Logic Based Approach to Web Services Discovery and Matchmaking

Simona Colucci, Tommaso Di Noia,
Eugenio Di Sciascio, Marina Mongiello
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Via Re David, 200
70125 BARI, Italy
s.colucci,t.dinoia,disciascio,
mongiello@poliba.it

Francesco M. Donini
Università della Tuscia
Via San Carlo 32
01100 Viterbo, Italy
donini@unitus.it

ABSTRACT

We present a logic based approach to web services discovery and matchmaking in an e-commerce scenario. In particular, we describe our framework, based on Description Logics formalization and reasoning, and its deployment in a prototype, which overcomes simple subsumption matching of services descriptions and allows match ranking and services classification. We also present results of first experiments testing the performance of our prototype in a realistic e-commerce scenario.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Decision Support; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages

General Terms

Algorithms, Languages

Keywords

E-commerce, Web-Services, Matchmaking, Knowledge Representation, Description Logics.

1. INTRODUCTION

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Web services have been gaining an increasing popularity, however, as they become more widespread new issues arise, such as automatic discovery and interoperability. Initially Web services were mainly intended to engage in dynamic business-to-business interactions with services deployed on behalf of other enterprises or business entities. However, with the evolution of Web service technology services will not only become increasingly sophisticated, but also move into the area of business-to-consumer or even 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 clear descriptions, so ontologies should be used to describe services to ease their discovery and selection. In this paper we investigate how Semantic and Web Services technologies can be used to support service advertisement and discovery in an e-commerce scenario. In particular, we describe our framework, based on Description Logics formalization and reasoning, and its implementation in a web services matchmaking prototype for an e-commerce scenario, which overcomes simple subsumption matching of services descriptions and allows match ranking and services classification.

Recently there has been a growing interest towards matchmaking engines and techniques, with emphasis placed either on e-marketplaces or generic Web services, in view of the promised transformation of the Web from human understandable to the Semantic, machine understandable, Web. Significant examples include [20] and [18] where a language, LARKS, is proposed specifically designed for agent advertisement. The matching process is carried out through five progressive stages, going from classical IR analysis of text to semantic match via Θ-subsumption. The notion, inspired by Software Engineering, of *plug-in* match is introduced to overcome in some way the limitations of a matching approach based on exact match. No ranking is presented but for what is called relaxed match, which basically reverts again to a IR free-text similarity measure. So a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match. In [21] and [13] a matchmaking framework is proposed, which operates on service descriptions in DAML+OIL and is based on the FaCT reasoner. Unfortunately as the authors admit, though very expressive, FaCT lacks of concrete datatypes, which are obviously extremely useful for, *e.g.*, e-commerce applications, and their prototype is incomplete. In [15] a framework for matchmaking based on DAML-S and describing an implementation for service discovery using Racer is

presented, which anyway reverts to basic subsumption based matchmaking. In [10] a knowledge-based system for person-to-person e-commerce is proposed. In [9] a logic based approach to matchmaking in an e-marketplace, which allows to categorize and rank matches is presented. Semantic service discovery via matchmaking in the Bluetooth framework is investigated in [19].

In the remaining of the paper we start with Description Logics basics in order to make the paper self-contained. In section 3 we outline our framework for matching of services and algorithm for matchmaking, which overcomes subsumption matching and allows logical ranking between descriptions. Then we describe technologies we adopt in our system. We present our prototype and describe its operation mode in section 5. Initial experiments are presented in the last section, outlining the system behavior.

2. DESCRIPTION LOGICS BASICS

Description Logics are a family of logic formalisms for Knowledge Representation [2, 12]. Basic syntax elements are *concept* names, *e.g.*, `book`, `person`, `product`, `apartment`, *role* names, like `author`, `supplier`, `hasRooms` and *individuals*, such as `NewYorkCity`, `BackYardGarden`, `TVset#123`. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts, as the role `author` that links books to persons (their writers). Individuals are used for special named elements belonging to concepts.

More formally, a semantic *interpretation* is a pair $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, which consists of the *domain* Δ and the *interpretation function* $\cdot^{\mathcal{I}}$, which maps every concept to a subset of Δ, every role to a subset of $\Delta \times \Delta$, and every individual to an element of Δ. We assume that different individuals are mapped to different elements of Δ, *i.e.*, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for individuals $a \neq b$. This restriction is usually called *Unique Name Assumption* (UNA).
Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DL include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations.
Roles can be combined with concepts using *existential role quantification*, *e.g.*, `book` $\sqcap \exists$`author.italian` which describes the set of books whose authors include an Italian, and *universal role quantification*, *e.g.*, `product` $\sqcap \forall$`supplier.japanese`, which describes products sold only by Japanese suppliers. Other constructs may involve counting, as number restrictions: `apartment` $\sqcap (\leq$ 1 `hasRooms`) expresses apartments with just one room, and `book` $\sqcap (\geq$ 3 `author`) describes books written by at least three people. Many other constructs can be defined, increasing the expressive power of the DL, up to n-ary relations [5].
Expressions are given a semantics by defining the interpretation function over each construct. For example, concept conjunction is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and also the other boolean connectives $\sqcup$ and $\neg$, when present, are given the usual set-theoretic interpretation of $\cup$ and complement. The interpretation of constructs involving quantification on roles needs to make domain elements explicit: for example, $(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta \mid \forall d_2 \in \Delta : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$

Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. For example, we could impose that books can be divided into paperbacks and hardcover using the two inclusions `book` $\sqsubseteq$ `paperbacks` $\sqcup$ `hardcover` and `paperbacks` $\sqsubseteq \neg$`hardcover`. Or, that books have only one title as `book` $\sqsubseteq (\leq$ 1 `title`). Definitions are useful to give a meaningful name to particular combinations, as in `doubleRoom` $\equiv$ `room` $\sqcap (=$ 2 `hasPlaces`). Historically, sets of such inclusions are called TBox (Terminological Box). In simple DLs, only a concept name can appear on the left-hand side of an inclusion.
The semantics of inclusions and definitions is based on set containment: an interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $C = D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *model* of a TBox T is an interpretation satisfying all inclusions and definitions of T.
In every DL-based system, at least two basic reasoning services are provided:

1. *Concept Satisfiability*: given a TBox T and a concept C, does there exist at least one model of T assigning a non-empty extension to C?
2. *Subsumption*: given a TBox T and two concepts C and D, is C more general than D in any model of T?

Our engine for service descriptions matching is based on an adapted version of the CLASSIC system [3, 4], which was developed at AT&T Bell Laboratories, where it was applied in several projects for configuration [22] and program repositories [6]. Its language has been designed with the goal to be as expressive as possible while still admitting polynomial-time inferences. So it provides intersection of concepts but no union, universal but not existential quantification over roles, and number restrictions over roles but no intersection of roles, since each of these combinations is known to make reasoning NP-hard [11]. Classic obviously provides subsumption and satisfiability reasoning services. Being a complete knowledge representation system (and not just a reasoner), Classic provides also data types as numbers and strings, and other services which are useful in a deployed prototype. Every Classic concept has an equivalent normal form, hence using such a normal form ensures syntax independence. In our current setting we use just a subset of CLASSIC constructors, *i.e.*, those needed in a $\mathcal{ALN}$ logic.

3. MATCHMAKING

The key question that has to be answered is how far is a request from the provided service? And which are the requirements that would eventually fulfill it? We consider descriptions of service advertisements and requests as logical descriptions, and move on to Description Logics (DLs). Hence, from now on we suppose that advertisements and requests are expressed in a DL $\mathcal{L}$, equipped with a model-theoretic semantics. We suppose also that a common ontology for service description is established, as a TBox $\mathcal{T}$ in $\mathcal{L}$. Now a match between a service description S and a request R could be evaluated according to $\mathcal{T}$. This framework ensures the first property that we would like to hold for matchmaking, namely, an open-world assumption. The

absence of a characteristic in the description of a service advertisement or a request should not be interpreted as a constraint of absence. Instead, it should be considered as a characteristic that could be either refined later, or left open if it is irrelevant for a user Obviously, also the algorithm employed for matchmaking should take this issue into account. Secondly, a matchmaking system may give different evaluations depending on whether it is trying to match a service description S with a request R, or R with S — *i.e.*, depending on *who* is going to use this evaluation. This requirement is already evident when characteristics are modeled as sets of words. in that case, underconstrained requirements of S from the point of view of R are expressed by $R - S$ (set difference) while underconstrained requirements of R from S's viewpoint are expressed as $S - R$. Let $T \models \ldots$ denote logical implication (truth in all models of T), and let $\sqsubseteq$ (subsumption) denote also implication between constraints of S and R. There are three relations between concepts expressing service descriptions and requests, that we consider meaningful in matchmaking: implication, consistency, and inconsistency.

In the first case, $\mathcal{T} \models (R \sqsubseteq S)$, *i.e.*, constraints imposed by R imply those of S, and vice versa if $\mathcal{T} \models (S \sqsubseteq R)$. This relation extends the previous set-based inclusion to general concepts. If both $\mathcal{T} \models (R \sqsubseteq S)$ and $\mathcal{T} \models (S \sqsubseteq R)$, then R and S should be considered equivalent in $\mathcal{T}$. This relation extends exact matching, making syntax differences irrelevant. In case of consistency, $R \sqcap S$ is satisfiable in $\mathcal{T}$. Then, there is a *potential* match, in the sense that the constraints of neither proposal exclude the other. This relation has been highlighted also by other researchers [21, 15]. However, they lack of a *ranking* between different potential matches. In the third case, $R \sqcap S$ is unsatisfiable in $\mathcal{T}$. Although also in this case a matching could be defined [8], we do not treat this case here for lack of space. Hence, from now on we concentrate on potential match only. We now highlight some principles that — we believe — every ranking function should have in logical matchmaking. First of all, if a logic is used to give some meaning to descriptions of services and requests, then proposals with the same meaning should have the same ranking, independently of their syntactic descriptions. We now state some properties that — we believe — every ranking function should have in logical matchmaking. First of all, if a logic is used to give some meaning to descriptions of services and requests, then proposals with the same meaning should have the same ranking, independently of their syntactic descriptions. Hence, a ranking for semantic matchmaking should be *syntax independent*. That is, for every pair of services S_1 and S_2, requests R, and ontology T, when S_1 is logically equivalent to S_2 then S_1 and S_2 should have the same ranking for R — and the same should hold also for every pair of logically equivalent requests R_1, R_2 with respect to every service S. Secondly, a ranking for semantic matchmaking should be *monotonic over subsumption*. That is, for every request R, for every pair of services S_1 and S_2, and ontology T, if S_1 and S_2 are both potential matches for R, and $T \models (S_2 \sqsubseteq S_1)$, then S_2 should be ranked either the same, or better than S_1. The same should hold also for every pair of requests R_1, R_2 with respect to a service S. Intuitively, this property could be read of as "A ranking of potential matches is monotonic over subsumption if the more specific, the better." When turning to partial matches, adding another characteristic to an unsatisfactory proposal may either worsen its ranking (when another characteristic is violated) or keep it the same (when the new characteristic is not in contrast). Note that this ranking should be kept different from the ranking for potential matches. Obviously, properties pointed out here are independent of the particular DL employed, or even the particular *logic* chosen. For instance, the same properties could be stated if propositional logic was used or also logics, such as DAML, for which reasoning systems are not yet fully available. Clearly, when the logic admits a normal form of expressions — as CNF or DNF for propositional logic, or the normal form of concepts for the DL of CLASSIC [3] — using such a normal form ensures by itself syntax independence. We remark that the properties and definitions we stated in this section are independent of the particular DL employed, or even the particular *logic* chosen. In this respect, we believe that the definition of our framework keeps its significance also if one chooses more expressive DLs such as $\mathcal{SHOQ}(\mathcal{D})$ [14] or DAML [18].

Based on our matching framework we developed an algorithm [9], which has been obtained adapting the original CLASSIC structural [4] subsumption algorithm, to classify and rank matches.

The algorithm can be proved to respect outlined properties. Complexity is obviously of paramount importance, for an algorithm to be of practical use. It is well known [16] that the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up. Nevertheless the expansion is exponential in the depth of the hierarchy of the TBox $\mathcal{T}$; it can be proved that if the depth of $\mathcal{T}$ is $O(\log |\mathcal{T}|)$, then the expansion is polynomial, and so also the algorithm [7]. This result further validates the use of CLASSIC , which although having a reduced expressiveness w.r.t. new reasoners *e.g.*, FAcT and Racer, is expressive enough for our purposes while maintaining in practical cases still a polynomial inference. On the contrary, time may become prohibitive in approaches such as the one devised in [15], where Racer is adopted, also for much simpler subsumption-based matching without ranking.

4. SERVICE DISCOVERY ELEMENTS

Services search needs at least tree phases to yield consistent and effective results:

Categorization: classification of the Service in a possibly standard category.

Description: representation of the Service in a semantic conveying language.

Deployment: storage of Service descriptions in a Marketplace.

Our prototype implements these phases using and at times extending three developing technologies: UNSPSC(United Nations Standard Products and Services Code) attends to Categorization, DAML-S(DARPA Mark-up Language for Services) attends to Description, UDDI(Universal Description Discovery and Integration) attends to Deployment.

The three technologies are detailed in the following.

4.1 UNSPSC: A Standard Product Classification

UNSPSC is a wide open-standard developed and maintained by *Dun & Bradstreet*, leader for development of information and finance standards.

UNSPSC provides a product classification, for E-commercepurposes, useful to identify the category of a selling item (and then of the selling item web service). It helps in the search and localization of Services identifying suppliers of a given product or service. Search 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 includes a two digits numeric code and a textual description, as shown in the following:
XX Segment identify the market segment of a product
XX Family identify a universally recognized product category
XX Class identify a group of products with the same functionality or usage.
XX Commodity identify a group of equivalent products
An example of UNSPSC classification is shown in Figure 1.

```
Hierarchy       Category Number and Name
Segment        44  Office Equipment, Accessories
                   and Supplies
                   10 Office Machines and their
                       supplies and accessories
                   11 Office and desk accessories
Family             12 Office supplies
                       15 Mailing supplies
                       16 Office supplies
                       17 Writing instruments
                       18 Correction media
Class                  19 Ink and lead refills
                           01 India ink
                           02 Lead refills
Commodity                  03 Pen refills

"Pen refills" = UNSPSC classification 44-12-19-03.
```

Figure 1: UNSPSC classification example

4.2 DAML-S: Semantic Mark-up for Web Services

DAML-S([1]) is an ontology describing Web localizable and accessible Services in a computer-interpretable way. Ontology basic structure is made up of three kinds of information: a *Profile* describing what Service takes from users and returns to them, a *Model* describing how Service functions and a *Grounding* describing how to use the Service.
The basic class of DAML-S ontology is *Service*, whose instances are deployed Services. *Service* is also the domain of properties *presents*, *describedBy* and *supports*, that respectively have classes *ServiceProfile*, *ServiceModel*, *ServiceGrounding* as range.
The class *Service*, then, *presents* a *ServiceProfile* describing Service capabilities; it is *describedBy* a *ServiceModel* showing Service behavior and *supports* a *ServiceGrounding* specifying Service access information.
Each **Service** may be described by at most one **ServiceModel**, linked to at least one **ServiceGrounding**.
The class **ServiceProfile** allows one to describe capabilities both provided by Services and searched by requestors. So it is the most interesting element of DAML-S ontology w.r.t. Service Discovery.
ServiceProfile structure, also extensible with the use of DAML subclasses, makes Service representable through the class **Profile**.
A DAML-S Profile describes a Service through:
Provider Information: general contact information
Functional Description: transformations produced by the Service from required inputs and pre-conditions to outputs and generated effects.
Properties: definition of Service main characteristics.
In the aims of the DAML-S authors, the Functional Description of the **Profile** class should be used to perform the service discovery. A detailed analysis of Profile Properties is out of the purpose of this paper and is available in [1]. We underline here only two Profile Properties we use in our prototype system. The first is **textDescription**, one of the human-readable properties providing a brief textual description of capabilities supplied by the Service. The latter, **ServiceCategory**, belongs to Profile additional attributes referring *i.e.*, to quality and classification of Service. The attribute describes Service category being based upon an eventually external code(UNSPSC in our prototype).

4.3 UDDI

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 knowlwdge:
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**).

UDDI Registry is used here to publish and retrieve web services semantic description. We do not consider here actual business transactions related to the web service use, which should be anyway kept into account in a practical use of the system.

5. A PROTOTYPE SYSTEM FOR SERVICE DISCOVERY

Our prototype for Web Service Discovery is able to find Services by evaluating the semantic match of provided capabilities. To perform such a match we implemented a framework which uses UNSPSC, DAML-S, DAML+OIL and UDDI. The search algorithm is made up of four steps, shown in the following:

- Classification: process of finding the UNSPSC code corresponding to the searched Service.
- Description: representation of the searched Service in an expressive and standard language.
- Extraction: selection of a list of servers providing the searched Service from a UDDI Registry.
- Matchmaking: semantic based assessment of closeness between Service request and descriptions.

The System basic Architecture is shown in Figure 2.

The user performing the discovery could be either a

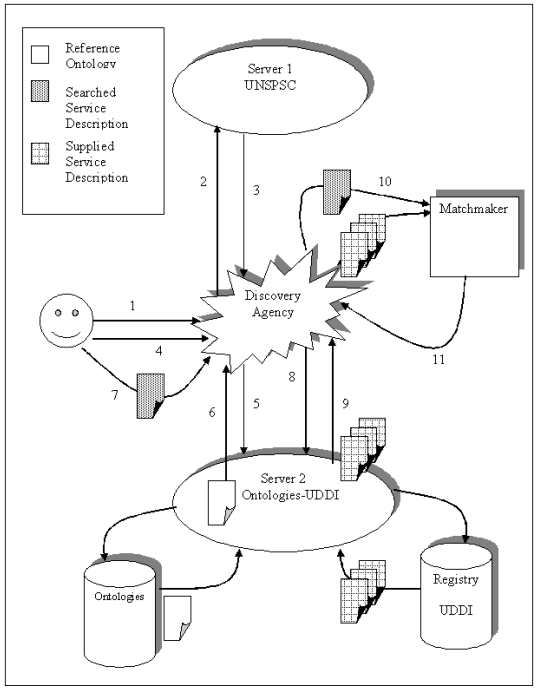

Figure 2: System Architecture

software agent or a human user. In the latter case an applet is used to guide her through the composition of the service description, as shown in Figure 3,

either to search for or to publish in UDDI. The applet dynamically loads a DL-based ontology identified by a UNSPSC code (see par. 5.1 and par. 5.2). With reference to the well-known *triangle diagram* for web services interaction, our framework represents a distributed *Discovery Agency* made up of a UDDI registry, an ontologies repository, a module for UNSPSC codes retrieving and

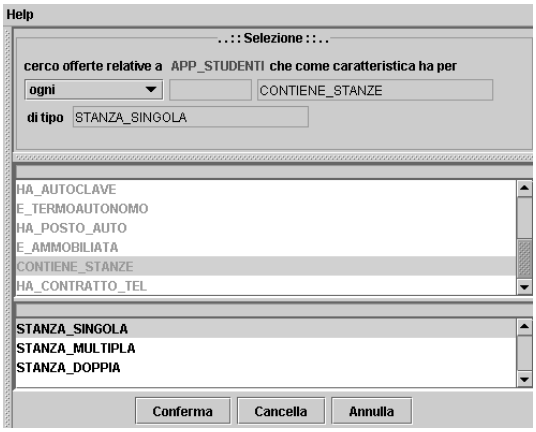

Figure 3: User Interface

a matching engine. The choice of using an ontologies repository rather than referring to their URIs is based on the remark that from time to time they are unreachable or are deprecated. As in our framework we refer to the same ontology to describe the service and to represent dynamically the request, as described below, we always need to have the ontologies availability.

We are going to explain the system behavior w.r.t. the four steps of the search algorithm.

5.1 Step 1: Classification

When finding a Web Service it is necessary to limit the search field by classifying the service in a category. Our system uses UNSPSC codes to classify searched Services. So this step takes in input words describing the request context and returns none, one or more UNSPSC codes as output. We preferably search UNSPSC codes corresponding to the demanded Service at a *commodity* level, the most detailed one. In this way there are less advertisements to analyze in the matchmaking process, but these are also the nearest to the request.

In our prototype a user searching for a Service accesses the *Discovery Agency* web page and submits a keyword expressing the searched product or service context. The Agency queries *Server1* asking for a list of UNSPSC codes, whose name contains submitted keyword. The listed codes may belong to every level of the UNSPSC tree; if a code corresponds to a high level node in the tree it is possible to browse the hierarchical structure of the taxonomy starting from the proposed node.

The list of codes is sent to the user, who chooses the item better corresponding to the searched product or service. The selected code may lay at a commodity level or, if it is on a higher level, the user can expand the item following the tree up to its bottom level.

5.2 Step 2: Description

The UNSPSC code selected by the user is the output of Step 1 and, obviously, is taken in input by Step 2. The output of Description process is a representation of the searched service in a standard and expressive language, able to convey the semantics underlying the request. Users have to express their Services requests according to the same

language by which the service description is represented. This common language is provided by standard reference ontologies which formalize the contexts represented in UNSPSC. In an ideal world we should have either an ontology for each ID in UNSPSC or a global common ontology describing the whole UNSPSC taxonomy. In the former case there would be about 10'000 ontologies related to each other, in the latter one there would be a unique ontology with a huge number of concepts and roles. Both cases are not easily manageable. To overcome these drawbacks we proposed to develop an ontology for each *Family* in UNSPSC. In our prototype the standard ontologies information are provided by the second server. *Discovery Agency* sends the UNSPSC code selected by the user in the previous step to *Server2 Ontologies-UDDI*, that returns to the Agency the reference ontology corresponding to the UNSPSC *Family* of the input code. The ontology is shown to the user, who returns to the *Discovery Agency* the output semantically endowed description of the searched Service according to the reference ontology.

5.3 Step 3: Extraction

In the previous steps we defined the request of the Web Service we are going to discover. This step, instead, aims to extract from the UDDI Registry a list of Service advertisements that can, to some extent, fulfil the request.
In our system *Server2* contains the UDDI Registry, so the *Discovery Agency* queries *Server2* asking for descriptions of Services classified with the UNSPSC code selected by the user in the first step.
UDDI allows providers to be accessed on the Web by deploying all information useful to know and get Services. But no information conveying the semantics can be deployed by means of UDDI. In [17]it is proposed to overcome this shortcoming by mapping DAML-S in UDDI.
DAML-S Profile and UDDI share, in fact, much information, even if it is differently stored by them. DAML-S handles information as literals filling ontology properties; UDDI, instead, makes use of Database records.
The correspondence between information shared by the two models is elicited in Figure 4. DAML-S information not included in UDDI can be conveyed using *Category Bag* and *T-Models* provided by UDDI. A T-Model is a data structure allowing one to specify further attributes for entities yet registered in a UDDI repository. Instances of T-Models are called *KeyedReferences* and include three fields:
KeyName: name of the attribute to elicit with T-Model;
KeyValue: value of the attribute;
TModelKey:key of used T-Model.
A *BusinessService* may be described by KeyedReferences belonging to different T-Models. A *CategoryBag* include all KeyedReferences describing the same BusinessService. So, in order to convey also in UDDI information provided only by DAML-S, we can create a T-Model endowing UDDI with semantics.
On the basis of so far explained models we can extract from the UDDI Registry advertisements semantically fitting the demand. Our extraction method creates a T-Model containing a KeyedReference, whose KeyValue is the UNSPSC code output of the first step. We can so extract only BusinessServices classified with the searched code.
UDDI Server has to return at least three data about extracted advertisements:
discoveryURL: Service socket
Service name
URI: URI of the semantic description of the Service.
The third information is crucial for the search process. It allows us to locate descriptions anywhere in the Web and retrieve them following the reference points provided by URI. This description is embedded in the DAML-S *SeviceProfile* class of the services. The aim of this modelling is to perform a two steps comparison of services capabilities (what the service does).We would like to compare: In the first step the service semantic description to the request semantic representation;
In the second step the semantic representation of service inputs and outputs to the user or agent requirements. The matchmaking approach is different between the two steps. In step one we assess closeness between service request and available service descriptions. For instance, the request "Apartment, Soho, 2 rooms, smoker, dog, garden" and service description "Apartment, center(Piccadilly), car place, fireplace, 2 rooms", could match even if the requestor is not completely satisfied. In the step two instead, the user has to be able to provide all service input constraints and vice versa for the service output constraints. This leads to a subsumption relation between the inputs provided by the user and service input constraints and vice versa for output. For instance, if the service requires credit-card number as input and outputs both a payment receipt and immediate availability, then it can be queried only by users providing at least their credit-card numbers and asking for at most a payment receipt and immediate availability.
Moreover the semantic description may be expressed without language constraints: DAML-S, DAML+OIL, or even WSDL(not endowed with semantics) may be used. The choice of description language obviously affects the degree of match with the request, but the chance of choosing the language of representation provides flexibility to the system.

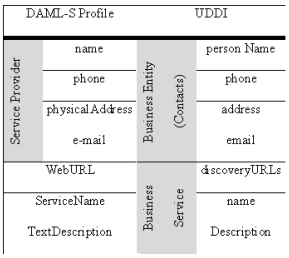

Figure 4: Information shared by DAML-S and UDDI

5.4 Step 4: Matchmaking

Discovery Agency now holds both the description of the Service requested by the user(output of the second step) and the advertisements of Services extracted in the previous step. These descriptions are the input to the Matchmaking step. The Agency is endowed with a Matchmaker module that processes input descriptions according to the algorithm previously outlined, trying to find the advertisement better cor-

```
 Request: student looking for a nice 1/2 bed flat, ch, furnished,
kitchen, washing machine. Price: 150

 Service1: large, fully furnished room, price includes cable TV
and bills. Price: 120
 Service2: double room, suit a couple or two girls, required
deposit. Price: 150
 Service3: room to rent, suit a non smoking female student with
worker, international preferred, deposit required. Price: 80
 Service4: single room in clean flat for nosmoker quiet student,
sharing with 2 others. Price: 120
 Service5: dbl room in shared house, suit single person, use of
lounge, kitchen , garden, rent includes council tax. Price: 85
 Service6: female to share a room in a residential area
flatshare, washing machine, TV, VCR. Price: 81
 Service7: large room in family houseshare with 2 adults, suit
prof / student female, viewing recommended. Price: 95
 Service8: 2 bed flat, perfect for student, double bed ADSL
computer beneath. Price: 600
```

Figure 5: User request and service descriptions in a human-readable representation

```
Request (and Bedroom (all toLetFor Student) (at-least 1 hasBed)
(at-most 2 hasBed) (at-least 1 hasFacilities) (all hasFacilities
(and WashingMachine NoAutonomousHeating FullyFurnished)) (all
hasServices Kitchen) (all price (maximum 150)))

Service1 (and Bedroom (all price (minimum 120)) (all priceIncludes
(and Bill TVPrice)) (all hasFacilities (and FullyFurnished
Spatious)))
Service2 (and (all price (minimum 150))(at-least 2
hasRoom)(at-most 2 hasRoom)(all hasRoom Room)(at-least 2
toLetFor)(at-most 2 toLetFor)(all toLetFor (and Couple
Student))(at-least 1 depositRequired)(all depositRequired Yes))
Service3 (and (at-least 1 depositRequired)(all depositRequired
Yes)(all price (minimum 80)) Bedroom (all toLetFor (and Student
NoSmoker Worker (all sex Female))))
Service4 (and SingleRoom (all toLetFor (and NoSmoker
Student))(at-least 2 occupants)(all price (minimum 120)))
Service5 (and DoubleRoom (all toLetFor Single)(all hasFacilities
(and Lounge Garden))(all hasServices Kitchen)(all price (minimum
85))(all priceIncludes CouncilTax))
Service6 (and Bedroom (at-least 1 occupants) (all price (minimum
81)) (all toLetFor Female)(all hasFacilities (and WashingMachine
TV VCR)))
Service7 (and Bedroom (all price (minimum 95)) (all occupants
Family) (at-least 2 occupants) (all toLetFor (and Student
Professional (all sex Female))))
Service8 (and Flat (at-least 2 hasBed) (at-most 2 hasBed) (all
toLetFor Student) (all hasFacilities ADSL) (all price (minimum
95))))
```

Figure 6: User request and service descriptions in a lisp-like representation

responding to the requested Service.
Our matching engine is based on Java servlets; it embeds a modified NeoClassic reasoner and communicates with it running as a background daemon. The system receives a Knowledge Representation System Specification (KRSS) or DAML+OIL string describing the request/service description.
The Reasoner checks the description for consistency; if it fails, based on the reasoner output, the system provides an error message stating the error occurred. Otherwise the proper matchmaking process takes place.
The module compares requests with service descriptions and ranks discovered advertisements on the basis of their degree of match with the searched Service.

6. EXPERIMENTS AND RESULTS

For test purposes we created eight different web services which were published in a UDDI registry using the proposed mapping of DAML-S in UDDI. For each service it was built a semantic description w.r.t. a reference ontology on *Real estate services* (the UNSPSC *Family* identified by the 80.13.00.00 id) in which it is specified what kind of service is offered, as shown in Figure 5 - 6. Of course we could have services referring to different ontologies. In this scenario we have all the elements to achieve a semantic service discovery:
UNSPSC server for a service category search.
test ontology for semantic service description identified by a UNSPSC id.
semantically described web services represented by a code which is the same or at most is strictly related to their reference ontology name (i.e. the UNSPSC *Family* code)
UDDI registry server in which is stored web services information to be searched.
The result of the service discovery is a set of ranked URI representing the services which best match the request as shown in Figure 7. Of course if the request is the same as, or sub-

Figure 7: Returned page with ranked services for the *Request* in Figure 6

sumes, a service description, we will have the corresponding item ranked with 0 (see the *Let's Let* service in Figure 8). During the discovery we can encounter basically two kinds

Figure 8: Returned page with ranked services for a Request description subsumed by the *Let's let* one

of error in two different phases of the search process: error due either to the absence in the UNSPSC taxonomy of the searched category or to the inconsistency of the request description w.r.t. the reference ontology. In the former case the error is catched by the *UNSPSC server* and a message of *No Items Found* is sent back to the client, in the latter case the error is catched by the *Matchmaker module* which sends back to the client a message containing the description

parts causing the inconsistency. In both cases the client is able to modify its search specification.

7. CONCLUSIONS

In this work we have proposed a framework for web service discovery based not only on a rigid categorization (UNSPSC) but also on the semantic model of a web service (DAML-S upper ontology) and of its domain (domain ontology). With the proposed system, which overcomes simple subsumption matching of services descriptions, we are able to classify web services matching their semantic descriptions in order to establish both how far the offered service is from the requested one and which are the differences in terms of logical constraints. Having a ranked list of candidate web services, it is possible then to assign a preference value useful for a further interaction with the "discovered" web services, that is the negotiation of logical constraints, in the request description, which are not satisfied by discovered ones. We are currently working on a new version of the system that will return *input* and *output* **DAML-S Service Profile** properties for a two steps search. In the first step the comparison is performed between requested service and offered descriptions using the algorithm in [9], implemented in the ***Matchmaker*** module of the proposed architecture; in the second step the comparison is performed between input and output descriptions using a subsumption algorithm.

8. REFERENCES

[1] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. Daml-s: Semantic markup for web services. In *Proc. of the Intl. Semantic Web Workshop*, 2001.

[2] A. Borgida. Description Logics in Data Management. *IEEE TKDE*, 7(5):671–682, 1995.

[3] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In *Proc. of ACM SIGMOD*, pages 59–67, 1989.

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

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proc. of PODS'98*, pages 149–158, 1998.

[6] P. Devambu, R. J. Brachman, P. J. Selfridge, and B. W. Ballard. LASSIE: A Knowledge-Based Software Information System. *Comm. of the ACM*, 34(5):36–49, 1991.

[7] T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proc. IJCAI-03*, pages 337–342, August 9–15 2003.

[8] T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In *Proc. SAC '03*, pages 582–586. ACM, 2003.

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

[10] E. Di Sciascio, F. Donini, M. Mongiello, and G. Piscitelli. A Knowledge-Based System for Person-to-Person E-Commerce. In *Proc. of KI-2001 Workshop on Applications of Description Logics (ADL-2001)*. CEUR Workshop Proceedings vol. 44, 2001.

[11] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR'91*, pages 151–162. Morgan Kaufmann, Los Altos, 1991.

[12] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

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

[14] I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In *Proc. of IJCAI 2001*, pages 199–204, 2001.

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

[16] B. Nebel. Terminological Reasoning is Inherently Intractable. *Artif. Intell.*, 43:235–249, 1990.

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

[18] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *The Semantic Web - ISWC 2002*, number 2342 in LNCS, pages 333–347. Springer-Verlag, 2002.

[19] S.Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, pages 96–99, 2002.

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

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

[22] J. R. Wright, E. S. Weixelbaum, G. T. Vesonder, K. E. Brown, S. R. Palmer, J. I. Berman, and H. H. Moore. A Knowledge-Based Configurator that Supports Sales, Engineering, and Manufacturing at AT&T Network Systems. *AI Magazine*, 14(3):69–80, 1993.