A Fuzzy Framework for Semantic Web Service Description, Matchmaking, Ranking and Selection

Ehsan Sharifi, Reza A. Moghadam
Dpt. of Computer Science
Payam noor University, Tehran, Iran
ehsasha@gmail.com, askari@pnu.ac.ir

Fernando Bobillo
Dpt. of Computer Science and S.E.
University of Zaragoza, Spain
fbobillo@unizar.es

Mohamad M. Ebadzadeh
Dpt. of Computer Science
Amirkabir University of Technology
Tehran, Iran, ebadzadeh@aut.ac.ir

***Abstract*—Semantic Web Services (SWSs) provide a semantic representation of web services that provides a well-defined semantics in order to make them computer-interpretable, use-apparent, and agent-ready. In the last years it is becoming more and more accepted the idea that current Semantic Web technologies are not appropriate to deal with imprecise and vague knowledge, inherent to several real world domains.**

This paper proposes a fuzzy logic-based framework that makes service description, discovery and selection more flexible, and that includes support for non-functional properties. In particular, we extend OWL-S services by including a fuzzy domain ontology, which improves the definition of functional properties, and a fuzzy Quality of Service ontology, which improves the definition of non-functional properties. We also show how to enhance the tasks of SWS selection and ranking by means of fuzzy reasoning.

I. Introduction

A *web service* is a software system which allows to access to a remote application through standard Web protocols. A major limitation of the Web services technology is that discovering and composing services requires a lot of manual effort. To solve this problem, some researchers proposed to extend Web services with ideas from the *Semantic Web* [2], an extension of the web to give information a well-defined meaning, enabling automatic information processing.

The extension of Web services with a semantic description of their functionality is referred to as *Semantic Web Services* (SWSs) [6]. SWSs allow software agents to automatically discover, invoke and compose services. With information described semantically, a service broker could satisfy a complex user request by searching a services registry, matching the most suitable service (or services) and, if necessary, composing the individual results to perform the full task.

SWS descriptions rely on two kinds of ontologies. Firstly, a generic ontology specifies the main aspects of the service regardless of the application domain. OWL-S [21] is a language for SWS representation, being actually an OWL ontology to annotate Web Services for adding semantic information. Secondly, a domain ontology provides specific knowledge about the domain of the service. Domain knowledge is represented in an OWL or OWL 2 ontology [10], [20].

Due to the increasing number of services offering similar functionalities, *non-functional properties* (NFPs) have become essential criteria to enhance the processes of service selection. Thus, a service description usually includes functional properties and non-functional properties.

Current approaches often use *Quality of Service* (QoS) to filter the SWSs discovery results based on user constraints of QoS descriptions [22], as services with equivalent functionality can be provided by different service providers with substantially different values of QoS. Nevertheless, different service providers and requesters may use different QoS concepts for describing service quality information, which leads to the issue of semantic interoperability of QoS. The solution is using a QoS ontology that supports not only describing QoS information in great detail but also facilitating various service participants expressing their QoS offers and demands at different levels of expectation [19]. During the last years, a lot of effort has been deployed to enhance current frameworks to describe NFP properties. For example, several QoS ontologies have been proposed [12], [14], [1], [19].

Today it is widely agreed that classical ontologies are not suitable to represent vague and imprecise knowledge that the humans are very used to, such as the notions of *cheap* or *small*. In particular, the values of some non-functional properties of services, such as some QoS properties, cannot be properly described with crisp ontologies. In other words, because of the imprecise nature of the values of this properties, users cannot describe their request properly.

As a solution, *fuzzy ontologies* [17] have been proposed as a combination of ontologies with techniques from fuzzy set theory and fuzzy logic [23]. Essentially, in fuzzy ontologies fuzzy concepts denote fuzzy sets of individuals, whereas fuzzy roles denote fuzzy binary relations among individuals. Axioms are also extended to the fuzzy case and, consequently, it is possible to add an upper bound or a lower bound to some axioms. For instance, it is possible to represent fuzzy concepts inclusions, such as "fuzzyConcept1 is a subconcept of fuzzyConcept2 with at least degree 0.5".

The main objective of our research is to enhance SWSs by considering *fuzzy SWS*, where fuzzy ontologies are used rather than classical ontologies. More precisely, we use OWL-S as the SWS description language and extend both the functional and non-functional aspect of our new semantic web service framework with fuzzy ontologies. Then, we show how to enhance the tasks of SWS selection and ranking by means of fuzzy reasoning. As an additional contribution, we propose a framework for fuzzy ontology generation as a considerable extension of our previous work in [7]. This framework is used

978-1-61284-181-6/11/$26.00 ©2011 IEEE

to generate the fuzzy domain ontologies for the fuzzy SWS, but can be used in another different frameworks.

The remainder of this work is organized as follows. Section II proposes a framework for fuzzy ontology generation. Then, Section III describes our architecture for fuzzy SWSs management. Next, Section IV describes the experimental evaluation of our approach. Finally, Section V sets out some conclusions and ideas for future research.

II. A Framework for Fuzzy Ontology Generation

In this section we address the problem of obtaining the fuzzy domain ontology. We have separated this framework from the architecture in the next section because it could be used outside fuzzy SWSs. Here, we extend and refine our previous work in [7], which is inspired by [15]. Our framework (depicted in Figure 2) comprises several steps:

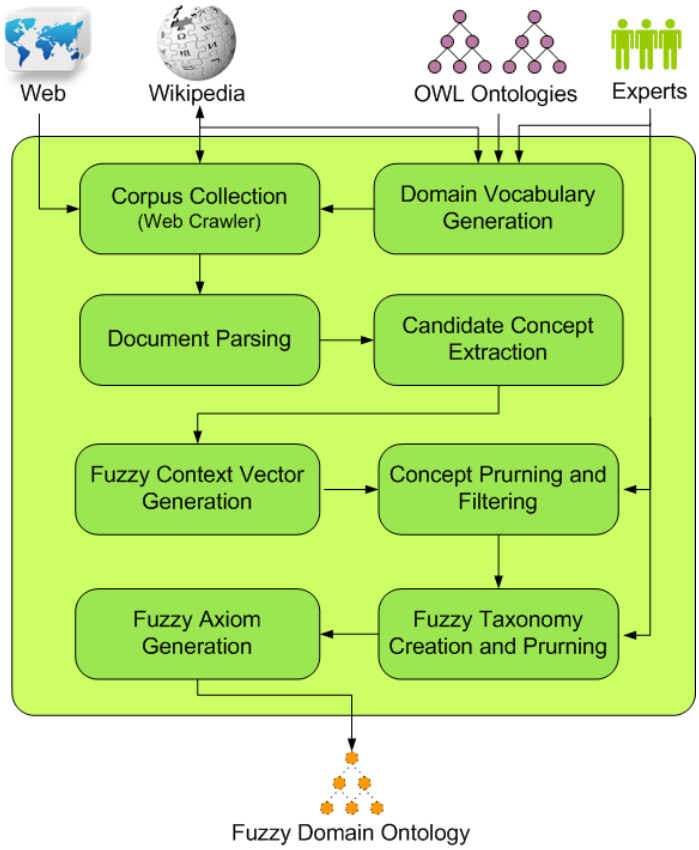

Fig. 1. Framework for fuzzy ontology generation.

a) **Domain vocabulary generation:** In this first step, we gather the domain vocabulary from multiple sources. The output of this step is an initial set of concepts and terms related to the application domain. These sources include experts knowledge, existing ontologies and `Wikipedia`[1]. The important feature of the selected information is the diversity. We can find appropriate ontologies by using a semantic search engine, as `Swoogle`[2]. Wikipedia is a specially important source of vocabulary due to its collaborative development. Finally, the experts can revise and extend the domain vocabulary.

b) **Corpus collection by web crawler:** We use the vocabulary generated at the previous step to collect a corpus of related web documents that are relevant for the domain. We use `JWPL` API to collect relevant web pages from Wikipedia, and by using a web crawler to retrieve relevant web pages from the Internet. Note that we use Wikipedia again, but this step produces documents instead of vocabulary. As usual, one has to find hubs that are suitable as starting points for the search process. Good hubs correspond to web pages that are linked by many web pages that are related to the domain. Note that this step is an enhancement with respect to [7].

c) **Document parsing:** Our corpus is converted into text format with `WP2Text`[3] tool. This step comprises three main parts [16]: *(i)* stop word removal, *(ii)* part-of-speech tagging, and *(iii)* word stemming.

d) **Candidate concept extraction:** A term consists of one or more words. A term is a candidate to become a concept if it carries recognizable meaning with respect to a context. Extracting concepts from our text corpus requires 3 steps: *(i)* basic preprocessing, *(ii)* pattern filtering, and *(iii)* computing statistical data between corpus terms.

e) **Fuzzy context vector generation:** In the field of information retrieval, context vectors have been proposed for concept representation [18]. In this step, we generate a fuzzy context vector for each candidate concept. We compute the degree of relatedness between terms and concepts by any of the measures discussed in [15]: BMI, JA, CP, KL, or ECH.

f) **Concept pruning and concept filtering:** This step is another enhancement with respect to [7]. We prune the concepts that have terms with membership degree below a specified threshold (obtained empirically after several experiments in [15]). Next, we eliminate concepts that have a high frequency in the domain but also in other domains. To this end, we compute the relevance score of a concept to a domain, because if a concept is significant for a particular domain, it will appear with a higher frequency than in other domains. Finally, experts can revise and modify the final list of concepts.

g) **Fuzzy taxonomy creation and pruning:** This step consists on discovering fuzzy subsumption relations among concepts. We compute the subsumption degree between two concepts based on their structural similarity [15]. When the taxonomy is built, we prune the subsumption relations below some threshold (again, obtained empirically in [15]).

h) **Fuzzy axiom generation:** Finally, the fuzzy context vector and the fuzzy taxonomy are used to obtain fuzzy concept inclusions. In particular, we use the syntax in [9]. Fuzzy assertions could also be obtained from fuzzy context vectors as in [7], but we will not consider them as they are not necessary for our fuzzy SWS framework. At this point, experts can complete the fuzzy ontology with new knowledge that could not be automatically generated in the previous steps.

[1] http://www.wikipedia.org

[2] http://swoogle.umbc.edu

[3] http://wp2txt.rubyforge.org

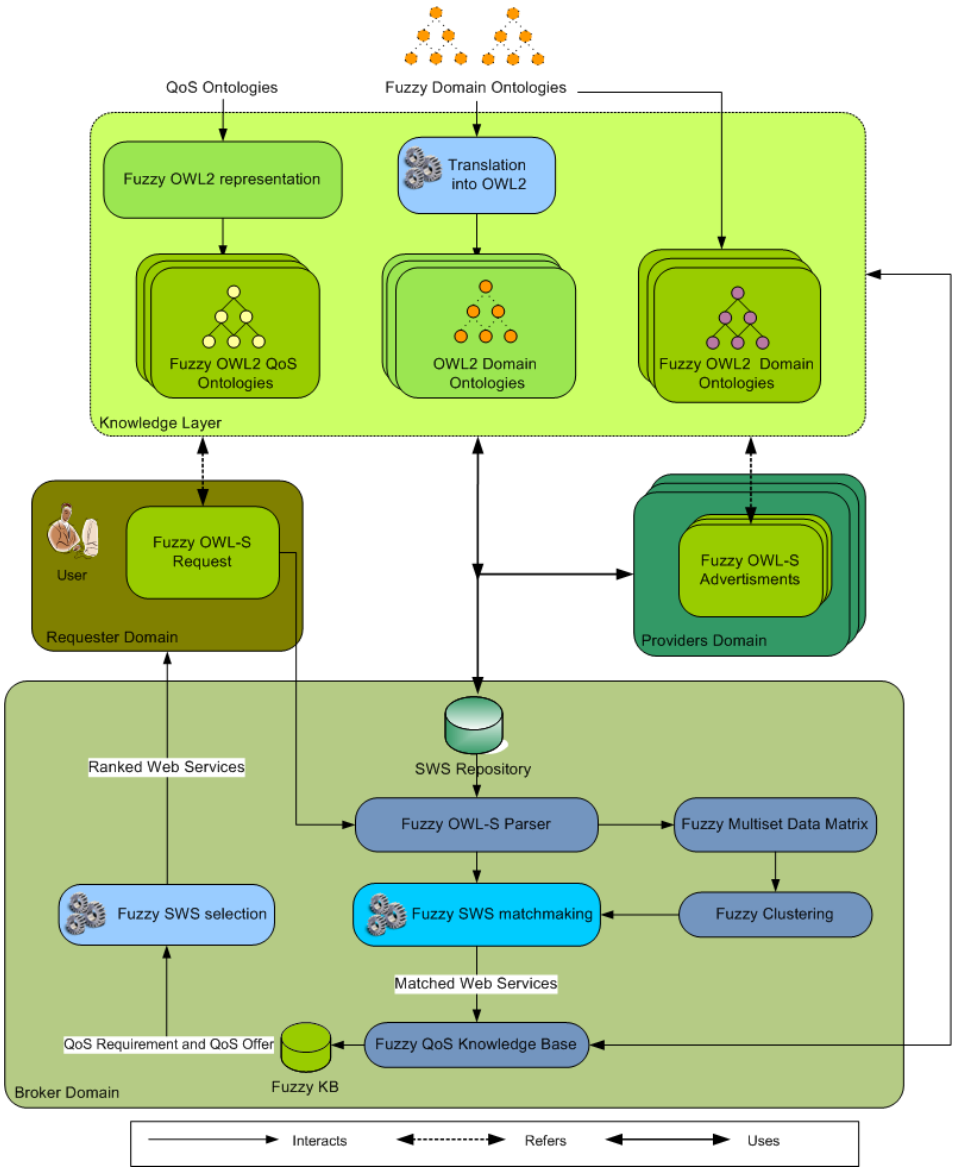

Fig. 2. Architecture for Fuzzy OWL-S.

III. An Architecture for Fuzzy SWS Management

This section describes our architecture for fuzzy SWS Description, Discovery and Selection (depicted in Figure 2). The main idea of the architecture is that, instead of using a traditional OWL-S upper ontology, we use multiple fuzzy ontologies. Both the service provider and requester can use fuzzy ontologies for the advertisement and the request. Fuzzy ontologies provide a more expressive description of functional and non-functional properties of SWSs.

There are several differences with our previous work [7]: we distinguish between functional and non-functional properties, we consider a tool for fuzzy ontology representation, and we deal with the tasks of fuzzy SWS matchmaking and selection.

On the one hand, functional aspects of a SWS (such as inputs and outputs) are described by means of fuzzy domain ontologies, which are built by using the framework described in Section II. On the other hand, non functional properties are described by means of fuzzy QoS ontologies. The benefit of these properties is that they allow more advanced service discovery and selection, although they require an extended service description and some way to use this new information to select the services. Such fuzzy ontologies are built by annotating existing ontologies. The framework in Section II is not used because of the different structure of the ontology.

For the moment, we are considering just one fuzzy QoS ontology, which is a fuzzy extension of the QoS ontology in [12], which integrates the Amigo ontology and the OWL-Q ontology [14], [1]. The Profile of OWL-S is linked to the QoS ontology parameters by using the property ServiceParameter and the QoS concept. Then, we provide a more powerful system to define quality properties values by using fuzzy

datatypes. For instance, while the property Reliability is originally defined as ≤ 50 (lessthanorequal50), we can define them by using a fuzzy left-shoulder membership function.

It is important to stress that we use two different fuzzy ontology reasoners: `DeLorean` [4] and `fuzzyDL` [8]. The reason is that they offer different features: `DeLorean` computes non-fuzzy OWL 2 ontologies, equivalent to the fuzzy ontologies [5], whereas `fuzzyDL` supports different logical operators that make it more appropriate for matchmaking [8].

Our architecture involves the following steps:

a) **Representation of fuzzy ontologies with fuzzy OWL 2:** We use `Fuzzy OWL 2` Protégé plug-in[4] to represent fuzzy QoS ontologies. The idea of the plug-in is to extend the elements of OWL 2 with annotation properties that represent the features of the fuzzy ontology that OWL 2 cannot directly encode. For further details, the interested reader is referred to [9]. Hence, this step receives as input some QoS ontologies and provide as output fuzzy QoS ontologies.

b) **Translation of fuzzy OWL 2 into equivalent OWL 2 ontologies:** As already noted, `DeLorean` reasoner is able to transform a fuzzy ontology into an equivalent OWL 2 ontology. In this step, we can translate some of the fuzzy ontologies into equivalent OWL 2 ontologies. The reason is that we can use for traditional (non-fuzzy) SWS advertisement and request. However, this is not a mandatory step, as the provider and the user could directly use the fuzzy ontologies.

c) **Fuzzy SWS advertisement:** Providers can describe their services by using fuzzy OWL-S. The functional properties are semantically described by using a fuzzy domain ontology (or an equivalent non-fuzzy ontology), whereas the NFPs are described by fuzzy QoS ontologies (or the equivalent non-fuzzy ontologies). Providers can register their advertisements in a SWS repository, which is a central and UDDI (Universal Description, Discovery and Integration) based database.

d) **Fuzzy SWS request:** Users can describe their requests in a more flexible manner with fuzzy OWL-S. The idea behind this step is the same as in the previous one.

e) **Fuzzy SWS matchmaking:** This step requires three previous steps: *(i)* translating fuzzy OWL-S functional properties into fuzzy multisets (using a fuzzy OWL-S parser), *(ii)* fuzzy multiset data matrix building, and *(iii)* fuzzy clustering based on the algorithm of the Fuzzy C-Means (FCM) [3]. We use a modified version of the SWS matchmaking algorithm proposed in [11], which takes as input a collection of fuzzy multisets (representing the information enclosed in OWL-S SWS descriptions) and a user request based on the functional description of the SWS, and returns a list of matched services, such that their advertisement produced a matching higher than given that a certain threshold. The difference is that we consider the representation in the fuzzy OWL-S descriptions.

f) **Fuzzy SWS selection:** In this last step, we *(i)* model NFPs (QoS properties) of the matched SWSs as fuzzy ontological axioms, *(ii)* rank the advertisements, according to their degree of satisfaction of the requests, by using a fuzzy ontology reasoner, and *(iii)* select the best one. More precisely, we use `fuzzyDL` reasoner and encode this task in similar way as the fuzzy matchmaking example in [8].

[4]http://www.straccia.info/software/FuzzyOWL/

IV. Evaluation

This section describes our experiments for evaluating our approach from an empirical point of view. Our approach has been tested on the test collections created for OWLS-MX [13], an hybrid matchmaker that complements logic based reasoning with approximate matching based on syntactic information retrieval-based similarity measures.

The contribution of this section is two-fold. On the one hand, we generate a fuzzy domain ontology, using the framework described in Section II, that we use for the semantic description of the functional properties. On the other hand, we replace the functional properties of the test collections with fuzzy properties from the fuzzy domain ontology, and we add fuzzy non-functional properties from fuzzy QoS ontologies.

In the following we describe further implementation details, the test case and the experimental results.

a) **Implementation.:** Our prototype for fuzzy domain ontology generation is implemented in `Java`. It uses several external libraries, namely the Stanford `NLP parser`[5], Part-Of-Speech (`POS`) tagger[6], JWPL[7], and `OWL-S API` 1.1[8]. `JWPL` (Java-based WikiPedia Library) is a Java-based application programming interface that allows to access Wikipedia. `OWL-S API` provides a programmatic access to OWL-S service descriptions for extending non-functional properties with respect to fuzzy QoS ontologies.

Our fuzzy matchmaking algorithm has been implemented in `Matlab`. We also use the `Fuzzy OWL 2` Protégé plug-in for fuzzy ontology representation (in OWL 2 with annotations) and the fuzzy DL reasoners `DeLorean` (to translate fuzzy ontologies) and `fuzzyDL` (for fuzzy SWS fuzzy selection).

b) **Test case.:** We have chosen the travel domain for our experiments. The first step was to generate a fuzzy domain ontology. To this end, we collected a corpus. For simplicity, it was simply based on Wikipedia, by collecting all the pages and subcategories related to the travel domain. Our corpus contains almost 500 pages. The travel ontology in the OWL-S Service Retrieval Test Collection (OWL-TC)[9] was used for the domain vocabulary generation part of our framework. As a result, our fuzzy domain ontology contains a fuzzy taxonomy between the concepts in the travel domain.

Then, we randomly selected some SWS advertisements in the test collection and replaced their concepts with fuzzy concepts from the fuzzy ontology.

We also added some non-functional properties (such as QoS). We developed a fuzzy extension of the Qos ontology in [12], by providing some annotations to describe the values of some properties by means of fuzzy datatypes.

[5]http://nlp.stanford.edu/software/lex-parser.shtml
[6]http://nlp.stanford.edu/software/tagger.shtml
[7]http://code.google.com/p/jwpl
[8]http://on.cs.unibas.ch/owls-api
[9]http://projects.semwebcentral.org/projects/owls-tc/

c) **Experiments.:** To evaluate our fuzzy SWS description framework we compared the retrieval performance of the fuzzy SWS matchmaking based on our fuzzy extension of OWL-S with the fuzzy matchmaking algorithm in [11] in one particular case (two service requests).

Our system is evaluated in terms of precision and recall measures. We consider the evaluation of micro-average of the individual precision-recall curves, using a number $\lambda = 20$ of steps up to reach the highest recall value [11]. The micro-averaging of recall and precision (at step λ) are defined as $Rec_\lambda = \sum_R |D_R \cap B_{\lambda,R}|/|D|$ and $Prec_\lambda = \sum_R |D_R \cap B_{\lambda,R}|/|B_\lambda|$, respectively, where D_R is the answer set of relevant service advertisements for a given request R, B_λ is the set of retrieved OWL-S descriptions at the step λ and $B_{\lambda,R}$ is the set of all relevant OWL-S descriptions at step λ.

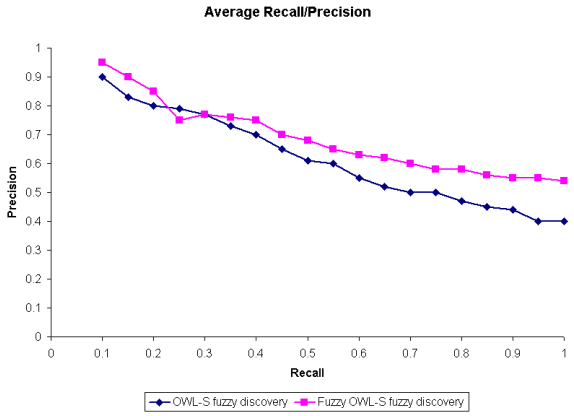

Fig. 3. Comparison of precision/recall with hybrid OWL-S approach.

We computed precision and recall of two small variants of two service requests from OWL-TC (citycountry-hotel-service and title-comedyfilm-service), modified to use fuzzy concepts from our fuzzy ontologies. As shown in Figure 3, both approaches show comparable results, but in this case our matchmaking using fuzzy ontologies produces a slightly better performance. The execution time is the same as the algorithm is the same (we only change the input matrix). Additionally, our approach allows to use NFPs to rank the services in a fuzzy SWS selection step.

V. Conclusions and Future Work

This paper presents a fuzzy framework for SWS description, discovery and selection. The purpose of this research is to expand the scope of the existing descriptions of Web services, enabling them to describe uncertain knowledge, and to provide an integrated solution for fuzzy description-based service organization and matching.

In particular, we extend the semantic description of the functional properties of SWSs by using a fuzzy domain ontology. We also enhance the semantic description of the non-functional properties by using fuzzy Quality of Service ontologies. Non-functional properties play an important role in Web service selection in order to evaluate and rank candidate SWSs, so extending them by using fuzzy ontologies produces a more powerful formalism for SWS description.

Up to now, we have shown that our approach may perform better in some cases. Future work will include a more rigorous evaluation of the fuzzy SWS selection, modifying more SWS in the OWL-S Service Retrieval Test Collection with fuzzy ontologies and non-functional properties.

References

[1] Amigo Project. Amigo middleware core: Prototype implementation and documentation. Deliverable D3.2, 2006. http://www.hitech-projects.com/euprojects/amigo/deliverables/amigo-d3.2-final.pdf.

[2] T. Berners-Lee, J. Hendler and O. Lassila. The semantic web. Scientific American, 284(5):34–43, 2001.

[3] J. C. Bezdek. Pattern recognition and fuzzy objective function algorithms. Plenum Press, 1981.

[4] F. Bobillo, M. Delgado and J. Gómez-Romero. DeLorean: A reasoner for fuzzy OWL 1.1. In Proceedings of the 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008), vol. 423. of CEUR Workshop Proceedings, 2008.

[5] F. Bobillo, M. Delgado and J. Gómez-Romero. Crisp representations and reasoning for fuzzy ontologies. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 17(4):501–530, 2009.

[6] F. Bobillo, J. Gómez-Romero and R. Pérez-Pérez. Semantic web services: A brief overview. In Proceedings of the IADIS International Conference WWW/Internet 2005, pp. 1–8, 2005.

[7] F. Bobillo, E. Sharifi, M. M.. Ebadzadeh and R. A. Moghadam. On fuzzy semantic web services. In Proceedings of the 19th IEEE Int. Conference on Fuzzy Systems (FUZZ-IEEE 2010), pp. 2418–2423, 2010.

[8] F. Bobillo and U. Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008), pp. 923–930. 2008.

[9] F. Bobillo and U. Straccia. Representing fuzzy ontologies in OWL 2. In Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010), pp. 2695–2700, 2010.

[10] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider and U. Sattler. OWL 2: The next step for OWL. Journal of Web Semantics, 6(4):309–322, 2008.

[11] G. Fenza, V. Loia and S. Senatore. A hybrid approach to semantic web services matchmaking. International Journal of Approximate Reasoning 48(3):808–828, 2008.

[12] Hydra project. Quality-of-Service enabled Hydra middleware. Deliverable D4.10, 2009. http://www.hydramiddleware.eu/hydra_documents/D4.10_QoS_Enabled_Hydra_Middleware.pdf.

[13] M. Khalid, K. Sycara, M. Klusch and B. Fries. OWLS-MX: Hybrid semantic web service retrieval. In Proc. of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web, AAAI Press, 2005.

[14] K. Kritikos and D. Plexousakis. Semantic QoS-based web service discovery algorithms. In Proceedings of the 5th European Conference on Web Services (ECOWS 2007), IEEE Computer Society, 2007.

[15] R. Y. K. Lau. Fuzzy domain ontology discovery for business knowledge management. Intelligent Informatics Bulletin, 8(1):29–41, 2007.

[16] R. Y. K. Lau. Towards a fuzzy domain ontology extraction method for adaptive e-learning. IEEE Transactions on Knowledge and Data Engineering, 21(6):800-813, 2009.

[17] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics, 6(4):291–308, 2008.

[18] H. Schütze. Automatic word sense discrimination Computational Linguistics 24(1):97–124, 1998.

[19] V. X. Tran, H. T. and Ryosuke Masuda. A new QoS ontology and its QoS-based ranking algorithm for Web services. International Journalof Simulation Modelling Practice and Theory 17(8):1378–1398, 2009.

[20] W3C. OWL Web Ontology Language overview, 2004. http://www.w3.org/TR/owl-features/.

[21] W3C. OWL-S: Semantic markup for web services, 2004. http://www.w3.org/Submission/OWL-S.

[22] B. Yin, H. Yang, P. Fu and X. Chen. A semantic web services discovery algorithm based on QoS ontology. In Proceedings of the 6th International Conference on Active Media Technology (AMT 2010), Lecture Notes in Computer Science 6335, pp.166–173, 2010.

[23] L. A. Zadeh. Fuzzy sets. Information and Control, 8:338–353, 1965.