Semantic Discovery and Selection: A QoS-Aware, Hybrid Model

José María García, David Ruiz, Antonio Ruiz-Cortés, and Manuel Resinas
Dept. Lenguajes y Sistemas Informáticos, University of Sevilla
E.T.S. Ingeniería Informática, Av. Reina Mercedes s/n, 41012 Sevilla, Spain

Abstract – *Most Semantic Web Services discovery approaches are based on Description Logics, allowing a limited expressiveness when describing Quality-of-Service preferences. Furthermore, DLs is not suited to perform selection tasks, because these are modeled as optimization problems. In this work, we present a hybrid discovery and selection model for Semantic Web Services that takes care of* QoS *preferences. Our approach splits the whole process into two stages, using the most suited engine in each one, depending on its search nature. In order to perform* QoS-*aware discovery and selection, user preferences have to be semantically described as the rest of the service description. Our model provides an ontology for user preferences, so instances can be transformed into optimization problems that can be solved by using the most suited engine.*

Keywords: Service Discovery, Quality-of-Service, Semantic Web Services, Ontology Languages, QoS-Aware Selection.

1 Introduction

Most approaches on automatic discovery of Semantic Web Services (SWS) use Description Logics (DLs) reasoners to perform the matching [8, 13, 16, 20, 27, 28]. These approaches have limitations regarding with the expressiveness of searches, especially when there are Quality-of-Service (QoS) conditions integrated within user preferences. For instance, a condition like "find a service which $availability \geq 0.9$, where $availability = MTTF/(MTTF + MTTR)$"[1] can not be expressed in DLs. However, there are some proposals that extend DLs with concrete domains [10], though they still have limitations on expressing complex conditions [1, 15]. Furthermore, selection tasks lead to optimization problems, so DLs are not suited in that stage of the whole discovery process.

[1] $MTTF$ stands for "Mean Time To Failure", while $MTTR$ stands for "Mean Time To Repair". Both of them are QoS parameters often used to define service availability.

Optimization problems can be handled by solvers based on different techniques, like Linear Programming (LP), Constraint Programming (CP), or Dynamic Programming (DP), among others. Thus, it is possible to split discovery and selection tasks in terms of functional and QoS preferences, so the former task can be performed by DLs reasoners, while the latter can be performed by solvers, taking a hybrid approach.

Our proposal presents a hybrid architecture to discover SWS. Thus, our solution splits that process into two stages: (1) functional discovery, which is usually performed by DLs reasoners using functional preferences; and (2) QoS-driven selection, where a solver obtain the best service in terms of QoS preferences that define an optimality criterion. Both *user* preferences have to be semantically described at the same level, so different, but possibly equivalent, service descriptions can be matched.

This model does not restrict the concrete technique to be used in any of its stages. Our proposal provides a user preference ontology that can be linked to any current SWS framework. Moreover, it uses a very expressive solution to define these preferences, i.e. utility functions and weights. Finally, our hybrid architecture is extensible, so it is possible to add more stages to the process, provided that the descriptions needed by that stage are described at the same semantic level than the rest of the service description.

The rest of the paper is structured as follows. In Sec. 2 we analyze current approaches on discovering and selecting SWS. Then, in Sec. 3 we present our hybrid discovery and selection model, explaining the proposed architecture, an ontology of user preferences using utility functions, and how to use that ontology to perform QoS-aware semantic selection. Finally, in Sec. 4 we sum up our contributions, and discuss our conclusions.

2 Related work

In this Section, we discuss related work on discovery and selection of SWS, describing the different approaches and analyzing their suitability to handle QoS-aware user preferences. Firstly, we review discovery-related proposals, which use DLs to match services with functional preferences, and then we present different approaches on selecting services by means of QoS-aware user preferences.

Concerning these preferences, in the following, service descriptions are considered as their *provider preferences* (what a provider offers and possibly its requirements to the user), and user requirements are referenced as *user preferences*. Nevertheless, each preference can be broken in two main parts: (1) functional preferences, that refer to what a service has to do; and (2) QoS preferences, that are used to rank services in terms of one or more QoS parameters.

2.1 Discovering SWS

In the context of DAML-S (the precursor of OWL-S [17]), Sycara *et al.* show how semantic information allows automatic discovery, invocation and composition of Web Services [28]. They provide an early integration of semantic information in a UDDI registry, and propose a matchmaking architecture. It is based on a previous work by Paolucci *et al.*, where they define the matching engine used [21]. This engine matches a demand and an offer when this offer describes a service which is “sufficiently similar” to the demanded service, i.e. the offered service provides the functionality demanded in some degree. The problem is how to define that degree of similarity, and the concrete algorithm to match both service descriptions. They update their work to OWL-S in [29].

Furthermore, there are proposals that perform the matchmaking of SWS using DLs [8, 13, 16]. Particularly, González-Castillo *et al.* provide an actual matchmaking algorithm using the subsumption operator between DLs concepts describing demands and offers [8]. They use existing DLs reasoners, as RACER [9] or FaCT [11], to perform the matchmaking. On the other hand, Lutz and Sattler [16] do not provide an algorithm, but give the foundations to implement it using subsumption, like Li and Horrocks [13], who also give hints to implement a prototype using RACER.

These three works define different matching degrees as in [28], from exactly equivalents to disjoint, so they perform a selection. All of them perform this matching by comparing inputs and outputs. However, Benatallah *et al.* propose to use the degree of matching to select the best offer in [2], but it results to be a NP-hard problem, as in any optimization problem [4].

On the other hand, Benbernou and Hacid realise that some kinds of constraints are necessary to discover SWS, including QoS related ones, so they formally discuss the convenience of incorporating constraints in SWS discovery [3]. However, instead of using any existing SWS description framework, their proposal uses an *ad-hoc Services Description Language*, in order to be able to define complex constraints. In addition, the resolution algorithm uses constraint propagation and rewriting, but performed by a subsumption algorithm, instead of a CSP solver.

2.2 Selecting SWS

An early approach on modeling QoS in the context of SWS discovery are found in [23]. In this work, Ran presents a UDDI extension and a catalog of QoS parameters that can be included in UDDI descriptions. Discovery is performed using queries with functional requirements, as well as conditions on QoS. However, the actual discovery algorithm is not defined, and queries that use QoS parameters are not shown, so their expressiveness is unknown. Additionally, UDDI only supports a keyword based search, so no form of inference or flexible match can be performed [28]. Apart from that, the resultant services are not ranked, so the user have to perform different, successive queries, filtering the result set in order to find the best suited service.

Although their proposal is not semantically defined, Liu *et al.* present a QoS computation model including a selection algorithm [14], which is adopted in other approaches [22,30]. They propose an extensible QoS model that comprises both generic and domain specific criteria. Selection is performed using an algorithm based on matrices normalization, where services are ranked in terms of their QoS matrix description and a vector of relative weights between QoS parameters, which express user preferences.

Pathak *et al.* also model mappings between ontologies in [22]. They propose to use domain specific ontologies to define QoS preferences among users and providers. In their work, selection is done using matching degrees at a first stage. Then, QoS parameters values are collected in a quality matrix, which is used to calculate a fixed, weighted utility function for each offer. Finally, offers whose utility function is above a given threshold, are ranked by one QoS parameter to obtain the optimal offer.

Wang *et al.* provide an extension to WSMO ontology [24] to handle QoS parameters [30]. They define a QoS selection model and an algorithm based on a quality matrix that contains values of QoS parameters. The user preferences are described in terms of tendencies, i.e. a demand may prefer parameters to be as small as possible, as large as possible, or around a given value. Thus, in conjunction with weights, they rank services to select the best one within a given set.

Maximilien and Singh present a framework and a QoS ontology for dynamic selection in [19]. They use an agent-based approach where QoS are modeled via a three-layer ontology: an upper ontology which defines basic concepts associated with a quality parameter, a middle ontology which defines the most frequent QoS parameters and metrics, and a user-defined lower ontology that depends on the domain of the service. Although it constitutes a well-defined framework to semantically describe QoS and it is referenced by many authors [6, 12, 22], it is not aimed at semantically describing user preferences.

An extension to DAML-S to include QoS profiles is proposed in [32] by Zhou *et al.* This proposal only allows order conditions between QoS parameters, so it performs discovery and selection using DLs. The QoS ontology is simple and can be easily linked to the DAML-S service profile. However, its selection algorithm uses matching degrees to rank the resulting set of services, so user preferences can only be expressed as ordering relations, which are inherent to that selection algorithm.

Another DAML-based proposal is also presented in [26], where S. Bilgin and Singh provide a DAML-based query language, instead of just extending OWL-S. Using this *Semantic Web Services Query and Manipulation Language*, they advertise QoS attributes and perform the selection. The main drawbacks of this approach are the same as in [23], with limitations on the expressiveness of queries, due to the use of DAML as its foundation. Thus, user preferences can not be expressed in those queries, and are inherent to their selection algorithm, as in [32].

Dobson *et al.* presents QoSOnt in [6], which is an ontology that extends OWL-S to describe QoS attributes and metrics. However, they do not explicitly explain how to perform selection, and their proposal suffers from OWL limitations, so they have to use an ad-hoc XML language to allow custom data ranges. User preferences are modeled using the preferred tendency of metric values (e.g. the higher the best).

On the other hand, Zeng *et al.* show a basic QoS model to Web services composition in [31], although it can be applied to discovery and selection. They propose an algorithm based on utility functions, which are already defined for all the contemplated QoS parameters. The optimization is implemented using Integer Programming, providing weights to the different QoS parameters involved. The main drawbacks of this proposal are that it do not take semantics into account and that the utility functions are fixed, so the user can define its preferences only by means of weights.

Ruiz-Cortés *et al.* describe a QoS-aware discovery using Constraint Programming, where optimization is modeled as a Constraint Satisfaction Optimization Problem that minimize a weighted composition of utility functions, which are defined by the client using QoS parameters from a catalog [25]. As in [31], this proposal does not provide semantics, but user preferences, described by utility functions, can be defined by the user with high expressiveness.

An extension to [18] is presented in [12] by Kritikos and Plexousakis. They propose an ontology similar to the proposed by Maximilien and Singh [19], mixing offers and demands within an OWL-S description. Moreover, they present a matching algorithm to infer equivalences between different named QoS parameters that are semantically equivalent, although it is generally undecidable. Concerning discovery and selection, they use CSPs to perform the matchmaking of compatible offers, and then select the best service by means of a weighted composition of utility functions, which balance the worst and best scenarios to compute the utility value. However, these user preferences are not semantically defined in their QoS ontology.

2.3 Motivation

Several conclusions can be obtained from the analysis of the related work. The main ones are the following, which conform the motivation of this work.

1. Discovery proposals, all of them based on DLs, generally use matching degrees to select the best service [8, 13, 16, 28]. However, they do not support QoS preferences, so they can not perform any optimization based on preferences.

2. There are a few proposals that uses utility functions to express user preferences [12, 25, 31], although only [25] allows the user to define complex utility functions. These three proposals use optimization techniques, as Integer Programming or Constraint Programming, to select the best offers. Therefore, utility functions become the natural choice to define highly expressive user preferences.

3. There are many proposals that provide a semantic framework to define QoS [6, 12, 19, 22, 26, 30, 32], although [19] do not handle user preferences in their ontology and [26, 32] have a fixed definition of user preferences, inherent to their selection algorithm. [12] is the most expressive when defining user preferences, followed by [6, 22, 30], that limit their preferences to weights and parameter tendencies. According to all those proposals, it is clear that QoS have to be defined semantically.

4. None of the analyzed proposals semantically define user preferences, although in [6, 30] the authors include in their ontology extension the tendency of QoS parameters. What is more, most of the proposals that perform selection tasks in terms of user preferences describe them using ad-hoc, non-semantic descriptions completely decoupled with the ones used to describe service functionality, causing a semantic gap between functional descriptions and user preferences.

These conclusions motivate this paper, because it is necessary to tackle the previous problems. Most recent proposals use utility functions to express user preferences, although they are not semantically defined, and there are many QoS ontologies which any solution should be able to linked with. Our discovery and selection model takes into account all these problems, providing a hybrid and QoS-aware solution.

3 A QoS-aware, hybrid model

The addition of QoS preferences to SWS descriptions, turns most approaches on selecting SWS insufficient, because they mainly use DLs, which are usually limited to logical and relational expressions when describing QoS conditions. Discovery and selection have to become independent, so the former can be performed by DLs reasoners, but the latter has to use an optimization technique, although functional and QoS-aware preferences have to be described at the same level. Thus, a hybrid solution arise as the most adequate option. In the following, this solution is presented, along with an ontology of user preferences that allows to describe both discovery and selection at the same semantic level. Although in [7] we present a generic n-stages hybrid discovery, in this paper we specify that early approach by using two concrete stages, as well as providing an user preference ontology that is used in the selection process.

Our selection proposal comes from mixing the expressiveness of utility functions and weights proposed by Ruiz-Cortés *et al.* [25], the semantic definition of QoS from Maximilien and Singh [19] or Kritikos and Plexousakis [12], and an extension to give semantics to utility functions. Furthermore, this QoS preferences ontology can be linked with functional preferences, so the whole discovery and selection process are performed within a hybrid architecture, where DLs is used to discover services as in the proposals from Sec. 2.1, and selection is treated as an optimization problem.

3.1 Hybrid semantic discovery

Fig. 1 shows the activity diagram of a hybrid discovery process performed by two different engines. This process begins with the discovery stage, where a set of SWS descriptions are matched with the user functional preferences. Thus, at this point, only compatible services, in terms of functionality, are returned to the next stage. That matching can be performed using available DLs reasoners that can take SWS descriptions and return those SWS which match with the functional preferences, that can be expressed as a service profile in OWL-S or as service capabilities in WSMO, for instance.

Then, the *discovered SWS* are further processed, ranking them in order to select the best service. In this

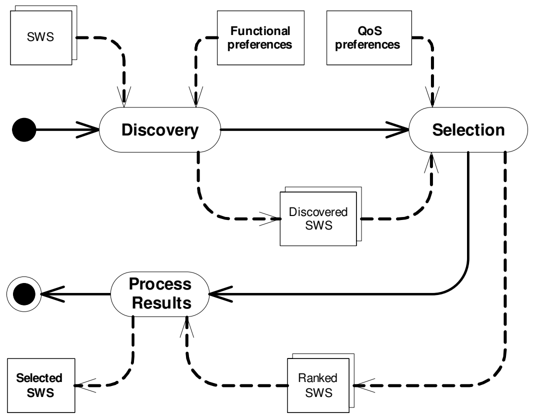


Figure 1: Activity diagram of the hybrid process.

case, selection process uses QoS-aware user preferences to rank services. These preferences are also semantically described, and linked with functional preferences to conform the whole user preference. However, QoS preferences are not semantically described in current proposals (cf. Sec. 2.2), so it is necessary to provide a conceptualization of QoS-related user preferences. Furthermore, in order to perform the selection stage, an optimization technique has to be used. Thus, discovered SWS descriptions and QoS preferences are transformed into an optimization problem that can be performed by different techniques, such as CP, LP or DP solvers. This stage is further explained in Sec. 3.2.

Finally, the list of *ranked SWS* are processed at the second stage, where the best service in terms of user preferences are returned, so it can be invoked or composed with others. This service is referenced as *selected SWS* in Fig. 1.

This hybrid discovery architecture has many advantages. It is loosely coupled, due to the possibility to use any discovery and selection engines. Also, user preferences expressiveness are not constrained to a specific selection technique, provided that a transformation from our conceptualization is available. Moreover, our proposed architecture can be applied to any existing SWS framework and corresponding repositories, taking benefit of the wide range of tools already implemented.

3.2 QoS-aware semantic selection

In order to decouple QoS-aware preferences descriptions with the concrete selection algorithm used, we propose to model these preferences as an upper ontology. This conceptualization allows the user to describe the whole service, including functional descriptions from a SWS framework, at the same semantic level. Furthermore, it provides semantic interoperability between user preferences based on differently named QoS parameters, because equivalences between those QoS parameters can be in-

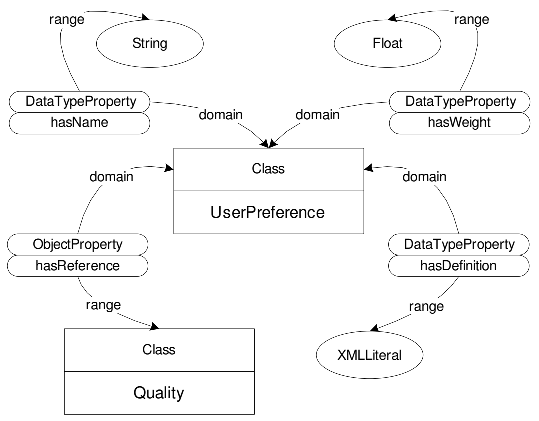


Figure 2: Proposed ontology to model user preferences.

ferred.

Our proposed model is shown in Fig. 2. The main concept (or *class*) is UserPreference, which references a Quality concept via the *hasReference* object property. This Quality concept is analogous to the defined in [19], and represents the QoS parameter which is used in the definition of the corresponding user preference. Furthermore, the UserPreference concept has a key datatype property, *hasDefinition*, which links the more generic preference concept with the utility function that defines it. Note that Quality class is the link to QoS parameters used in the semantic definition of QoS. This definition can be performed using the ontology from Kritikos and Plexousakis [12] or from Maximilien and Singh [19], for instance.

The utility function is initially modeled as a property that contains an XML expression that describe the definition of each function in terms of OpenMath standard [5], as used in [12], allowing the evaluation of the function with a proper compiler or a mathematical tool, such as Mathematica.

Finally, our main concept UserPreference has two datatype properties: *hasName* and *hasWeight*. The former is used as an identifier of a given instance. The latter is a real number which corresponds to the relative weight associated with the corresponding QoS parameter, used to compute the global utility function of an offer.

Fig. 3 shows two instances of our proposed ontology, in the case of a composed user preference about $MTTF$, with an associated weight of 0.7 (Fig. 3c), and $MTTR$ with its corresponding weight of 0.3 (Fig. 3d). On the one hand, the instance PreferredMTTF references an instance UserMTTF of MTTF class, that is a subclass of Quality class from [19]. On the other hand, PreferredMTTR references an instance UserMTTR of MTTR class. Moreover, concrete utility functions are specified as OpenMath objects that represent the showed in Fig. 3a and Fig 3b, respectively, using XML.

Selection process has to take all the instances from the proposed ontology to compose a global user preference. Thus, in the example from Fig. 3, the concrete optimization technique used has to take both user preferences into account to perform the actual selection, according to the relative weights associated with each QoS preference.

4 Conclusions

In this work, we show that using a unique engine to discover SWS is not appropriate, due to each engine is usually designed for a concrete kind of search. For instance, DLs reasoners are well suited when discovering SWS in terms of concepts and relations, but they can not handle complex numerical QoS preferences. Although there are extensions to allow concrete domains in DLs, reasoners have to implement them, and they may bring undecidability results.

We present a hybrid solution that consists in a two-stages discovery process, where each stage is performed using the most appropriate technique. Furthermore, we provide a semantic framework to define user preferences on semantically defined QoS parameters, provided that it is used in conjunction with another proposal that semantically defines those QoS parameters, like [12, 19]. Thus, all facets of SWS description (functional, non-functional, and user preferences) are described at the same semantic level, so discovery and selection tasks are completely done within a single semantic framework, allowing interoperability between different service definitions.

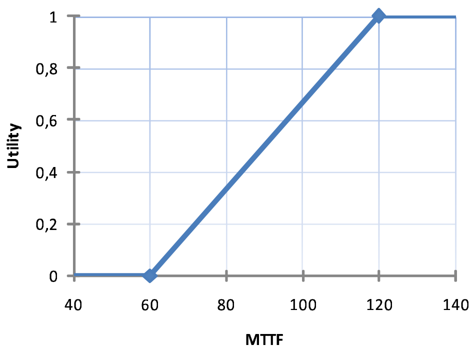
In addition, our proposed architecture is extensible and loosely coupled, allowing to define complex QoS conditions, and to use utility functions based on QoS parameters to obtain the best service. This architecture does not impose any restriction on the SWS framework and repository to use, allowing its materialization as a discovery component for current SWS implementations. Moreover, it is independent on the concrete optimization technique used in the selection stage.

Acknowledgment

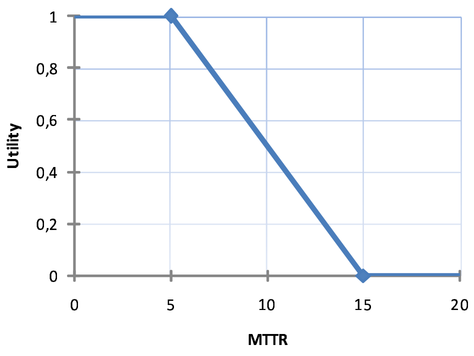
This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472) and by the Andalusian Government under project ISABEL (TIC-2533).

References

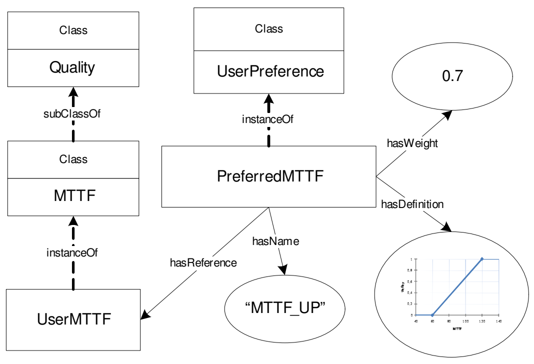
[1] F. Baader and U. Sattler. Description logics with aggregates and concrete domains. *Information Systems*, 28(8):979–1004, December 2003.



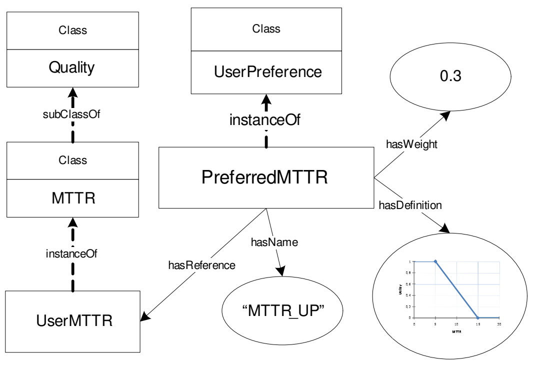
(a) $MTTF$ utility function.



(b) $MTTR$ utility function.



(c) User preference instance about $MTTF$.



(d) User preference instance about $MTTR$.

Figure 3: Conceptualization of $MTTF$ and $MTTR$ user preferences.

[2] B. Benatallah, M. Hacid, C. Rey, and F. Toumani. Semantic reasoning for web services discovery. In *WWW Workshop on E-Services and the Semantic Web*, 2003.

[3] S. Benbernou and M. Hacid. Resolution and constraint propagation for semantic web services discovery. *Distributed and Parallel Databases*, 18(1):65–81, 2005.

[4] P. Bonatti and P. Festa. On optimal service selection. In *14th international conference on World Wide Web*, pages 530–538, 2005.

[5] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase. The OpenMath standard. Technical Report Version 2.0, The OpenMath Society, 2004.

[6] G. Dobson, R. Lock, and I. Sommerville. QoSOnt: a QoS ontology for service-centric systems. In *EUROMICRO-SEAA*, pages 80–87. IEEE Computer Society, 2005.

[7] J. M. García, D. Ruiz, A. Ruiz-Cortés, O. Martín-Díaz, and M. Resinas. An hybrid, QoS-aware discovery of semantic web services using constraint programming. In B. Krämer, K.-J. Lin, and P. Narasimhan, editors, *ICSOC 2007*, volume 4749 of *LNCS*, pages 69–80. Springer, 2007.

[8] J. González-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. Technical Report HPL-2001-265, Hewlett Packard Labs, 2001.

[9] V. Haarslev and R. Möller. RACER system description. In *Automated Reasoning, First International Joint Conference, IJCAR 2001*, pages 701–706, 2001.

[10] V. Haarslev and R. Möller. Practical reasoning in RACER with a concrete domain for linear inequations. In *Int. Workshop on Description Logics*, 2002.

[11] I. Horrocks. FaCT and iFaCT. In *Int. Workshop on Description Logics*, 1999.

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

[13] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Int. World Wide Web Conference*, pages 331–339, 2003.

[14] Y. Liu, A. H. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In *WWW (Alternate Track Papers & Posters)*, pages 66–73, 2004.

[15] C. Lutz. Description logics with concrete domains – a survey. In *Advances in Modal Logic*, pages 265–296, 2002.

[16] C. Lutz and U. Sattler. A proposal for describing services with DLs. In *Int. Workshop on Description Logics*, 2002.

[17] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, et al. OWL-S: Semantic markup for web services. Technical Report 1.1, DAML, 2004.

[18] O. Martín-Díaz, A. Ruiz-Cortés, D. Benavides, A. Durán, and M. Toro. A quality-aware approach to web services procurement. In *4th. VLDB Workshop on Technologies for E-services TES'03*, pages 42–53, 2003.

[19] E. M. Maximilien and M. P. Singh. A framework and ontology for dynamic web services selection. *Internet Computing, IEEE*, 8(5):84–93, 2004.

[20] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: A framework and infrastructure for semantic web services. In *Int. Semantic Web Conference*, pages 306–318, 2003.

[21] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Int. Semantic Web Conference*, pages 333–347, 2002.

[22] J. Pathak, N. Koul, D. Caragea, and V. G. Honavar. A framework for semantic web services discovery. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 45–50, New York, NY, USA, 2005. ACM Press.

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

[24] D. Roman, H. Lausen, and U. Keller. Web service modeling ontology (WSMO). Technical Report D2 v1.3 Final Draft, WSMO, 2006.

[25] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán-Toro, and M. Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst*, 14(4):439–468, 2005.

[26] A. Soydan Bilgin and M. Singh. A DAML-based repository for QoS-aware semantic web service selection. In *IEEE International Conference on Web Services*, pages 368–375, 2004.

[27] N. Srinivasan, M. Paolucci, and K. Sycara. Semantic web service discovery in the OWL-S IDE. In *Hawaii International Conference on Systems Science*, 2006.

[28] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *J. Web Sem.*, 1(1):27–46, 2003.

[29] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet Computing*, 8(3):66–73, 2004.

[30] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-aware selection model for semantic web services. In A. Dan and W. Lamersdorf, editors, *ICSOC 2006*, volume 4294 of *LNCS*, pages 390–401. Springer, 2006.

[31] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.

[32] C. Zhou, L. Chia, and B. Lee. DAML-QoS ontology for web services. In *IEEE International Conference on Web Services*, pages 472–479, 2004.