm. An extension to this work would realize QoS-based WS composition by enforcing global QoS constraints and using for every QoS metric, metric evaluation functions imposed on any possible workflow (WS) composition construct to produce global QoS metric values [17] through construct reduction.

References

[1] D. F. et. al. *Web Service Modelling Ontology*. ESSI WSMO working group, http://www.wsmo.org, 2004.

[2] L. Jin, V. Machiraju, and A. Sahai. Analysis on service level agreement of web services. Technical Report HPL-2002-180, Software Technology Laboratories, HP Laboratories, 2002.

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

[4] K. Kritikos and D. Plexousakis. Semantic qos metric matching. In *ECOWS*, pages 265–274. IEEE Computer Society, 2006.

[5] 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.

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

[7] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK, 2002. Springer-Verlag.

[8] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma. Meteor-s web service annotation framework. In S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, editors, *WWW*, pages 553–562. ACM, 2004.

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

[10] F. Rossi. Preference reasoning. In P. van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 9–12. Springer, 2005.

[11] K. Sivashanmugam, K. Verma, A. P. Sheth, and J. A. Miller. Adding semantics to web services standards. In Zhang [18], pages 395–401.

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

[13] 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.

[14] 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.

[15] V. Tosic, B. Pagurek, and K. Patel. Wsol - a language for the formal specification of classes of service for web services. In Zhang [18], pages 375–381.

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

[17] T. Yu and K.-J. Lin. Service selection algorithms for composing complex services with multiple qos constraints. In B. Benatallah, F. Casati, and P. Traverso, editors, *ICSOC*, volume 3826 of *Lecture Notes in Computer Science*, pages 130–143. Springer, 2005.

[18] L.-J. Zhang, editor. *Proceedings of the International Conference on Web Services, ICWS '03, June 23 - 26, 2003, Las Vegas, Nevada, USA*. CSREA Press, 2003.

[19] 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.