AN AGENCY FOR SEMANTIC-BASED AUTOMATIC DISCOVERY OF WEB-SERVICES

Simona Colucci,[1] Tommaso Di Noia,[1] Eugenio Di Sciascio,[1] Francesco M. Donini,[2] Marina Mongiello,[1] Giacomo Piscitelli,[1] and Gianvito Rossi[1]

[1]*Politecnico di Bari. Via Re David, 200, 70125 Bari. Italy*[*]

{s.colucci, t.dinoia, disciascio, mongiello, gv.rossi, piscitel}@poliba.it

[2]*Universitą della Tuscia. Via S. Carlo 32, 01100 Viterbo. Italy*

donini@unitus.it

Abstract With the evolution of Web service technology, services will not only become increasingly sophisticated, but also move into the area of business-to-consumer and peer-to-peer interactions. Because of todays wide variety of services offered to perform a specific task, there is a need for mediation infrastructures able to support humans or agents in the eventual selection of appropriate services. It is a common opinion that such issues should be solved adopting semantically rich unambiguous descriptions. Hence, ontologies should be used to describe services, to ease their discovery and selection. In order to perform such a selection, a matchmaking procedure, based on semantic descriptions similarity, is needed. Technologies developed for the Semantic Web based on theoretical studies on Artificial Intelligence, particularly on Description Logics, can help in this sense. As the Semantic Web is conceived as an extension of the current one, technologies developed explicitly for they both must be used synergically in order to provide a semantic layer to approaches such as UDDI registries, using OWL formatted descriptions. In this paper we present a framework for discovery of services, stored in an UDDI registry, which exposes a description whose semantic can be modeled using OWL-DL based formalism. In order to perform this task, methodologies to compute semantic differences between two descriptions and non-standard inference services have been investigated and exploited in an implemented system.

1. Introduction

The discovery process of a Web Service can be defined as: ”The act of locating a machine-processable description of a Web service that may have been

[*]Partial funding provided by projects PON CNOSSO and MS3DI

previously unknown and that meets certain functional criteria" [Web Services Glossary, 2003]. The previous definition points out that finding a web service requires the identification through a description modeled using a machine-processable language. It is also emphasized that the only constraint needed on the description semantics is a functional one. In [OWL-S overview, 2003] automatic web service discovery is introduced as: "Automatic Web service discovery involves the automatic location of Web services that provide a particular service and that adhere to requested constraints. [...] Currently, this task must be performed by a human who might use a search engine to find a service, read the web pages, and execute the service manually, to determine if it satisfies the constraints". In the latter definition, the human interposition required to prune the searching space, is highlighted: "read the web pages". During such a pruning process, a user has to decide how good the service is with respect to what s/he is looking for. In other words, the user bears the burden to decide how the meaning (semantics) of the available services description is similar to the searched one, and in the presence of several available services sort them some way.

Automation of the web services discovery process requires as a first step providing service descriptions –what the service actually offers– that have to be well defined, machine understandable and processable. It is a common opinion that such issues should be solved adopting semantically rich clear descriptions, so ontologies should be used to describe services to ease their discovery and selection [S.A.McIlraith and Martin, 2003].

Because of todays wide variety of services offered to perform a specific task, there is a need for mediation infrastructures able to support humans or agents in the eventual selection of appropriate services. A semantic-based matching mechanism is then needed, based on the automatic analysis of descriptions similarity. This process is usually called *matchmaking*. Several recent works formalize with Description Logics (DLs) the matchmaking of descriptions (see the Realted work section for more detail). DLs, in fact, allows to model structured descriptions of supplies and demands as concepts, usually sharing a common ontology. Furthermore DLs allow for an open-world assumption. Incomplete information is admitted, and absence of information can be distinguished from negative information.

Usually, standard reasoning services of a DL system — subsumption and (un)satisfiability — are employed to determine a match. In brief, if a supply is described by a concept S and a demand by a concept D, unsatisfiability of the conjunction of S and D noted as $S \sqcap D$ identifies the incompatible proposals, satisfiability identifies potential partners — that still have to agree on underspecified constraints — and subsumption of S and D noted as $S \sqsubseteq D$ means that requirements on D are completely fulfilled by S.

As a matter of fact the flat classification into compatible and incompatible matches can be of little help in the presence of, say, some hundred compatible proposals. In previous works [Di Noia et al., 2003b; Di Noia et al., 2003c; Di Noia et al., 2003a] we introduced a logic-based formal framework and reasonable algorithms to classify and rank matches into classes, *i.e.*, *Exact match:* all requested characteristics are available in the description examined;*Potential match:* some part of the request is not specified in the description examined;*Partial match:* some part of the request is in conflict with the description examined.

The algorithms are modified versions of the structural subsumption algorithm originally proposed in [Borgida and Patel-Schneider, 1994] and compute a distance between each description w.r.t. a requested service according to the following criteria. *Potential match*: how much of the request is not specified in the description examined. *Partial match*: how much of the request is in conflict with the description examined. Notice that an *Exact match* can obviously be considered a special case of potential match where subsumption is true.

In this paper we present our semantic-based framework, based on Description Logics formalization and reasoning, and its deployment in a prototype, which overcomes simple subsumption matching of services descriptions, providing information on their similarity and allowing services classification and ranking with reference to a given service request.

The system embeds an adapted NeoClassic system, which communicates via DIG, and carries out service discovery through matchmaking of the exposed service descriptions with requested ones. With reference to the well-known triangle diagram for web-services interaction our system plays the role of *Discovery Agency*.

The remaining of the paper is organized as follows. Next Section recalls some of the relevant literature. In Section 3 we briefly revise basics of description logics and our approach to description matching using description logics. In Section 4 we present our Discovery Agency and its operating mode. Concluding remarks close the paper.

2. Related Work

The Retsina Multiagent infrastructure [Sycara et al., 2003] includes a matching agent [Sycara et al., 2002] and [Paolucci et al., 2002b] with a language, LARKS, specifically designed for agent advertisement. No ranking is presented but for what is called relaxed match, which basically reverts again to a free-text similarity measure. So a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match. Standard Information retrieval techniques have been also used in the GRAPPA matchmaking framework [Veit et al., 2001]. An extension to the approach in [Paolucci et al., 2002b] is proposed in [Li and Horrocks, 2003] where two new levels for service profiles matching are introduced. JADE agent platform for semantic web services discovery is used there on a test ontology based on DAML-S. The approach presented does not introduce a ranking method to measure proximity of service descriptions. In [Di Sciascio et al., 2001] an initial setting for logical matchmaking was presented in a person-to-person framework, based on subsumption matchmaking. In [Gonzales-Castillo et al., 2001] and [Trastour et al., 2002] an initial matchmaking framework was proposed, which operated on service descriptions in DAML+OIL and was based on the FaCT reasoner. IBM's Websphere-SilkRoad matchmaking environment is based on a matchmaking engine that describes supplies / demands as properties and rules. Matching is accomplished by simply comparing properties and verifying rules. No notion of distinction between full, partial potential and inconsistent matches are present [Hoffner et al., 2000]. A similar approach, with descriptions defined in XML and again a rule based decision system is in [Casati and Shan, 2001]. In [Ströbel and Stolze, 2002] an extension to the original Websphere matchmaker is proposed, which introduces users' specification of negotiable constraints when no total match is available. Also in [Benatallah et al., 2003] web services matchmaking was tackled. An approach was proposed, which is based on the Difference operator in DLs, followed by a set covering operation optimized using hypergraph techniques. Anyway, to the best of our knowledge there is no algorithm able to compute an exact Concept Difference in a DL endowed of the negation constructor. In [Di Noia et al., 2003c] a logic based approach to matchmaking in an e-marketplace, which allows to categorize and rank matches is presented. Its initial deployment also in a web-service-oriented system is reported in [Colucci et al., 2003b]. In [Di Noia et al., 2003a; Colucci et al., 2003a] novel DL services, namely Concept Abduction and Concept Contraction are proposed in the framework of e-marketplaces matchmaking to overcome limitations of presently available inferences.

3. Description Logics for Semantic Discovery

Description Logics (DLs) [Baader et al., 2002] are a family of logic formalisms for knowledge representation. We assume readers be familiar with them and just provide some insight into the specific constructs and system we adopt. Several DL-based reasoning systems have been implemented, such as Loom, Kris, FAcT, Racer,among others. Our system embeds a modified version of CLASSIC system [Borgida and Patel-Schneider, 1994]. Although CLASSIC DL is not very expressive w.r.t. other proposed systems, its language has been designed with the goal to be as expressive as possible while still admitting polynomial-time inferences. It manages an $\mathcal{ALN}$(**A**ttributive **L**anguage with unqualified **N**umber restrictions) DL. Constructs allowed in an

$\mathcal{ALN}$ DL are:
A *atomic concepts*.Sets of objects.
$\top$ *universal concept.* All the objects in the domain.
$\bot$ *bottom concept.* The empty set.
$\neg A$ *atomic negation.* All the objects not belonging to the set represented by A.
$C \sqcap D$ *intersection.* The objects belonging both to C and D.
$\forall R.C$ *value restriction.* All the objects participating to the R relation whose range are all the objects belonging to C.
$(\geq\ n\ R)|(\leq\ n\ R)$ *unqualified number restrictions.* Respectively the minimum and the maximum number of objects participating in the relation R.
The CLASSIC system uses a $simple - TBox$ in order to express the relations among objects in the domain. With a $simple - TBox$ in all the axioms (for both inclusion and definition) the left side is represented by a concept name. As for each atomic concept only one axiom is allowed, in the CLASSIC system, it is possible to use a normal form and each C concept has an equivalent normal form as $C_{names} \sqcap C_{\sharp} \sqcap C_{all}$, in which C_{names} is a conjunction of names, $C_{\sharp}$ of number restrictions, and C_{all} of universal role quantifications. Thanks to the normal form, the management of the hierarchical structure of an ontology, within its nested levels, is possible without traversing the semantic graph represented by the axioms.

Matchmaking is a process by which given an object (*O*) belonging to a set, a sub-set is searched, whose elements share some characteristics with *O*, *i.e.*, match *O*. Obviously, in our setting, objects to be found are web-services descriptions, based on a request submitted to a facilitator. In previous works [Di Noia et al., 2003b; Di Noia et al., 2003c] we introduced a logic-based formal framework to classify and rank matches into classes, *i.e.*, *Exact match:* all characteristics in the requested object are available in the offered one; *Potential match:* some characteristics in the requested object are not specified in the offered one; *Partial match:* some characteristics in the requested object are in conflict with the offered ones. In order to perform the matchmaking process, we identified some properties a facilitator should have.
[Open-world descriptions] The absence of a characteristic in the description of an offered or requested object should not be interpreted as a constraint of absence, instead it should be considered as a ”don't care for the moment”.
[Non-symmetric evaluation] A matchmaking system may give different evaluations depending on whether it is trying to match O with R, or R with O — *i.e.*, depending on who is going to use this evaluation.
If requested and offered objects are modeled using a logic, then objects with the same meaning should have the same ranking, independently of their syntactic descriptions.
[Syntax independence in ranking matches] A ranking of matches is *syntax independent* if for every pair of offered objects O_1 and O_2, requested object D,

and ontology $\mathcal{T}$, when O_1 is logically equivalent to O_2 then O_1 and O_2 have the same ranking score for R, and the same holds also for every pair of logically equivalent requested object R_1, R_2 with respect to every available one O. In order to show the relation between ranking and implications, let us consider a description with sets of words. Let R be a requested object and O_1, O_2 be two offered objects defined as follows:

$$\begin{aligned} R &= \{computer, intel, hard\_disk, LCDmonitor\} \\ O_1 &= \{computer, intel, browser, linux\} \\ O_2 &= \{computer, intel, browser, linux, word\_processor\} \end{aligned}$$

In this case, the characteristics that O_2 adds to O_1 are irrelevant for R. Hence, whatever the rank for O_1, the one for O_2 should be the same. If instead we let

$$O_3 = \{computer, intel, browser, linux, LCDmonitor\}$$

then O_3 should be ranked better than O_1 since it adds a characteristic required by R.

[Monotonicity of ranking potential matches over subsumption] A ranking of potential matches is *monotonic over subsumption* whenever for every requested object R, for every pair of offered objects O_1 and O_2, and ontology $\mathcal{T}$, if O_1 and O_2 are both potential matches for R, and $\mathcal{T} \models (O_2 \sqsubseteq O_1)$, then O_2 should be ranked either the same, or better than O_1, and the same should hold for every pair of requested objects R_1, R_2 with respect to an offered one O. The point of view is flipped over when turning to partial matches. In such matches we already have some characteristics in conflict between R and O; adding new characteristic to R may only make the match worse than before (introducing new conflicting characteristic) or keep it the same (if new characteristic are not in conflict).
[Antimonotonicity of ranking partial matches over implication] A ranking of partial matches is *antimonotonic over implication* whenever for every requested object R, for every pair of offered objects O_1 and O_2, and ontology $\mathcal{T}$, if O_1 and O_2 are both partial matches for R, and $\mathcal{T} \models (O_2 \sqsubseteq O_1)$, then O_2 should be ranked either the same, or worse than O_1, and the same should hold for every pair of requested objects R_1, R_2 with respect to an offered one O.
In the following we briefly illustrate the behavior of the algorithms we adopt in our system (for a complete description see [Di Noia et al., 2003c]). Given a requested object R and an offered one O we have a *potential* match if $R \sqcap O$ is satisfiable in $\mathcal{T}$, *i.e.*, the properties of neither concepts exclude the other. The *rankPotential* algorithm takes as inputs two concepts to be matched *i.e.*, R vs. O, in normal form, such that $R \sqcap O$ is satisfiable and returns a score n. The algorithm adds to n the number of concept names in R that are not among the

concept names of O and number restrictions of R that are not implied by those of O and for each universal role quantification in R adds to n the result of a recursive call. A total match, *i.e.*, concept implication, yields a $n = 0$ score, while $n \geq 0$ of O w.r.t. R, *i.e.*, the score increases (worsens) as the two descriptions are, though still compatible, more different. Notice the rationale of the approach, which penalizes generic descriptions, which in simple subsumption matching would be unfairly advantaged, as the algorithm ranks better more specific descriptions of O matching R. It can be proved the algorithm is syntax independent, and monotonic over subsumption. Also notice that *rankPotential* can be used to compute a metric for the length of a concept. In fact the length of a concept R can be weight up as the score returned by *rankPotential* algorithm when $O = \top$. Given a requested object R and an offered one O we have a *partial* match if $R \sqcap O$ is unsatisfiable in $\mathcal{T}$, *i.e.*, some of the properties of concepts are in conflict. In the *rankPartial* algorithm we are hence looking for inconsistencies. Also this time the algorithm takes as input two concepts in normal form, but this time the score accounts for inconsistencies.
The basic idea of the Semantic Web initiative is to structure information with the aid of markup languages, based on the XML language, such as RDF and RDFS (http://www.w3.org/RDF/), DAML+OIL (www.daml.org/2001/03/)and more recently OWL [McGuinness et al., 2002]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of domain ontologies, and aim at increasing openness and interoperability in the web environment. A subset of the OWL-DL language is the obvious candidate for a framework based on DLs, as we propose in the following section.

4. Discovery Agency

The theoretical framework summarized in the previous section has been deployed in a complete Discovery Agency. The agency, which is a web service itself, exposes a SOAP over HTTP interface.

The system carries out two main activities: publication of web-services and semantic search on published web-services. To this aim it exploits available standards such as UDDI (Universal Description, Discovery and Integration) (www.uddi.org), which defines a facilities set supporting description and discovery of Web Services providers, Web Services, and technical access interfaces to Services trough a structured registry and UNSPSC (United Nations Standard Products and Services Code) (www.unspsc.org), a wide open-standard developed and maintained by *Dun & Bradstreet* to categorize families of services (and consequently, in our approach, domain ontologies). UDDI aims to define a facilities set supporting description and discovery of Web Services providers, Web Services itself, and technical access interfaces to Services. UDDI uses widespread standards such as HTTP, XML, XML Schema and SOAP, providing a set of functions allowing one to register companies, services and their access information, to modify or cancel registrations and to search in the registrations database. The deployment of a Web Service is made up of three phases: *registration of the provider as company*; *deployment of a description of provided service*; *deployment of service invocation information*. These information are usually grouped in categories: *white pages*(Businesses), *yellow pages*(Services), *green pages*(technical information). User may search a Service, then, by company or by capability. UDDI Registries are available on the Web and are themselves Web Services: the user have to know access information to a given UDDI to use it. Business UDDI Registries store in their records four types of knowledge: *businessEntity*: non technical information about providers (white pages); *businessService*: non technical information about provided capabilities (yellow pages); *bindingTemplate*: technical information about services (green pages); *tModel*: Service access details (depending on *bindingTemplate*). UDDI provides also API(Application Programming Interfaces) for information and Services search(Inquiry API) and deployment(Publishing API). UNSPSC provides a product classification, for E-commerce purpose, useful to identify the category of a selling item (and then of the selling item web service). It helps us to search and localize Services identifying suppliers of given product or service. Searching by code avoids the shortcomings of textual retrieval: results of searching process are Services providing capabilities classified under the given code and not irrelevant Services whose name contains the searched product. UNSPSC is a hierarchical classification made up of four levels. Each level include a two digits numeric code and a textual description, as follows: *Segment: identifies the market segment of a product*; *Family: identifies a universally recognized product category*; *Class: identifies a group of products with the same functionality or usage; Commodity: identifies a group of equivalent products.*

Obviously the added value we pursue is that, having obtained a subset of the search space using UNSPSC categorization, a semantic match of users request with available web-services description is possible.

MAMAS

The core module of the discovery agency is the MAtchMAker Service: MAMAS. It embeds a version of the original NEOCLASSIC the C++ version of CLASSIC modified in order to perform the matchmaking functionalities. The interface exposed by MAMAS is based on DIG 1.0 specifications [Bechhofer et al., 2003] for an $\mathcal{ALN}$ Description Logic. Hence, through a *tell* and *ask* mechanism it is able to dynamically load an ontology, request web service description and offered web service description stored in the UDDI registry, as DL individuals, and return the matchmaking results.

MAMAS behavior can be simply explained with the aid of an example. Suppose to have two published web services offering rental service:
$ws1 = rent \sqcap \forall related\_to.(bedroom \sqcap \forall contains\_bed.wedding\_bed$
$\sqcap \forall is\_contained.sea\_house) \sqcap \forall available.summer \sqcap$
$\forall has\_price.((\geq\ 200\ euro) \sqcap (\leq\ 350\ euro))$
$ws2 = rent \sqcap \forall related\_to.(bedroom \sqcap \forall is\_contained.mountain\_house \sqcap$
$\forall has\_appliances.TV \sqcap VCR \sqcap Internet)$
$ws1$ could be a good choice for room finding if you want to spend your summer holiday in a site near the sea; on the other hand $ws2$ offers rooms with accessories in houses sited on the mountains. Now we present two possible users of rental services: a human being looking for a double room in a site near the sea
$hb = rent \sqcap \forall related\_to.(double\_room \sqcap \forall is\_contained.$
$(house \sqcap \forall is\_located.sea)) \sqcap \forall available.august$
$\sqcap \forall has\_price.(\leq\ 300\ euro)$
and the personal software agent of a touristic agency looking for particular rooms available in June in order to compose ”all inclusive” supplies:
$sa = rent \sqcap \forall related\_to.(bedroom \sqcap \forall is\_contained.apartment$
$\sqcap \forall has\_appliance.fully\_furnished) \sqcap \forall available.june$
$\sqcap \forall has\_price.(\leq\ 300\ euro)$
It is easy to observe that in both cases, $ws1$ and $ws2$ do not completely satisfy hb and sa. Nevertheless, $ws1$ is a good solution for hb, and both $ws1$ and $ws2$ are good solutions for sa, while $ws2$ offers a service which is in conflict with the one serched by hb. MAMAS catches this behavior and, computing both hb and sa lengths, provides the following results:

- for hb:
  $ws1$ is a **potential** match with a *8%* of mismatch degree
  $ws2$ is a **partial** match with a *2%* of incompatibility degree
- for sa:
  $ws1$ is a **potential** match with a *21%* of mismatch degree
  $ws2$ is a **potential** match with a *15%* of mismatch degree

The DIG interface for NeoClassic is provided by the NEODIG Java servlet. The DIG standard is basically an XML Schema describing the language used to interact with a DL reasoner, in order both to introduce new knowledge and to query a DL knowledge base, using HTTP POST requests. The DIG standard allows only qualified number restriction ($\mathcal{Q}$) and not unqualified number restriction ($\mathcal{N}$), such as the one supported by CLASSIC, in the <language> TAG there are generic <atMost/ > and <atLeast/ >. It is well known that an

unqualified number restriction is reducible to a qualified number restriction. In fact $(\geq\ 2\ R) = (\geq 2R).\top$ and $(\leq\ 2\ R) = (\leq 2R).\top$.
As DIG was thought for standard DL inference we added auxiliary TAG to perform the **matchMake** ask. A matchmaking request, identified by an id, of a description *I1* versus another description *I2* is:

```
<matchMake  id="q">
          <individual name="I1"/>
          <individual name="I2"/>
</matchMake>
```

where *I1* is the description of an offered web service, stored in the registry, and *I2* is the description of the searched one. The reply includes the category type of match obtained, *i.e.*, potential –no conflict– or partial –elements in conflict– and the score determined for the match. For potential match:

```
<matchMake  id="q">
          <type name="rankPotential"/>
          <result num="potentialScore"/>
</matchMake>
```

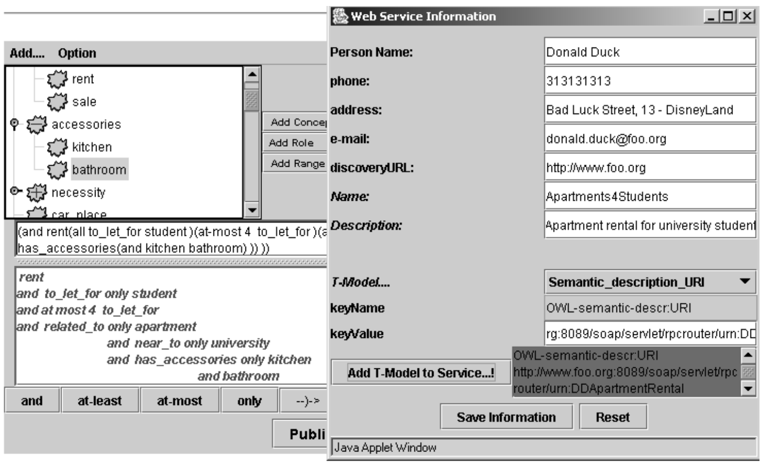
for a partial match:

```
<matchMake  id="q">
          <type name="rankPartial"/>
          <result num="partialScore"/>
</matchMake>
```

The Knowledge Bases (Kbs) Repository contains ontologies, written using a subset of OWL-DL, representing knowledge domains corresponding to UNSPSC items. A classification of services within a taxonomy is useful to limit the searching space of the requested service, *i.e.*, the domain of the service. An issue that arose was whether to build an ontology for each item in the taxonomy or to build a single ontology embedding the whole taxonomy. In the former case there would be about 10’000 ontologies related to each other, in the latter one there would be one ontology with a huge number of concepts and roles. Both cases are not easily manageable. Trading-off pros and cons in our approach, we assume the development of an ontology for each *Family* level in UNSPSC.

Web Service Publication

A typical problem in knowledge management systems is how to expose the knowledge to a human user, in a way as friendly as possible in order to keep a high degree of flexibility. To face this issue, once the domain has been identified via the UNSPSC code, if the user interacts using a browser, the system sends an applet to guide the composition of the description both to publish or



(and rent(all to_let_for student)(at-most 4 to_let_for)(a
has_accessories(and kitchen bathroom)))))

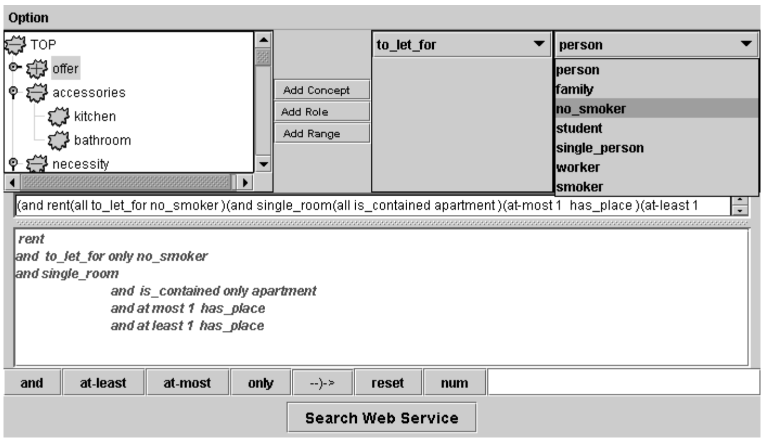
rent
and to_let_for only student
and at most 4 to_let_for
and related_to only apartment
and near_to only university
and has_accessories only kitchen
and bathroom

Person Name:	Donald Duck
phone:	313131313
address:	Bad Luck Street, 13 - DisneyLand
e-mail:	donald.duck@foo.org
discoveryURL:	http://www.foo.org
Name:	Apartments4Students
Description:	Apartment rental for university student
T-Model....	Semantic_description_URI
keyName	OWL-semantic-descr:URI
keyValue	rg:8089/soap/servlet/rpcrouter/urn:DD

OWL-semantic-descr:URI
http://www.foo.org:8089/soap/servlet/rpc
router/urn:DDApartmentRental

Figure 1. Graphical User Interface during web service publication.

to search a service description. The applet loads the corresponding domain ontology and the user is guided in the description of the service. The composition area of the GUI is divided in: 1. an area diplaying the whole concepts taxonomy; 2. a list which is dynamically populated as soon as a concept is selected (The list items are the role whose domain is the selected concept); 3. a list populated by the concept which are the range of the selected role. The description composition is supported by a translation of the DL expression into a format closer to the human language, as shown in Figure 1. Each Web Service published in the UDDI registry, must expose both *setDescription()* and *getDescription()* methods. These methods are invoked via a SOAP RPC. With the former method, the OWL-DL formatted version of the description is binded to the Web Service, with the latter one such description is returned. The publication of a web-service in our system goes through various steps. The publisher provides first the UNSPSC code of the service application domain; then with the aid of the above described applet s/he is guided in the description of the service and in the definition of service location and interface parameters. The activities that can be requested to the system in the publication stage are hereafter summarized. *Publication of an OWL-DL ontology to be classified within UNSPSC taxonomy*. As the CLASSIC reasoner used in the agency manages an $\mathcal{ALN}$ DL, the ontology has to be modeled using only the constructors allowed by such a logic. The file name representing the ontology must be in the form XX.XX.owl, where the two groups of digits represent the UNSPSC *family* level.
Search of an item in the UNSPSC taxonomy and of its corresponding level within the taxonomy. It is possible to surf the UNSPSC taxonomy in order to



```
(and rent(all to_let_for no_smoker )(and single_room(all is_contained apartment )(at-most 1  has_place )(at-least 1
```

```
rent
and  to_let_for only no_smoker
and single_room
                and  is_contained only apartment
                and at most 1  has_place
                and at least 1  has_place
```

Figure 2. Graphical User Interface during a service request composition.

find the exact code representing the service to be published.
Publication of a web service, in the UDDI registry, whose semantic description is built based on a published ontology. The publication consists of both an instantiation of two tModels in the registry and the RPC of the *setDescription()* method for the Web Service being published. The first tModel represents UNSPSC information, the second one is related to the URI of the OWL-DL formatted semantic description.
We would like to point out that, although our framework is generic, a simple mapping between UDDI and OWL-S is possible and a semantic description can be well embedded into an OWL-S (formerly DAML-S) [OWL-S overview, 2003] based description of the publishing service. The semantic description can be interpreted as the **effect** part of the IOPE (Input, Output, Precondition, Effect) model proposed in OWL-S for the *Service Profile* class. Using the mapping proposed in [Paolucci et al., 2002a] the UDDI tModel representing the semantic description field of the service is formatted as a part of OWL-S description of the service.

Web Service Discovery

The search process also goes through various stages and it is initiated again providing the UNSPSC code of the searched service, which can be evoked also using a keyword based approach, and defined up to the commodity level navigating through the hierarchical structure of the classification levels. Having selected a code the applet again guides the user in the description of the requested service, in accordance with the domain ontology. Once the user has found the UNSPSC code corresponding to the search domain, the ontology

corresponding to the family level of the code is selected and its class taxonomy and roles constraints are sent back to the user. Again the user is helped by an applet that guides her/him in the request description composition, see figure 2. Both the description and the reference domain ontology ID are sent to the Communication Service that provides interaction with MAMAS and finally returns a ranked list of services matching the request, and corresponding service invocation information. The discovery process of a web service can be hence summarized as:
Search a UNSPSC code (ID) using a keyword-based approach;
Build a semantic description of the web service to be searched, using the ontology identified by ID;
Send the description to the Discovery Agency;
Find in the UDDI registry all the services with a semantic description identified by ID;
Call the getDescription() for all the Web Services identified in the previous step;
Match services OWL-DL descriptions with the requested one;
Send back to the user the ranked list of services information.
Computing the length of the requested description, the ranked list is expressed in terms of percentage degree of mismatch (for *rankPotential* results) or of incompatibility (for *rankPartial* results).

5. Conclusion

We have investigated web-services advertisement and discovery in a framework based on DLs formalization and reasoning, which overcomes simple subsumption matching, providing information on service similarity and allowing match ranking and classification. Based on the theoretical work we have implemented a fully functional agency for semantic-based web-service discovery, which exploits state of art technologies and protocols.

References

Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., and Patel-Schneider, P., editors (2002). *The Description Logic Handbook*. Cambridge University Press.

Bechhofer, S., Mller, R., and Crowther, P. (2003). The DIG Description Logic Interface. In *Proc. DL'03*, volume 81 of *CEUR Workshop Proceedings*.

Benatallah, B., Hacid, M.-S., Rey, C., and Toumani, F. (2003). Request Rewriting-Based Web Service Discovery. In *Proc. ISWC*, volume 2870 of *LNCS*, pages 242–257.

Borgida, A. and Patel-Schneider, P. F. (1994). A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *J. of Artificial Intelligence Research*, 1:277–308.

Casati, F. and Shan, M. C. (2001). Dynamic and Adaptive Composition of E-Services. *Information Systems*, 26:143–163.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003a). Concept Abduction and Contraction in Description Logics. In *Proc. DL'03*, volume 81 of *CEUR Workshop Proceedings*.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003b). Logic Based Approach to web services discovery and matchmaking. In *Proc. E-Services Workshop at ICEC'03*.

Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003a). Abductive matchmaking using description logics. pages 337–342, Acapulco. MK.

Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003b). Semantic matchmaking in a P-2-P electronic marketplace. In *Proc. Symposium on Applied Computing (SAC '03)*, pages 582–586. ACM.

Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003c). A system for principled Matchmaking in an electronic marketplace. In *Proc. WWW '03*, pages 321–330, Budapest. ACM Press.

Di Sciascio, E., Donini, F., Mongiello, M., and Piscitelli, G. (2001). A Knowledge-Based System for Person-to-Person E-Commerce. In *Proc. KI-2001 Workshop ADL-2001*, volume 44 of *CEUR Workshop Proceedings*.

Gonzales-Castillo, J., Trastour, D., and Bartolini, C. (2001). Description Logics for Matchmaking of Services. In *Proc. KI-2001 Workshop ADL-2001*, volume 44. CEUR Workshop Proceedings.

Hoffner, Y., Facciorusso, C., Field, S., and Schade, A. (2000). Distribution Issues in the Design and Implementation of a Virtual Market Place. *Computer Networks*, 32:717–730.

Li, L. and Horrocks, I. (2003). A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. WWW '03*, pages 331–339, Budapest. ACM Press.

McGuinness, D., Fikes, R., Hendler, J., and Stein, L. (2002). DAML+OIL: An Ontology Language for the Semantic Web . *IEEE Intelligent Systems*, 17(5):72–80.

OWL-S overview (2003). www.daml.org/services/owl-s/1.0/.

Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. (2002a). Importing the semantic web in uddi. In *Proc. of Web Services, E-business and Semantic Web Workshop*.

Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. (2002b). Semantic Matching of Web Services Capabilities. In *Proc. ISWC*, number 2342 in LNCS, pages 333–347.

S.A.McIlraith and Martin, D. (2003). Bringing Semantics to Web Services. *IEEE Intelligent Systems*, pages 90–93.

Ströbel, M. and Stolze, M. (2002). A Matchmaking Component for the Discovery of Agreement and Negotiation Spaces in Electronic Markets. *Group Decision and Negotiation*, 11:165–181.

Sycara, K., Paolucci, M., Van Velsen, M., and Giampapa, J. (2003). The RETSINA MAS infrastructure. *Autonomous agents and multi-agent systems*, 7:29–48.

Sycara, K., Widoff, S., Klusch, M., and Lu, J. (2002). LARKS: Dynamic Matchmaking Among Heterogeneus Software Agents in Cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203.

Trastour, D., Bartolini, C., and Priest, C. (2002). Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *Proc. WWW '02*, pages 89–98. ACM.

Veit, D., Muller, J., Schneider, M., and Fiehn, B. (2001). Matchmaking for Autonomous Agents in Electronic Marketplaces. In *Proc. Agents '01*, pages 65–66. ACM.

Web Services Glossary (2003). http://www.w3.org/TR/2003/WD-ws-gloss-20030808/.