Bridging the Gap Between Service Description Models in Service Matchmaking[1]

Alberto Fernández, Zijie Cong, Analay Baltá

CETINIA, University Rey Juan Carlos, Móstoles, Spain
alberto.fernandez@urjc.es, zijie@ia.urjc.es, analay@ia.urjc.es

Abstract. In service-oriented architectures (SOA) the process of service discovery involves the matchmaking between service advertisements and service requests. Most approaches assume that both service advertisements and service requests are expressed using the same description language as well as domain ontologies. However, in large-scale open environments that assumption does not hold. In this paper we present a method that addresses the semantic mismatches at service description level. In particular, we focus on the alignment of service description models and service matchmaking on the unified request and advertisements. We propose a matchmaking method that encompasses different semantic, syntactic and hybrid service description languages.

Keywords: service discovery, matchmaking, semantic web services, service oriented architecture, agreement technologies, semantic alignment.

1 Introduction

In multi-agent systems, agents communicate with the aim of achieving their objectives. An individual agent may require a service to be performed by another entity. There are several stages since an agent identifies a given need until the service that provides it is eventually executed. First, the agent identifies some functionality that it is not able to perform or that might be executed more efficiently by an external entity. Then, candidate service providers must be located. Once a set of potential providers is known, the agent must choose one among them. This decision can be made based on several factors such as quality of service, price, reputation, etc. After the selection is made the two agents might engage in a negotiation about the conditions under which the service is going to be performed.

In this paper we concentrate on the phase of provider location. Distributed service directories and efficient decentralised matching techniques are essential for dynamic and scalable Web service discovery.

[1] Work partially supported by the Spanish Ministry of Science and Innovation through grants TIN2009-13839-C03-02 and CSD2007-0022 (CONSOLIDER-INGENIO 2010)

The process of discovering and interacting with a Semantic Web Service includes [4] *candidate service discovery* (match advertised service descriptions against specifications from requesters), *service engagement*, and *service enactment*.

Several description frameworks to annotate provided services on the one hand and express service requests on the other have been proposed. They range from logic-based complex and expressive semantic service descriptions (e.g. OWLS [25], WSMO [3]) to syntactical ones (WSDL [5], keywords, tag clouds), with some approaches in between (SAWSDL [9]). In this context, several frameworks to semantically match service advertisements and requests have been presented in the literature [11, 14, 23, 28].

In open environments the mechanisms for locating appropriate services have to struggle with the additional problem of semantic mismatches among the *service description models or languages* as well as *domain ontologies* used to specify the concepts in the descriptions. Note that most approaches assume the use of the same language for both service advertisements and requests. Therefore, semantic alignment mechanisms need to be purposefully integrated into the service discovery process.

The contributions of this paper are: i) an architecture for service discovery where semantic alignment mechanisms are integrated into, ii) an alignment technique for different service description models, and iii) a service matchmaking method that uses the aligned services.

The rest of the paper is organized as follows. In the next section we present an abstract architecture for semantic service discovery, which pays especial attention to semantic alignments of service descriptions. In section 3, we propose a service description model alignment including an integrated model which contains all the significant characteristics of the most important existing models. We propose a service matching framework in section 4, which includes two complementary methods. Finally, we review related works in section 5 and present some conclusions and future work.

2 Service Discovery Architecture

Figure 1 illustrates the proposed architecture that defines the service discovery functionality. The architecture comprises the building components to match a service request against a service advertisement. In particular, this proposal pays special attention to the problem of semantic mismatches between descriptions. Semantic mismatches are considered at two different levels:

- *Service description models.* Services (advertisements and requests) might be described using different languages or models (e.g. OWL-S, WSMO, SAWSDL, ...).
- *Domain ontology concepts.* Since semantic service descriptions rely on the use of domain ontologies, the second type of mismatch is due to the use of different domain ontologies to specify the concepts used in the descriptions.

Note that both options can be combined. For instance, two services might share the same service model (e.g. OWL-S) but use different domain ontologies, or they might use the same domain ontology but different service models.

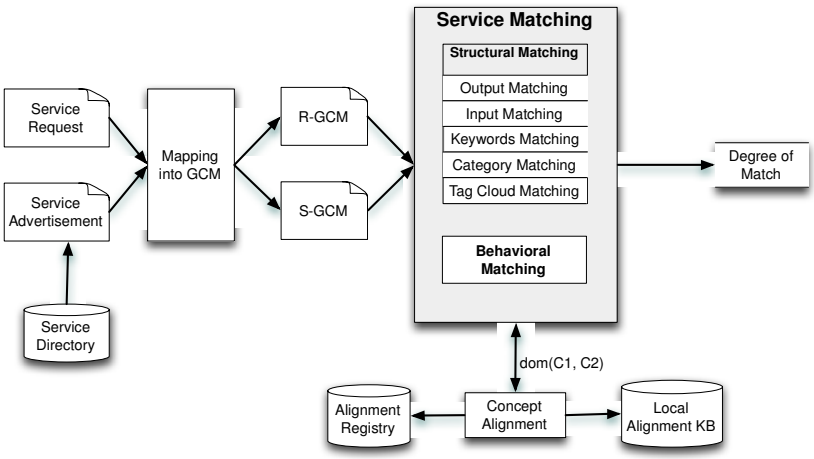

Fig. 1 Service Discovery Architecture

In the first step, we describe the mapping from the original models into a general common model (*GCM*) for service discovery purposes that integrates the relevant characteristics of service description languages. Therefore, service model alignment consists of mapping both descriptions (request and advertisement) into the GCM. Note that service description approaches not only differ in the language in which they are written. They are classified at different levels of expressiveness, ranging from complex, formal, logic-based semantic descriptions to lightweight syntactical ones.

The *Service Matching* module takes those two descriptions and returns the degree of match between them. As we will see in section 4, service matchmaking may include syntactic and semantic matching of description fragments (depending on the source descriptions). Semantic matchmaking algorithms usually include a semantic concept matching process to analyze the (similarity) relation between concepts used in advertisements and requests. In particular, we propose two complementary approaches: a *structural matching* focused on the structure of descriptions and a second one that focuses on the behavior of services. As it was pointed out above, those concepts might belong to different ontologies so a *Semantic Concept Alignment* is carried out to solve this problem. In this case, we keep a local knowledge base of alignments and assume the existence of an external registry of alignments. The *local knowledge base* acts as a cache of alignments used in previous matching. The *external alignment registry* is consulted to avoid carrying out a process of ontology matching if there is an alignment published by third parties.

In the next sections we detail the building blocks of the proposed architecture.

3 Service Description Model Alignment

As commented above, service model alignment consists of defining a common model for two different ones (the request and the candidate services), and mapping descriptions expressed in those source models into the common model. This transformation may produce a loss of expressiveness in at least one of the original descriptions, especially if they use models with different expressiveness power. Here, we do not aim at the design of a unified model with the intersection of all the existing ones, which would probably lead us to the definition of a very simple (lightweight) model to account for all the different source models. However, we envision the definition of mappings between pairs of them, thus keeping that particular common model as close to the original ones as possible. Besides, this approach is more modular and flexible to consider new models. Model-to-Model alignments consist of three steps:

1. *Conceptual analysis* of characteristics finding mappings between models. Note that this task can be done focusing on service matchmaking, i.e. only the aspects considered by the matching techniques have to be mapped.
2. Definition of a *common model language* (CML), which might be one of the originals. The use of standard languages is encouraged here.
3. Implementation of a tool for *automatic transformation* of service descriptions from the original models to the CML.

In the next subsection we describe the characteristics of most popular service description approaches, so as to propose a model, which encompasses the different common models. Afterwards, we present a unified common model for matchmaking that integrates the specific CMLs as well as mappings from the different service descriptions models.

3.1 Service description approaches

In this section we describe different approaches to service description. We include semantic models (OWL-S, WSMO), syntactic models (WSDL), hybrid (SAWSDL), as well as other lighter approaches (*keyword*-, *cloud*-, and *text*-based service descriptions). For each approach, we present a brief description and a formal specification of the selected characteristics from a matchmaking point of view.
Semantic service descriptions include concepts defined in some domain ontologies. In the following, we consider such domain ontology Q, and $\mathcal{N}$ represents the set of concept names. Q can be considered as the union of all domain ontologies used by the different service descriptions.

OWL-S. Web Ontology Language for Services (OWL-S) [25] is an OWL ontology characterized by three modules: Service Profile, Process Model and Grounding. The *service profile* is used to describe what the service does; it takes a global view of the service independently of how this function is realized by the service. The *process model* is used to describe how the service is used; and the *grounding* is used to describe how to interact with the service. The service profile and process model are

thought of as abstract characterizations of a service, whereas the grounding makes it possible to interact with a service by providing the necessary concrete details related to message format, transport protocol, and so on. The service profile is crucial in the web service discovery process since it describes the capabilities of web services.

Formally, we define a service S, described in OWL-S by $S = <\mathcal{I}, \mathcal{O}, \mathcal{P}, \mathcal{E}, \mathcal{C}, \mathcal{T}>$ where:

- $\mathcal{I} \subseteq \mathcal{N}$ represents a set of inputs (defined by the property *hasInput*), where each input $\mathcal{I}_k \in \mathcal{I}$ is a concept defined in a domain ontology ($\mathcal{I}_k \in \mathcal{N}$).
- $\mathcal{O} \subseteq \mathcal{N}$ represents a set of outputs (defined by the property *hasOutput*), where each output $\mathcal{O}_k$ is a concept of an ontology ($\mathcal{O}_k \in \mathcal{N}$).
- $\mathcal{P} \subseteq \mathcal{N}$ represents a set of preconditions (defined by the property *hasPrecondition*), where each precondition $\mathcal{P}_k$ is a concept of an ontology ($\mathcal{P}_k \in \mathcal{N}$). When there are more than one precondition they must be interpreted as logical conjunctions expressing conditions over the state of the world, e.g. $\mathcal{P} = \{p_1, p_2\} \equiv p_1 \sqcap p_2$.
- $\mathcal{E} \subseteq \mathcal{N}$ represents a set of effects (defined by the property *hasResult*), where each effect $\mathcal{E}_k$ is a concept of an ontology ($\mathcal{E}_k \in \mathcal{N}$). Analogously, in case of several effects, all of them must be fulfilled.
- $\mathcal{C} \subseteq \mathcal{N}$ represents a set of categories (defined by the property *hasCategory*), where each category $\mathcal{C}_k$ is a concept of an ontology ($\mathcal{C}_k \in \mathcal{N}$) (e.g. NAICS [27] or UNSPSC [40]).
- $\mathcal{T}$: string is a plain text description of the Web service (defined by the property *textDescription*).

WSMO. Web Service Modeling Ontology (WSMO) [**¡Error! No se encuentra el origen de la referencia.**] offers four key components to model different aspects of Semantic Web Services in WSML: *ontologies*, *goals*, *services*, and *mediators*. Besides these main elements, non-functional properties such as accuracy, network related QoS, performance, scalability, and reliability are used in the definition of WSMO elements that can be used by all its modelling elements. *Ontologies* provide the terminology used by other elements to describe the relevant aspects of the domains of discourse. *Goals* state the intentions that should be solved by web services and are representations of one or more objectives which need to be fulfilled. WSMO *web service* descriptions consist of functional, non-functional and the behavioural aspects necessary for web service discovery. Web Service descriptions are defined into WSMO capability by their *precondition, postcondition, assumption, effect*, and their *nonFunctionalProperties* (*title, subject, description*). Finally, *mediators* resolve interoperability problems and describe elements to overcome incompatibility problems between different elements on data, process and protocol level.

For our service discovery approach, WSMO and OWL-S share the same formalization, i.e., $S = <\mathcal{I}, \mathcal{O}, \mathcal{P}, \mathcal{E}, \mathcal{C}, \mathcal{T}>$, although these components are obtained in WSMO from different fields. In particular: $\mathcal{I}$ (*precondition*), $\mathcal{O}$ (*postcondition*), $\mathcal{P}$ (*assumption*), $\mathcal{E}$ *(effect*), $\mathcal{C}$ (*subject*), $\mathcal{T}$(*description*).

WSDL. Web Service Description Language (WSDL) [5] is an XML-based language for syntactically describing the service, including the service name, functions, and input and output parameters. WSDL is most commonly used with SOAP and XML to make these services available over the Internet. This allows the reuse of abstract definitions: *messages*, which are abstract descriptions of the data being exchanged, and *port types* which are abstract collections of *operations*. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable *binding*, and a collection of *ports* define a *service*. Frequently, WSDL describes the public interface to the web service in UDDI (Universal Description, Discovery and Integration) [39], which provides a web wide registry of web services. Currently web services are described using WSDL descriptions which provide operational information and it is limited in its search services by its inability to extend beyond the keyword-based matches.

Formally, we define a WSDL service $S = <\mathcal{I}, \mathcal{O}, \mathcal{T}>$ where:

- $\mathcal{I} \subseteq \Sigma^*$, where Σ={a, …, z}, represents a set of inputs (strings extracted from the *wsdl:input* element defined in the *messages* of operations).
- $\mathcal{O} \subseteq \Sigma^*$, where Σ={a, …, z}, represents the set of outputs (*wsdl:output*).
- $\mathcal{T}$ is a string containing the Web service documentation (*wsdl:documentation*).

SAWSDL. Semantic Annotations for WSDL and XML Schema (SAWSDL) [9] introduces three new extension attributes for using in WSDL and XML Schema documents, and discusses some of their possible uses. The extension attribute *modelReference* is used to specify the association between a WSDL or XML Schema component and a concept in some semantic model. It is used to annotate XML Schema type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults. The schema mapping attributes, *liftingSchemaMapping* and *loweringSchemaMapping*, are intended for specifying mappings between semantic data and XML. SAWSDL allows service discovery via a direct annotation of the *types* (simple or complex) and *elements* that express the content of inputs and outputs of WSDL operations. The addition of these attributes requires no other changes to existing WSDL or XML Schema documents, or the manner in which they had been used previously. Note that it is possible that some of the elements are not semantically annotated.

Formally, we define a service, S, described in SAWSDL by $S = <\mathcal{I}, \mathcal{O}, \mathcal{T}>$ where:

- $\mathcal{I}=<\mathcal{I}_{syn}, \mathcal{I}_{sem}>$ is a pair of sets containing the inputs not semantically annotated ($\mathcal{I}_{syn} \subseteq \Sigma^*$, Σ={a, …, z}) and those annotated with concepts from ontologies ($\mathcal{I}_{sem} \subseteq \mathcal{N}$).
- $\mathcal{O}=<\mathcal{O}_{syn}, \mathcal{O}_{sem}>$ represents the set of annotated ($\mathcal{O}_{sem}$) and not annotated ($\mathcal{O}_{syn}$) outputs
- $\mathcal{T}$, of type string, is the Web service documentation.

Keyword-based descriptions. A typical way of describing resources is by annotating them using keywords (or tags). Service descriptions are also possible following this approach. For example, *seekda!*[2] web service search engine includes free text and tag descriptions provided by service users, as well as WSDL files. Although keywords are usually considered free text, they may be defined using concepts from ontologies (usually taxonomies). Thus, we admit two kinds of keyword-based descriptions (K): syntactic (free text) and semantic (ontology-based).

Formally, $\mathcal{K} = <\mathcal{K}_{syn}, \mathcal{K}_{sem}>$, where:

- $\mathcal{K}_{syn} \subseteq \Sigma^*$, with Σ={a, …, z}, is a set of syntactic keywords
- $\mathcal{K}_{sem} \subseteq N$, is a set of semantic keywords

Note that, depending on the service description annotation framework, $\mathcal{K}$ may only include syntactic or semantic keywords. For example, for services annotated in *seekda!*, which only include free text keywords, $\mathcal{K}_{sem} = \phi$.

Tag Clouds based descriptions. Tags are short informal descriptions, often one or two words long, used by Web users to describe online resources. The advantage of using tags in the context of service discovery is that they supply a user-defined vocabulary based on a consensus of how the service is perceived or used in the world [10, 30]. Unlike simple keyword search, we envision tag clouds associated with services themselves as semantic descriptions carrying collaborative knowledge about the service.

A tag-cloud-based service description is a set of pairs $\mathcal{TC} = \{<t, n> | t \in \Sigma^*, \Sigma$={a, …, z}, $n \in N\}$, where t is a free text tag and n is the frequency of the tag t in the cloud.

Tag clouds are used in faceted browsing and often displayed graphically there, emphasizing the weight n by the font size of a tag. The frequency n, which denotes the weight of a particular tag, might be obtained from explicitly annotations of the services or, in frameworks where users can annotate their perception of service, by counting the number of users that included the tag t in their annotation of that service.

Textual descriptions. Services can be defined by free text human-readable information. For these kinds of descriptions service discovery is based on information retrieval techniques (IR).

3.2 Common Model for Service Discovery

In order to establish the relationships between the different service description models, we set out from existing conceptual comparisons between OWL-S and WSMO [21, 33], OWL-S, SAWSDL, WSDL [19, 29] and simple descriptions such keywords and free text to obtain a general model description of services that facilitates their discovery.

From the analysis of the different approaches to service descriptions (section 3.1) we define the *GCM* with the following elements: *inputs, outputs, preconditions, effects, keywords, textual description, category* and *tag cloud*.

[2] http://webservices.seekda.com/

Definition 1. Let $\mathcal{N}$ be a set of concepts of domain ontologies, a *general common model* (*GCM*) for service discovery is a tuple $<\mathcal{I}_{GCM}, \mathcal{O}_{GCM}, \mathcal{P}_{GCM}, \mathcal{E}_{GCM}, \mathcal{K}_{GCM}, \mathcal{C}_{GCM}, \mathcal{T}_{GCM}, \mathcal{TC}_{GCM}>$, where:

- $\mathcal{I}_{GCM} = <I_{syn}, I_{sem}>$ is the set of syntactic ($I_{syn} \in \{a, .., z\}^*$) and semantic ($I_{sem} \subseteq \mathcal{N}$) inputs of the service.
- $\mathcal{O}_{GCM} = <O_{syn}, O_{sem}>$ is the set of syntactic ($O_{syn} \in \{a, ..., z\}^*$) and semantic ($O_{sem} \subseteq \mathcal{N}$) outputs.
- $\mathcal{P}_{GCM}$ is the set of preconditions. $\mathcal{P}_{GCM} \subseteq \mathcal{N}$
- $\mathcal{E}_{GCM}$ is the set of effects. $\mathcal{E}_{GCM} \subseteq \mathcal{N}$
- $\mathcal{K}_{GCM} = <\mathcal{K}_{syn}, \mathcal{K}_{sem}>$ is the sets of syntactic and semantic keywords, where $\mathcal{K}_{syn} \subseteq \{a, \ldots, z\}^*$, $\mathcal{K}_{sem} \subseteq \mathcal{N}$.
- $\mathcal{C}_{GCM}$ is a set of categories of the service, described semantically ($C_{sem} \subseteq \mathcal{N}$) (e.g. NAICS or UNSPSC).
- $\mathcal{T}_{GCM}$ is a textual description of the service.
- $\mathcal{TC}_{GCM}$ is a tag cloud. $\mathcal{TC}_{GCM} = \{<t, n> \mid t \in \{a, \ldots, z\}^*, n \in \mathbb{N}\}$.

We opt for using RDF as the representation language for the *GCM*. RDF is a W3C recommendation for representing data on the Web. A lot of RDF contents and tools to process them are available. Although RDF is less expressive than OWL-S, WSMO and SAWSDL, it is enough for representing the information needed for semantic service search. Furthermore, the use of RDF may also allow the exploitation of SPARQL [41] to query service descriptions.

3.3 Mapping service descriptions into the GCM

Table 1 shows how the different elements of the *GCM* can be obtained from each source service description model. The first column specifies the element of the *GCM*, while each cell contains the value mapped from the model shown in the first row.

There are many straightforward mappings that consist of simple associations between parameters in both models. For instance, in OWLS/WSMO $\mathcal{I}_{GCM} = < \emptyset$, $pt(\mathcal{I})>$ because they only provide semantically described inputs $\mathcal{I}$ ($\mathcal{I}_{sem}$), where $pt(\mathcal{I}) = \{ t \mid t = parameterType(i)\ \forall i \in \mathcal{I}\}$. The contrary applies to WSDL, where only the syntactic values are filled ($\mathcal{I}_{GCM} = <\mathcal{I}, \emptyset>$). However, SAWSDL may contain both syntactic and semantic descriptions explicitly, thus $\mathcal{I}_{GCM} = <I_{syn}, I_{sem}>$. The same is applied to the outputs. Trivial mappings apply to preconditions ($\mathcal{P}_{GCM}$), effects ($\mathcal{E}_{GCM}$), categories ($\mathcal{C}_{GCM}$) and textual descriptions ($\mathcal{T}_{GCM}$).

However, some fields (tag-clouds, keywords) may not be explicitly described by a given model but they can be obtained from the rest of the description. To reduce the possibility of loss of expressiveness when matching two services described in models of different expressiveness level, these fields are discovered and filled automatically using information from other available fields. For example, **Tag-clouds** can be calculated from textual descriptions by means of a function $\Delta(\mathcal{T})$, which returns the *k most relevant* words from the text $\mathcal{T}$ as well as their frequency. We adopt information retrieval (IR) techniques to obtain that information through a process of (i) word

extraction using TF-IDF or quadgram-based methods [35], (ii) stemming, and (iii) filtering out non-relevant terms (chosen heuristically). In addition, The set of input concept names $\mathcal{N}(\mathcal{I})$ and output concept names $\mathcal{N}(\mathcal{O})$ in semantic descriptions (OWL-S, WSMO, SAWSDL) are considered for the cloud with non-character symbols removed and converted to lowercase. The case of keyword-based service descriptions (where no text is included), a plain cloud is created with frequency 1 for every keyword in the description.

Syntactic **keywords** can be easily obtained from tag clouds (either original or calculated with Δ), by simply adopting the k most relevant words (function $\tau(TC)$, being TC a tag-cloud). The set of input and output concept names as well as their parameter types ($pt(\mathcal{I})$ and $pt(\mathcal{O})$) are also adopted as syntactic and semantic keywords, respectively. It should be noticed that the syntactic keywords and semantic keywords are two disjoint sets, essentially, "semantic keywords" is a set of ontological concepts while syntactic keywords carries no semantic information. We treated these two components differently in the matchmaking process which can be seen in Sect.4.

The complexity of the mapping depends largely on the original description model. Semantic fields are extracted directly from the original description, no additional processing is required. On the other hand, syntactic information often needs to be pre-processed, e.g. stemming and filtering out non-relevant terms, due to the nature that syntactic information in service descriptions are usually short, the time required for pre-processing the syntactic information is considered negligible for both human users and software agents.

Table 1. Service(S)-to-*GCM* mapping

GCM	**OWL-S / WSMO**	**SAWSDL**	**WSDL**	**Keyword (tag)**	**Tag Cloud**	**Text**
$\mathcal{I}_{GCM}$	$<\emptyset, pt(\mathcal{I})>$	$<\mathcal{I}_{syn}, \mathcal{I}_{sem}>$	$<\mathcal{I}, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$
$\mathcal{O}_{GCM}$	$<\emptyset, pt(\mathcal{O})>$	$<\mathcal{O}_{syn}, \mathcal{O}_{sem}>$	$<\mathcal{O}, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$
$\mathcal{P}_{GCM}$	$\mathcal{P}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{E}_{GCM}$	$\mathcal{E}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{C}_{GCM}$	$\mathcal{C}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{T}_{GCM}$	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{T}$	$\emptyset$	$\emptyset$	S
$\mathcal{TC}_{GCM}$	$\Delta(\mathcal{T}) \cup \mathcal{N}(\mathcal{I}) \cup \mathcal{N}(\mathcal{O})$	$\Delta(\mathcal{T}) \cup \mathcal{I}_{\dagger\setminus} \cup \mathcal{O}_{\dagger\setminus}$	$\Delta(\mathcal{T}) \cup \mathcal{I} \cup \mathcal{O}$	$\{<t, 1> \mid t \in \mathcal{K}_{syn}\}$	S	$\Delta(S)$
$\mathcal{K}_{GCM}$	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{N}(\mathcal{I}) \cup \mathcal{N}(\mathcal{O}), pt(\mathcal{I}) \cup pt(\mathcal{O})>$	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{I}_{syn} \cup \mathcal{O}_{syn}, \mathcal{N}(\mathcal{I}_{sem}) \cup \mathcal{N}(\mathcal{O}_{sem})>$	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{I}_{syn} \cup \mathcal{O}_{syn}, \phi>$	$\mathcal{K}$	$<\tau(S), \emptyset>$	$<\tau(\Delta(S)), \emptyset>$

Figure 2 sumarises the characteristics of the *GCM* that can be obtained from each original service description model.

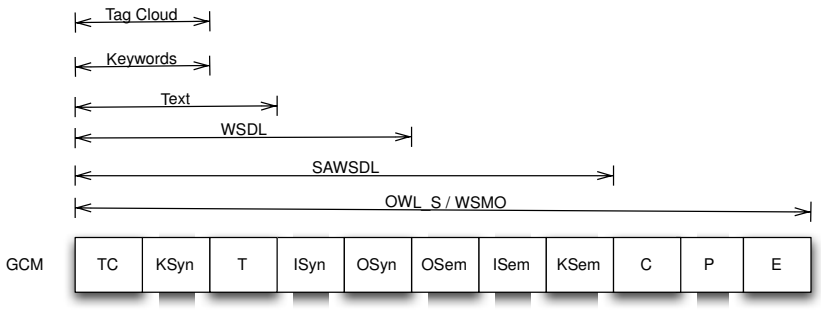

Fig. 2 Mapping to GCM

3.4 Examples

In this section we show our proposal with an example of a novel reservation service[3], a service that returns information of person who has booked a given novel. We present an OWL-S and a SAWSDL representation of that service, as well as their mapping to the *GCM*.

3.4.1 OWL-S

Figure 3 shows a partial view of the OWL-S profile for that service. In particular, it shows the definition of the input *_NOVEL* and output *_PERSON*. Thus, we can see that the semantic input/output concepts are obtained from the value of the *parameterType* property (in this case *books*[4]*:Novel* and *books:Person*) of OWL-S input *_NOVEL* and output *_PERSON* in the service.

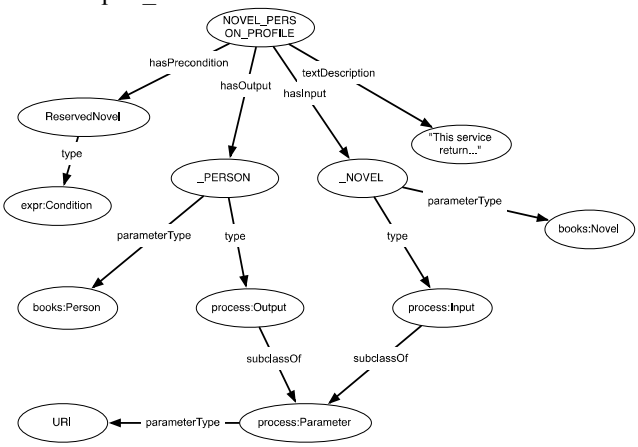

Fig. 3 OWL-S description

[3] From OWL-S test collection version 4 (for OWL-S) and SAWSDL test collection version 3 (for SAWSDL) http://www.semwebcentral.org/projects/owls-tc/ or /sawsdl-tc/

[4] books: http://(OWL-TC_URI)/ontology/books.owl

We define the OWL-S novel reservation example service $S = <\mathcal{I}, \mathcal{O}, \mathcal{P}, \mathcal{E}, \mathcal{C}, \mathcal{T}>$ as follows:

$\mathcal{I}$ = {npr[5]:_NOVEL}

$\mathcal{O}$ = {npr:_PERSON}

$\mathcal{P}$ = {npr:ReservedNovel}

$\mathcal{E} = \emptyset$

$\mathcal{C} = \emptyset$

$\mathcal{T}$ = "*This service returns information of person who has booked the given novel.*"

From the OWL-S service description we can obtain the service mapping described in the *GCM* according to Table 1. There are no syntactic inputs/outputs, and the semantic inputs would be the semantic concepts (parameter types) of inputs $\mathcal{I}$. The tag cloud includes the most relevant words of the text $\mathcal{T}$ as well as their frequency. The syntactic keywords are obtained using tag clouds with frequency 1 as well as the inputs and outputs concept names. As for the semantic keywords the semantic concepts of inputs and outputs (parameter types of inputs $\mathcal{I}$ and outputs $\mathcal{O}$) are taken. Therefore:

$S_{GCM} = <\mathcal{I}_{GCM}, \mathcal{O}_{GCM}, \mathcal{P}_{GCM}, \mathcal{E}_{GCM}, \mathcal{K}_{GCM}, \mathcal{C}_{GCM}, \mathcal{T}_{GCM}, \mathcal{TC}_{GCM}>$ where:

$\mathcal{I}_{GCM}$ = <{∅},{books:Novel}>

$\mathcal{O}_{GCM}$ = <{∅},{books:Person}>

$\mathcal{P}_{GCM}$ = npr:ReservedNovel

$\mathcal{E}_{GCM} = \emptyset$

$\mathcal{K}_{GCM}$ = <{novel, person, book, reserved},{books:Novel, books:Person}>

$\mathcal{C}_{GCM} = \emptyset$

$\mathcal{T}_{GCM}$ = "*This service returns information of person who has booked the given novel*"

$\mathcal{TC}_{GCM}$ = {<novel,2>, <person,2>, <reserve,1>, <book, 1>}.

3.4.2 SAWSDL

We define a novel reservation service[6] in SAWSDL as $S=<\mathcal{I},\mathcal{O},\mathcal{T}>$, where:

$\mathcal{I}$ = <{novel},{books:Novel}>

$\mathcal{O}$ = <{person, date},{books:Person, support[7]: Calendar-Date}>

$\mathcal{T}$ = "*This service returns information of person who has booked the given novel*"

Note that we consider semantic annotation for every input and output. In this case, we use *books:Novel*, and *books:Person, support:Calendar-Date* to annotate inputs and outputs related to airports and dates, respectively.

---

[5] npr: http://(OWL-TC URI)/services/1.1/novel_person__Reserverservice.owls

[6] This service is adapted from SAWSDL TC Ver.3.0 novelperson_reservationservice.wsdl

[7] support: http://(OWL-TC_URI)/ontology/support.owl

From the SAWSDL service description we obtain the following service mapping described in the *GCM*.

S_{GCM} = <$\mathcal{I}_{GCM}$, $\mathcal{O}_{GCM}$, $\mathcal{P}_{GCM}$, $\mathcal{E}_{GCM}$, $\mathcal{K}_{GCM}$, $\mathcal{C}_{GCM}$, $\mathcal{T}_{GCM}$, $\mathcal{TC}_{GCM}$ > where:

$\mathcal{I}_{GCM}$= `<{novel},{books:Novel}>`

$\mathcal{O}_{GCM}$= `<{person, date},{books:Person, support: Calendar-Date}>`

$\mathcal{P}_{GCM}$ = ∅

$\mathcal{E}_{GCM}$ = ∅

$\mathcal{K}_{GCM}$ = `<{novel, person, book, reserved},{books:Novel, books:Person , support: Calendar-Date}>`

$\mathcal{C}_{GCM}$ = ∅

$\mathcal{T}_{GCM}$ = "`This service returns information of person who has booked the given novel and the date of reservation.`"

$\mathcal{TC}_{GCM}$ = `{<novel,2>, <person,2>, <reserve,1>, <date,3>, <book,1>}`

4 Service Matchmaking

In this section, we first present a unified method for matching both semantic and syntactical similarities. Then we introduce two service-matching approaches: the first approach presented in subsection 4.2 concentrates on the services' structural level, which is fundamental for the successful invocation and execution of services; the second approach is complementary, it tackles the "I/O" pitfall problem and reduces the loss of expressiveness during matching.

4.1 Concept Matching

Since service descriptions are composed by several elements or fields, each of them is defined by concept descriptions (syntactical or semantic), the process of matchmaking includes a concept matching task. This function takes two concepts, C_A (advertisement) and C_R (request), and returns the degree of match between them.

For semantic concepts, many of the current proposals for defining the degree of match between service advertisements and requests are based on subsumption checking of concepts present in inputs and outputs of service descriptions. We adopt the four degrees of match proposed by Paolucci et al. [28]: *exact* ($C_A = C_R$), *plug-in* (C_R subsumes C_A), *subsumes* (C_A subsumes C_R) and *fail* (otherwise).

Several similarity (or distance) measures for concept matching have been proposed in the literature, although their application to the concrete domain of service matching is very limited. One of the most well known distance measures between concepts is the length of the shortest path between them in the taxonomy, proposed by Rada et al [34]. Other proposals further refine that approach ([22, 24, 42]).

We propose a combination of service matching and concept similarity. In [11] we describe how both approaches can be combined into a unified service selection framework which returns a numeric value that can be used for ranking services. The ranking function compares the level of match first (exact, plugin, subsumes, fail), and then the level is refined with the (numerical) similarity value. The degree of match between two concepts is defined as follows.

Definition 2. The degree of match between two concepts C_A (advertisement) and C_R (request) is given by:

$$conceptMatch(C_R, C_A) = \begin{cases} 1 & if \quad C_A = C_R \\ \frac{1}{2} + \frac{1}{2} sim(C_A, C_R) & if\ C_R\ subsumes\ C_A \\ \frac{1}{2} sim(C_A, C_R) & if\ C_A\ subsumes\ C_R \\ 0 & otherwise \end{cases}$$

Where concept similarity (*sim*) is calculated as [24]:

$$\mathrm{sim}(\mathrm{C}_1, \mathrm{C}_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if\ C_1 \neq C_2 \\ 1 & otherwise \end{cases}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of the shortest path length (l) between the two concepts and the depth (h) of the least common subsumer in the concept hierarchy, respectively.

This method can also be applied to syntactical values and category information by using lexical database, such as WordNet [26] and public classification taxonomies, like UNSPSC and NAICS, respectively. Most of these systems are hierarchically organized. Therefore Definition 2 and *sim* can be utilized.

4.2 Structural Matching

Since service descriptions consist of several components (inputs, outputs, ...), the similarity between services must be defined based on its individual elements (e.g. each of its inputs) and aggregation operators.

For service matching, given the *GCM* representation of the service advertisement (GCM_A) and the service request (GCM_R), we analyze the common elements defined in them. Only those non-empty fields belonging to the intersection of both ($GCM_A \cap GCM_R$) are considered here, as they define the common model to A and R (CM_{AR}).

For each pair of the common elements we propose a matching algorithm taking into account the semantic concept matching, being boolean (0, 1) in the case of syntactic values. Finally an aggregation function is applied to combine the results.

Definition 3. Given a service request GCM_R and a service advertisement GCM_A where, according to Definition 1, $GCM_R = \langle \mathcal{I}_R, \mathcal{O}_R, \mathcal{P}_R, \mathcal{E}_R, \mathcal{K}_R, \mathcal{C}_R, \mathcal{T}_R, \mathcal{TC}_R \rangle$ and $GCM_A = \langle \mathcal{I}_A, \mathcal{O}_A, \mathcal{P}_A, \mathcal{E}_A, \mathcal{K}_A, \mathcal{C}_A, \mathcal{T}_A, \mathcal{TC}_A \rangle$. Then, the *degree of match* between GCM_R and GCM_A is calculated as:

$$dom(GCM_R, GCM_A) = \mathrm{f}(IM(\mathcal{I}_R, \mathcal{I}_A), OM(\mathcal{O}_R, \mathcal{O}_A), KM(\mathcal{K}_R, \mathcal{K}_A), CM(\mathcal{C}_R, \mathcal{C}_A), TCM(\mathcal{TC}_R, \mathcal{TC}_A))$$

IM, OM, KM, CM and *TCM* are defined below.

As a service matching example, we will use the services described in sections 3.4.1 and 3.4.2 (S^1_{GCM} and S^2_{GCM}), acting the OWL-S example as the service advertisement ($GCM_A = S^1_{GCM}$) and the SAWSDL as the service request ($GCM_R = S^2_{GCM}$).

Output matching (*OM*).

In line with Paolucci's proposal [28], an **output** matches if and only if for each output of the request there is a matching output in the service description, i.e. the service provides all the outputs required.

Given $\mathcal{O}_R = <O^R_{syn}, O^R_{sem}>$ and $\mathcal{O}_A = <O^A_{syn}, O^A_{sem}>$, $OM(\mathcal{O}_R, \mathcal{O}_A)$ is calculated as a combination of the semantic match (*OSemM*) and the syntactic output match (*OSynM*). If O_R is empty, the result of matching is always exact since the service request does not require any outputs. It may occur that the sets are partially empty. In that case, we apply only the semantic match if both semantic descriptions (O^R_{sem} and O^A_{sem}) are available. Otherwise, only the syntactic match is calculated. The next equation gives the details.

$$\mathrm{OM}(\mathrm{O_R}, \mathrm{O_A}) = \begin{cases} 1 & if\, O_R = \varnothing \\ OSemM(O^R_{sem}, O^A_{sem}) & if\, O^R_{sem} \neq \varnothing \wedge O^A_{sem} \neq \varnothing \\ OSynM(O^R_{syn}, O^A_{syn}) & if\, (O^R_{sem} = \varnothing \vee O^A_{sem} = \varnothing) \wedge (O^R_{syn} \neq \varnothing \vee O^A_{syn} \neq \varnothing) \\ 0 & otherwise \end{cases}$$

The semantic match is obtained by taking, for each output in the request, the best match (section 4.1) against the ones in the advertisement. The worst case (minimum value) is then chosen to combine the best matches, i.e.:

$$\mathrm{OSemM}(\mathrm{O^R}_{sem}, \mathrm{O^A}_{sem}) = \min_{\mathrm{o^R} \in \mathrm{O^R}_{sem}} \max_{\mathrm{o^A} \in \mathrm{O^A}_{sem}} \{\mathrm{conceptMatch}(\mathrm{o^R}, \mathrm{o^A})\}$$

The syntactical match is obtained in a similar way. To apply function *conceptMatch* to syntactical information, a third-party lexical database with taxonomy, such as WordNet, is required. The syntactical outputs are first mapped to a concept in WordNet and then *conceptMatch* can be applied using the taxonomy provided by WordNet. Essentially, OSynM is a composed function:

$$OSynM(O^R_{syn}, O^A_{syn}) = OSemM(Map(O^R_{syn}), Map(O^A_{syn}))$$

where *Map* is a function that maps syntactic outputs to concepts in the external taxonomy.

Input matching (*IM*).

For input matching (*IM*), an analogous approach is followed, but with the order of request and advertisement reversed.

Precondition/Effect matching (*PM/EM*).

Since preconditions and effects can usually not be expressed in a taxonomy or ontological hierarchy, rather more complex formalisms (e.g. SWRL rules) are proposed to describe these. Several approaches have been proposed for service constraints matching (Preconditions/Effects) [15, 38, 16], most of these approaches are based on theta-subsumption checking between two sets of clauses, originally proposed in [38]. However, theta-subsumption is NP-complete in general, the efficiency and practicability is yet to be proven. On the other hand, constrains are rarely seen in practice among service descriptions [18].

For these reasons we consider preconditions and effects specified by concept definitions, and keep them as they were specified in the original model. Thus, the preconditions and effects will not participate in the structural matching. In next section, we propose an approach that exploits preconditions and effects to reveal the behavioural information of services.

Keyword matching (*KM*)

Given R = $\mathcal{K}_R$ = <K^R_{syn}, K^R_{sem}> and $\mathcal{A}$ = $\mathcal{K}_A$ = <K^A_{syn}, K^A_{sem}>, keyword matching is based on the syntactic and semantic keywords:

$KM(K_R, K_A) = \alpha * KMatch(K^R_{sem}, K^A_{sem}) + \beta * KMatch(K^R_{syn}, K^A_{syn})$, with $\alpha + \beta = 1$.

We adopt Ehrig [8] measure to compare sets of concepts:

$$KMatch(R, A) = \frac{\sum_{r \in R} \vec{r}}{\left|\sum_{r \in R} \vec{r}\right|} \cdot \frac{\sum_{a \in A} \vec{a}}{\left|\sum_{a \in A} \vec{a}\right|}$$

with $\vec{r} = (sim(r, r_1), sim(r, r_2), ..., sim(r, a_1), sim(r, a_2)...)$, $\vec{a}$ analogously. Where r_n and a_n denote the *n*-th keywords in service request and advertisement respectively.

Tag-cloud matching (*TCM*)

In the context of service matching, **tag-clouds** have a natural correspondence to the typical vector space model used in standard information retrieval. Using the vector-space model, each tag-cloud is represented as a vector in term space and each term in the vector is weighted according to the standard *TF-IDF* weighting scheme [36]. In the vector-space model, similarity is calculated using the cosine measure.

This is similar to the calculation of keyword matching defined in the previous section, if we consider keywords as a tag-cloud with frequency 1 applied to each element. Therefore, the tag-clouds matching is calculated as:

$$TCMatch(R, A) = \frac{\sum_{r \in R} \delta_r \vec{r}}{\left|\sum_{r \in R} \delta_r \vec{r}\right|} \cdot \frac{\sum_{a \in A} \delta_r \vec{a}}{\left|\sum_{a \in A} \delta_r \vec{r}\right|}$$

with $\vec{r} = (sim(r, r_1), sim(r, r_2), ..., sim(r, a_1), sim(r, a_2)...)$, $\vec{a}$ analogously. And δ is the frequency of the corresponding tag.

Category matching (*CM*)

For the **category** field, which may be described by a concept from ontology, [0...1] value or the result of the semantic concept similarity function is taken, respectively.

We require that for each category in the request *R* at least there is one matching category in *A*. Therefore, similar to output matching:

$$CM(C_R, C_A) = \min_{c^R \in C_R} \max_{c^A \in C_A} \{\text{conceptMatch}(c^R, c^A)\}$$

Aggregation Function

Finally, service matching must combine the similarity value for each of these fields, Definition 3 ($dom(GCM_R, GCM_A)$). Different options can be considered for *f:*

$$f(W_{IM} \cdot IM(I_R, I_A), W_{OM} \cdot OM(O_R, O_A), W_{PM} \cdot PM(P_R, P_A), W_{EM} \cdot EM(E_R, E_A),$$
$$W_{KM} \cdot KM(K_R, K_A), W_{CM} \cdot CM(C_R, CA), W_{TCM} \cdot TCM(TC_R, TC_A))$$

and

$$W_q = \alpha \cdot w(n_c), where$$
$$q \in \{IM, OM, PM, EM, KM, CM, TCM\},$$
$$c \in \{I, O, P, E, K, C, TC\}, and$$
$$\alpha = \frac{1}{\sum_c w(n_c)} \text{ where } n_c \text{ denotes the number of elements in component C}$$

One should notice that the textual information would not participate in the aggregation calculation as it is always transformed into tag-clouds. If we consider those fields as a conjunctive set (i.e. all are expected to be matched) then a triangular norm (e.g. the *minimum*) can be used.

For the moment, a more general approach is taken: a weighted sum of each similarity, where the weighting parameters are the contribution of the corresponding components of the GCM. The contribution of each component is calculated using a logistic function:

$$w(n_c) = \frac{1}{1 + e^{(1 - \frac{n_c}{0.5\overline{N}})}}$$

where n_c denotes the number of elements in component C, and $\overline{N}$ denotes the average number of non-empty elements in both service models.

Function *w* is a logistic function, which makes the weights of the components with number of elements close to the average increase rapidly. Also, logistic function prevents the over-influence caused by components with excessive number of elements. The graph of function *w* with $\overline{N}$=3 is shown in figure 4.

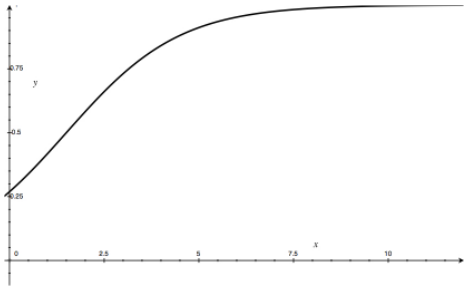

Fig. 4 Function w when $\overline{N}$=3

4.2.1 Structural Matching Example

In this section we demonstrate a running example of abovementioned matching algorithm. For simplicity's sake, we purposely eliminated some elements from the GCM models.

This example calculates the structural matching degree of two web services. For service advertisement, S_a, we use service from section 3.4.1, which is described in GCM as:

$\mathcal{I}_{GCM}$= <{∅},{books:Novel}>
$\mathcal{O}_{GCM}$= <{∅},{books:Person}>
$\mathcal{P}_{GCM}$ = ∅
$\mathcal{E}_{GCM}$ = ∅
$\mathcal{K}_{GCM}$ = <{novel, person}>
$\mathcal{C}_{GCM}$ = ∅
$\mathcal{T}_{GCM}$ = "*This service returns information of person who has booked the given novel*"
$\mathcal{TC}_{GCM}$ = ∅.

And for service request , we use another service from the publication field, S_r, described in GCM as:

$\mathcal{I}_{GCM}$= <{∅},{books:Book}>
$\mathcal{O}_{GCM}$= <{∅},{books:Price}>
$\mathcal{P}_{GCM}$ = ∅
$\mathcal{E}_{GCM}$ = ∅
$\mathcal{K}_{GCM}$ = <{book, price}
$\mathcal{C}_{GCM}$ = ∅
$\mathcal{T}_{GCM}$ = "*This service returns price of a given Book*"
$\mathcal{TC}_{GCM}$ = ∅.

The matchmaking start by calculating the semantic inputs using *ISemMatch(A, R),* in this case the result obtained is 0.140 based on the ontology showed in figure 5.

The output match returns a zero as concept *books:Price* and *books:Person* has no direct subsumption relation between them.

For the syntactic keywords, the similarity between the two sets of the keywords from service description is calculated using *simple path* similarity measure with WordNet as a lexical database, and the result obtained is 0.5667.

To aggregate these three degree of match (OSemMatch(A,R)=0, ISemMatch(A,R)=0.140 and KMatch(A,R)=0.5660), the weight of each component is calculated:

$$W_{OM} = \alpha \cdot w(n_O) = 0$$
$$W_{IM} = \alpha \cdot w(n_I) = 0.4140$$
$$W_{KM} = \alpha \cdot w(n_K) = 0.5859$$

The final degree of match of S_r and S_a is:

$$f(OM(A,R) \cdot W_{OM}, IM(A,R) \cdot W_{IM}, KM(A,R) \cdot W_{KM})) = 0.3895$$

4.3 Behavioral Matching

In this previous section we proposed several different algorithms for calculating the degree of match of each component in GCM independently, then used an aggregation function to combine the results. The computation of this approach is straightforward and computationally efficient.

However, some of the components are difficult to be used solely, such as preconditions and effects; some are limited under special circumstances such as the "I/O pitfall" problem stated in [14].

In our work [6], a graph-based matching algorithm for semantic web services is proposed. This approach intends to fully utilize all components in service description and reveal the behavioral information by extracting a subgraph from the ontology based on the service description.

When both service request and advertisement consist of semantic concepts, this approach can be used complementarily to the structural matching approach defined in the previous section.

4.3.1 Service Behavioral Graph

To exploit the behavioral information of a service, we consider a service as a mapping from its inputs to outputs. For semantic web services, inputs and outputs are instances of certain ontological concepts.

As ontology can be represented by a *multi-relational graph*, where each vertex denotes a concept and each edge denotes a relation between concepts, a service thus can be further considered as a sub-graph of ontology.

This sub-graph is a set of relations and concepts that connect inputs and outputs. These relations and concepts are referred as *critical elements*. Besides inputs and

outputs, other components in GCM such as preconditions, effects, tags and textual information may also contribute in identifying these critical elements.

For example, figure 5 illustrates a partial view of the ontology used by the second novel reservation service defined in SAWSDL. By using components of GCM, we identified the concepts *Novel*, *Person, Calendar-Date* and the object property *isReserved* as critical elements.

Then a path that connects input, *Novel*, to output, *Person*, and pass through relation *isReserved* is established, this path (strong elements in figure 5) is thus called the "Service Behavioral Graph".

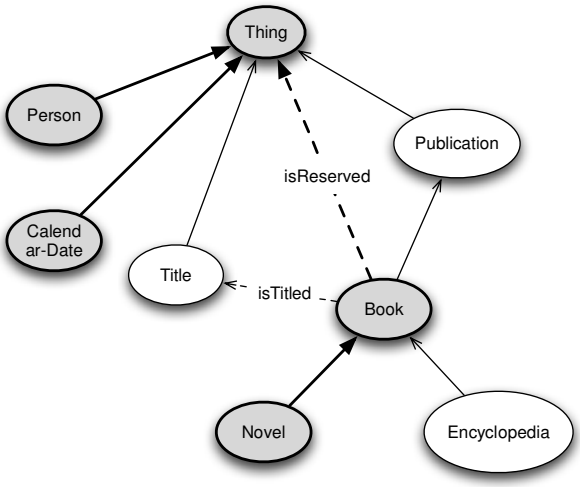

Fig. 5 Book Ontology (Solid edges depict *isA* relation)

4.3.2 Matching Algorithm

The degree of behavioral match of two services is calculated using the difference between path pairs in their service behavioral graphs. As a path is a sequence of concepts and relations, the difference can be considered as the edit distance (number of operations required to transform one of them into the other) between two paths.

Various distance metrics can be employed, such as Hamming distance, Levenshtein distance and Damerau-Levenshtein distance.

Function *dif* shows how the difference between two paths is calculated.

$$dif(P^A, P^B) = \frac{EditDist(P^A, P^B)}{P^A.len() + P^B.len()}$$

The final degree of match between services' behavioral graphs is the lowest degree of match of the best-matched path pairs.

This result can be used complementarily to the method described in the previous section.

This approach is limited by the quality of the ontologies used by services, therefore, one of our ongoing work is to take into account the quality of ontology during the calculation of services' behavioral similarity. One intuitive measurement of ontologies' quality is the ratio of number of concepts to number of relations.

4.4 Concept Alignment

In the previous section we saw that current service matchmaking algorithms are based on checking the relations between the concepts that appear in the different fields of semantic service descriptions. If the concepts being compared are defined in different ontologies then semantic alignments must be considered instead of obtaining a *fail* match.

An alignment (or mapping) between two ontologies *O* and *O'* can be described as a quadruple [7]: *<e, e', n, R>*

where:

- *e* and *e'* are the entities between which a relation is asserted by the mapping (e.g., formulas, terms, classes, individuals)
- *n* is a degree of trust (confidence) in that mapping
- *R* is the relation associated to a mapping, where *R* identifies the relation holding between *e* and *e'*.

In this work we are not concerned about ontology matching techniques, but on the use of alignments. Thus, we are interested in representing and querying mappings between ontologies. We propose using RDF as the language for expressing alignments, so that they can be published on the web and queried using SPARQL. In particular, we use the format of the Ontology Alignment Evaluation Initiative[8].

```
<rdf:RDF xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#">
….
   <ao:map>
     <ao:Cell>
      <ao:entity1 rdf:resource='http://www.example.com/ontology/books.owl#Novel'/>
      <ao:entity2 rdf:resource='http://dbpedia.org/resource/Novel'/>
      <ao:relation>=</ao:relation>
      <ao:measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</ao:measure>
     </ao:Cell>
    </ao:map>
</rdf>
```

Fig. 6 A segment of concept alignment representation

[8] http://oaei.ontologymatching.org/, An example of OAEI alignment can be found at http://alignapi.gforge.inria.fr/tutorial/tutorial1/results/equal.rdf

Fig. 6 shows an OAEI-like concept alignment segment represented in RDF/XML format, the "measure" of concept similarity could be obtained by human evaluation or other automated/semi-automated ontology alignment approaches (although it is out of the scope of this work). This repository then can be queried using SPARQL as shown in Fig. 7, where possible similar ("=") concepts are queried.

```
SELECT ?concept ?measure
WHERE {
  ?alignment ao:entity1 <http://www.example.com/ontology/books.owl#Novel>.
  ?alignment ao:entity2 ?concept.
  ?alignment ao:relation "=".
  ?alignment ao:measure ?measure
}
```

Fig. 7 An example of SPARQL of concept alignment

In case that two concepts involved in a service matchmaking process are contained in different ontologies, or the similarity value obtained using the function *sim(C1,C2)* is zero (FAIL), our system will query the concept alignment repository to attempt to obtain an existing similarity measure. The confidence measure obtained from the alignment is considered the *conceptMatch*(C_R,C_A) value between the concepts in *sameAs* relations ("="). In the case of subclass relations, note that we cannot calculate *sim* since we only know the relation but not the distance between C_1 and C_2. For the moment, the values 0.75 and 0.25 are considered for C_R subsuming C_A and vice versa, respectively. This is a matter of future research.

5 Related Work

Some (not many) other efforts have been made trying to align or compare different service description approaches. As we mentioned in section 3.2, we set out from existing conceptual comparisons between popular semantic web service languages [19, 21, 29, 33] to obtain a general model description of services that facilitates their discovery.

Giantsiou et al. [12] propose a service meta-model in which found services are transformed and represented in RDF. Their meta-model and discovery approach is influenced by light weight approaches (SAREST [37] and SAWSDL). In [17], Klush et al. proposed another meta-model called PIM4SWS which is similar to OWL-S model and focuses mainly on semantic information carried by web service description. Differently, we focus on both lightweight and semantic techniques, allowing other description models.

Most of the current approaches to Semantic Web Services matching, particularly those based on OWL-S, are based on subsumption reasoning on concepts included in the descriptions (e.g. [23, 28]). Klusch et. al [14] present a hybrid matchmaker that complements logic based reasoning with approximate matching techniques from Information Retrieval. In this sense we propose a hybrid approach, which combines subsumption checking, concepts similarity, and information retrieval. However, we focus on the integration of several different service description models.

The directory service using a common model (AT-GCM) in the same direction as iServe [31] uses the minimum service model to address interoperability, the difference is that our board to consider Tag-Cloud, and keywords free text for use in the directory.

Ambite et al introduced a system (DEIMOS) for constructing semantic web service from online sources automatically in [1]. DEIMOS uses an existing semantic web service as a seed, by calculating the syntactic similarity and a brute-force invocation-observation learning process, DEIMOS semantically annotated an external source. Differently to our approach they use only inputs/outputs to characterise services. Also, they use the Local-As-View (LAV) [20] datalog rules to describe the sources. We use RDF instead, although this does not reduce expressivity against LAV, in fact DEIMOS generates an RDF graph from LAV descriptions.

In addition, A. Heß introduced a web service classification approach using machine-learning techniques in [13]. Even though the evaluation showed a remarkable accuracy, no information about computational efficiency was shown. As techniques such as Naïve-Bayes and SVM could be noticeably computationally expensive, this approach might not be entirely suitable for service discovery in a large, open environment.

6 Conclusion

In this paper we have dealt with the problem of service discovery in open systems. We proposed an architecture that has semantic alignment as a first citizen component. In particular, we discussed in detail the alignment of service description models, and the transformation of them into a RDF common model. Although we provided with an alignment mechanism for a set of service description languages, other languages can be easily integrated into. In effect, if such new model fits into the proposed *GCM* only the adequate mappings have to be specified. Otherwise, new characteristics might be added to the *GCM* to account for those new languages, and considering them empty for the previous models (additionally those legacy models might be completed with the corresponding mappings to the new characteristics).

Regarding computational aspects, note that the mapping of service advertisements to the *GCM* can be done at registration time, so we only need to process the service request at run time (as well as the matchmaking algorithm).

We also proposed the combination of service matching and concept similarity into an integrated service-matching framework.

Domain ontologies can be specified using different languages. We address that aspect at the semantic concept matching implementation level, where we consider (for the moment) ontologies written in RDFS, OWL and WSML. At the service model mapping level we only need to copy the concept without any reasoning with the ontology.

The implementation and evaluation of the proposed framework is part of our ongoing work, including adapting an appropriate lexical database for syntactic information handling (for example, an customized WordNet dedicated to web services), automated category discovery and other model enrichment techniques to fill

the missing components from service descriptions. We plan to adapt, transform and extend existing test collection (e.g. the OWL-S TC, SAWSDL TC) as a starting benchmark for our experiments, for evaluation, existing and mature frameworks for semantic evaluation at large scale, such as SEALS[9] and SME2[10] are ideal choices for our future work.

7 References

1. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.: Automatically constructing semantic web services from online sources. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pages 17–32. Springer, Heidelberg, 2009.
2. Botelho, L., Fernandez, A., Fries, B., Klusch, M., Pereira, L., Santos, T., Pais, P., Vasirani, M.: Service Discovery. In M. Schumacher, H. Helin, H. Schuldt (Eds.) CASCOM - Intelligent Service Coordination in the Semantic Web. Birkha□user, Verlag. 2008.
3. Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., K¨onig-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., and Stollberg, M. Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005.
4. Burstein, M., Bussler, C., Zaremba, M., Finin, T., Huhns, M. N., Paolucci, M., Sheth, A. P., and Williams, S. A Semantic Web Services Architecture. *IEEE Internet Computing* vol. 9, 5, pages 72-81. 2005.
5. Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001
6. Cong, Z., Fernandez, A. Behavioral Matchmaking of Semantic Web Services. In Proceedings of the 4th International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2), Shanghai (China), CEUR Workshop Proceedings, vol. 667, pages 131-140, 11/2010.
7. David, J., Euzenat, J., Scharffe, F., Trojahn dos Santos, C.: The Alignment API 4.0, Semantic Web Journal 2(1), pages 3-10, 2011
8. Ehrig M. Ontology Alignment: Bridging the Semantic Gap. Springer. 2007.
9. Farrell, J. and Lausen, H. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C Recommendation 28 August 2007. http://www.w3.org/TR/sawsdl/
10. Fernandez, A., Hayes, C., Loutas, N., Peristeras, V., Polleres, A., and Tarabanis, K. Closing the service discovery gap by collaborative tagging and clustering techniques. In 2nd International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2), pages 115–128, Karlsruhe, 10/2008.
11. Fernandez, A., Polleres, A., and Ossowski, S. Towards Fine-grained Service Matchmaking by Using Concept Similarity, ISWC-2007 Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2). 2007
12. Giantsiou, L., Loutas, N., Peristeras, V. and Tarabanis, K. Semantic Service Search Engine (S3E): An Approach for Finding Services on the Web. In LNCS, Volume 5736/2009, pages 316-325. Springer, 2009.
13. Heß, A., Johnston, E., Kushmerick, N.: Machine Learning for Annotating Semantic Web Services. In: Semantic Web Services: Papers from the 2004 AAAI Spring Symposium Series. AAAI Press , 2004.

[9] www.seals-project.eu/

[10] http://semwebcentral.org/projects/sme2/

14. Klusch, M., Fries, B., and Sycara, K. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, 2, Pages 121-133, 4/2009,..
15. Kaufer, F. and Klusch, M. WSMO-MX: A hybrid Semantic Web service matchmaker. Web Intelligence and Agent Systems.vol. 7, pages 23-42, 2009.
16. Klusch, M. and Kapahnke, P. iSeM: Approximated reasoning for adaptive hybrid selection of semantic services. The Semantic Web: Research and Applications, pages 30-44, 2010.
17. Klusch, M.; Nesbigall, S.; Zinnikus, I.: Model-Driven Semantic Web Service Matchmaking for Collaborative Business Processes. Proceedings of 2nd International Workshop on Semantic Matchmaking and Resource Retrieval (SMR2) at ISWC, Karlsruhe, Germany, 2008.
18. Klusch, M. and Zhing, X. Deployed semantic services for the common user of the web: A reality check., IEEE International Conference on Semantic Computing. pages 347-353, 2008.
19. Kourtesis, D., and Paraskakis, I. Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. Springer Berlin / Heidelberg. pages 614-628. 2008.
20. Levy, A.Y.: Logic-based techniques in data integration. In: Minker, J. (ed.) Logic-Based Artificial Intelligence. Kluwer Publishers, Dordrecht, 2000.
21. Lara, R., and Polleres, A. D4.2v0.1 Formal Mapping and Tool to OWL-S, WSMO working draft 17 december 2004. http://www.wsmo.org/2004/d4/d4.2/v0.1/
22. Leacock, C. and Chodorow, M. Combining local context andWord- Net similarity for word sense identification. In Christiane Fellbaum, editor, WordNet: An Electronic Lexical Database, pages 265–283. MIT Press, 1998.
23. Li, L. and Horrocks, I. A software framework for matchmaking based on semantic web technology. Int. J. of Electronic Commerce, vol. 8, 4, pages 39–60, 2004.
24. Li, Y., Bandar, Z., and McLean, D. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. Knowl. Data Eng., vol 15, 4, pages 871–882, 2003.
25. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDemott, D., McIlraith, D., Narayanan, D., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K.. OWL-S: Semantic Markup for Web Services. W3C Member Submission, 2004. Available from http://www.w3.org/Submission/OWL-S/.
26. Miller, G.A. WordNet: a lexical database for English, Communications of the ACM, pages 39-41, 1995. Association for Computational Linguistics.
27. NAICS Association. NAICS code searching. http://www.naics.com/search.htm, 2004.
28. Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. Semantic Matching of Web Service Capabilities. In ISWC, pages 333–347. Springer Verlag, 2002.
29. Paolucci, M., Wagner M., and Martin M. Grounding OWL-S in SAWSDL. Springer Berlin / Heidelberg. pages 416-421. 2007.
30. Papathanasiou, M., Loutas, N., Peristeras, V., and Tarampanis, K. Combining Service Models, Semantic and Web 2.0 Technologies to Create a Rich Citizen Experience. M.D. Lytras et al. (Eds.): WSKS 2009, LNAI 5736, pages 296–305, 2009
31. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J. and Domingue, J. iServe: a Linked Services Publishing Platform, Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference, 2010.
32. Pedersen, T., Patwardhan, S., and Michelizzi, J. WordNet::Similarity: measuring the relatedness of concepts. In Demonstration Papers at HLT-NAACL 2004 on XX (HLT-NAACL '04). Association for Computational Linguistics, Morristown, NJ, USA, 38-41.
33. Polleres, A., and Lara, R. A Conceptual Comparison between WSMO and OWL-S, WSMO Working Group working draft, 2005. http://www.wsmo.org/2004/d4/d4.1/v0.1/.

34. Rada, R., Mili, H., Bicknell, E., and Blettner, M. Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics, pages 17–30, 1989.
35. Renz, I., Ficzay, A., and Hitzler, H., Keyword Extraction for Text Characterization, Proceedings of Natural Language Processing and Information Systems. 2003.
36. Salton, G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley. 1989.
37. Sheth, A., Gomadam, K., and Lathem, J. SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups. IEEE Internet Computing, vol. 11, 6, pages 91–94 ,2007.
38. Sycara, K. and Widoff, S. and Klusch, M. and Lu, J. LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace. Autonomous agents and multi-agent systems, Vol. 5, 2, Pages 173-203, 2002.
39. UDDI: The UDDI Technical White Paper, http://www.uddi.org, 2000
40. UNDP, Dun and Bradstreet Corporation. UNSPSC code searching. http://www.unspsc.org/, 2004.
41. W3C World Wide Web Consortium. SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/.
42. Wu, Z. and Palmer, M. Verbs semantics and lexical selection. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pages 133–138, Morristown, NJ, USA, 1994. Association for Computational Linguistics.