The mediator centric approach to Automatic Web Service Discovery of COCOON Glue

Emanuele Della Valle, Dario Cerizza and Irene Celino

CEFRIEL - Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
dellavalle@cefriel.it, cerizza@cefriel.it, celino@cefriel.it

Abstract. Automatizing the Web Service Discovery is crucial for truly implementing a Service Oriented Architecture. Semantics has been shown to be useful. Several initiative, namely OWL-S, WSMO and WSDL-S, are successfully employing ontologies, but we believe that WSMO is right in highlighting mediation as *the* missing element. In this paper we provide a mediator centric refinement of the conceptual model for WSMO discovery and the related architecture as well as the prototypical implementation (named Glue) we are using in the projects COCOON and Nomadic Media.

1 Introduction

In a Services Oriented Architecture (SOA) a requester entity may not know which provider entity to engage. A requester entity should only know the functional criteria of the service it wishes to interact with and it can find out suitable candidate services by inquiring a *discovery agency*. In this way such discovery agency decouples requester entities from provider ones.

Web Services are meant to enable the implementation of a SOA. As reported in the Web Service glossary[1], **discovery** is "*the act of locating a machine-processable description of a Web Service that may have been previously unknown and that meets certain functional criteria*". UDDI [1] provides a general purpose model for Web Service discovery by gathering metadata about a collection of Web Services and making that information available in a searchable way (white, yellow and green pages). But, as stated in section 1.4.5 "Overview of Engaging a Web Service" of W3C Note on WSA [2], such meta-data requires initial knowledge about both Web Service existence (e.g., which tModel specified by EAN/UCC it implements) and location (e.g., the name of a green page category). UDDI does not provide any formal and explicit way to exchange such initial knowledge; therefore today the most common approach to obtain it is through e-mail exchanges or word of mouth. This requires to keep humans in the loop and limits scalability as well as economy of Web Services.

The Semantic Web is making available technologies which support (and, at some degree, automate) knowledge sharing. In particular, several initiatives (e.g.

[1] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

OWL-S[2], WSMO[3], WSDL-S[4]) provide evidence that ontologies, with their ability to interweave human understanding of symbols with their machine processability, can play a key role in automating Web Service Discovery [3][4][5][6][7]. All approaches share the idea that:

- *at publishing time*, a set of relevant domain ontologies can be used to semantically annotate Web Service descriptions (i.e., describing the capabilities of such Web Services),
- *at discovery time*, the *same set of ontologies* can be used to describe the functional criteria of the service the requester entity wishes to interact with; therefore, the Discovery engine, accessing the knowledge modeled in the ontologies, is not limited to syntactic matching techniques, but it can take into consideration also semantic matching techniques (e.g. subsumption base matching [5], transaction logic [8]).

The problem is that such approach may work in a small controlled environment, but it is unrealistic in open environments in which actors suffers from various form of **polarization** (e.g., philosophical position, sense of belonging, etc.). We observe that in the real world:

1. in the same domain *different proposals for domain ontologies supported by competing cartels* are under development and are likely to be maintained also in the future.

   For instance, in the health care field, in which the need for sharing a common knowledge has been addressed for more than a decade, several competing terminology standards are available for describing pathologies, e.g. International Classification of Disease[5] (ICD) and SNOMED Clinical Terms[6].
2. *the points of view of providers and requesters on the shared knowledge are different*, and this is reflected in different ontologies respectively used for annotating Web Services and for formulating requests.

   For instance, consider the knowledge of date-time used in the field of arranging meetings for consultancy. A week-based calendar (e.g. each Thursday afternoon and Friday morning) may be preferred by provider entities in stating the nominal availability of a consultant, whereas a Gregorian calendar (e.g. April, 9th from 10.00 to 12.00) may be preferred by a request entity in requiring advice. Even if both provider and requester entities share the same knowledge on date-time, they may prefer to use different ontologies.

A possible objection is that OWL already provides constructs to reason on equality (e.g. `sameAs`, `equivalentClass`, `equivalentProperty`, etc.), but those are not sufficient to deal with ontology alignment.

The Web Service Modeling Ontology (WSMO) working group has moved a step forward in the direction of semantic empowered Web Service Discovery [8][9]

[2] http://www.daml.org/services/
[3] http://www.wsmo.org/
[4] http://lsdis.cs.uga.edu/projects/wsdl-s/
[5] http://www.who.int/classifications/icd/en/
[6] http://www.snomed.org/

by introducing the notion of *mediation* [10]. Mediation in WSMO is, together with strong decoupling, a foundation principle. *Mediators address the problem of handling the heterogeneities that naturally arise in open environments.* WSMO proposes a classification of mediators according to their role in WSMO conceptual model: Ontology to Ontology mediators (ooMediators), Goal to Goal Mediators (ggMediators), Web Service to Goal Mediators (wgMediators), etc[7].

A model for automatic location of services (i.e. service discovery plus service contracting) in WSMO is introduced in [9] and in [12]. Such model provides the basis for a variety of solutions that are easy to use for requesters, and that provides efficient pre-filtering of relevant services and accurate contracting of services that fulfill a given requester goal. Moreover the preliminary architecture for WSMO compliant discovery engines is presented in WSMO Discovery Engine D5.2 [13] and in WSMX Discovery D10 [14].

In this paper we provide a **refinement of WSMO discovery conceptual model** centered on mediators (cf. section 2):

- by making the notion of class of goals and class of Web Service descriptions explicit,
- by using ggMediators for automatically generating a set of goals semantically equivalent to the one expressed by the requester but expressed with a different form or using different ontologies;
- by *making wgMediators the conceptual element responsible for evaluating the matching*;
- by using *ooMediators* for solving any terminological mismatch that can appear with different polarized ontologies for the domains, and
- by redefining the *discovery mechanism as* ***a composite procedure*** *where the discovery of the appropriate mediators and the discovery of the appropriate services is combined.*

Moreover, in section 3, we provide a description of the **refinement of WSMX Discovery Engine architecture** according to our refined WSMO discovery conceptual model both in terms of components and execution semantics (cf. section 3.1) and we show how we **implemented it in** Glue[8] using F–logic[15] (cf. section 3.2). In section 4 we introduce a *use case of Service Discovery in the healthcare field* and we show how we put Glue at work in the projects COCOON[9] and Nomadic Media[10]. Finally in section 5 and 6 we respectively present related works and we draw some conclusions providing insight view of our future developments.

[7] We do not provide here a complete list of them and we prefer to refer to WSMO D.2 [11] for a detailed explanation of their usage.

[8] http://glue.cefriel.it

[9] COCOON is a 6th Framework EU integrated project aimed at setting up a set of regional semantics-based healthcare information infrastructure with the goal of reducing medical errors.

[10] The Nomadic Media project resides under the Eureka/ITEA program. Within this project we addressed the problem of out to provide mobile access to healthcare services.

2 The concept

In this paper we strictly refer to the automatic location of services proposed in [9] for WSMO. We state this because we agree with [16] that when talking about Web Services many notions (including *service*) are semantically overloaded and one should commit to a precise meaning when using such notions.

We agree on the definition of Service (including the distinction between *abstract* and *concrete* service) and of requester need (that reflects the conceptual element of WSMO named *goal*). We hold with the idea of pre-defined goals and with parameterizing them, so that a requester has only to locate (eventually unconsciously by interacting with a standard application) one of the pre-defined goals and to provide concrete values for its parameters. But we prefer to reserve the name goal for a concrete goal whereas we suggest to call **class of goals** a specific *parametrized and pre-defined goal*.

We agree with the idea of having parametrized and pre-defined Web Service descriptions that can be instantiated by the provider entities, so that two concrete descriptions differ in the values given to the parameters. Consider, for instance, a scenario where there are some technical specialists offering virtual meetings through a collaboration platform. This platform exposes one distinct Web Service for booking a meeting with each specialist. The structures of all these Web Services are similar but the Services provided are different since they depends on the technical capabilities of each specialist. The Web Service descriptions of these Web Services are multiple instances of the same class of parametrized pre-defined Web Service description. As we suggest for goals, we prefer to make explicit the notion of **class of Web Service description** (a pre-defined Web Service description) and of *instance of Web Service description* (a Web Service description).

Having in mind this class-based approach, we can adopt the inheritance model typical of Object-Oriented programming languages, where a class can be subsequently refined in child classes providing a more specific structure and behavior. In this way, it is possible to extend classes of Web Service descriptions and classes of goals in a more fashionable manner, useful in domains where a lot of similar but non-identical Web Services need to be discovered.

In order to describe our concept, we consider the whole process in three subsequent phases: *Set up time*, *Publishing time* and *Discovery time*.

During *Set up time*, the system is being initialized with all the necessary information for performing automatic service discovery in a generic domain $\mathbf{D}$. This information includes domain ontologies, Web Service description classes, goal classes and the required mediators. As discussed in the introduction, given a specific domain $\mathbf{D}$, it is natural that provider's point of view is reflected in a particular polarized $\mathbf{D}^+$ domain understanding and, on the other hand, requester's point of view is reflected in a differently polarized $\mathbf{D}^-$ domain understanding. Thus, having different points of view the standard process of agreement on an ontology for the domain $\mathbf{D}$ would require a lot of effort. Instead of developing one ontology, we suggest to follow the activities shown in figure 1. The Semantic Web service expert (SWS Expert) is the main actor involved, since he is able

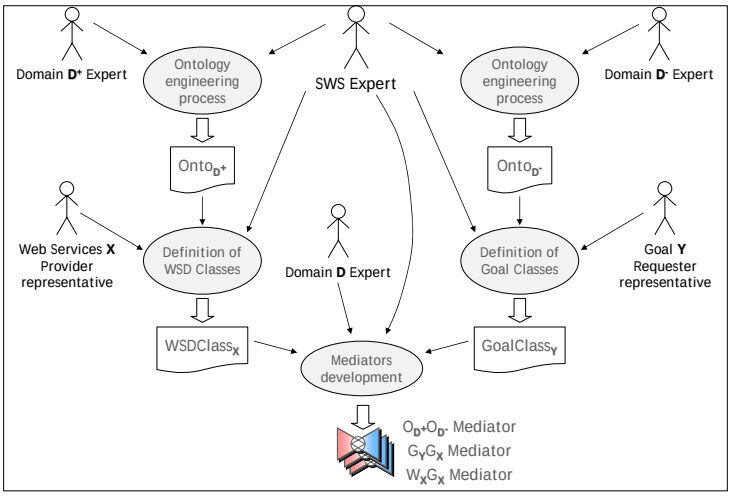

Fig. 1. The sequence of activities performed by actors at Set up time

to understand the internal semantic notation of the system. Domain ontologies for $\mathbf{D}^+$ and $\mathbf{D}^-$ are developed (integrating already available sources) during separated engineering processes that involve the relative domain experts with the SWS Expert. When the ontologies are ready, a provider representative and the SWS Expert define the class of Web Service descriptions for the class **X** of Web Services using the ontology for domain $\mathbf{D}^+$. Meanwhile, a requester representative and SWS Expert define the class **Y** of goals. As highlighted, the SWS Expert does not have neither to ask provider and requester entities to commit to the same set of ontologies (which may prove to be an unworkable plan) nor to develop a complete mediator between all the conflicting ontologies (which may be unnecessary if only a small portion of such ontologies is involved in the matching process). For this reason, the SWS Expert needs to be supported by a non-polarized expert for the domain **D**, that can help him solving ontology alignment problems between the two different domains. **This process generally involves the definition of various mediators**. In our case, a ooMediator, a ggMediator and a wgMediator will work, but more complex scenarios can be imagined. The first mediator implements the domain-specific rules for solving terminology mismatches between the involved portion of ontology $\mathbf{D}^+$ and ontology $\mathbf{D}^-$. The ggMediator is developed in order to translate the instance of goal **Y** into semantically equivalent instances of goals that are more suitable for discovery instances of Web Services **X** (later named goal **X**). This mediator uses the functionalities provided by the previously developed ooMediator. The latter wgMediator implements the domain-specific rules for matching instances of just-translated goal **X** with instances of Web Service description **X**.

At *publishing time*, provider entities publish instances of Web Service descriptions simply by referring to the correct class of Web Service description

and providing values to all the necessary parameters. The system creates the instances and register them in its internal repository in order to retrieve them when a wgMediator will require it.

Finally, *at discovery time*, when a goal instance is created and submitted, a *look up* mechanism can be used for selecting the class of goals (being the submitted goal an instance of a class). Similarly, a look up mechanism can be used for selecting the set of appropriate ggMediators able to translate the user goal in a set of semantically equivalent goals. Then again, the appropriate wgMediators can be looked up and the instances of Web Services descriptions can be roughly *filtered* on the basis of the target of the detected wgMediators. A set of ooMediators is used to solve any terminological mismatch. And, finally, the ***similarity rules coded in the wgMediators can be used to match*** the given goal against the instances of Web Service descriptions obtained by filtering, returning an ordered list of references to the concrete Web Services.

3 Glue Architecture and Implementation

The goal of our work was to design and build a system suited for medium scale deployment (up to some tens of classes of Web Service Description and of classes of goals, using ontologies of a couple of thousand concepts each, but with some hundreds of instances of Web Service Description in each class) while providing a lightweight stand alone implementation[11].

3.1 Architecture

Having a such mediator–centric vision in mind, we propose to refine the architecture described in WSMO/X Discovery [13][14] accordingly to the refinement we propose in section 2. WSMO/X discovery architecture envision the presence of a *Communication Manager* (which handles the incoming requests), two *WSMO Element Repositories* (one for goals and one for Web Service descriptions), a *Message Parser* (which analyzes the incoming message), a *Proof Generator* (which constructs logical formulas for running the matchmaker) and a *Reasoner Manager* (which abstracts from the different interfaces of the reasoners).

We propose to extend the WSMO Discovery Engine architecture, refining several **components** (see figure 2). First, we decomposed the Message Parser into three components: Web Service description constructor, Goal Constructor and Goal Translator. The two *Constructor Components* serve for instantiating respectively the instances of Web Service descriptions and goals of classes registered at set up time. The *Goal Translator* looks up ggMediators that are appropriate for the goal instance passed by the goal constructor and translates such goal in a set of provider–oriented goals. Secondly, we refine the *Proof Generator* component giving it also some of the responsibilities of the *Message Parser*. The part we include in the Message Parser is responsible for looking up the

[11] In WSMX the discovery engine is part of a much larger architecture and is not implemented as a stand alone component.

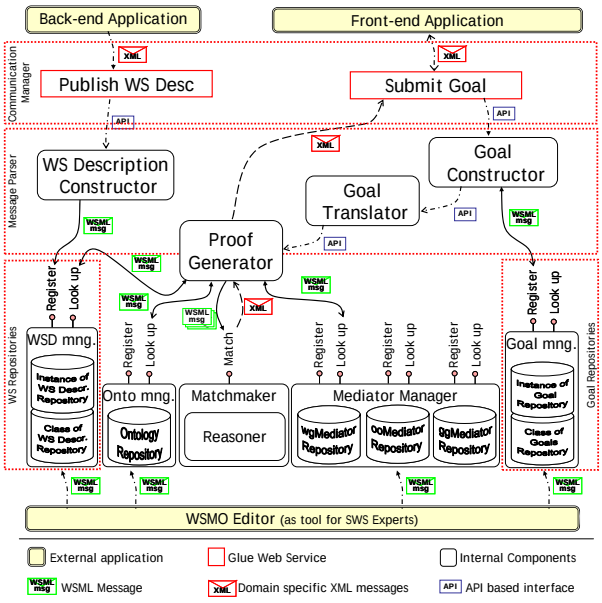

Fig. 2. The architecture of the mediator centric extension to the WSMO Discovery Engine we propose in this paper. Dashed lines represent the components described in the original WSMO Discovery Engine described in [14], while the other components are those we introduce in this paper.

mediators and the ontologies required for executing the matching whereas the part outside the Message Parser roughly filters the Web Services descriptions, it loads them and it executes the logical formulas for running the matchmaker. Aside, we introduce all those components that satisfies our need for a lightweight implementation of Glue, even if they are generally envisioned as part of the entire WSMX architecture [17]: the two Repositories, one for ontologies and one for mediators, and a WSMO Editor (such as WSMO Studio) as tool for SWS Experts.

In figure 3, we depict the **execution semantics** describing the interactions among the components both in case a new Web Service description is published (the interactions labeled with 'a', 'b' and 'c') and when a goal is submitted (the interactions labeled with numbers).

The first set of interactions describe how the "publish Web Service description communication component" invokes the WS Description Constructor passing a class identifier and a set of parameters (cf. line 'a'), how this component uses the class identifier to look up the class of Web Service description (cf. line 'b'), how it instantiates the Web Service description and how it registers the constructed instance in the Web Service description instances repository (cf. line 'c').

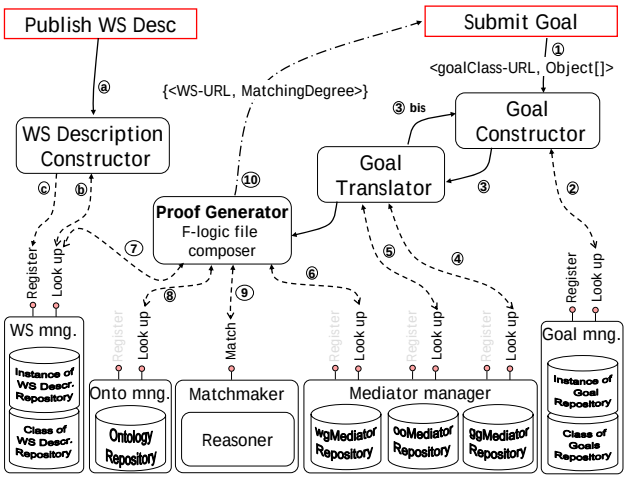

Fig. 3. This collaboration diagram depicts the components of Glue and their role in supporting the discovery process.

The second set of interactions describes our *composite discovery procedure* that expects as input an identifier of a class of goals together with a set of structured parameters (cf. line 1) and provides, as result (cf. line 10) a list of pairs in which the first element is a reference to a Web Service and the second element is the degree of matching [12].

The first part of the procedure (cf. line 2) consists in looking up the WSML description of the class of goals corresponding to the given identifier and instantiating the concrete representation of the goal using the given parameters.

The second part of the procedure concerns the use of ggMediators in translating the constructed instance of goal in an equivalent instance expressed using a different terminology (cf. line 4). In turn, each ggMediator optionally uses an ooMediator for reconciliating semantic heterogeneities (cf. line 5). If necessary, the Goal Translator component can call back the Goal Constructor (cf. line "3 bis") for instantiating other goals instances.

The third part of the procedure is undertaken by the Proof Generator. This component is responsible for constructing the formulas for running the matchmaking. For each instance of goal, it starts looking up the wgMediators that *has as target* the class of goal having such goal as an instance (cf. line 6). Then, for each wgMediator, the Proof Generator looks up the Web Service description instances of the class that *is the source* for such wgMediator (cf. line 7) and it also looks up all the required ontologies (cf. line 8). All the retrieved ontologies and instances are sent to the matchmaker (cf. line 9), that uses the reasoner for

[12] exact, subsumed, plug-in and intersection as proposed in WSMO D5.1 and in many other earlier works related to OWL-S [5][6]

evaluating the similarity rules coded in each wgMediator, and returns references to the discovered Web Services and the degree of matching as a list of pairs (cf. line 10).

3.2 The prototype implementation of Glue

In developing Glue, a prototype of this proposed architecture, we faced two major choices: the reasoner and the format for the logical language. We knew that we needed both an ontological language and a rule language, but we were also constrained by the efficiency we aim at.

WSML working group[13] proposes in [18] a subset of OWL (named OWL$^-$) that can be translated into f–logic [19]. This, according to WSML working group, allows for efficient query answering and for easy implementation of a rule on top of the ontology. Moreover, we were looking for an expressive datatype support and WSML working group showed in [20] that OWL–E [21] (a proposal for extending OWL with expressive datatype expressions) can be added to OWL$^-$ and the resulting ontological language, named OWL-Flight, can still be translated into f–logic. But, at the time we started developing Glue (September 2004), WSML efforts, in providing a language for formalizing WSMO, were a work–in–progress and the tools for translating WSML into reasoner–specific formats were missing. Therefore we decided to directly use a dialect of f–logic implemented in Flora–2, an open source f–logic inference engine that runs over XSB[14], an open source implementation of tabled-prolog and deductive database system. This choice allowed for an early proof of concept without constraining compatibility with WSML.

Plugging Flora–2 in Glue involved some consequent implementation choices: the proof generator in the prototype is mainly a *f–logic file composer*; all the information exchanges, which we envision in the architecture as based on WSML, are implemented directly using f–logic; and all the WSMO element repositories manages directly f–logic files, keeping in the internal SQL syntax the necessary relationships between mediators, ontologies and classes.

4 A case of mediator centric Discovery for eHealth

Most of times, Service Discovery is treated as a back-end problem, but actually, if Semantic Web Services efforts will succeed, the use of Service Discovery tools in the future will be as frequent as using search tools today. Therefore, in COCOON project, beside envisioning a clear back-end use of Service Discovery (in combination with healthcare application protocols such as HL7), we also envision a realistic use case of its daily employment. In this section, we describe the usage scenario for Service discovery as implemented in COCOON project and some details about the real usage of Glue in this scenario.

[13] http://www.wsmo.org/wsml/
[14] http://xsb.sourceforge.net/

4.1 Usage scenario

In COCOON project, we are evolving a usage scenario[15] of Web Service Discovery that describes an interaction between a General Practitioner (GP) and COCOON platform with the intent to find *medical advice* and *teaching* services offered by specialists organized in communities of practice (CoP). In particular, Glue takes responsibility for enabling on demand access to services for arranging virtual meetings; the actual arrangement and the subsequent meeting is supported by the collaboration services provided by the COCOON platform that acts as a front-end application from the Glue point of view (see figure 2).

The general criteria for matching a GP goal against the description of the services offered by a CoP are the correspondence between the GP's problem and CoP's medical capabilities and the matching between the GPs' date–time preferences and the nominal availability time of each CoP.

Medical advice will predictably be the most frequent reason for a GP to start an interaction, as it normally could be triggered during the practice time (e.g. questions by patients). Most of times, the *clinical capabilities* of the CoP may be the ones the GP wishes. But, sometimes (e.g. for more difficult and rare patient cases) *research capabilities* of the specialists involved in a CoP may be sought.

Teaching, on the other hand, will be predictably less frequent and a reason to start a request to the system could happen in the 1-hour/week time that is normally reserved for contacting peers[16], as it normally could be triggered by GP's reflection on his/her week's practice.

In order to facilitate the understanding of this scenario, in figure 4, we show Glue surrounded by a set of CoPs (which are provider entities) and a COCOON platform (which is a requester entity). Each CoP exposes the functionality of arranging the two types of meeting as a Web Service. As discussed in 2, the process that enables a GP to arrange a meeting with the most suitable CoP can be broken down in the following tasks:

- Set up time:
  1. the service provider and requester entities **agree on the ontologies to use** for modeling pathologies (e.g. they may agree on using the ICD), drugs (e.g. they may choose International Nonproprietary Names for Pharmaceutical Substances – INN[17]), advice services, date–time, etc.;
  2. **if they cannot agree** on the use of a specific set of common ontologies, the use of mediators is required as discussed in 2 . In this scenario, the CoP providers and the requester entities cannot agree on the use of a common date–time ontology. The CoP provider entities prefer to express the nominal availability of each CoP using a *week-based calendar* (e.g. the advice service is available on Thursday afternoon and Friday morning),

[15] The various versions of our usage scenarios are publicly available at http://cocoon.cefriel.it/RD2/usecases/

[16] As reported in a February 2005 national survey of Italian GPs.

[17] http://www.who.int/medicines/organization/qsm/activities/qualityassurance/inn/orginn.shtml

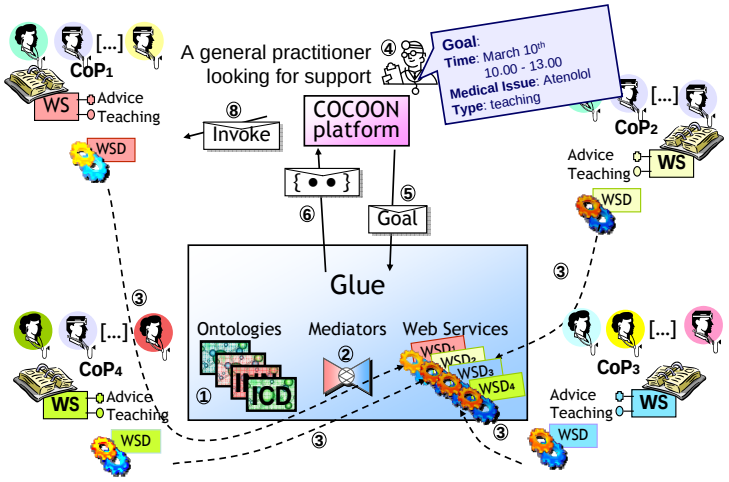

Fig. 4. A case of Service Discovery that enables a general practitioner to find the most appropriate medical advice/teaching service.

whereas the requester entity prefer to express users' preferences using a *Gregorian calendar* (e.g. is the service available on April, 9th from 10.00 to 12.00?);

– Publishing time:

3. each CoP provider entity **publishes** inside Glue its Web Service descriptions for arranging a meeting, describing the clinical capabilities the CoP holds and the date–time intervals the CoP is normally available (the nominal availability for advice and teaching may differ). For instance, a CoP provider entity may register its CoP as "*a community that delivers intervention based on alpha and beta blockers with nominal availability on Monday, Tuesday and Friday in the afternoon for advice and on Tuesday for teaching*";

– Discovery time:

4. similarly, a GP can discover the most suitable CoP by using a GUI, provided by COCOON platform, in order to **express his/her goal** in term of the available ontologies. For instance the GP asks "*a teaching session on the use of Atenolol, preferring the meeting to be arranged on June 8th from 10.00 to 13.00 or on June 9th from 13.00 to 16.00*";
5. COCOON platform **submits** the goal to Glue;
6. Glue **performs the composite discovery process** described in previous sections **matching** the GP goal against the descriptions of the advice/teaching services offered by each CoP; then it returns a list of references to Web Services for arranging a meeting, ordered by decreasing relevance;
7. the **results list is displayed** to the GP;

8. the GP interactively **selects** one of CoPs untill he/she finds one to arrange a meeting with.

4.2 Putting Glue at work

In order to test Glue, we modeled in WSMO the use case just illustrated. We used f–logic to describe the ontologies, the classes/instances of Web Services descriptions, the classes/instances of goals and the wgMediators. Then, we populated Glue with some tens of realistic descriptions of Web Services for arranging meetings with a CoP.

The **ontologies** necessary to support this use case are the COCOON medical ontology, the advice/teaching services ontology and two calendar ontologies.

COCOON ontology is a demonstrative ontology of hypertension and breast cancer domains derived from ICD-10 and INN. It contains the definition of a hundred concepts (like `disease`, `hypertension`, `breast neoplasm`, etc., `medication`, `beta-blockers`, etc., `part of the body`, `heart`, etc., `specialist`, `cardiologist`, etc.) and the relations among them (like beta blockers `control` hypertension, cardiologists `deal with` heart, hypertension `affects` heart and arteries, etc.).

The *advice/teaching ontology* describes the concepts of clinical, research and teaching capabilities of a Community of Practice.

- *Clinical Capabilities* describes the CoP in terms of:
  - `hasClinicalSpecialists`: the list of the kind of specialists grouped by the CoP (e.g. Cardiologist, Urologist, Pneumatologist, Dermatologist, etc.),
  - `managesDiseases`: the list of diseases managed by the CoP as ICD codes (e.g. Diabetes – ICD9CM 250.00), and
  - `deliversInterventions`: the list of the diagnostic / therapeutic / preventive interventions (including pharmaceuticals) delivered by the CoP;
- *Clinical Research Capabilities* describes the CoP in terms of
  - `hasResearchSpecialists`: the list of the kind of specialists grouped by the CoP (e.g. Statistician, Social worker, Psychologist),
  - `studiesDiseases`: the list of diseases which are actively researched by the CoP (e.g. Gastric ulcer [ICD10–K25] Prevention), and
  - `studiesInterventions`: the list of the diagnostic / therapeutic / preventive interventions (including pharmaceuticals) which are actively researched by the CoP; and
- *Teaching Capabilities* describes the CoP in terms of
  - `hasTeachingExpertise`: the list of teaching roles that the CoP can fulfill (e.g. Teacher, OnlineTeacher, Tutor, OnlineTutor, etc.)
  - `hasAuthoringExpertise`: states the availability of online/offline collaborative working tools (i.e. for teaching) within the CoP (e.g. NetMeeting, Skype, Messenger, etc.)

Finally, as discussed in 4.1 two *calendar ontologies* are necessary in our use case to express the date–time intervals. Therefore an *ooMediator* has been employed in translating the date–times from the Gregorian calendar to the week-based one. In our implementation, this ooMediator was realized with a Java program exposed as a Web Service used at discovery time by Glue.

Having these ontologies, we were able to describe in WSMO the capabilities of the class of **Web Services** for arranging a meeting with a CoP. We define a class hierarchy of Web Service descriptions with a generic class on top (for meeting arrangement in a given set of date–time intervals) and two specific classes below (for arranging respectively advice and teaching meetings).

The description of the generic meeting arrangement class of Web Service asserts that:

- the *pre-conditions* are: the input has to be the information about an advice request, the general practitioner has to ask an advice on one of the medical issues treated by the various CoPs; and the booking date has to be after the current date;
- the only *assumption* is that the general practitioner has the right to use the advice service;
- the *post-conditions* describe the possible meetings the CoP is available for: it can offer support that regards its capabilities and it can provide support only during its nominal available times;
- the *effect* is that the agendas of both the GP and the specialists in the CoP are updated with a reference to the scheduled meeting.

In a similar manner, we defined a hierarchy of **classes of goals** that asserts GP's need of finding a CoP that can provide advice or teaching support on a given medical issue in the date–times intervals the GP prefers.

As described in section 4.1, in our use case no agreement was reached in the date–time ontology to use. To bypass such heterogeneity we defined also a parallel hierarchy of classes of goals that express the GP goal in terms of the week-based calendar ontology (the one chosen by CoP providers) and we used a **ggMediator** for translating instances of goal from one class to the other. This ggMediator, when invoked, simply rewrites the goal formulated by the GP using Gregorian dates (e.g. June, 8th 2005), translating it into days of the week (e.g. Wednesday) through the ooMediator illustrated above.

Finally, as we described in section 3.2, we expect SWS Expert to encode in Glue a set of **wgMediators** with the similarity rules for matching a class of goals against a class of Web Services descriptions. For instance, the rule that performs an exact match between what the `GP is asking for` and the medical capabilities of a CoP says that there is an exact matching when:

- the `GP is asking for` a specialist and
  - the CoP `has that clinical specialist`,
  - or the CoP `manages a disease` that affects a body part dealt by the specialist the GP is asking for,
  - or the CoP `delivers an intervention` that controls one of diseases treated by the specialist the GP is asking for,
- the `GP is asking for` a disease and
  - the CoP `has a clinical specialist` that deals with a body part affected by the disease the GP is asking for,
  - or the CoP `manages the disease` that the GP is asking for,
  - or the CoP `delivers an intervention` that controls the disease the GP is asking for,

– the `GP is asking for` an intervention and
  - the CoP `has a clinical specialist` that deals with a body part affected by a disease controlled by the intervention the GP is asking for,
  - or the CoP `manages a disease` controlled by the intervention the GP is asking for,
  - or the CoP `delivers the intervention` that the GP is asking for.

The rules for subsume and plug-in matching mainly differ from the one presented above because they broaden the search space to subconcepts and superconcepts respectively, navigating the COCOON domain ontology. Beside these rules that match medical capabilities, there are other different rules that matches date-time intervals between goal and Web Services description.

Having two parallel hierarchies of classes, we wrote three wgMediators: one links a generic service for arranging a meeting to a generic goal for requesting support, while the other two link respectively a service for arranging advice meetings to a request for advice and a service for arranging a teaching meeting with a request for teaching support. Since the rules in the three wgMediators largely overlap, we found useful the possibility of building also hierarchies of wgMediators, so that the two specific wgMediators can be defined by extending the generic one and reusing its rules.

For lack of space we don't present in this paper all the internal f–logic syntax of our scenario. Readers can refer to Glue Web site[18] for more detailed information.

5 Related works

The work we present in this paper is strictly related to the activities of WSMO/L/X working groups. Like other articles proposed in the context of WSMO, it moves away from OWL-S. In OWL-S approach goals and Web Service description must be defined using the same ontologies and the automation of Web Service discovery normally relays on subsuption reasoning (e.g. [4][22][6]). In WSMO, on the other hand, goals and Web Services can be annotated using different ontologies and the relationships between them can be captured by the mean of wgMediators. This, clearly, requires to step aside the lack of rules in the OWL languages. In this paper, we proposed to use f–logic rules within the wgMediator to capture the knowledge for reconciliating the conflicts that arise when goal and Web Service descriptions are not annotated using the same ontologies. This approach is similar to an early work on OWL-S, in which the service discovery problem is formulated as a rewriting problem where requests are attempted to be rewritten in terms of available services [3], but it takes a much easier approach in coding the matching rules directly in the wgMediator.

The relatively easiness in describing classes of Web Service description and of goals and in coding the matching rules makes the approach we propose very efficient. We easily modeled the presented use case and the performances[19] of

[18] http://glue.cefriel.it/

[19] the machine we used for the tests is a 2800MHz Pentium 4 processor with 1 gigabyte of RAM

the COCOON Glue Discovery Engine with 50 Web Service descriptions remains under 3 seconds. However one can move to our approach the criticism that it does not relies on generic notion of matching such as subsumption reasoning or transaction logic. Our position is that such criticism is only partially true, because one can encode in the wgMediator a subsumption based matching, but at the time we are writing this article we cannot provide evidence that such approach is convenient.

6 Conclusion and future work

The main lesson we are learning bin applying WSMO in the healthcare field is that the clear separation between the ontologies used by each entity simplifies and speeds up the gathering of consensus, which is alway difficult to reach in large groups, and especially in healthcare field. This is mainly due to the adoption, in WSMO, of mediators. In particular, wgMediators appear to offer a flexible way for describing similarities between goals and Web Service descriptions, hence for enabling a semantic match between them.

Finally, the tasks that are currently being the subject of our research are:

- WSMO discovery
  1. extending our approach with the notion of *intention* as presented in [9], such extension will provide COCOON Glue WSMO Discovery Engine with more degrees of matching and it will enable future support for contracting;
  2. aligning our work with the WSML family of languages that are currently being defined as part of the WSML working group activity, in particular with WSML-rule; and
  3. aligning our work with the WSMX architecture providing COCOON Glue WSMO Discovery Engine as a WSMX plug-in.
- the COCOON project
  1. selecting and adjusting the ontologies required for describing the other healthcare services offered in COCOON, through the development of (possibly ad-hoc) mediation services to overcome heterogeneity of the various healthcare related ontologies;
  2. extending the test cases of our Discovery Engine to include the other components still under development in COCOON project (i.e. semantic information retrieval and clinical guideline based decision support system); and
  3. extending the approach to the regional eHealth services (starting from the SISS in Lombardy - Italy).

Acknowledgements

The research has been supported by the COCOON (IST FP6-507126) integrated project and Nomadic Media ITEA (Information Technology for European Advancement) project, financed by the Italian Public Authorities. We thank Prof. Stefano Ceri, Doc. Michele Tringali, Lara Gadda, Maria Rodriguez and Irene Celino for their precious comments and support.

References

1. OASIS: The uddi technical white paper. Technical report, OASIS (2004)
2. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. Technical report, W3C (2004)
3. Benatallah, B., Hacid, M.S., Rey, C., Toumani, F.: Request rewriting-based web service discovery. In: International Semantic Web Conference. (2003) 242–257
4. Trastour, D., Bartolini, C., Gonzalez-Castillo, J.: A semantic web approach to service description for matchmaking of services. In: SWWS. (2001) 447–461
5. L. Li, I. Horrocks: A software framework for matchmaking based on semantic web technology. (2003)
6. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Semantic matching of web services capabilities. In: International Semantic Web Conference. (2002) 333–347
7. Oldham, N., Thomas, C., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework with machine learning classification. In: SWSWPC. (2004) 137–146
8. Kifer, K., Lara, R., Polleres, A., Zhao, C., Keller, U., Lausen, H., Fensel, D.: A logical framework for web service discovery. In: Semantic Web Services Worshop at ISWC. (2004)
9. Keller, U., Lara, R., Lausen, H., Polleres, A., Fensel, D.: Automatic location of services. In: ESWC. (2005) 1–16
10. Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Computer **25** (1992) 38–49
11. Roman, D., Lausen, H., Keller, U.: D2 web service modeling ontology (WSMO). Technical report (2005)
12. Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., Fensel, D.: D5.1 WSMO Web Service Discovery. Technical report (2004)
13. Keller, U., Lara, R., Lausen, H., Polleres, A., Predoiu, L., Toma, I.: D5.2 WSMO Discovery Engine. Technical report (2004)
14. Keller, U., Lara, R., Lausen, H., Polleres, A., Predoiu, L., Toma, I.: D10 WSMX Discovery. Technical report (2005)
15. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. J. ACM **42** (1995) 741–843
16. Preist, C.: A conceptual architecture for semantic web services. In: International Semantic Web Conference. (2004) 395–409
17. Cimpian, E., Moran, M., Oren, E., Vitvar, T., Zaremba, M.: D13.0 overview and scope of WSMX. Technical report (2005)
18. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: Owl-. Technical report, WSML (2004)
19. Angele, J., Lausen, G.: Ontologies in f-logic. In: Handbook on Ontologies. (2004) 29–50
20. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: D20.3 OWL flight. Technical report, WSML (2004)
21. Pan, J. Z., Horrocks, I.: OWL-E: Extending owl with expressive datatype expressions. Technical report, IMG/2004/KR-SW-01/v1.0, Victoria University of Manchester (2004)
22. K. Verma, K. Sivashanmugam, A. Sheth, A. Patil: METEOR-S WSDI: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. (2003)