Semantic Service Publication and Retrieval in the MAIS Project

Devis Bianchini[1], Valeria De Antonellis[1], Barbara Pernici[2], Pierluigi Plebani[2]

[1] *Università di Brescia*
Via Branze, 38 - 25123 Brescia, Italy
[2] *Politecnico di Milano*
Piazza Leonardo da Vinci, 32 - 20133 Milano, Italy

Abstract. According to the Service Oriented Computing paradigm, a lot of services have been recently implemented by several companies. Since the number of services are growing, tools able to aid people and software to look for a service are required. Current solutions, such as UDDI, even if compliant with the Service Oriented Computing paradigm, do not provide an efficent way to search for services, since they mainly rely on keyword-based approaches.
In this paper we present an approach based on a semantic service description supporting the service publication and retrieval processes. Such an approach is based on an UDDI extension in which a service ontology is considered.

1 Introduction

In the last few years the Service Oriented Architecture (SOA) has become very popular. The availability of technological standards such as WSDL for service description and UDDI for service registry has facilitated the development of an ever-growing number of Web services. One of the major limitations of these standards is due to the fact that service semantic aspects are not considered in the service description and only keyword-based search can be supported [6]. Current research work has therefore focused on the definition of advanced methods and tools to enable effective service usage in the different activities of publication, discovery, composition, invocation, monitoring and recovery. First of all, it is necessary to provide tools to express service semantics in a formal way making easier for machines to discover and use the right service at the right time. Recently the SemanticWeb technologies have suggested ontological tools to provide explicit description of service semantics and the ontology description languages OWL [19] and OWL-S [10] have been proposed. OWL language is based on the formal Description Logics that are endowed with a sound, complete and decidable inference procedure [1]. OWL-S is a service ontology specified in OWL. Several ontology-based approaches have already been proposed for describing service semantic aspects. In [17] service profiles are described in terms of inputs, outputs, pre- and post- conditions, as in OWL-S, and domain ontologies are used to annotate service concepts with additional semantic information in order to assess service matching. In [15] service profiles are instead described by means of Description Logics and inference mechanisms are exploited to establish service matching. In [13] a similarity-based approach is proposed for searching Web services described in WSDL. Similarity matching has been originally studied in the framework of software reuse [21] and process re-engineering [8]. In [2] a comparison and discussion of the different types of approach (keyword-based, concept-based, similarity-based and deductive) is provided. In [16] semantic Web service search is described as a multistep process including namespace filters and subsumption.

The aim of this paper is to present the approach developed within the MAIS Project [18], in which an extended service description is used as a basis for providing service publication and retrieval facilities in an enhanced UDDI Registry: the MAIS Registry. Service description results from the co-occurrence of several components: (i) a UDDI registry is responsible for handling offered service descriptions, (ii) a Domain Ontology provides the general knowledge about concepts of the business domain in which services are used, and (iii) a Service Ontology organizes services at different levels of abstraction. For service publication and retrieval two matching strategies are proposed: a deductive strategy with a reasoning procedure exploiting ontology knowledge to assess the type of match among services [4]; a similarity-based strategy exploiting retrieval metrics to measure the degree of match among services [7, 12]. The similarity approach is applied after the deductive one in order to rank selected services according to the measured matching degree.

The paper is organized as follows: in Section 2 we present the service description proposed in the ambit of the MAIS Project; Section 3 describes the deductive and the similarity-based matching strategies; Section 4 suggests a combined use of those two strategies for service publication and retrieval. Section 5 introduces an architecture, which extends UDDI Registry, able to support the service publication and retrieve processes. Finally, conclusions and related work are presented in Section 6.

2 Service description in MAIS

The aim of the service description in MAIS is twofold. On one hand, for publication purpose, the model specifies what a service is and which are the elements to describe it. On the other hand, the description model also considers in which way the published services are organized in order to support the retrieval process.

For service description, in our approach a *service descriptor* is defined, as usually done in the software components retrieval [11]. A descriptor is composed of information directly extracted from the service signature expressed in the related WSDL specification [9]. Here, the service name, the operation names, and the names of the parameters involved in those operations, are considered.

Once a service provider publishes its services, for each of them a service descriptor is automatically generated starting from the service WSDL specification.

The set of service descriptors are organized in a *service ontology* where they are classified according to the functionalities the services provide.

In more detail, the service ontology is organized in three levels as shown in Figure 1, where each box represents a service descriptor. From the bottom to the top, in the first level the published services are grouped in clusters. These clusters include the services which perform the same functionalities, and can be considered compatible. For this reason, we introduce the term *compatibility classes* to define such clusters. The upper level is populated by services able to represent the compatibility classes. Whereas the services at bottom level are services which can be invoked, the services at upper level are built to represent the cluster, therefore we refer to *concrete services* and *abstract services* to respectively describe such a distinction. In particular:

- *Concrete services* are real directly invocable services published by service providers. The result of the discovery phase is one or more of these services.
- *Abstract services* are not directly invocable services, which represent the capabilities of the concrete services belonging to the same compatibility class. Moreover, abstract services in the service ontology are semantically organized according to two kinds of semantic relationships:
  - an abstract service $\mathcal{S}_a$ is a *generalization* of another abstract service $\mathcal{S}_b$ if $\mathcal{S}_b$ provides at least the operations of $\mathcal{S}_a$;
  - an abstract service $\mathcal{S}_a$ *isComposedOf* a set $\Phi = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ of other abstract services if the operations of $\mathcal{S}_a$ are included in the union set of operations of $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$; in particular, Φ must be minimal (no redundancy); the service $\mathcal{S}_a$ is often called the *composite service*, while $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are called the *component services*.

Finally, at the top level, according to UDDI Registry structure, the abstract services refer to categories as defined in well known taxonomies, such as UNSPSC and NAICS.

The service ontology is one of the main elements composing the MAIS Registry, a tool extending UDDI able to support the service publication and retrieval process, presented in more detail in Section 2. In fact, in the MAIS Registry, in addition to the functional description expressed in WSDL and service provider information, for each concrete service its QoS characteristics are published, using the WSOL [20] language to specify quality dimensions, ranges, functions, and relationships among different quality dimensions [5]. However these aspects are out of the scope of the present paper.

3 Service matching

As presented above, the service ontology is based on the compatibility classes. For this reason we need a way to evaluate how much a service matches to another one, in order to identify in which services belong to which compatibility class. In this section we present two different approaches which will be used during the

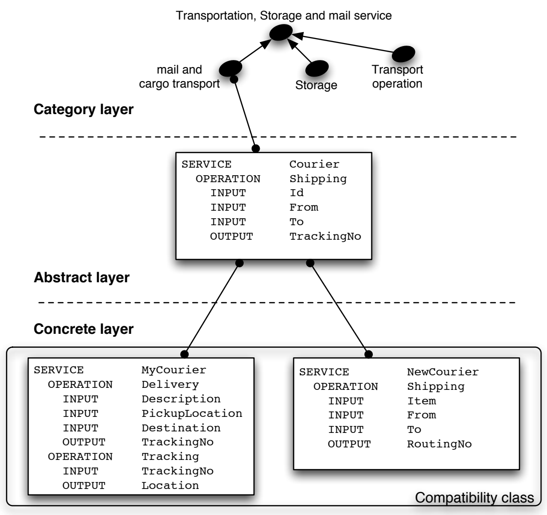

Fig. 1. Service ontology structure

service publication and retrieval process as presented in Section 4. The first one is based on a deductive process, whereas the second one on a semantic analysis of the involved terms.

3.1 The deductive approach to service matching

In this section we propose a deductive approach in which, with the help of reasoning on domain ontological knowledge, different kinds of match can be recognized among service descriptions. We formalize service descriptions by means of the $\mathcal{SHOIN}(\mathcal{D}+)$ Description Logic that provides formal foundations and reasoning support for the automation of service discovery process and has influenced the development of the semantic ontology web language OWL-DL. Semantic service descriptions are organized in a service ontology according to generalization/specialization hierarchies and composition relationships (see Section 4).

In this approach, a service is formally described as a conjunction of a concept in the form $\exists\texttt{hasCategory}.CAT$, where CAT is the service category representing the domain of interest of the service, and one or more concepts in the form $\exists\texttt{hasOperation}.OP$ for each provided operation OP that represents service capability. Most of the required information are directly obtained by the service descriptor in the service ontology. Each operation OP is described as a conjunction of:

- the operation name, expressed by means of an atomic concept;
- a conjunction of one or more concepts in the form $\exists hasInput.IN$, where IN is a concept representing an input parameter;

- a conjunction of one or more concepts in the form $\exists hasOutput.OUT$, where OUT is a concept representing an output parameter.

Service categories, operation names and I/O parameter names refer to concepts defined in an underlying *domain dependent ontology*, handled by a domain expert which collects all the relevant terms and organizes them according to their semantic relationships.

Matching between the request $\mathcal{R}$ and the advertisement $\mathcal{S}$ is checked by verifying the satisfaction with respect to the domain ontology of semantic relationships for concepts to which the service description elements refer. Following current directions in the literature [3], we consider several kinds of match, that can be intuitively described as follows:

- *pre-filtering*, if categories of $\mathcal{R}$ and $\mathcal{S}$ are the same or are related in any generalization hierarchy in the domain ontology, then the other kinds of match are investigated, otherwise the match fails;
- *exact match*, to denote that $\mathcal{S}$ and $\mathcal{R}$ have the same capabilities, that is, they have names of operations, input parameters and output parameters that are equivalent in the domain ontology;
- *plug-in match*, to denote that $\mathcal{S}$ offers at least the capabilities of $\mathcal{R}$, that is, the operations in $\mathcal{R}$ can be mapped into operations of $\mathcal{S}$ and, in particular, their names, input parameters and output parameters are related in any generalization hierarchy in the domain ontology;
- *subsume match*, to denote that $\mathcal{R}$ offers at least the capabilities of $\mathcal{S}$, like in plug-in match, but with the roles of $\mathcal{R}$ and $\mathcal{S}$ exchanged;
- *intersection match*, to denote that $\mathcal{S}$ and $\mathcal{R}$ have some common operations and some common I/O parameters, that is, some pairs of operations and some pairs of parameters respectively are related in any generalization hierarchy in the domain ontology;
- *mismatch*, otherwise.

The proposed deductive approach ensures matching flexibility, since it would be unrealistic to expect only exact match between the request and the advertisement, as emphasized in [17].

3.2 The similarity-based approach to service matching

The service matching presented in this section aims at measuring the similarity deep among two services by analyzing the terminological relationships (e.g., hyperonomy, synonymy) among the terms included in the service descriptors.

This approach can rely not only on a *domain dependent lexical ontology* in which the relevant terms are organized, but also on *domain independent lexical ontology* such as WordNet [14]. In particular, we can state how much a term is similar to another term, by computing properly defined similarity coefficients. The similarity is defined by a value in the range $[0, 1]$.

The similarity between two services $\mathcal{S}_a$ and $\mathcal{S}_b$, is defined by the *Global similarity coefficient* (G_{Sim}) defined as a composition of two other similarity coefficients, i.e. Entity-based similarity coefficient (E_{Sim}), and Functionality-based similarity coefficient (F_{Sim}) [5]:

$$G_{Sim}(\mathcal{S}_a, \mathcal{S}_b) = w_1 \cdot NormE_{Sim}(\mathcal{S}_a, \mathcal{S}_b) + w_2 \cdot NormF_{Sim}(\mathcal{S}_a, \mathcal{S}_b) \quad \in [0,1] \quad (1)$$

$NormE_{Sim}$ and $NormF_{Sim}$ are respectively the values of E_{Sim} and F_{Sim} normalized to the range $[0,1]$. In more detail, E_{Sim} considers all the I/O parameters defined in the service descriptor, independently from the operation with which a particular parameter is associated. In this way, E_{Sim} measures how much two services are based on the same information set. At this stage, such a similarity is performed only considering the name affinity of the parameter names. Further study will consider the data type as well.

On the other side, F_{Sim} performs a more functional analysis. In this case, each operation in $\mathcal{S}_a$ is compared with all the operations in $\mathcal{S}_b$ in order to identify which is the operation in $\mathcal{S}_b$ more similar to the considered operation in $\mathcal{S}_a$. Such a comparison is performed considering name affinity between the operation names and the affinity among the information set involved, i.e. the I/O parameters. Thus, F_{Sim} returns a value which collects all the best comparison measures and, in this way, reflects how much the two services perform the same functionalities.

Weigths w_1 and w_2, with w_1, $w_2 \in [0,1]$ and $w_1 + w_2 = 1$, are introduced to assess the relevance of each kind of similarity in computing the G_{Sim}. The use of weights in G_{Sim} is motivated by the need of flexible comparison strategies. For instance, to state that the E_{Sim} and F_{Sim} similarity have the same relevance, we choose $w_1 = w_2 = 0.5$.

4 Service publication and retrieval process

The presented deductive and similarity-based approaches to service matching can be (separately or in combination) applied in the publication and retrieval processes.

In the retrieval process, inference is used to classify the match between the desired service $\mathcal{R}$ (defined by a set of requested functionalities) and available abstract services $\mathcal{S}_a$ into one of the five kinds of match illustrated in Section 3.1. Successively, similarity evaluation can be exploited to further refine and quantify the functional similarity between $\mathcal{R}$ and $\mathcal{S}$, according to the following rules:

– if *exact* or *plug-in match* occurs, from the request viewpoint the abstract service provides completely the required functionalities, so $G_{Sim}(\mathcal{R}, \mathcal{S}_a)$ is set to 1 (full similarity) without computing the similarity coefficients;
– if *mismatch* occurs, $G_{Sim}(\mathcal{R}, \mathcal{S})$ is directly set to zero;
– if *subsume* or *intersection match* occurs, the abstract service fulfills the request only partially and similarity coefficients are computed to quantify how much the abstract service satisfies the request; in this case, $G_{Sim}(\mathcal{R}, \mathcal{S}_a) \in (0,1)$.

Only available services for which the $G_{Sim}(\mathcal{R}, \mathcal{S}_a)$ is equal or greater than a given threshold are proposed among the search results, ranked with respect to the G_{Sim} values.

The abstract services are obtained according to a semi-automatic procedure supervised by the domain expert. Such a procedure can follow both a bottom-up and a top-down approach.

According to a bottom-up approach, once the clusters are built using the $FSim$, the abstract service is defined with a "minimal" set of operations, i.e. the set of operations in the abstract service interface are only those common to all the services in the cluster. It is worth noting that this set of operations is composed only by the semantically common operations. Two operations with synonym names are considered equal. The designer can force additional capabilities to the abstract service also including operations belonging to a large number of services in the cluster.

On the other hand, in the top-down approach, the domain expert defines a set of abstract services on the basis of his knowledge about the domain. In the publication process, when a service provider publishes the service in the Registry, according to the $FSim$ value, the service is assigned to one of the existing compatibility classes. It might happen that none of the existing abstract service is close enough to the service being registered. In this case, the domain expert is in charge of defining a new abstract service.

Actually, a good approach to create and to manage abstract and concrete services relies on a mix of the two approaches. When a compatibility class becomes too populated, indeed, it means that it could be useful to split the class in two subclasses, each of them assigned to an abstract service. These abstract services will be specializations of the former abstract service. According to that, the abstract layer organization can become more complex and organized in several levels.

Semantic relationships between abstract services can be exploited to make more efficient the retrieval procedure, according to the following intuition: if an abstract service $\mathcal{S}_a$ matches with a given service request $\mathcal{R}$, then also abstract services that provide the same capabilities of $\mathcal{S}_a$ (as expressed by means of semantic relationships) match with $\mathcal{R}$. According to this intuition, the following rules are applied:

- if $\mathcal{S}_a$ presents an *exact* or a *plug-in match* with $\mathcal{R}$ and $\mathcal{S}_a$ is a generalization of another abstract service $\mathcal{S}_b$, then also $\mathcal{S}_b$ presents a *plug-in match* with $\mathcal{R}$ and the application of matching algorithm to $\mathcal{S}_b$ is not required; we can set $G_{Sim}(\mathcal{R}, \mathcal{S}_a) = G_{Sim}(\mathcal{R}, \mathcal{S}_b) = 1$;
- if $\mathcal{S}_a$ presents a *mismatch* with $\mathcal{R}$ and another abstract service $\mathcal{S}_b$ is a generalization of $\mathcal{S}_a$, then we can say that also $\mathcal{S}_b$ presents a *mismatch* with $\mathcal{R}$;
- the same procedure applies when $\mathcal{S}_a$ is a composite service and $\mathcal{S}_b$ is the union of its component ones;
- otherwise, we cannot say anything about $G_{Sim}(\mathcal{R}, \mathcal{S}_b)$ and the matching procedure must be applied also to it.

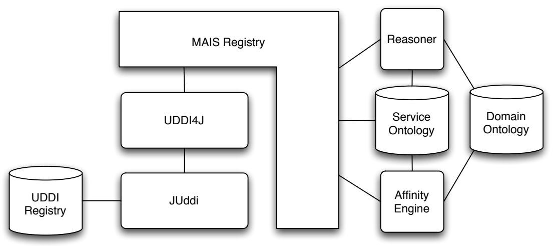

Fig. 2. MAIS Registry architecture

Finally, after abstract services that match with the request are identified, concrete services belonging to the corresponding clusters are included into the search results, by setting the G_{Sim} value of each concrete service with respect to $\mathcal{R}$ equal to the G_{Sim} value of the corresponding abstract service.

5 Architecture

The publication and retrieval processes described above have been implemented as a UDDI Registry extension. Since one of the most discussed weakness of UDDI Registry is about the limited retrieval method, with our implementation we aim at providing a new way of searching services. In particular, the new searching method allows the user to submit a WSDL expressing the desired service, in order to obtain the list of services able to perform the requested functionalities.

Figure 2 shows the architecture supporting the publication and retrieval process described in the previous section. Such an architecture is designed to be completely compliant with the current UDDI v.2 implementations. For this matter, the system relies on Juddi, an open source implementation of UDDI which also exposes its functionalities according to UDDI4J API. In particular, the MAIS Registry redefines the functionalities about the service publication and introduces new functionalities which allow the user to perform the advanced retrieval functions based on the service semantics evaluation.

The Affinity Engine performs the similarity evaluation and the Reasoner performs the deductive matching by exploiting the domain ontology and the service ontology.

During the publication phase, the MAIS Registry is able to read a user publication request and, before performing the standard publication steps required by UDDI, identifies the corresponding abstract service and updates the Service Ontology. In this way the Service Ontology can organize the published services according to the layers discusses in Section 2.

On the other hand, in case a typical service retrieval method is requested, the related functionality supported by JUddi is called. Otherwise, if the user searches for a service according to the new retrieval method, the new functionality offered by the MAIS Registry is directly invoked. In this case the Affinity Engine, as well as the Service Ontology, are invoked in order to perform the retrieval process discussed in Section 4.

6 Conclusions and future work

The paper presents a novel approach to service publication and retrieval, based both on deductive techniques and on matching techniques derived from the information retrieval domain. The current implementation is based on UDDI v2 and partially implements the functionality described in the paper. Experimentation has shown that semantic affinity performs well for identifying clusters in which services are published, even using a term ontology not specialized in given application domains. For retrieval purposes, on the other hand, the need of asymmetric similarity evaluation has emerged in case single operations (or given groups of operations) are to be found against a user request. Experimentation with different matching algorithms is on going.

Another extension to the basic UDDI registry involves the publication of quality of service information by providers for each service and for each operation, and the selection of services according to preferences expressed by requestors and providers. On going research work studies how to associate quality information to services and mechanisms to evaluate and monitor the quality levels declared by providers and the ones actually provided during service execution.

Acknowledgments

Part of this work has been supported by Italian MIUR under the MAIS (Multichannel Adaptive Information Systems) FIRB project.

References

1. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press, 2003.
2. A. Bernstein and M. Klein. Towards High-Precision Service Retrieval. In *Proc. of the International Semantic Web Conference (ISWC 2002)*, pages 84–101, Sardinia, Italy, June 2002.
3. A. Bernstein and M. Klein. Towards high-precision service retrieval. *IEEE Internet Computing*, pages 30–36, 2004.
4. D. Bianchini and V. De Antonellis. Ontology-based Semantic Interoperability Tools for Service Dynamic Discovery. In *Proc. of the First Int. Conference on Interoperability of Enterprise Software Applications (INTEROP-ESA'05)*, Geneva, Switzerland, February, 23rd-25th 2005.
5. D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. Ontology-based Methodology for *e*-Service discovery. *Accepted for publication on Journal of Information Systems, Special Issue on Semantic Web and Web Services*, 2004.

6. F. Casati, M-C. Shan, and D. Georakopoulos. The vldb journal: Special issue on e-services. *Springer-Verlag Berlin Heidelberg*, 10(1), 2001.
7. S. Castano and V. De Antonellis. A Framework for expressing Semantic Relationships between Multiple Information Systems for Cooperation. *Information Systems*, 19(4):33–54, 1998.
8. S. Castano, V. De Antonellis, and M. Melchiori. A Methodology and Tool Environment for Process Analysis and Reengineering. *Data and Knowledge Engineering*, 31(3):253–278, November 1999.
9. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1.* World Wide Web Consortium (W3C), March 2001. `http://www.w3.org/TR/2001/NOTE-wsdl-2001 0315`.
10. The OWL Service Coalition. *OWL-S 1.1 beta release*, July 2004. `http://www.daml.org/services/owl-s/1.1B`.
11. E. Damiani and M.G. Fugini. Fuzzy Identification of Distributed Components. In *Proceedings of the 5th Fuzzy Days International Conference*, LNCS 1226, pages 550–552, 1997.
12. V. De Antonellis, M. Melchiori, B. Pernici, and P. Plebani. A Methodology for *e*-Service Substitutability in a Virtual District Environment. In *Proceedings of the 2003 Conference on Information Systems Engineering (CAiSE 2003)*, Velden, Austria, 2003.
13. X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In *Proc. of the 30th Int. Conference on Very Large Data Bases (VLDB2004)*, Toronto, Canada, August 29th - September 3rd 2004.
14. C. Fellbaum (ed.). *Wordnet: An Electronic Lexical Database and Electronic Commerce.* MIT Press, 1998.
15. I. Horrocks and L. Li. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, USA, October 2003.
16. T. Kawamura, J.-A. D. Blasio, T. Hasegawa, M. Paolucci, and K. Sycara. Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry. In *Proc. of First Int. Conf. on Service Oriented Computing (ICSOC 2003)*, pages 208–224, Trento, Italy, 2003.
17. T. Kawamura, M. Paolucci, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proc. of First Int. Semantic Web Conference (ISWC)*, 2002.
18. The MAIS Project Home Page. `http://www.mais-project.it`.
19. D. L. McGuinness and F. van Harmelen. *OWL Web Ontology Language Overview.* World Wide Web Consortium (W3C) Recommendation, February 10th 2004. `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.
20. V. Tosic, K. Patel, and B. Pagurek. WSOL - Web Service Offerings Language. In *Web Services, E-Business and the Semantic Web, CAiSE 2002 International Workshop on Web Services, E-business, and the Semantic Web (WES 2002)*, Toronto, Canada, May 27-28th 2002.
21. A. M. Zaremski and J.M. Wing. Specification matching of software components. In *Proc. the 3rd Int. ACM Symposium on Foundations of Software Engineering (SIGSOFT)*, pages 6–17, Washington DC, USA, 1995.