Mapping a Fuzzy Logic Approach for QoS-aware Service Selection on Current Web Service Standards

Ioana Şora, Gabriel Lazăr and Silviu Lung
Department of Computers
Politehnica University of Timisoara, Romania

***Abstract*—We propose FQ (*F*uzzy-*Q*oS), a complete architecture for including user preferences and quality of service characteristics in the selection process of web services. Besides the flexibility of the selection and ranking algorithm, we consider of equal importance the properties of the implementation: compliance with standards, backwards compatibility and compatibility with non-FQ users, and performance of the selection mechanisms implementation.**

We present our approach that relies on a combination of two standards from the domain of web service and semantic web technologies: UDDI, for storing and retrieving syntactic and semantic information about web services and SAWSDL, for creating semantically annotated web service descriptions.

I. Introduction

Service Oriented Architectures are based on the notion that programs can be constructed by composing independent services. A service is defined as "a loosely-coupled, reusable software component that encapsulates discrete functionality which may be distributed and programmatically accessed over standard Internet protocols" [1], [2].

The general scenario in service oriented computing is that Service Providers design and implement services, hosting them as network-accessible modules, and advertise them by defining service descriptions that are published in Service Registries. Clients or Service Requesters use the published service descriptions in order to find the needed services.

The major fact that assured the succes of service oriented computing is that the development of service technologies has been governed from the beginning by an active standardisation process. Two important standards for web services are:

- UDDI (the Universal Description Discovery and Integration specification) that defines how to publish and discover information about web services
- WSDL(Web Services Description Language) - for the description (functional aspects) of web services [3]

In the field of web service discovery and selection, the major research challenges include enhancing web service discovery and selection with semantic aspects [4]. Achieving automated web service discovery and selection requires adding semantic annotations to web service definitions and including user preferences - quality of service characteristics and non-functional properties - in the selection process. When several functionally equivalent web services are available, their quality of service characteristics and non-functional properties become important. These factors should have the final word in the selection process when several web services provide the same functionality, but they could also be a decision factor in the choice between several alternative functionalities.

In our previous works [5], [6], we have developed a novel fuzzy logic approach for the specification, selection and ranking of services according to individual QoS preferences. Our approach uses fuzzy inference for ranking the candidates, but based on sets of automatically generated fuzzy rules for each set of individual preferences. Fuzzy rules have a big expressive power, and the fact that they are generated automatically makes this approach user-friendly. As we have shown in [5], the advantage of using fuzzy inference with generated rules for service ranking is that it is more flexible and has a bigger and more controllable expressive power than the classical fuzzy multicriteria decision making (FMCDM), especially in the case when several properties with different importance degrees are considered.

The goal of our current work is to fully prove the feasability of our fuzzy selection and ranking approach, both from the point of view of the integration with current standards that govern the domain of web services and the performance of the implementation of fuzzy selection and ranking in service registries. In Section II we present the global architecture of FQ (*F*uzzy-*Q*oS). Section III summarizes the characteristics of our FQ ontology. Sections IV and V describe the implementation using UDDI data structures and SAWSDL annotations. Experimental results measuring the performance of the implementation of our fuzzy selection and ranking mechanism are presented in section VI. Section VII discusses related work in the field of QoS-aware selection of services. Conclusions are drawn in section VIII.

II. Global FQ architecture

In this work, we propose an approach that supports semantic web service descriptions using registries conforming to UDDI V2 specification. We present how we store QoS-related semantics in the UDDI data model and how we make use of that information in the fuzzy selection and ranking process.

Our approach is completely backwards compatible supporting service requestors and service providers who wish to take advantage of FQ's semantic capabilities and also those that do not. Not requiring any modification to the existing UDDI registries is considered an important advantage.

We propose an architecture where the UDDI server stands independently to the semantically-enabled modules, as depicted in Figure 1.

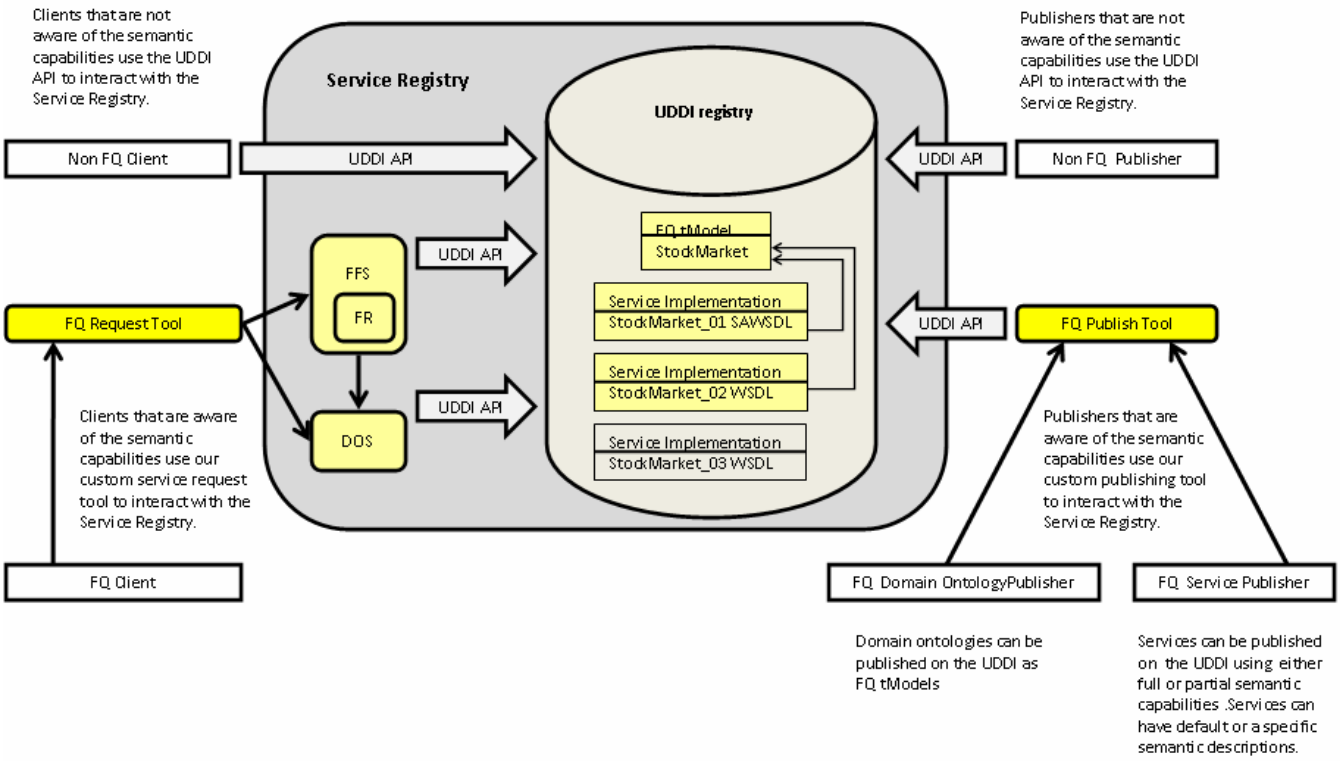

Fig. 1. Arhitecture of our approach

The service registry component of the classical service computing architecture is formed, in our approach, by the UDDI registry together with a FunctionalityFindingService and a DomainOntologyService.

The Domain Ontology Service (DOS) is a server component responsible for storing in a local database all the known ontologies gathered from the UDDI registry. It is the enabling element for our QoS-aware approach. QoS ontologies define the semantics of QoS parameters for different service types.

The Functionality Finding Service (FFS) is a server component responsible for retrieving the web services from the UDDI registry in response to requests coming from clients. Here the functionally equivalent web service candidates are ranked according to their QoS properties matching the individual client request. In order to complete its information about the semantics of QoS parameters found in the description of service candidates, it interacts with the Domain Ontology Service to obtain additional semantic information. The ranking is done the FuzzyRanking subsystem (FR), which implement the approach based on automatically generating fuzzy rules starting from individual user preferences and using them in a fuzzy inference process ranking the web service candidate, as described in our previous work [5].

The FQPublisherTool component allows providers to add/remove/update web services or ontologies. The FQPublisher tool is to be used by Service Providers that adhere to our proposed approach. Service providers that are not aware of the approach access the UDDI registry via its regular API and mechanisms. Also, authorised Domain Definers can use it in order to create and publish domain ontologies.

The FQRequestTool component allows users to construct valid web service requests and to receive an ordered list of candidates descriptions for their request. In order to ensure the validity of the request, it interacts with the Domain Ontology Service. The request is sent and results are received from the Functionality Finding service. The FQRequestTool is to be used by Service requestors that want to make use if the facilities of expressing individual QoS preferences and of the facility of fuzzy selection and ranking. Other service requestors can access the service discovery facility via the usual UDDI API.

III. FQ Domain Ontologies

Characteristic to our approach is that we need domain ontologies that can help in defining value categories for different non-functional or QoS properties through linguistic variables. For example, the response time of a service could be described with the terms *very fast*, *fast*, *slow*, *very slow*. The membership functions of these terms can be established by domain experts, separately for each type of service - as the expectations are different for different types of services.

For each functionality (service type), the domain expert establishes the valid non-functional properties. For each non-functional property, following description items have to be provided, in the context of describing the given functionality:

- domain: the range of values for measurable quantities, or a default range 0-100 for description-only properties.

```
<?xml version="1.0" encoding="utf-8" ?>
<ontology xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="ontology.xsd">
 <functionality name="StockMarket">
   <property name="Availability" start="0" end="100">
      <term name="Low" start="0" left="0" right="66" end="80" />
      <term name="Normal" start="70" left="80" right="90" end="95" />
      <term name="High" start="90" left="95" right="100" end="100" />
   </property>
  <property name="ResponseTime" start="Infinity" end="0">
      <term name="Slow" start="Infinity" left="Infinity" right="20000" end="15000" />
      <term name="Normal" start="20000" left="15000" right="7500" end="2500" />
      <term name="Fast" start="7500" left="2500" right="0" end="0" />
  </property>
```

Fig. 2. Example of QoS ontology: excerpt of the StockMarket FQ ontology

```
<tModel tModelKey="uuid:a9ff70e4-32a2-48dd-a38b-86f7b28d52eb" operator="UDDI" authorizedName=*>
  <name>StockMarket</name>
  <overviewDoc>
      <overviewURL>http://193.226.12.174/Ontologies/StockMarket.xml</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey=uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4
                    keyName="QoS" keyValue="ontology" />
  </categoryBag>
</tModel>
```

Fig. 3. Exemple of QoS tModel: tModel of the StockMarket FQ ontology

- optimization direction: specifies what is better, for the current property, a bigger value (as for availability) or a smaller value (as for response time)
- fuzzy terms: the fuzzy terms which can be used to describe this property have to be enumerated, for each term providing its name and the shape of its membership function as defined by a domain expert. In our approach, we assume that the terms have to be defined of trapezoidal shape.

For example, a type of web services providing stock market informations can be described by following non-functional properties: *Availability, ResponseTime, Cost, PublisherReputation, UserRating, UpdatesPerDay, ListedCompaniesNo, RequestLimitPerDay, TotalAssetsValue*. The value categories of these properties in the context of stock market services are defined in the StockMarket FQ-Ontology. An excerpt of this ontology for the first two properties is shown as example in Figure 2, where we can see that in this example both properties have 3 fuzzy terms of trapezoidal shapes defined by their coordinates. The domain ontology can be published by authorized domain definers with help of the FQPublisher tool.

IV. FQ UDDI DATA STRUCTURES

The UDDI specification defines an XML-based data model for storing descriptive information about web services and their providers, and a web service-based API for publishing this type of information to the registry and performing inquiries.

Two important structures of the UDDI data model are tModels and bindingTemplates. A tModel is a data structure representing a service type in the UDDI registry. Each business registered with UDDI categorizes all of its Web services according to a defined list of service types. The tModel is an abstraction for a technical specification of a service type. Another UDDI data structure, the bindingTemplate organizes information for specific instances (particular implementations) of service types. When businesses want to make their specification-compliant services available to the registry, they include a reference to the tModel key for that service type in their bindingTemplate data.

In our approach we use TModels for the storage of ontology information. The QoS Ontology tModel, used in our implementation, serves as a place holder for the ontology as a whole. It contains information including the ontology name and URL of external descriptions. The QoS Ontology tModels can be published in the UDDI registry in order to allow providers to state that their web services are semantically described using concepts from these ontologies.

Published using general keywords (KeyName = QoS, KeyValue = ontology), the QoS Ontology tModels are used to specify the functionality of web services and the valid vocabulary used for describing the quality of service characteristics and non-functional properties allowed to occur in web service descriptions (ex: availability, reliability, response time, cost, publisher reputation).

Figure 3 presents as an example the QoS ontology tModel for the StockMarket domain.

V. FQ SAWSDL ANNOTATIONS

The Web Services Description Language (WSDL) describes Web services on a syntactic level. Semantic Annotations for WSDL and XML Schema (SAWSDL) [7] defines a set of extension attributes for WSDL and XML Schema definition

```
<bindingTemplate serviceKey="a2a16918-bc88-48d4-a6d6-f69f558298e9"
                                                        bindingKey="c2fceee7-a3bd-44b7-8acb-5aeff7b50b00">
  <accessPoint URLType="http">http://193.226.12.174/WebServices/StockMarket/StockMarket_01.wsdl</accessPoint>
    <tModelInstanceDetails>
          <tModelInstanceInfo tModelKey="uuid:a9ff70e4-32a2-48dd-a38b-86f7b28d52eb" />
    </tModelInstanceDetails>
</bindingTemplate>
```

Fig. 4. Example of service implementation adhering to the FQ StockMatket ontology

```
<wsdl:service name="StockMarket_01"
              sawsdl:modelReference="http://193.226.12.174/Annotations/StockMarket_01.xml">
  <wsdl:port name="StockMarket_01Soap" binding="tns:StockMarket_01Soap">
              <soap:address location="http://193.226.12.174/WebServices/StockMarket/StockMarket_01.asmx" />
  </wsdl:port>
  <wsdl:port name="StockMarket_01Soap12" binding="tns:StockMarket_01Soap12">
               <soap12:address location="http://193.226.12.174/WebServices/StockMarket/StockMarket_01.asmx" />
  </wsdl:port>
</wsdl:service>
```

Fig. 5. Example of SAWSDL annotated StockMarket implementation

```
<?xml version="1.0" encoding="utf-8"?>
<service>
          <property name="Availability" crisp="30" />
          <property name="ResponseTime" crisp="14000" />
          <property name="Cost" crisp="1200" />
           <property name="PublisherReputation" crisp="4" />
           <property name="UserRating" crisp="3" />
           <property name="UpdatesPerDay" crisp="75" />
           <property name="ListedCompaniesNo" crisp="200" />
           <property name="RequestLimitPerDay" crisp="450" />
           <property name="TotalAssetsValue" crisp="125" />
</service>
```

Fig. 6. Example of SAWSDL annotation of a StockMarket implementation

language that allows description of additional semantics of WSDL components. The specification defines how semantic annotation is accomplished using references to semantic models, e.g. ontologies. SAWSDL does not specify a language for representing the semantic models, but it provides mechanisms by which concepts from the semantic models, typically defined outside the WSDL document, can be referenced from within WSDL components using annotations.

We use semantic annotations to describe web services using concepts from ontologies, represented as tModels in the UDDI registry.

In order to specify that a web service has the functionality and makes use of concepts from a published ontology, the tModel representing the corresponding ontology is referenced by the bindingTemplate of that service. For example, a web service(bindingTemplate) may be advertised as having the StockMarket functionality simply by referencing the StockMarket QoS Ontology tModel, like in Figure 4.

Generally, non-functional properties are specific details to a web services implementation or running environment, thus we attach references pointing to semantic concepts on the WSDL's service element. Figure 5 presents how the service element of a stockmarket service has a model reference to a SAWSDL document. The annotation concepts from the StockMarket ontology and an excerpr is presented in Figure 6.

Web services can be published both by publishers that are aware of the FQ approach and also by publishers that are not using FQ. A web service(bindingTemplate) can be published in the following ways:

- no tModel referenced and WSDL document as accessPoint (service providers not aware of FQ)
- tModel referenced and WSDL document as accessPoint (service providers aware of FQ but who do not annotate their service descriptions, thus a default semantic description will be assumed)
- tModel referenced and SAWSDL document as accessPoint (service providers aware of FQ and who provide a specific semantic description)

VI. Experimental validation

A. Validation objectives

Our approach uses fuzzy inference for ranking the candidates, fuzzy inference performed on sets of automatically generated fuzzy rules for each set of individual preferences. Since this process of generating fuzzy rules from custom user preferences and using them in a fuzzy inference process to rank candidate services seems computational intensive, we want to measure its influence on the server's response time,

in the conditions of a high number of concurrent requests. In order to do this, we created also a second version of the system, where the fuzzy ranking is moved from the FunctionalityFindingService (FFS) to the FQRequestTool. In this second version, the FFS sends the client only the list with all the functionality-matching web services found in the UDDI repository, then the client is responsible to do the ranking. We compare the response times obtained by the client in the two versions of the system, in conditions of many concurrent requests to the server.

B. Experimental setup

The service registry is populated with 300 web services, representing instances of 2 service types (StockMarket and Processor), for each service type there are 150 functional equivalent implementations characterised by different QoS properties. A query will have to rank through fuzzy inference up to 150 functional equivalent services.

Out of the 150 services implementing the same functionality, 120 have references to SAWSDL descriptions, 24 are linked to the ontology but have no links to specific semantic descriptions, thus they will be considered worst cases for all properties. The services with SAWSDL descriptions are annotated in such a way that they present a well balanced distribution of values from all ranges for all their properties.

The tests were done on a network of 7 Dell dual core, 2GB RAM. One computer hosted the UDDI registry, the Functionality Finding Service and the Domain Ontology Service. The other 6 computers were used to run special "spammer" programs that created a number of concurrent threads, with each thread issuing queries to the Functionality Finding Service.

As far as testing is concerned, we took into consideration three varying parameters in order to represent objectively run time variation in the two versions of our system. The parameters are:

- number of user-specified properties (NP) in each query: this parameter affects the inference and rule generation process because as the client (user) assigns values (crisp or fuzzy) to more properties that describe a chosen functionality, the more complex the inference process is and the more rules are generated and thus this process takes more time.
- number of functional equivalent web services in the UDDI repository (NWS): this parameter affects the inference process, that has to be repeated for each candidate. Also the parameter affects run time when the inference machine is placed on the client side and the FFS has to send the client all candidates, this time being also increased by delays due to network traffic.
- number of threads used by each spammer (NT): this parameter affects the run time when the inference machine is placed on the server side (the FFS) because it simulates multiple simultaneous requests from clients.

C. Results

Figures 7, 8 and 9 show results of the measurements.

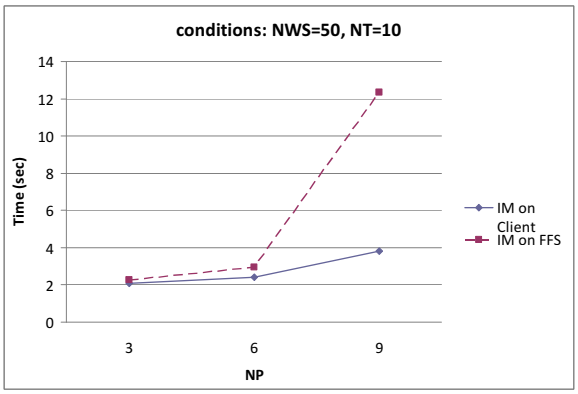

Fig. 7. Response time for a service selection

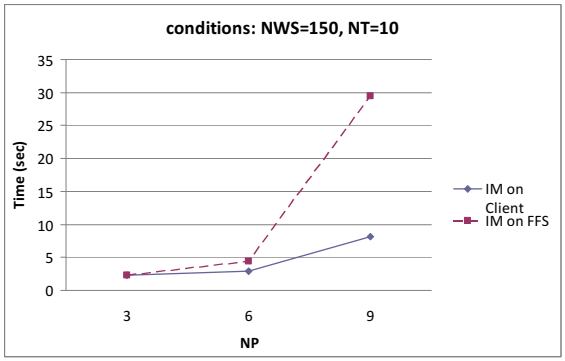

Fig. 8. Response time for a service selection

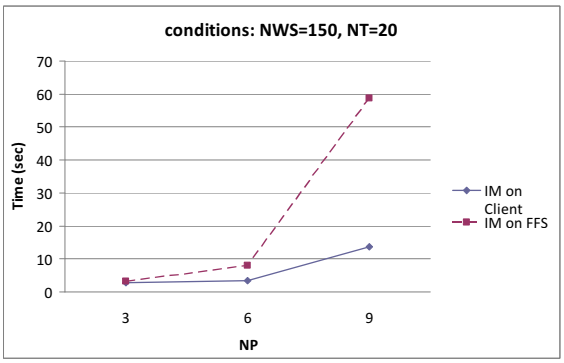

Fig. 9. Response time for a service selection

As we can observe, in the worst case, corresponding to the figure 9, when the server has to proces 6*20=120 concurrent requests of ranking 150 equivalent services while taking into account 9 different properties, it takes 4 times more in the case where the fuzzy ranking is done by the central FFS server compared to the case when the ranking is done at each client side. We consider this result satisfactory for the performance of the fuzzy ranking implemented in the Functionality Finding Service.

We also can notice that the critical parameter which influences the most the increase of the response time is NP, the

number of user-specified properties in the request. Requests with up to 6 specified properties are solved in very good times. We believe that this case corresponds to the most frequent user scenario and there are rarely requests that specify more properties.

VII. Related work

QoS-aware selection of services is a hot topic in today's research. We analyse related work grouping it around the three main characteristics present in our work: ontologies, selection and ranking mechanisms, and technologic implementations.

In order to enable automatic semantic matching while keeping a low degree of formalism in the description of services, most approaches rely on the concept of domain ontologies [8] [9]. Several approaches of ontologies for semantic or QoS -aware services have been proposed, as for example in [10], [9],[11], [12]. None of these ontology approaches is fully appropriate for the goal of facilitating imprecise (fuzzy) matching, where we need a domain ontology that can help in defining value categories through linguistic variables, as it is the case in the FQontology proposed as part of our approach.

For selection and ranking, different approaches are used: [13] uses Singular Value Decomposition SVD, based on decomposing the Quality and Web Services matrix. [14] considers service selection based on maximizing a utility function under cost constraints. Utility functions reflect the importance of different attributes in a request. Other approaches use constraint programming to check QoS conformance [12], or hybrid approaches like [15], which combine Integer Programming, genetic algorithms, and case-based reasoning to tackle the QoS-aware service composition problem. There are also approaches based on fuzzy logic [16] as a solution for both the computational complexity and the matching with imprecise QoS constraints. Most of the fuzzy approaches are variants of Fuzzy Multi Criteria Decision Making ([17], [18], [19]) or a version of Fuzzy decision by a committee of evaluators [20].

While many approaches remain theoretical investigations of selection algorithms or modelling of QoS concepts, a few researches give full solutions that are integrated with the curent standards for web services, as for example in [21], [22].

VIII. Conclusions

We propose FQ (*F*uzzy-*Q*oS), a complete architecture for including user preferences and quality of service characteristics in the selection process of web services.

The implementation of our approach relies on a combination of two standards from the domain of web service and semantic web technologies: UDDI, for storing and retrieving syntactic and semantic information about web services and SAWSDL, for creating semantically annotated web service descriptions. This makes our approach fully compatible with current standards, while assuring backwards compatibility and compatibility with users that are not using FQ.

The experimental validation proved that our selection and ranking algorithm can be implemented with a reasonable overhead in the Functionality Finding Server.

References

[1] J. Sommerville, *Software Engineering*, 8th ed. Addison Wesley, 2006.

[2] T. Erl, *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[3] *Web Services Description Language (WSDL) Version 2.0*, W3C Std. Recommendation 26 June 2007. [Online]. Available: http://www.w3.org/TR/wsdl20

[4] M. Papazoglu, P. Traverso, S. Dustdar, and F. Leyman, "Service-oriented computing: State of the art and research challenges," *IEEE Computer*, Nov. 2007.

[5] I. Sora, D. Todinca, and C. Avram, "Translating user preferences into fuzzy rules for the automatic selection of services," in *SACI*, 2009, pp. 497–502.

[6] I. Sora and D. Todinca, "Specification-based retrieval of software components through fuzzy inference," *Acta Politehnica Hungarica*, vol. 3, no. 3, pp. 121–132, 2006.

[7] *Semantic Annotations for WSDL and XML Schema*, W3C recommendation Std., 2007. [Online]. Available: http://www.w3.org/TR/sawsdl/

[8] E. Giallonardo and E. Zimeo, "More semantics in QoS matching," in *Service-Oriented Computing and Applications, 2007. SOCA '07. IEEE International Conference on*, 2007, pp. 163–171. [Online]. Available: http://dx.doi.org/10.1109/SOCA.2007.30

[9] G. Dobson, R. Lock, and I. Sommerville, "QoSOnt: a QoS ontology for service-centric systems," in *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE Computer Society, 2005, pp. 126–133.

[10] Y. Liu, A. H. Ngu, and L. Z. Zeng, "QoS computation and policing in dynamic web service selection," in *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM, 2004, pp. 66–73.

[11] G. Dobson, S. Hall, and G. Kotonya, "A domain-independent ontology for non-functional requirements," in *IEEE International Conference on E-Bussines Engineering*. IEEE Computer Society, 2007, pp. 563–566.

[12] Q. Ma, H. Wang, Y. Li, G. Xie, and F. Liu, "A semantic QoS-aware discovery framework for web services," in *ICWS*, 2008, pp. 129–136.

[13] H. Chan, T. Chieu, and T. Kwok, "Autonomic ranking and selection of web services by using single value decomposition technique," in *ICWS*, 2008, pp. 661–666.

[14] D. A. Menascé and V. Dubey, "Utility-based QoS brokering in service oriented architectures," in *ICWS*, 2007, pp. 422–430.

[15] X. Ye and R. Mounla, "A hybrid approach to QoS-aware service composition," in *ICWS*, 2008, pp. 62–69.

[16] V. X. Tran and H. Tsuji, "QoS based ranking for web services: Fuzzy approaches," in *Proceedings 4th International Conference on Next Generation Web Services Practices, NWESP '08*, Seul, Korea, Oct. 2008, pp. 77–82.

[17] P. Xiong and Y. Fan, "QoS-aware web service selection by a synthetic weight," in *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD (3)*, 2007, pp. 632–637.

[18] M.-F. Chen, T. Gwo-Hshiung, and C. Ding, "Fuzzy MCDM approach to select service provider," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2003.

[19] M. De Cock, S. Chung, and O. Hafeez, "Selection of web services with imprecise QoS constraints," in *Web Intelligence, IEEE/WIC/ACM International Conference on*, 2007, pp. 535–541. [Online]. Available: http://dx.doi.org/10.1109/WI.2007.92

[20] P. Wang, K.-M. Chao, C.-C. Lo, C.-L. Huang, and Y. Li, "A fuzzy model for selection of QoS-aware web services," in *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on*, 2006, pp. 585–593.

[21] D. Kourtesis and I. Paraskakis, "Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery," in *ESWC*, ser. Lecture Notes in Computer Science, S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds., vol. 5021. Springer, 2008, pp. 614–628.

[22] J. Luo, B. Montrose, A. Kim, A. Khashnobish, and M. Kang, "Adding OWL-S support to the existing UDDI infrastructure," in *Web Services, IEEE International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 153–162.