

Matchmaking by difference in Service Discovery ^{*}

Devis Bianchini¹ Valeria De Antonellis¹ Michele Melchiori¹

Università degli Studi di Brescia - Dip. di Elettronica per l'Automazione
Via Branze, 38 - 25123 Brescia - Italy
{bianchin|deantone|melchior}@ing.unibs.it

Abstract. Service discovery over the Web is considered a crucial issue; in particular, flexibility of the discovery process, that is, the ability of recognizing not only exact matches between the requests and offers, but also partial ones, needs to be enhanced. We have defined a composite approach to flexible service matchmaking with different matching models. In this paper we present the novel difference-based matching model. The approach is based on an ontological framework adding semantics to service descriptions. Optimization and ranking techniques are provided.

1 Introduction

Enterprise interoperability in distributed environments has been recently improved by adopting the emerging Web Service standards and technology, making available on the net an ever-growing number of services. Searching for specific service capabilities, many services can be found able to fully or partially satisfy the request. As a consequence, advanced service discovery approaches are being developed in order to find the best suitable offers for a given request. In particular, flexibility of the discovery process, that is, the ability of recognizing not only exact matches, but also partial ones by evaluating the degree of match between the request and each offer, needs to be considered. In fact, according to the degree of flexibility it is possible to reduce or augment the likelihood of finding services matching a given request.

In the literature several service discovery approaches have been presented, based on different service matchmaking techniques. In particular, we distinguish techniques that rely on deductive reasoning and techniques based on similarity evaluation. In general, techniques based on deductive reasoning (e.g., [5, 8]), present high precision and recall, but are often characterized by low flexibility. On the contrary, similarity-based techniques (e.g., [7]) are characterized by high flexibility, but limited precision, because, for example, if a partial match is established, there is no way to know if more functionalities are required than the provided ones or viceversa. A comparison of deductive and similarity-based approaches shows that the former ones are able to distinguish between the request and the offer viewpoints, but do not provide a quantification of how much the offer matches with the request, while the latter approaches are symmetric, not distinguishing between the request and the offer, but provide a quantification of the degree of match.

^{*} This work has been partially supported by the ESTEEM PRIN Project funded by the Italian Ministry of Education, University and Research, and by NoE INTEROP IST Project n. 508011 - 6th EU Framework Program.

With respect to the previous approaches, we have proposed in [2] a novel hybrid matchmaking strategy that first uses a reasoning procedure based on Description Logics to precisely establish the kind of match between the request and each offered service, then ranks the partially matching services on the basis of their similarity. In this paper we focus on a novel deductive approach [3] that, instead of evaluating the degree of similarity between the request and the offers, is devoted to compute the difference between the requested capabilities and the offered ones. Main advantages of this matching model is that users can choose the most suitable offers having knowledge about exceeding/missing capabilities. This kind of matching by difference can be exploited in P2P environments for complex service composition. In such a context, peers can maintain information about contents provided by their neighbors. When a peer receives a request, it finds out the capabilities that cannot provide by applying the difference matching. At this point, the peer formulates a new request, containing only missing capabilities, and forwards it to the peers that are able to provide them.

The paper is organized as follows: in Section 2 the ontological framework is presented, while in Section 3 the matchmaking approach is described; Section 4 presents the algorithm for matching by difference and Section 5 concludes the paper.

2 Service description and the ontological framework

Starting from the WSDL service specification [6], the expected capabilities of the service can be identified in the operations offered by the service and in the output parameters associated to operations. These elements are automatically extracted from the WSDL documents together with service category and a Description Logic (DL) expression is built to represent a service. The DL expression is a conjunction of:

- a concept in the form $\exists\text{hasCategory}.CAT$, where CAT is a concept which represents the associated service category;
- one or more concepts in the form $\exists\text{hasOperation}.OP$, where OP is a concept described as a conjunction of:
 - an atomic concept representing the name of a service operation;
 - a conjunction of one or more concepts $\exists\text{hasOutput}.OUT$, where OUT is a concept representing an output parameter of the operation and can be defined as an atomic concept, an enumeration $\{o_1, o_2, \dots, o_n\}$ of nominals or a complex concept obtained by applying the *intersection* operator (\sqcap), the *union* operator (\sqcup) and the *negation* operator (\neg).

Example 1. We consider a running example in the domain of geographic information services, where are required services displaying maps with different kinds of information such as aerial photos, streets, gas pipes, water pipes and so on. Let consider the following Description Logic expressions representing a request \mathcal{R} and two advertisements $\text{DisplayGasInfrastructure}$ and $\text{DisplayTransportInfrastructure}$ in this domain:

$$\begin{aligned}
 \mathcal{R} &\equiv \exists\text{hasCategory}.GeographyInformationService \sqcap \\
 &\quad \exists\text{hasOperation}.(\text{viewDetailedMap} \sqcap \\
 &\quad \quad \exists\text{hasOutput}.gasPipes \sqcap \\
 &\quad \quad \exists\text{hasOutput}.waterPipes \sqcap \\
 &\quad \quad \exists\text{hasOutput}.urbanStreets) \\
 \text{DisplayGasInfrastructure} &\equiv \exists\text{hasCategory}.GeographyInformationService \sqcap \\
 &\quad \exists\text{hasOperation}.(\text{displayMoreGrainedMap} \sqcap \\
 &\quad \quad \exists\text{hasOutput}.gasPipes \sqcap \\
 &\quad \quad \exists\text{hasOutput}.streets) \\
 \text{DisplayTransportInfrastructure} &\equiv \exists\text{hasCategory}.GeographyInformationService \sqcap \\
 &\quad \exists\text{hasOperation}.(\text{viewDetailedMap} \sqcap \\
 &\quad \quad \exists\text{hasOutput}.gasPipes \sqcap \\
 &\quad \quad \exists\text{hasOutput}.transportWays) \quad \square
 \end{aligned}$$

To improve the effectiveness and the efficiency of service matchmaking, additional semantics is associated to service description. In particular, an ontological framework is defined for semantic enrichment [2]. The ontological framework is constituted by three components: (i) a Domain Ontology $DomONT$, used to conceptualize the domain knowledge related to the names of elements used in service descriptions (operation names and output parameters), expressing it in terms of *concepts* and *semantic relationships* between them; (ii) a Thesaurus \mathcal{TH} (automatically derived from available lexical systems such as WordNet), used to relate names of concepts of the Domain Ontology to other terms by means of terminological relationships (e.g., *synonymy*, *hypernymy*, etc.), in order to extend matching possibilities between the concept names used in the service request and the names used in the descriptions of provided services; (iii) a Service Ontology $ServONT$, that organizes services on three layers of abstraction; in the middle layer we have the *Abstract services*, that are introduced to summarize the functionalities of sets of similar *Concrete services*; these ones are positioned in the lower layer of the ontology and are directly invocable services that implement the functionalities represented by Abstract ones; at the top layer of the ontology, we have *Subject Categories*, that organize Abstract services into standard available taxonomies to provide a topic-driven access to them. Abstract services can be organized into generalization hierarchies: we say that an Abstract service is a *specialization* of another one if it provides at least the same outputs and performs the same capabilities or provides/performs more specific outputs/capabilities.

Example 2. Figure 1 shows a portion of the ontological framework in the domain of geographic information services, where are represented the two advertised Abstract services considered in the Example 1. \square

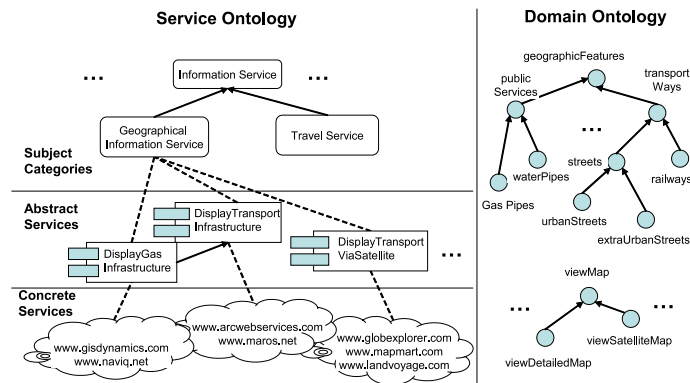


Fig. 1. A portion of ontological framework for geographic information service domain.

3 Service matchmaking

The service matchmaking framework is characterized by the following components.

Matching model - Different models have been defined: (i) a similarity-based model, where retrieval metrics are applied to measure the degree of match between services [4] (ii) a deductive model, exploiting reasoning algorithms performed on service descriptions [2] and (iii) a hybrid model, that combines the similarity-based and the deductive models [2]. In this paper we define a new deductive model for reasoning on service descriptions by difference evaluation.

Metrics - Different metrics are introduced by means of suitable coefficients apt to compute similarity and difference between request and offers.

Ranking - A ranking scheme is defined based on the established degree of match between the service request and each suitable offer.

Optimization - An optimization policy is used to reduce the number of comparisons to be performed during the matchmaking process.

In the following we briefly review main aspects of the deductive model focusing then on the difference-based model.

3.1 The deductive matching model

In the deductive matching model presented in [2] both the thesaurus \mathcal{TH} and the semantic relationships in the Domain Ontology are exploited to classify the kind of match between the request and each advertisement; following general guidelines in the current literature, we consider five kinds of match, that can be intuitively described as follows:

- *exact match*, when the request and the offer present the same functionalities (this is a strong condition);
- *plug-in match*, when the offer provides at least the required functionalities and possibly adds new ones;
- *subsume match*, when the functionalities provided by the offer are less than the required ones (it is like *plug-in match*, but with the roles of request and offer exchanged);
- *intersection match*, when the request and the offer present some common functionalities;
- *mismatch*, when no common functionalities exist between the request and the offer.

For the deductive strategy we proposed a non standard subsumption test (named $C \sqsubseteq_{\mathcal{TH}} D$) that is based on both the semantic relationships between concepts in the Domain Ontology and the terminological affinity according to the Thesaurus.

Definition 1 (Affinity-based subsumption test). *Given the Domain Ontology $Dom\mathcal{ONT}$, the thesaurus \mathcal{TH} and a pair of concepts C and D with names c_n and d_n , respectively, C is subsumed by D with respect to \mathcal{TH} , denoted by $C \sqsubseteq_{\mathcal{TH}} D$, if and only if one of the following conditions holds:*

- $C, D \in Dom\mathcal{ONT}$ and $(C \sqsubseteq D)$ is satisfied in $Dom\mathcal{ONT}$;
- only $C \in Dom\mathcal{ONT}$ and $GC_N(C) \cap D_{\mathcal{TH}} \neq \emptyset$, where $GC_N(C) = \{Name(X) \mid X \in Dom\mathcal{ONT} \text{ and } C \sqsubseteq X\}$ is the set of names of the concepts ancestors of C in $Dom\mathcal{ONT}$ and $D_{\mathcal{TH}}$ is the set of terms that have name affinity (in symbols, \sim) with d_n , that is, $D_{\mathcal{TH}} = \{y \in \mathcal{TH} \mid (d_n \sim y)\}$;
- only $D \in Dom\mathcal{ONT}$ and $SC_N(D) \cap C_{\mathcal{TH}} \neq \emptyset$, where $SC_N(D) = \{Name(X) \mid X \in Dom\mathcal{ONT} \text{ and } X \sqsubseteq D\}$ is the set of names of the concepts descendant of D in $Dom\mathcal{ONT}$ and $C_{\mathcal{TH}}$ is the set of terms that have name affinity with c_n , that is, $C_{\mathcal{TH}} = \{y \in \mathcal{TH} \mid (c_n \sim y)\}$.

Note that we pose $C \equiv_{\mathcal{TH}} D$ if and only if both $C \sqsubseteq_{\mathcal{TH}} D$ and $D \sqsubseteq_{\mathcal{TH}} C$ hold. \square

In [2] we proposed DL-based reasoning procedures to classify the five kinds of match. In the following section, we describe a new deductive procedure, that is able not only to identify the kind of match, but also to return what are the exceeding and the missing capabilities among the request and the offer. Main advantage of this matching model is that it is possible for the users to choose the most suitable offers for a given request having knowledge about single offer lacks.

3.2 The deductive difference-based matching model

In this model the matching problem is to find the information contained in \mathcal{R} and not in \mathcal{S} and viceversa, to verify if all the required capabilities are supplied by the advertisement, if only a portion of them is provided or no requirement is satisfied. To do this, we take inspiration from the *difference operator* proposed in [1] for comparing DL expressions of natural language queries and we adapt it to the problem of service matchmaking.

Definition 2 (Difference Operator). *The difference of two Description Logic expressions, that is, the information contained in the first expression and not in the second one, is defined in [9] as the syntactic minimum and expressed as:*

$$\mathcal{C} - \mathcal{D} := \min\{\mathcal{X} \mid \mathcal{X} \sqcap \mathcal{D} \equiv \mathcal{C} \sqcap \mathcal{D}\} \quad (1)$$

where *min* is defined with respect to a subdescription ordering. \square

The intuitive meaning of the difference operator is that it allows for removing from a given description \mathcal{C} all the information contained in the description \mathcal{D} .

The idea behind our approach is that, if we represent the capabilities of \mathcal{R} and \mathcal{S} by means of DL-based expressions and compare them, we can obtain three kinds of information:

- \mathcal{MD} (*Missing Description*), it is the information required, but not provided by the advertisement;
- \mathcal{ED} (*Exceeding Description*), it is the information provided, but not explicitly required;
- \mathcal{CD} (*Common Description*), it is the information required and provided, that is, common to the request \mathcal{R} and to the advertisement \mathcal{S} .

Actually, the expressions of \mathcal{MD} , \mathcal{CD} and \mathcal{ED} can be obtained from the Description Logic representation of \mathcal{R} and \mathcal{S} by means of the differences:

$$\mathcal{MD} := \mathcal{R} - \mathcal{S} \quad (2)$$

$$\mathcal{ED} := \mathcal{S} - \mathcal{R} \quad (3)$$

$$\mathcal{CD} := \mathcal{R} - \mathcal{MD} := \mathcal{S} - \mathcal{ED} \quad (4)$$

We can now precisely define the five kinds of match previously introduced. Firstly, we consider the service categories CAT_R of the request and CAT_S of the offer and we verify if $CAT_R \sqsubseteq_{\mathcal{TH}} CAT_S$. If this is not verified, a **Mismatch** is established, otherwise other matches can be progressively defined in terms of the \mathcal{MD} , \mathcal{CD} and \mathcal{ED} differences.

Match	Definition in terms of \mathcal{MD} and \mathcal{ED}
Exact	if $(\mathcal{MD} \sqsubseteq_{\mathcal{TH}} \perp)$ and $(\mathcal{ED} \sqsubseteq_{\mathcal{TH}} \perp)$
Plug-in	if not(Exact) and $(\mathcal{MD} \sqsubseteq_{\mathcal{TH}} \perp)$
Subsume	if not(Plug-in) and $(\mathcal{ED} \sqsubseteq_{\mathcal{TH}} \perp)$
Mismatch	if not(Subsume) and $(\mathcal{MD} \equiv_{\mathcal{TH}} \mathcal{R})$
Intersection	Otherwise

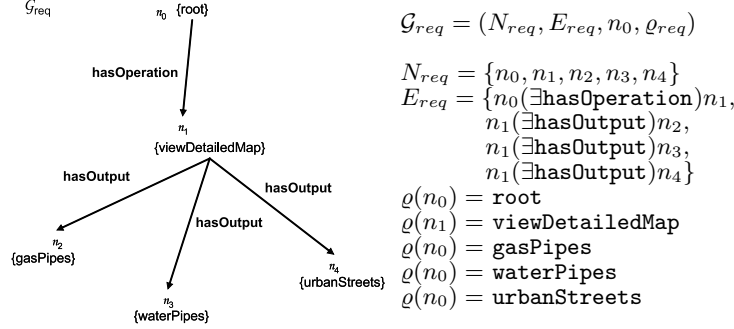


Fig. 2. An example of description tree.

4 The DIFFERENCE-MATCH algorithm

An algorithm that performs the difference between two Description Logic expressions using a tree-based characterization of subsumption has been proposed in [1]. In our case, the definition of *description tree* is introduced for the description of services given in Section 2. The algorithm to compute the difference between service descriptions is then defined exploiting the affinity-based subsumption test for reasoning on description trees.

Definition 3 (Description tree). A description tree for a Description Logic representation of a service \mathcal{S} is a tree of the form $\mathcal{G}_{\mathcal{S}} = (N_{\mathcal{S}}, E_{\mathcal{S}}, n_0, \varrho_{\mathcal{S}})$, where:

- $N_{\mathcal{S}}$ is a finite set of nodes of $\mathcal{G}_{\mathcal{S}}$;
- $E_{\mathcal{S}} \subseteq N_{\mathcal{S}} \times \{\text{hasOperation}, \text{hasOutput}\} \times N_{\mathcal{S}}$ is a finite set of edges labeled with role names $r \in \{\text{hasOperation}, \text{hasOutput}\}$ (\exists -edges); an \exists -edge from n to m labeled r is written as $n(\exists r)m$; by construction, we can have the following kinds of \exists -edges:
 - $n_0(\exists \text{hasOperation})n_i \in E_{\mathcal{S}}$, with $i = 1 \dots |OP_{\mathcal{S}}|$, where $OP_{\mathcal{S}}$ is the set of the operation names;
 - $n_i(\exists \text{hasOutput})p_{ih} \in E_{\mathcal{S}}$, with $i = 1 \dots |OP_{\mathcal{S}}|$ and $h = 1 \dots |OUT_{\mathcal{S}}^i|$, where $OUT_{\mathcal{S}}^i$ is the set of names of the output parameters associated to the operation $op_{\mathcal{S}}^i \in OP_{\mathcal{S}}$;
- $n_0 \in N_{\mathcal{S}}$ is the root of $\mathcal{G}_{\mathcal{S}}$;
- $\varrho_{\mathcal{S}} : N_{\mathcal{S}} \rightarrow [\bigcup_{i=1}^{|OP_{\mathcal{S}}|} OUT_{\mathcal{S}}^i] \cup OP_{\mathcal{S}} \cup \{\text{root}\}$ is a labeling function mapping the nodes in $N_{\mathcal{S}}$ to the concepts related to service description elements; in particular, $\varrho(n_0) = \text{root}$.

Example 3. The description tree for the request in the Example 1 is shown in Figure 2. \square

The algorithm that computes the difference between the description tree representations of two services \mathcal{R} and \mathcal{S} and that uses the $\sqsubseteq_{\mathcal{TH}}$ subsumption test to establish semantic mappings among single elements of the services is shown in Figure 3. The algorithm returns the type of match and possible differences (missing/exceeding elements). Here, the notation $\mathcal{G}.subtree(n)$ denotes the subtree with root node n in the description tree \mathcal{G} and, given $n_0(\exists \text{hasOperation})m_i \in E_{\mathcal{R}}$ and $n_0(\exists \text{hasOperation})n_j \in E_{\mathcal{S}}$, $\mathcal{G}_{\mathcal{R}}.subtree(m_i)$

```

procedure DIFFERENCE-MATCH( $\mathcal{G}_R, \mathcal{G}_S, DomONT, \mathcal{TH}$ )
(1) inputs: the description tree  $\mathcal{G}_R = (N_R, E_R, m_0, \varrho_R)$  of a request  $\mathcal{R}$ 
(2)         the description tree  $\mathcal{G}_S = (N_S, E_S, n_0, \varrho_S)$  of an offer  $\mathcal{S}$ 
(3)         a Domain Ontology  $DomONT$  and a thesaurus  $\mathcal{TH}$ 
(4) outputs:  $\mathcal{MD} = \text{diff}_{aff}(\mathcal{G}_R, \mathcal{G}_S)$ 
(5)          $\mathcal{ED} = \text{diff}_{aff}(\mathcal{G}_S, \mathcal{G}_R)$ 
(6)          $Mtype \in \{\text{'exact'}, \text{'plug-in'}, \text{'subsume'}, \text{'intersection'}, \text{'mismatch'}\}$ 

(7)  $\mathcal{MD} = \mathcal{G}_R; \mathcal{ED} = \mathcal{G}_S;$ 
(8) if not ( $CAT_R \sqsubseteq_{\mathcal{TH}} CAT_S$ )
(9)      $Mtype = \text{'mismatch'}$ ; return  $\mathcal{MD}, \mathcal{ED}, Mtype;$ 
(10) foreach ( $m_0(\exists hasOperation)m_i \in E_R$ )
(11)     if  $\exists (n_0(\exists hasOperation)n_j \in E_S)$  such that  $[\mathcal{G}_R.subtree(m_i) \sqsubseteq_{\mathcal{TH}} \mathcal{G}_S.subtree(n_j)]$ 
(12)         delete  $\mathcal{G}_R.subtree(m_i)$  from  $\mathcal{MD}$ ;
(13)     else if  $\exists (n_0(\exists hasOperation)n_j \in E_S)$  such that  $[\varrho_R(m_i) \sqsubseteq_{\mathcal{TH}} \varrho_S(n_j)]$ 
(14)         foreach ( $m_i(\exists hasOutput)p_{ih} \in E_R$ )
(15)             if  $\exists (n_j(\exists hasOutput)p_{jk} \in E_S)$  such that  $[\varrho_R(p_{ih}) \sqsubseteq_{\mathcal{TH}} \varrho_S(p_{jk})]$ 
(16)                 delete  $\mathcal{G}_R.subtree(p_{ih})$  from  $\mathcal{MD}$ ;
...
// repeat rows (10)-(16) exchanging the roles of  $\mathcal{G}_R$  and  $\mathcal{G}_S$  to find  $\mathcal{ED}$ 
...
(17) if ( $\mathcal{MD} \sqsubseteq \perp$ ) and ( $\mathcal{ED} \sqsubseteq \perp$ )
(18)      $Mtype = \text{'exact'}$ ;
(19) else if ( $\mathcal{MD} \sqsubseteq \perp$ )
(20)      $Mtype = \text{'plug-in'}$ ;
(21) else if ( $\mathcal{ED} \sqsubseteq \perp$ )
(22)      $Mtype = \text{'subsume'}$ ;
(23) else if ( $\mathcal{MD} \equiv \mathcal{G}_R$ )
(24)      $Mtype = \text{'mismatch'}$ ;
(25) else
(26)      $Mtype = \text{'intersection'}$ ;
(27) return  $\mathcal{MD}, \mathcal{ED}, Mtype;$ 

```

Fig. 3. Difference algorithm between description tree representations of services.

$\sqsubseteq_{\mathcal{TH}} \mathcal{G}_S.subtree(n_j)$ if and only if:

- (i) $\varrho_R(m_i) \sqsubseteq_{\mathcal{TH}} \varrho_S(n_j)$, and
- (ii) for each $m_i(\exists hasOutput)p_{ih} \in E_R$ there exists $n_j(\exists hasOutput)p_{jk} \in E_S$ such that $\varrho_R(p_{ih}) \sqsubseteq_{\mathcal{TH}} \varrho_S(p_{jk})$.

The generalization hierarchy of Abstract services is exploited to make more efficient the service discovery procedure, according to the following intuition: if an Abstract service \mathcal{S}_a matches with a given service request \mathcal{R} , then also Abstract services that provide at least the same capabilities of \mathcal{S}_a (that is, are specializations of \mathcal{S}_a) match with \mathcal{R} . Once the desired Abstract service is found, its corresponding Concrete services are proposed among the searching results without any further application of the discovery algorithm.

Example 4. Applying the DIFFERENCE-MATCH algorithm to the service descriptions in the Example 1, we obtain:

```

 $\mathcal{R} - \text{DisplayGasInfrastructure} := \exists hasOperation.(viewDetailedMap \sqcap \exists hasOutput.waterPipes)$ 
 $\text{DisplayGasInfrastructure} - \mathcal{R} := \perp$ 

 $\mathcal{R} - \text{DisplayTransportInfrastructure} := \exists hasOperation.(viewDetailedMap \sqcap \exists hasOutput.waterPipes)$ 
 $\text{DisplayTransportInfrastructure} - \mathcal{R} := \perp$ 

```

Then, the request \mathcal{R} presents a subsume match both with `DisplayGasInfrastructure` and `DisplayTransportInfrastructure` and the difference operator returns the same results for the two comparisons. \square

In order to measure the degree of difference between a request and available offers we define a difference coefficient.

Definition 4 (Difference coefficient). *The difference coefficient between the services \mathcal{R} and \mathcal{S} is size of their difference.*

$$diff(\mathcal{R}, \mathcal{S}) := |\mathcal{MD}(\mathcal{R}, \mathcal{S})| \quad (5)$$

where $|A|$ is the number of symbols in the expression A .

According to the difference coefficient offers can be ranked and the more an offer fulfills the request, the better is the rank. For a given threshold of desired covering, 1-1 and 1-N mappings between a request and available offers can be determined.

5 Conclusions

In this paper we have presented one of the matching models characterizing our service matchmaking framework defined to find services that best fit a given request. The different matching models aim at improving searching results and can be used in conjunction with optimization and ranking strategies. The application of the different models produces different results depending on the level of flexibility expected from the requester. Future work will investigate in more detail the design of a toolbox based on different matching models to support user in choosing the best flexible discovery strategy.

References

1. S. Benbernou, M.S. Hacid, N. Karam, and M. Schneider. Semantic Matching of Natural Language Web Queries. In *Proceedings of ICWE2004*, pages 416–429, Munich, Germany, July 2004.
2. D. Bianchini, V. De Antonellis, and M. Melchiori. Capability Matching and Similarity Reasoning in Service Discovery. In *CAiSE Int. Workshop on Enterprise Modeling and Ontologies for Interoperability, EMOI 2005*, Porto, Portugal, June 2005.
3. D. Bianchini, V. De Antonellis, and M. Melchiori. Evaluating Similarity and Difference in Service Matchmaking. In *CAiSE Int. Workshop on Enterprise Modeling and Ontologies for Interoperability, EMOI 2006*, Luxemburg, June 2006.
4. D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. Ontology-based Methodology for e-Service discovery. *Journal of Information Systems, Special Issue on Semantic Web and Web Services*, 31(4-5):361–380, June-July 2006.
5. A. Calì, D. Calvanese, S. Colucci, T. Di Noia, and F.M. Donini. A logic based approach for matching user profiles. In *Proc. of the 8th Int. Conf. on Knowledge-Based Intelligent Information & Engineering Systems (KES 2004)*, volume 3215 of *Lecture Notes in Artificial Intelligence*, pages 187–195, September 2004.
6. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1*. World Wide Web Consortium (W3C), March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-2001_0315.
7. X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In *Proc. of the 30th Int. Conference on Very Large Data Bases (VLDB2004)*, pages 372–383, Toronto, Canada, August-September 2004.
8. I. Horrocks and L. Li. A Software Framework for Matchmaking Based on Semantic Web Technology. *Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Michael Wellman and John Riedl, editors, Int. Journal of Electronic Commerce (IJEC 2004)*, August 2004.
9. R. Kusters. Non-Standard Inferences in Description Logics. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 2100, 2001.