A Semantic and QoS-aware Broker for Service Discovery

Malika Charrad, Nadia Yacoubi Ayadi and Mohamed Ben Ahmed
RIADI Laboratory
National School of Computer Science,
Manouba University, 2010 La Manouba
Email: malika.charrad.1@ulaval.ca, nadia.yacoubi@riadi.rnu.tn, mohamed.benahmed@riadi.rnu.tn

As the Web is increasingly used not only to find answers to specific information needs but also to carry out various tasks by the means of Web services, enhancing the capabilities of current Web search engines with effective and efficient techniques for Web service discovery becomes an important issue. In this paper, we propose a semantic and QoS-aware broker, namely SemQoS, that enables users to discover Web services based on both functional and non-functional criteria. The SemQoS broker relies on a semantic canonical description meta-model that allows describing services in terms of functional and non-functional properties. First, SemQoS discovery approach selects a set of services matching a discovery query with different degrees of approximate matching. The approach relies on deductive reasoning to relax the discovery query’s constraints based on semantic service descriptions and domain knowledge. As a second step, the SemQoS broker refines the set of selected services to generate a set of service clusters based on their QoS attributes values. This step allows to prune the search space of a discovery query and determine the best sub-set (cluster) of Web services satisfying user’s constraints. We evaluate our approach experimentally using a real Web services dataset.

Keywords: Web Service discovery, broker, reasoning, Quality of Service, clustering, ranking
ACM Classification: H.3.5, H.3.3

1. Introduction

Web services constitute a key technology for realizing Service Oriented Architectures (SOA), enabling loose coupling and interoperability among heterogeneous information systems and platforms. As the technology of Web services proliferates, it becomes more problematic to find and locate a specific service that can perform a given task. The Web Service Discovery (WSD) process involves the matchmaking of a set of functional and other non-functional criteria with a set of service descriptions. The WSD problem becomes especially important and challenging as the number of services is increasing. We distinguish two prominent steps in the WSD process. First, a requester discovers services based on functional properties that may include input and output parameters, preconditions, postconditions or service category. This phase ensures that the returned services meet the requester’s functional requirements. The second phase is to identify the most appropriate service among those selected in the first step. This step is usually performed based on non-functional properties, such as information about the service’s provider and QoS...
A Semantic and QoS-aware Broker for Service Discovery

(Quality of Service) attributes. QoS properties are used during the service discovery and selection process as additional criteria to provide a ranking of the matched results. A service rank is a quantitative metric that in some way shows the importance of a service within a Web service sub-set.

On another side and during the last ten years, Semantic Web Services (SWS) technology (McIlraith 

et al., 2001) has tried to overcome the WSD problem by enriching Web service descriptions with semantic annotations in order to better support their smooth discovery, composition and invocation by machines. Services annotations are highly valuable information that may inform about existing reusable functionalities exposed somewhere. Semantic WSD proposes to make use of services annotations and the background ontological knowledge used to create them. However, the impact of SWS has been minimal for several reasons. First, research has mostly focused on devising highly expressive conceptual models (OWL-S (Martin et al., 2007) and WSMO (Roman et al., 2005)) and has given birth to a number of diverging and sometimes incompatible solutions of service matchmakers. Moreover, these frameworks introduce complexity and demand additional effort from users to understand their conceptual foundations.

In this paper, we present the SemQoS broker as a semantic and QoS-aware broker enabling the selection of services based on both their functional and non-functional properties. SemQoS relies on an RDF-S semantic description model presented in Ayadi and Ahmed (2010). Our semantic model represents core service semantics by a set of lightweight semantic annotations. Existing semantic Web services expressed in various formats such as WSMO, OWL-S or SAWSDL are mapped to canonical descriptions according to the proposed semantic model. Thus, the SemQoS broker is able to perform uniform search over different semantic service formats by automatically generating the corresponding set of RDF statements annotations which are stored in the SemQoS Web service catalog. Considering a discovery query, functional matchmaking in SemQoS is performed based on ontology-based deductive reasoning that enables to select a set of services that satisfy the query with different degrees of approximate matching. Moreover, we adopt in SemQoS two ranking strategies. The ranking is performed in terms of QoS properties. The first strategy generates, in a first time, a set of service clusters based on the agglomerative clustering method (Hastie et al., 2009). The second ranking strategy enables to deduce an ordered set of services belonging to the same cluster based on a distance metric.

The SemQoS ranking approach has the advantage to rank service clusters (instead of individual services) such as in case a given service is not available or its QoS has decreased, a user could select another service from the cluster.

The paper is structured as follows. Section 2 reviews the most prominent works that investigated the WSD problem. In Section 3, we present the SemQoS broker architecture. In Section 4, we present the functional matchmaker component of the SemQoS broker. Section 5 presents clustering and ranking techniques adopted in the SemQoS broker. In Section 6, we describe experimentations conducted with a real-world services dataset. Finally, Section 7 outlines our conclusions and our future work.

2. Background Research

The state-of-the-art of WSD approaches ranges from keyword-based search engines to ontology-based frameworks for discovering semantic Web services. Keyword based search approaches as well as category browsing such as Google or seekda!^1 provide a significantly higher number of

1 http://webservices.seekda.com/
WSDL files for a given keyword-based user request which usually lead users to choose manually the proper service according to its semantics (Bachelechner et al., 2006). Obviously, the lack of semantic description in WSDL prevents automatic discovery, selection, invocation and composition. To deal with these issues, several approaches were developed in order to offer rich and semantic descriptions of Web services. Top-down approaches such as WSMO (Roman et al., 2005) and OWL-S (Martin et al., 2007) use high-level ontologies as frameworks for describing Web services. On the other hand, bottom-up approaches such as SAWSDL (Kopecky et al., 2007) and YASA (Chabeb and Tata, 2008) adopt an incremental approach to adding semantics to existing Web service standards by specifying specific extensions to WSDL that connect the syntactic definitions to semantic annotations. Several semantic discovery approaches, such as those proposed in Paolucci et al. (2002), Domingue et al. (2008) and Facca et al. (2009), rely on deductive reasoning to infer logical equivalence and subsumption between query’s concepts and service descriptions. Hybrid approaches such as OWLS-MX (Klusch et al., 2009) and OWLS-SLR (Meditskos and Bassiliades, 2010) complements logic-based semantic matchmaking of services with information retrieval (IR)-based syntactic similarity measurements in case the former fails. The results of the experimental evaluation of hybrid approaches provide strong evidence for the claim that logic-based semantic matching of services can be significantly improved by incorporating non logic-based IR techniques.

In this paper, we propose a canonical and lightweight model to describe Web services. Similar to our model, recent research work tend to propose lightweight representation of semanticWeb services such as WSMO-Lite (Vitvar et al., 2008) or MicroWSMO dedicated to Web API and RESTful services. The SemQoS broker, proposed in this paper, incorporates a reasoning engine that enables approximate matchmaking of Web services. We complement logic-based matchmaking with clustering process that consider the QoS values of Web services to perform services clustering and ranking.

3. SemQoS Broker: Architecture and Overview

The architecture of the SemQoS broker is presented in Figure 1. It is composed of a Web service catalog, a reasoning engine and a clustering and ranking engine. In the SemQoS catalog, Web services are described in terms of a set of functional and non-functional properties according to the canonical semantic Web service description model presented in Ayadi and Ahmed (2011).

The service description model consists on an RDF-S data model that captures the core semantics of a Web service. As depicted in Figure 2, a service is declared by an RDF triple as follows (s, rdf:type, isService). Each service has a possibly empty set of inputs declared by a set of RDF triples (c, hasInput, input) and a non empty set of outputs defined by a set of RDF triples (c, hasOutput, output).

Also, a service s has a possibly empty set of preconditions and effects, and an optional Category or service task. The inputs and outputs of a service describe respectively data necessary for the execution of the service and resulting from the execution of the service. A service is also characterized by a category, which is the description of the provided functionality.

The RDF-S model allows also to describe services from the non-functional view. A service is also described by a possibly empty set of non-functional properties defined by a set of RDF triples (s, hasQoSproperty, QoSPropertyName) which is Quality-of-Service (QoS) property. Each QoS property is defined by a triple (p, v, u) where p is QoS parameter name, v is a value’s parameter and u is the unit in which a value v is expressed. Non-functional properties are represented by metrics that evaluate the Web service performance, dependability and reputation. The performance of the Web
A Semantic and QoS-aware Broker for Service Discovery

Figure 1: The SemQoS Broker Architecture

Figure 2: The RDF-S Service Data Model
service is measured by metrics such as throughput, response time, execution time and resource utilization. The dependability of a Web service integrates different attributes including reliability, availability and security. The reputation of a Web service reflects the consumer perception towards the service. In this paper, we have considered the following QoS metrics, which are usually relevant for Web services (Wu and Wu, 2010).

- **Availability** which is the probability that the system is up and can respond to consumer requests. Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time.

- **Reliability** which is the ability of a service to perform its required functions under stated conditions for a specific period of time. The number of successfully used per month represents a measure of reliability of a Web service.

- **Performance** is the measure of the speed to complete a service request. It is measured by latency, throughput and response time.
  - **Throughput** represents the number of Web service requests served at a given time period.
  - **Latency** is the round-trip time between sending a request and receiving the response.
  - **Response time** is the time required to complete a Web service request.

- **Successibility** represents the number of request messages that have been responded.

- **Compliance** represents the extent to which a WSDL document follows WSDL specification.

Given a set of service descriptions comprised by the SemQoS broker catalog, discovery queries express functional and non-functional user’s constraints defined as follows:

**Definition 1. A Discovery Query.** A query $Q$ is a 2-tuple $(F, NF)$, where $F$ represents a set of functional requirements; $NF$ represents a set of non-functional requirements. The functional requirement $F$ is represented by a pair $(I, O)$, where $I$ is a set of input concepts and $O$ is a set of output concepts that are returned by the requested service. The non-functional requirement $NF$ is represented by a set of 4-tuples $(P, C, V, U)$, where, $P$ is a QoS parameter, $C$ is a comparator; $V$ is a value and $U$ is a unit; the evaluation of the query $Q$ has to respect the conjunction of the conditions expressed in the set $NF$.

The discovery query processing is performed in two steps. First, the reasoning engine performs the functional matchmaking of $Q$ and then, the clustering and ranking engine prune the search space of the query to deduce the best subset of candidate services in terms of their corresponding QoS properties. In the following, we define the functional and non-functional satisfiability of a query $Q$.

**Definition 2. Functional satisfiability.** Let $Q=(F, NF)$ and $S=(f, nf)$ be a discovery query and a canonical service description, respectively. The service $S$ satisfies the functional requirements of the query $Q$, - If $F = (I, O)$, then $I_S \subseteq I$ and $O \subseteq O_S$

**Definition 3. Non-functional satisfiability.** Let $Q=(F, NF, Dk, k)$ and $S=(i,o, nf)$ be a discovery query and a service description, respectively. The service $S$ satisfies the non-functional requirements of a query $Q$ if $NF$ is a set of 4-tuple $(P, C, V, U)$, $nf$ is a set of triples $(p, v, u)$ then, for each 4-tuple $(P, C, V, U) \in NF$, there is a triple $(p, v, u) \in nf$, such that:

- $P$ and $p$ are equal
- The expression $v \ C \ V$ is valid, where $C \in \{>, \geq, <, \leq, =, \neq\}$
- Both parameters are expressed in the same unit.
4. Functional Service Matchmaking in SemQoS

Functional matchmaking in SemQoS is performed thanks to reasoning services. We present in this section the reasoning services offered by the SemQoS reasoning engine. Indeed, we distinguish hierarchy relaxation axioms that exploit concept taxonomy and predicate relaxation axioms that exploit semantic relations between concepts in a domain ontology (Table 1). We explain in the following the different types of reasoning services.

1. Hierarchy relaxation which exploits subsumption relationships defined relating concepts in a domain ontology O. We distinguish two hierarchy relaxation axioms:

- **The input specialisation axiom:** (A1) asserts that if a concept C is declared as an input concept of a service S then all sub-classes of C are asserted as input types of the service. For instance, the service Blastn accepts as input a nucleotide sequence, an organism identifier, and a database name. RNA and DNA are two specific types of nucleotide sequences. mRNA (messenger RNA) is a specific type of RNA sequence. The input specialisation axiom allows to infer new input types for the Blastn service, i.e., RNA, DNA, mRNA. This means that the Blastn service might be invoked with a more specific input than expected (Figure 3).

- **The output generalisation axiom:** (A2) asserts that if a concept C1 is declared as an output of a service S then all super-classes of C2 are asserted as output types of the service. The Blastn returns a set of nucleotide sequences which are similar to the input nucleotide sequence. The output generalisation allows to infer that if a service returns a collection of nucleotide sequences then it is possible to deduce that the service returns a collection of genomic sequences. This allows to consider a service that returns a more specific output than specified by the query.

\[
(A_1) \quad (S, \text{hasInput}, i_1)(i_1, \text{rdf:type}, C_1)(C_2, \text{subClassOf}, C_1) \\
(S, \text{hasInputType}, C_2)
\]

\[
(A_2) \quad (S, \text{hasOutput}, o_1)(o_1, \text{rdf:type}, C_1)(C_1, \text{subClassOf}, C_2) \\
(S, \text{hasOutputType}, C_2)
\]

\[
(A_3) \quad (S, \text{hasInput}, i_1)(i_1, \text{rdf:type}, C_1)(C_1, P, C_2)(P, \text{rdf:type}, \text{funcProperty}) \\
(S, \text{hasInputType}, C_2)
\]

\[
(A_4) \quad (S, \text{hasOutput}, o_1)(o_1, \text{rdf:type}, C_2)(C_1, P, C_2)(P, \text{rdf:type}, \text{invfuncProperty}) \\
(S, \text{hasOutputType}, C_1)
\]

Table 1: Relaxation axioms

2 Nucleotide Blast compares a nucleotide query sequence against a nucleotide sequence database using the Washington University BLAST algorithm. The service might be invoked from http://www.biocatalogue.org/soap/services/2551/latest.wsdl
2. Predicate relaxation exploits semantic relations defined between concepts in a domain ontology \( O \) specified by a triple \((C_1, P, C_2)\) where \( C_1 \) and \( C_2 \) are concepts, \( P \) is the predicate that relates \( C_1 \) and \( C_2 \) and \( P \) is a functional or inverse functional property. We distinguish two types of axioms: the predicate to domain axiom and predicate to range relaxation axiom defined as follows:

- **Predicate to domain axiom**: \((A3)\) allows to infer a new triple \((S, \text{hasInputType}, C_2)\) from the triples \((c, \text{hasInput}, i_1)\), \((i_1, \text{rdf:type}, C_1)\), \((C_1, P, C_2)\), \((P, \text{dom}, C_1)\), and \((P, \text{rdf:type}, \text{functionalProperty})\). For example, the WSInterProScan\(^3\) is a Web service that allows to query the InterPro database. The Web service accepts an Amino Acid Sequence which could be functionally related to a Protein Identifier in the InterPro database.

- **Predicate to range axiom**: \((A4)\) allows to infer a new triple \((S, \text{hasOutputType}, C_1)\) from the triples \((S, \text{hasOutput}, o_1)\), \((o_1, \text{rdf:type}, C_2)\), \((C_1, P, C_2)\), \((P, \text{range}, C_2)\), and \((P, \text{rdf:type}, \text{InversefunctionalProperty})\). Given a protein identifier, the PDB\(^4\) Web service returns a 3D structure of a known protein. A 3D structure is modelized from an Amino Acid Sequence. IsModelized is an inverse functional property because of the fact that to a given amino acid sequence, it is possible to visualize one 3D structure. The predicate to range axiom allows to infer that the PDB Web service returns the amino sequence corresponding to the 3D structure.

---

\(^3\) WSInterProScan is a Web service for protein domain and family signature searches (see http://www.ebi.ac.uk/Tools/webservices/services/interproscan).

\(^4\) The PDB Web service offers RESTful interface to search and retrieve data from the Protein Data Bank (PDB). The Protein Data Bank is the single worldwide repository of information about the 3D structures of large biological molecules, including proteins and nucleic acids.
5. Services Clustering and Ranking in SemQoS

In this section, we propose to cluster Web services satisfying functional requirements of a discovery query, denoted $S_f$. The clustering is based on non-functional attributes. Clusters obtained are then ranked to select the best clusters $C_i \subset S_f$ composed of services that meet non-functional user’s requirements. The last step in our approach allows ranking Web services inside the best clusters. This double ranking aims to reduce the search space $S_f$ into $C_i$ denoting the cluster of services having the best QoS attribute values among all Web services that satisfy functional requirements.

5.1 QoS Values Normalization

QoS attributes are divided into two classes: negative attributes (e.g., response time, latency) and positive attributes (e.g., availability, reliability, throughput, successibility, compliance). The first class of attributes has a negative effect on QoS, hence they need to be minimized. On the contrary, positive QoS attributes need to be maximized since they increase the overall QoS. Normalization of QoS attributes consists in transforming it into a value between 0 and 1 with respect to the formulas below presented in Zeng et al (2004) and Mabrouk et al (2009).

Negative attributes:
$$q'_{ij} = \begin{cases} \frac{q_{ij}^{\text{max}} - q_{ij}}{q_{ij}^{\text{max}} - q_{ij}^{\text{min}}} & \text{if } q_{ij}^{\text{max}} - q_{ij}^{\text{min}} \neq 0 \\ 1 & \text{else} \end{cases}$$

Positive attributes:
$$q'_{ij} = \begin{cases} \frac{q_{ij} - q_{ij}^{\text{min}}}{q_{ij}^{\text{max}} - q_{ij}^{\text{min}}} & \text{if } q_{ij}^{\text{max}} - q_{ij}^{\text{min}} \neq 0 \\ 1 & \text{else} \end{cases}$$

Where $q'_{ij}$ denotes the normalized value of QoS attribute $j$ associated with service candidate $s_i$. It is computed using the current value $q_{ij}$ and $q_{ij}^{\text{max}}$ and $q_{ij}^{\text{min}}$ which refer respectively to the maximum and minimum values of QoS attribute $j$ among all service candidates.

5.2 Web Service Clustering

Clustering is the process of organizing objects into groups whose members are similar in some way. It is an unsupervised learning technique that is widely used in Artificial Intelligence and Data Mining. A cluster is a collection of objects that are “similar” between them and are “dissimilar” to the objects belonging to other clusters. Clustering is used in this work to identify related or similar Web services based on QoS values. In the present paper, we use bottom-up Hierarchical Clustering method (Hastie et al, 2009), called hierarchical agglomerative clustering or HAC. In this method, each Web service starts in its own cluster. Then, we use a similarity matrix of Web services to determine the nearest neighbours. Nearest clusters are then merged into one cluster as one moves up the hierarchy until all the Web services merge to a single cluster. Indeed, Web services described by $m$ QoS attributes are organized into a matrix $A$. $A = [V(a_{ij})]$, $1 \leq i \leq m; 1 \leq j \leq n$, refers to a collection of quality attribute-values for $n$ Web services that satisfy functional requirements of a query, such that, each row of the matrix corresponds to a particular candidate service and each column refers to the value of a particular QoS attribute (in which the user is interested). $V(a_{ij})$ represents the value of the $i^{th}$ QoS attribute for the $j^{th}$ candidate service. These values are mapped to a scale between and after normalization.
In order to decide which clusters should be combined, a measure of dissimilarity between sets of services is required. This is achieved by use of an appropriate metric (a measure of distance between pairs of services), and a linkage criterion (average linkage in this work) which specifies the dissimilarity of sets as a function of the pairwise distances of services in the sets. In this work, we use the average linkage clustering as linkage criteria and the squared Euclidian distance as metric.

After clustering, a new matrix $A'$ is obtained. $A' = \{(V(a'_{ij}), 1 \leq i \leq m; 1 \leq j \leq K}\}$. $K$ is the number of clusters.

Instead of comparing each service in the search space with the requested service, only services that belong to the best cluster are compared. Each cluster is represented by its centroid vector. The best cluster is selected on the base of values of attributes in the centroid vector using an additive value function, $f_{QoS\text{cluster}}$. We notice that, in order to give relative importance to the various attributes, the users can specify a weight value for each attribute, which are used along with the QoS attribute values to give relative score to each cluster using $f_{QoS\text{cluster}}$.

5.3 Web Service Ranking

Inside a given cluster, we introduce the notion of ranking attributes and a ranking function (based on those attributes), which are used to rank the candidate service that match both functional and nonfunctional user’s requirements.

Definition 4. Ranking Function

Let $C$ be a cluster of candidate services which match the functional and non-functional requirements of the user, $O \in \{\text{ascending, descending}\}$ is the ranking order, $D = \{d_1, d_2, \ldots, d_m\}$ is the set of distances between each service $s \in C$ and the centroid $c$ of the cluster $C$. $f_{\text{Rank}}(C,D,O) = C'$, is called the ranking function, which produces $C'$, the ordered set of candidate services.

For example, let $C' = \{s_1, s_2, s_3\}$ be a cluster of services selected based on the desired QoS properties (from the previous section/example), and $O = \text{ascending}$.

Assuming $d_3 < d_1 < d_2$ i.e. $\text{distance}(s_3, c) < \text{distance}(s_1, c) < \text{distance}(s_2, c)$, we have, $f_{\text{Rank}}(C,D,O) = \{s_3, s_1, s_2\} = C'$.
6. Experimental Results

The performance of the reasoning engine was presented in Goncalves et al. (2010). The authors have shown how relaxation techniques can enlarge the search space of a discovery query. In this paper, we concentrate on the performance of clustering and ranking engine. We conducted our experiments using the QWS dataset\(^5\) publicly available. These services were collected from public sources on the Web, including UDDI registries, search engines and service portals, and their QoS values were measured using commercial benchmark tools. More details about this dataset can be found in Al-Masri and Mahmoud (2008). Each Web service is described by its name and seven non-functional properties, namely: response time, availability, throughput, successability, reliability, compliance and latency.

For evaluation purpose, we have considered a scenario where a user is seeking bioinformatic services enabling protein data retrieval from different databases.

The set of candidate services obtained is comprised by 40 services that match functionally a discovery query. The clustering algorithm has generated five clusters based on similarity of these Web services. The cluster dendrogram is depicted in Figure 4.

If we assume that all attributes have the same weight (weight\(_i\) = 1, 1 ≤ i ≤ m), the top-2 best clusters are cluster 2 and cluster 3. Cluster 2 (f\(_{QoS}\)\(_{cluster} = 5.79\)) is composed of only one service “SeqReferenceInfoEJB”. Cluster 3 (f\(_{QoS}\)\(_{cluster} = 5.4\)) is composed of 15 Web services (Figure 4).

$$A' = \begin{bmatrix}
\text{Avail.} & \text{Succ.} & \text{Throu.} & \text{Re sp.} & \text{Re lia.} & \text{Compl.} & \text{Laten.} & f_{QoS}\text{cluster} \\
C_1 & 0.69 & 0.72 & 0.45 & 0.78 & 0.12 & 0.54 & 0.58 & 3.88 \\
C_2 & 0.84 & 0.95 & 1.00 & 1.00 & 1.00 & 1.00 & 5.79 \\
C_3 & 0.80 & 0.93 & 0.25 & 0.83 & 0.80 & 1.00 & 0.79 & 5.40 \\
C_4 & 0.50 & 0.51 & 0.00 & 0.00 & 0.13 & 0.50 & 0.00 & 1.64 \\
C_5 & 0.00 & 0.00 & 0.53 & 0.86 & 0.73 & 0.00 & 0.94 & 3.06 \\
\end{bmatrix}$$

Figure 4: Cluster Dendrogram

\(^5\) http://www.uoguelph.ca/~qmahmoud/qws/index.html
Table 2 provides a ranking of services belonging to cluster 3. The best services that fit functional and non-functional requirements are S11, S32 and S35, which represent respectively fuzzyproService, pepcoilService and sigcleaveService. GarnierService (S8) is in the second place ranking (distance to the cluster 3 centroid is $d(S8,c3) = 0.074$). In this case, the user can choose one of these three services.

### 7. Conclusion

The main goal of this work has been to address the WSD problem by considering both functional and non-functional properties of Web services. Thus, we have proposed the semantic and QoS-aware SemQoS broker. The SemQoS discovery approach is two-fold. First, the SemQoS reasoning engine performs relaxation of Web services descriptions based on domain knowledge in order to deduce approximate matching between query’s constraints and service descriptions. Second, the clustering and ranking engine organize the set of selected services into clusters based on their non-functional attributes in order to guide users to select the most appropriate (best) services. In our approach, user preferences are not considered. Thus, as future work, we aim to propose techniques to represent and exploit user preferences and feedbacks in order to bootstrap the quality of the SemQoS broker answers. Finally, to improve the quality of service annotations, we propose to make use of machine learning techniques to automatically generate annotations from textual descriptions of services and considering domain ontologies as background knowledge.
A Semantic and QoS-aware Broker for Service Discovery

References


Biographical Notes

Malika Charrad received the PhD degree in computer science from Conservatoire National des Arts et Métiers, France, in 2010. She is currently a postdoctoral researcher at Université Laval. Her principal research interests include machine learning and Web mining.

Nadia Yacoubi Ayadi received the PhD degree in computer science from Conservatoire National des Arts et Métiers, France, in 2010. She is currently assistant professor at High Institute of Management, Tunisia. Her research interests include semantic Web and Web services paradigms.

Mohamed Ben Ahmed is currently émeritus professor at Manouba University. His research interests include natural language processing and web services discovery.