# Ranked Matching for Service Descriptions using DAML-S

Michael C. Jaeger and Stefan Tang

Technische Universität Berlin, FG iVS,
Einsteinufer 17, Sek. EN-6, 10587 Berlin, Germany
mcj@ivs.tu-berlin.de, steftang@stanford.edu

**Abstract.** The vision of Semantic Web services is that computer systems shall find eligible services autonomously. This can be realised with providing semantic description about advertised and requested services. In this scenario a so-called matchmaker finds matches between service requirements and advertisements according to their description.
Based on the ontology language DAML+OIL a proposal for an upper ontology for the semantic description of services called DAML-S exists already. Also, approaches were introduced, which act as matchmakers for DAML-S descriptions. The contribution of this work is a new algorithm, which ranks the level of matching for DAML-S descriptions. To determine the different degrees of matching, elements of the DAML-S description are considered and matched individually.
With ranking a criterion is available to select a service among a large set of results. Consider the result of flat matchmaking that consists of a set of matching and another set of non-matching services. If an autonomous system must still choose one of the set of matching services. An ordered list of services provides a decision support to autonomously choose the best service possible.

## 1 Introduction

The Semantic Web working group of the W3C develops technologies and standards for the semantic description of the Web. The goal of these efforts is to make the Web understandable by machines in order to increase the level of autonomous interoperation between computer systems [6]. Several standards and languages were already released to follow this objective. For a brief introduction, the most relevant are XML, RDF [4], and OWL [8]. First, everything must be written in XML notation and specified with XML Schema. Building on top of this, RDF and its extension RDF Schema are provided to express basic statements. The primary goal of RDF is to structure and describe existing data. Thus, RDF is also called a metadata language. RDF Schema is a vocabulary extension to RDF, which allows expressing categorisations in a standardised way. Based on RDF and RDF Schema, OWL - the successor of DAML+OIL - increases the level of expressiveness with a richer vocabulary with retaining the decidability.

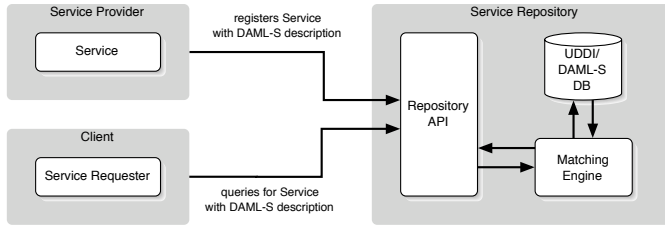


**Fig. 1.** Matchmaking in a Server-Sided Scenario

These languages are primarily used for content. The next logical step is to describe the semantics of services for optimising their platform- and organisation-independent interoperability over the Internet. Referring to the Semantic Web, the research field addressing this objective is named *Semantic Web services* [9]. It's vision is to apply semantic description for Web services in order to provide relevant criteria for their automated selection. Based in the predecessor of OWL – DAML+OIL – an upper ontology for the description of Web services named DAML-Services (DAML-S) is introduced [3]. Basically, the DAML-S ontology defines three elements: a service profile to describe the functionality of a service, a service model to model the structure of the service, and a service grounding to map the abstract interface to a concrete binding information.

Using DAML-S for the description of Web services can increase the ability of computer systems to find eligible services autonomously. A service provider describes his advertised services in a DAML-S description and a service requester queries for services with a DAML-S description expressing his requirements. In this scenario a socalled matchmaker finds matches between service requirements and advertised services according to their descriptions.

We will introduce previous work already addressing the matching of DAML-S descriptions more precisely in section 4. Our approach is an algorithm, which ranks the level of matching for advertisements with requirements. The ranks are determined by considering individual elements of the DAML-S description and the aggregation of their individual matchmaking.

## 2 Background

For the application of the matching algorithm, we can distinguish two scenarios: In the first, the semantic description of individual services is stored and processed on a central server. For this, existing repositories managing information and addresses of Web services could be extended to be compatible with DAML-S descriptions. A service requester provides with his query also his description and the server processes this request by matching the requirements with the description of eligible services. Then the server returns the best match. This scenario pretends that a generally accepted organisation maintains the repository and service providers are willing to submit a DAML-S description about their services. Figure 1 outlines this schema. An existing specification for a repository

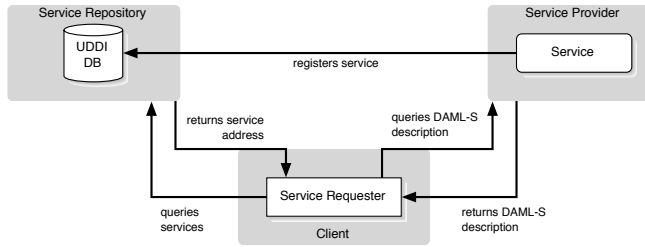


**Fig. 2.** Matchmaking in a Client-Sided Scenario

to find Web services in the Internet is UDDI [14]. The approach to extend UDDI repositories with semantic description can be found in [10] and [1].

In the second case the service requester obtains the DAML-S description for eligible services and performs the matching process on its own. The address of the service and the information how to obtain the DAML-S description can be found in a repository. Figure 2 outlines a scenario, where the service requester contacts an UDDI repository to find eligible services. Then, the individual services are contacted and their descriptions are requested. The service requester processes the descriptions and ranks the service by their level of matching. To favour either the server-centirc solution the following considerations are relevant:

**Simple Client:** The implementation of a client, which acts as the service requester can be kept very simple. Thus, the effort for finding services on the client side is very low.

**Centralised Logic:** Since currently DAML-S is an evolving proposal, it is likely that languages, standards and maybe the matching algorithm may become revised. Applying the necessary changes only on the server-side means lower efforts than keeping various applications up-to-date.

These two advantages are also well known for other approaches, which have lead to a server-oriented architecture in general. However, some advantages exist, which might be deciding to prefer a client-sided approach:

**Custom Matching:** A service requester wants to use an own matching algorithm instead of the implementation of the central server. In our matching algorithm we also provide to extend the matching with user-defined modules. These user-defined modules can define constraints that must be present in the DAML-S description of the advertised service. Only with the client-sided execution of such modules the personalisation of the matching process is possible.

**Organisational Limitations:** To perform the matchmaking of DAML-S descriptions on a central server pretends the cooperation of the service providers to submit their DAML-S descriptions. A decentralised approach would mean less organisational efforts and a quicker adoption within an organisational domain.

Since our approach includes user-customisable modules to enhance the matching algorithm, we favour the client-sided scenario. In our work a client application

representing the service requester queries the DAML-S description from an already obtained list of services to find the best match.

### 2.1 Short Introduction to DAML-S

The DAML-S ontology defines three basic elements: (1) a service profile to describe the functionality of a service, (2) a service model to model the structure of the service, and (3) a service grounding to map the abstract interface to a concrete binding information. As the focus of this work is on the discovery and selection of Web services, our main interest lies in the profile, which provides the necessary information for the matching algorithm. The profile provides three basic types of information. First, it presents a textual description and contact information, which is mainly intended for human users.

The next information provided is a functional description of the service. This functional description is expressed in terms of the input parameters and the output parameters generated by the service. Additionally, the functional description is described by two sets of conditions, namely preconditions, which have to hold before the service can be executed properly, and effects, which are conditions that hold after the successful execution of the service. These four functional descriptions are also referred to as *IOPE* (Input Output Precondition Effect). The IOPEs are realized through an object property called parameter. This property assigns a class of type ParameterDescription to the profile. The following four object properties are subproperties of parameter, and describe the functional behaviour of the Web service:

| | |
|---|---|
| input | Specifies one input parameter of the profile. A profile can have as many inputs as required (including none). |
| output | Specifies one output parameter of the profile. A profile can have as many outputs as required (including none). |
| precondition | Specifies one precondition of the profile. There can be as many preconditions as necessary. |
| Effect | Specifies one effect of the profile. A profile can have as many effects as required. |

Finally, the profile comes with a set of additional properties that are used to describe features of the service. From these properties we use in our algorithm the service category, which is used to classify the service with respect to some ontology or taxonomy of services. The value of the property is an instance of the class ServiceCategory.

## 3 Ranked Matching for DAML-S Descriptions

As defined in the DAML-S specification, a Web service is semantically described by an instance of the class Profile. Although a service can have multiple profiles as a Web service can provide multiple operations, in the following the words service and profile are used synonymously, meaning one specific profile definition.

An instance of this class holds all the inputs and outputs associated with that service as object properties; it is therefore possible to find a relation between two services by directly finding the semantic relationship between the instances of the referring Profile classes, which would include all the existing relationships between the input and output parameters, as these are properties of the profile. Moreover, if the Web service is described by an instance of a subclass of the class Profile, a matching algorithm can also perform a subsumption reasoning to determine the relation between the two services.

The result of this consideration is a matching algorithm, which is divided into three stages: (a) the matching of inputs, (b) the matching of outputs, and (c) the matching of the service profile classification itself. This algorithm determines the matching for each of the stages individually. The result is aggregated with a fourth stage, where user-defined constraints or functionality can complete the matching result. Most parts of the matching algorithm will build on the semantics of the services. Using DAML-S means that the semantics are defined in a DAML+OIL ontology. Our first task is thus classifying the different relations between two concepts and two properties to determine the degree of matching. From sections 3.1 to 3.3 we will explain the different three stages and the forth user-defined stage. In section 3.5, we present how the results of the different stages are aggregated to form the whole algorithm.

**Concept Matching**

Let $A$, $B$ denote two concepts which originate from a given set of ontologies. Then the following three relationships between $A$ and $B$ are considered:

| | |
|---|---|
| $Fail(A, B)$ | The concepts $A$ and $B$ are in no relation with each other. |
| $Subconcept(A, B)$ | The concept $B$ is subsumed by the concept $A$, meaning that $A$ denotes a more general concept than $B$. |
| $Equivalent(A, B)$ | $A$ and $B$ are equivalent, meaning that both denote exactly the same concept. |

The matching algorithm queries the underlying knowledge base to determine the relationship between the concepts. For checking the equivalence of two concepts, it is not sufficient to check if A = B. Two concepts can be equal even if they have different names and are defined in two different ontologies, as it is possible to equalise two concepts with usage of the <sameClassAs> element.

**Property Matching**

Let $R$ and $S$ denote two properties which originate from a given set of ontologies. We then have the following relationships between $R$ and $S$:

| | |
|---|---|
| $Fail(R, S)$ | The two properties $R$ and $S$ are in no relation to one another with respect to all referenced ontologies. |
| $Subproperty(R, S)$ | The property $R$ is a subproperty of the property $S$. |
| $Equivalent(R, S)$ | $R$ and $S$ are equivalent which means that both denote exactly the same property. |

The matching algorithm provides a functionality to determine the above mentioned relation between two given properties. As with concepts, two dif-

ferent properties can be defined to be equivalent by using the tag element <samePropertyAs>. In the implementation the matching algorithm queries the knowledge base to determine if two properties are equivalent.

### 3.1 Matching of Inputs

At this matching stage it is determined how well each input of the advertised service matches with an input of the requested service. As a general condition for using the advertised service properly, every input of the advertised service has to be matched. Since any input itself is defined as a property, there are two possibilities of finding a match: property matches and type matches. For the matching of the input properties table the following matching results can be distinguished:

| | |
|---|---|
| *FAIL* | At least one of input properties of the advertised service is not matched by the requested service. Thus, the service cannot be invoked properly. |
| *SUBPROPERTY* | The input of the requested service is a subproperty of the input of the advertised service. This means that the more specific property of the requested service will at least provide the same characteristic as required for proper invocation of the advertised service. |
| *EQUIVALENT* | Both inputs denote the same property. |

For the matching of input types as defined in the service profile, subsumption reasoning can be applied, because the types are described as concepts:

| | |
|---|---|
| *FAIL* | At least one of input types of the advertised service has not been matched or the requested service subsumes the input type of the advertised service. This means that a more specific input type is required than advertised. In this case the demands of the service requester are not met. |
| *SUBSUMES* | The input type of the requested service is subsumed by the input type of the advertised service. This means that the advertised service would be invoked with a more specific input than expected. However, the proper service execution is guaranteed. |
| *EQUIVALENT* | Both input types denote the same concept. |

For the ranking of the matching result of this stage, the result of property matching and type matching must be combined. In this combination, the property matching is given a higher priority than the type matching, because the classification of an input gives more insight to it's purpose than it's type definition. The table 1 lists the different combinations of results from the property- and type-matching and the applied ranks.

| Rank | Result for Property- and Type-Matching | Interpretation |
|---|---|---|
| 0 | FAIL | Any |
| | Any | FAIL |
| | There is at least one input of the advertised service for which there exists no matching input of the requested service | |
| 1 | SUBPROPERTY | SUBSUMES |
| | Every input of the advertised service could be matched with one input of the requested service. For every input pair the property-match degree is either Equivalent or Subproperty, but at least one input pair is matched with degree Subproperty, and the type-match degree is Subsumes. | |
| 2 | SUBPROPERTY | EQUIVALENT |
| | Every input of the advertised service could be matched with one input of the requested service. For at least one input pair the property-match degree is Subproperty, but for all type match pairs the degree is equivalent. | |
| 3 | EQUIVALENT | SUBSUMES |
| | Every input of the advertised service could be matched with one input of the requested service. For every input pair the property-match degree is Equivalent, but at least for one input pair the type-match degree is Subsumes. | |
| 4 | EQUIVALENT | EQUIVALENT |
| | Every input of the advertised service could be matched with one input parameter of the requested service. For every input pair the property-match degree and type-match degree is Equivalent. | |

**Table 1.** Ranking for the Matching of Input Parameters

### 3.2 Matching of Outputs

In this stage of matching it is determined how well the outputs of the advertised service correspond to the outputs of the requested service. During the output parameter matching, the algorithm tries to find a match for each output of the requested service. Analogously to the matching of inputs, the matching of outputs considers property-match and type-match degrees to find a relationship between two outputs.

However, the direction of the generalisation is now reversed, meaning that a Subproperty degree is recognized when the output of the advertised service is a subproperty of the output of the requested service and a Subsumes degree is found whenever the type of the output parameter of the advertised service is subsumed by the type of the output parameter of the requested service. In other words the output of requested service considers more general definitions than the output of the advertised service. This ensures that the requirements of the service requester will still be met. The definition of matching results for the property-matching and type-matching is omitted, because of it's mentioned

| Rank | Result for Property- and Type-Matching | Interpretation |
|---|---|---|
| 0 | FAIL | Any |
| | Any | FAIL |
| | There is at least one output of the requested service for which there exists no matching output of the advertised service | |
| 1 | SUBPROPERTY | SUBSUMES |
| | Every output of the advertised service could be matched with one input of the requested service. For every output pair the property-match degree is either Equivalent or Subproperty, but at least one output pair is matched with degree Subproperty, and the type-match degree is Subsumes. | |
| 2 | SUBPROPERTY | EQUIVALENT |
| | Every output of the requested service could be matched with one output of the advertised service. For at least one output pair the property-match degree is Subproperty, but for all type match pairs the degree is Equivalent. | |
| 3 | EQUIVALENT | SUBSUMES |
| | Every output of the requested service could be matched with one input of the advertised service. For every output pair the property-match degree is Equivalent, but at least for one output pair the type-match degree is Subsumes. | |
| 4 | EQUIVALENT | EQUIVALENT |
| | Every output of the requested service could be matched with one output parameter of the advertised service. For every output pair the property-match degree and type-match degree is Equivalent. | |

**Table 2.** Ranking for the Matching of Output Parameters

similarities to the input matching. Table 2 summarises the ranking levels for output parameter matching like table 1 for the input parameter matching. For every output parameter of the requested service the algorithm tries to find the matching output parameter of the advertised service that would result in the highest rank. As expected, these two output parameters then form a (matched) output pair.

### 3.3 Matching of the Service Profile

In DAML-S a service can be classified with certain categories described in a DAML+OIL ontology. In this ontology the individual service is an instance of a subclass of the class ServerCategory. With profile matching we try to determine how well the service category of the advertised service fits into the service category that the requested service demands. Let reqServiceCat denote the service classification of the requested service, and advServiceCat denote the service classification of the advertised service. Both reqServiceCat and advServiceCat are

thus concepts defined in some service classifying ontology. The identified matching results when matching the profile of the service request and the advertised profile are shown in table 3.

| Rank | Degree of Match | Interpretation |
|---|---|---|
| 0 | FAIL | The two concepts are in no relation with each other. |
| 1 | UNCLASSIFIED | Either the description of the advertised or the required service is not classified or one of the both are instances of the class Profile. |
| 2 | SUBSUMES | advServiceCat is an instance of class, which is subsumed by the class of the instance of reqServiceCat. In this case the classification of advServiceCat is more more specific than the classification of reqServiceCat. This means that the advertised service offers more specific functionality than required. |
| 3 | MATCH | reqServiceCat and advServiceCat are instances of the same class. |

**Table 3.** Ranking for the Matching of Service Categories

To clarify the meaning of the Subsumes degree, let us consider a small example: Assume a computer program which acts as a broker wants to let its user buy computers from different web stores. Therefore the program needs to make use of functions by web merchants who offer Web services, which allow to buy computers. The broker program thus looks for Web services which are classified into the service category BuyComputers, for example. Any advertised Web service which falls into the categories BuyDesktopComputers or BuyLaptopComputers, which both are subcategories of the category BuyComputers, would still be useful to the computer agent, allowing its users in turn to buy more specific kind of computers.

### 3.4 User-defined Matching

In addition to the three matching stages also a user-defined matching stage is provided. In the implementation, this matching stage consists of one or more additional modules, which can define specific requirements or conditions, which must be found in the description of the advertised service. As a result each of the modules returns either true or false. All matching results from the modules are interpreted conjunctively. If one user-defined matching function fails, the whole user-defined matching fails. For example, an important aspect that comes with the user-defined matching is the ability to verify that Quality of Service

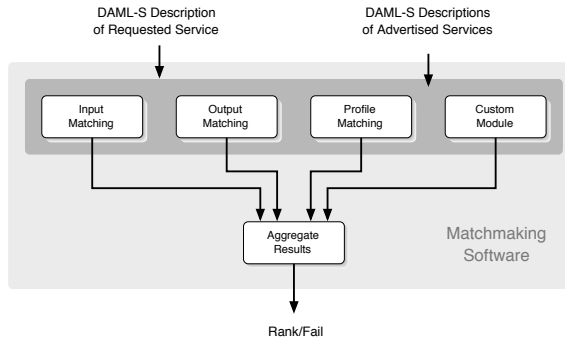


**Fig. 3.** Aggregation of the Matching Result

(QoS) constraints are met. A profile in DAML-S allows the specification of QoS for a given Web service via its class QualityRating. Thus, with given service definitions, it is possible to create plugins that performs a quality of service matching based on the available additional information in a service profile.

### 3.5 Combination of the four Stages

The result of the matching algorithm is an aggregation of the individual four stages. The simplest approach is to add up the ranks of the first three stages and complete the sum with the result of the user-defined module(s). The aggregated result is either the sum of the rank of the individual stages, which is an abstract measure of the matching quality, or the conclusion, that the descriptions did not match. However, the aggregation must consider, that if any of the individual stages will fail, the aggregated result must be regarded as fail as well. A perfect match of an individual matching stage cannot compensate a fail of another stage. This aggregation schema is outlined in figure 3.

## 4 Discussion and Related Work

Our proposal has elements from matching approaches for DAML-S, which were already introduced. A variation of the matching algorithm is to match complete Service Profile descriptions as found in [7]. Our motivation for refining the matching process is that it might easily occur that two profiles will be declared as non-compatible because one (probably less important) property in a profile is not matching to a property in the other profile. An optimal matching algorithm should indicate the lack of matching quality, but should not disqualify the advertised service completely.

Furthermore, DAML-S specifications can be arbitrarily complex. By splitting up the profile and determining the matches individually for the subparts, the algorithm can identify matches between service requests and advertisements more closely. The disadvantage that parts of the profile are ignored can be compensated with the proper usage of user-defined modules. For example, a module

that evaluates QoS statements could easily be added to the matching algorithm. Another characteristic of our work is that we consider the direction of the inheritance hierarchy between service requests and advertisements ("which concept subsumes which one"). This aspect can be considered as an refinement of existing matchmaking approaches as found in [15].

The matchmaker ATLAS [11] and the work of Paolucii et al. [10] consider the service profile and its inputs and outputs for determining matches between requests and advertisements as well. Our extensions to these approaches are (*a*) the explicit identification of individual elements and (*b*) the aggregation of the result from individual matching stages. A different approach for matching DAML-S descriptions is found in the work of Bansal and Vidal. A matchmaker is proposed that covers only the service model of DAML-S [2]. We do not agree with this idea, because as described in the introduction of DAML-S, the service model is not primarily provided to express requirements for finding matches with advertisements [3].

The matching of service descriptions is also addressed in other fields, which are not related with Web services or DAML-S. As an example, we refer to a work where conceptual graphs (CG) are used to describe services in the PhD thesis of Arno Puder [13], and his work about the AITrader [12]. This work has been applied in a CORBA environment and proposes an extension to the IDL. An approach for matching capabilities of Web services not using DAML-S but a custom RDF ontology is found in [15].

As already mentioned, some approaches also propose the integration of DAML-S descriptions with the UDDI standard, which we have introduced as an application scenario for the matching algorithm ([10] and [1]). We also plan to supplement with a defined procedure and a tool for the creation of DAML-S descriptions as introduced in [5]. A defined procedure for the description of services using DAML-S ensures, that relevant elements are well defined and therefore increase the efficiency of our matching algorithm.

## 5 Conclusions

We have shown that individual elements of the DAML-S upper ontology can be used to provide a fine grained ranking of matching results rather than flat subsumption reasoning for the service profile description as a whole unit. Also this work indicates the potential to provide even finer grained ranks with considering more elements of DAML-S.

For further research this field offers many opportunities: we plan to discuss the complexity of the algorithm, and to define a procedure for the creation of DAML-S descriptions. Other activities will focus on testing the algorithm to find an optimal weighting of the matching stages, and to identify more relevant elements of DAML-S for matchmaking. As the very next step, this work will be updated with the successor of DAML-S, which is based on OWL.

## Acknowledgments

The authors wish to thank Kurt Geihs, and Gregor Rojec-Goldmann for many fruitful discussions.

## References

1. Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A method for semantically enhancing the service discovery capabilities of uddi. In *In Proceedings of the Workshop on Information Integration on the Web*, pages 87–92, August 2003.
2. Sharad Bansal and Jose M. Vidal. Matchmaking of web services based on the daml-s service model. July 2003.
3. Anupriya Ankolenkar et al. Daml-s: A semantic markup language for web services. In *Proceedings of 1st Semantic Web Working Symposium (SWWS' 01)*, pages 441–430, Stanford, USA, August 2001. Stanford University.
4. Frank Manola et al. RDF Primer. W3C, http://www.w3.org/TR/rdf-primer/, 2004.
5. Michael Klein and Birgitta Koenig-Ries. A Process and a Tool for Creating Service Descriptions based on DAML-S. In *Proceedings of 4th International Workshop Technologies for E-Services, TES 2003*, pages 143–154. Springer, September 2003.
6. Marja-Riitta Koivunen and Eric Miller. W3c semantic web activity. In *Semantic Web Kick-Off in Finland*, pages 27–44, November 2001.
7. Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the twelfth international conference on World Wide Web (WWW2003)*, pages 331–339. ACM Press, May 2003.
8. Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. W3C, http://www.w3.org/TR/owl-features/, 2004.
9. Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18:90–93, January/February 2003.
10. M. Paolucci, T. Kawamura, T. Payne, , and K. Sycara. Semantic matching of web service capabilities. In *Proceedings of 1st International Semantic Web Conference. (ISWC2002)*, pages 333–347. Springer-Verlag, Berlin, 2002.
11. Terry R. Payne, Massimo Paolucci, and Katia Sycara. Advertising and matching daml-s service descriptions. In *Position Papers for SWWS' 01*, pages 76–78, Stanford, USA, July 2001. Stanford University.
12. A. Puder, F. Gudermann S. Markwitz, and K. Geihs. AI-based Trading in Open Distributed Environments. In *Proceedings of the 5th International Conference on Open Distributed Processing (ICODP'95)*, Brisbane, Australia, February 1995. Chapman and Hall.
13. Arno Puder. *Typsysteme für die Dienstvermittlung in offenen verteilten Systemen.* PhD thesis, Computer Science Department, Johann Wolfgang Goethe University, Frankurt/M., 1997.
14. UDDI Spec TC. Uddi version 3.0.1. OASIS, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm, 2003.
15. David Trastour, Claudio Bartolini, and Chris Preist. A semantic web approach to service description for matchmaking of services. In *Proceedings of the eleventh international conference on World Wide Web*, pages 89–98, Honolulu, USA, May 2002. ACM Press.