Importing the Semantic Web in UDDI

Massimo Paolucci[1], Takahiro Kawamura[1,2], Terry R. Payne[1], Katia Sycara[1]

[1] Carnegie Mellon University
Pittsburgh, PA, USA
{paolucci, takahiro, terryp, katia}@cs.cmu.edu
[2] Research & Development Center, Toshiba Corp.
1, Komukai Toshiba-cho, Saiwai-ku,
Kawasaki 212-8582, Japan
takahiro@isl.rdc.toshiba.co.jp

Abstract. The web is moving from being a collection of pages toward a collection of services that interoperate through the Internet. A fundamental step toward this interoperation is the ability of automatically locating services on the bases of the functionalities that they provide. Such a functionality would allow services to locate each other and automatically interoperate. Location of web services is inherently a semantic problem, because it has to abstract from the superficial differences between representations of the services provided, and the services requested to recognize semantic similarities between the two.
Current Web Services technology based on UDDI and WSDL does not make any use of semantic information and therefore fails to address the problem of matching between capabilities of services and allowing service location on the bases of what functionalities are sought, failing therefore to address the problem of locating web services. Nevertheless, previous work within DAML-S, a DAML-based language for service description, shows how ontological information collected through the semantic web can be used to match service capabilities. This work expands on previous work by showing how DAML-S Service Profiles, that describe service capabilities within DAML-S, can be mapped into UDDI records providing therefore a way to record semantic information within UDDI records. Furthermore we show how this encoded information can be used within the UDDI registry to perform semantic matching.

1 Introduction

Web services provide a new model of the web in which sites exchange dynamic information on demand. This change is especially important for the e-business community, because it provides an opportunity to conduct business faster and more efficiently. Indeed, the opportunity to manage supply chains dynamically to achieve the greatest advantage on the market is expected to create great value added and increase productivity. On the other hand, automatic management of supply chain opens new challenges: first, web services should be able to locate automatically other services that provide a solution to their problems, second,

services should be able to interoperate to compose automatically complex services.

In this paper we concentrate on the first problem: the location of web services on the basis of the capabilities that they provide. The solution of this problem requires a language to express the capabilities of services, the specification of a matching algorithm between service advertisements and service requests that recognizes when a request matches an advertisement and the deployment of registries that implement such algorithm. We leverage on on going work to specify DAML-S, a service description language that provides a logically grounded view of web services, [6, 4], and we leverage on on going work to develop an algorithm for capability matching for DAML-S Profiles [7]. Specifically, in this paper we show how DAML-S Profiles can be compiled into UDDI representations [11] and how UDDI registries can be modified to use the semantic information provided by DAML-S.

DAML-S ability of describing the semantics of web services is in stark contrast with emerging XML [12] based standards related with web services. Standards such as SOAP [13] and WSDL [3] are designed to provide descriptions of message transport mechanisms, and for describing the interface used by each service. However, neither SOAP nor WSDL are of any help for the automatic location of web services on the basis of their capabilities. Another emerging XML based standard is UDDI [11]; it provides a registry of businesses and web services. UDDI describes businesses by their physical attributes such as name and address and the services that they provide; business services are associate them with a set of attributes called `TModels`. TModels can be associated with description standards such as WSDL, or taxonomies such as NAICS [2]. Because UDDI does not represent service capabilities, it is of no help to search for services on the basis of what they provide.

A limitation shared by the XML based standards described above is their lack of an explicit semantics: two identical XML descriptions may mean very different things depending on when and who uses them. This proves to be a major limitation for capability matching: in fact, one crucial aspect of capability matching is that it can be done only at the semantic level. This is the case because the requester does not know what services are provided at any given time, otherwise it could contact the providers directly without need to search them; furthermore, advertisers and requesters have very different perspectives and different knowledge about the same service. The major problem with capability matching is that it is unrealistic to expect advertisements and requests to be equivalent, or even that exists a service that fulfills exactly the needs of the request. For example a service may advertise as a financial news provider, while a requester may need a service that report stock quotes. The task of the matching engine is to use its knowledge of the world and its semantic understanding of the advertisement and request to recognize their degree of mismatch and retrieve the advertisements that more closely match the request. It will be a task of the requester to select the provider that better suits its needs and how to deal with the mismatches between the request and the services that are available.

DAML-S supports our need for semantic representation of services through its tight connection with DAML+OIL [5]. DAML+OIL supports subsumption reasoning on taxonomies of concepts. Furthermore, DAML+OIL allows the definition of relations between concepts so that, for instance, it is possible to express statements like `X is part of Y` or more generally that a relation R exists between X and Y. The only limitations of DAML+OIL are its lack of a definition of well formed formulae and an associated theorem prover. While these limitations limit the expressivity of advertisements and requests, the language and the reasoning that can be associated are rich enough to allow the description of a wide range of services and matches between these descriptions.

In the rest of the paper, we describe DAML-S profiles to some detail; we will then describe how DAML-S profiles can be compiled into UDDI records; finally, we will then describe the main features of the matching algorithm adopted and how it can be integrated with UDDI.

2 DAML-S Profiles

The objective of a Service Profiles is to describe the functionality of services so that it can be selected to provide its functionalities. Service Profiles describe the "public face" of a service, or, in other words, what the service decides to provide to the community. Indeed, each service may have many functionalities, but not all of them have to be advertised. For example, a book-selling service may provide two different functionalities: it allows other services to browse its site to find books of interest, and it allows them to buy the books they found. The book-seller has the choice of advertising just the book-buying service or both the browsing functionality and the buying functionality. In the latter case the service makes public that it can provide browsing services, implicitly allowing other services to browse its registry without buying a book. In contrast, by advertising only the book-selling functionality, but not the browsing, the service discourages browsing by requesters that do not intend to buy. The decision as to which functionalities to advertise determines how the service will be used: a requester that intends to browse but not to buy would select a service that advertises both buying and browsing capabilities, but not one that advertises buying only.

From the technical standpoint, Service Profiles consist of three types of information: a description of the provider of the service, or, as DAML-S calls it, of the *actor* of the service; a specification of the *functionalities* that are provided by the service [10, 9] expressed in terms of the transformation produced by the service; and a host of *functional attributes* which provide additional information and requirements about the service and constraints on its use. For example, a news service that reports information about the current news on some topic could be described as follows: the actor describes the company that runs the service, its address, contact information and so on; the functionality would be described as a transformation that takes as input the news topic and as output

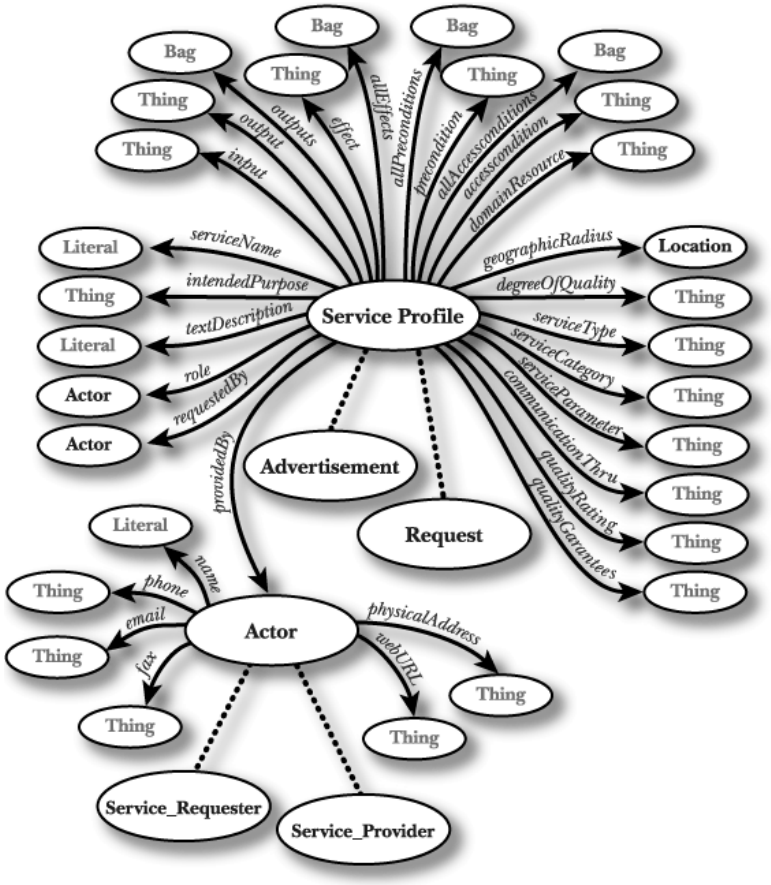

Fig. 1. Upper Ontology of Service Profiles

the news relevant to the topic; and the functional attributes could represent the response delay of the service, or its cost.

A service profile also describes the request for service, i.e. the expectations of the requester. For instance, a requester may look for a news service that reports stock quotes with no delay with respect to the market. The role of the registries is to match the request against the profiles advertised by other services and identify which services provide the best match.

Figure 1 shows the upper ontology for Service Profiles. The figure is logically divided in three parts: the bottom consists of the definition of *Actor*: it represent, for example, the type of information that is specified about the provider of the service. The middle part describes the *Functional Attributes*: they range from Quality Rating, that is the rating assigned to the service, to Geographic radius, that specifies whether there are geographic constraints to the service. Geographic

Constraints are used to prevent that the requesters are beyond geographic scope the service providers. Such a constraint, for example, prevents that a request for Chinese food issued in Pittsburgh is served by a restaurant Shanghai.

The top part of the figure represents the *Functional Description* of the service. It describes the capabilities of the service in terms of inputs, outputs, preconditions and effects. An input is what is required by a service in order to produce a desired output. For example, the input to a book buying service are the title and the author of the desired book. The output is a confirmation that the order has been received and successfully processed. The preconditions represent conditions in the World that should be true for the successful execution of the service. In the book buying example a precondition would be a valid credit card. The execution of the service may result in actions in the World, these conditions are described as the effects of the agent. In the buying of a book example, the credit card is charged and the book changes property.

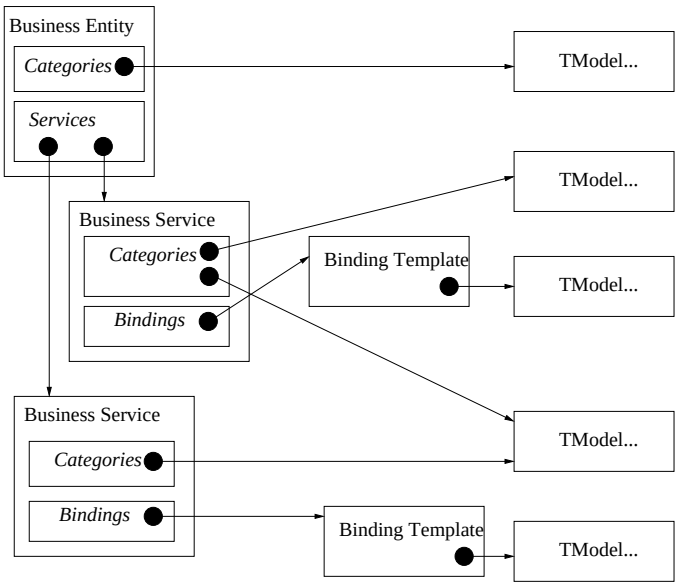

Fig. 2. UDDI Service Representation

3 From DAML-S to UDDI

The Universal Description Discovery and Integration (hereafter UDDI)[11] is an industrial initiative whose goal is to create an Internet wide network of registries of web services. UDDI allows businesses to register their presence on the web by specifying their points of contact both in terms of the ports used by the service

to process requests and in terms of the physical contacts with people that can answer questions about the service. In addition, UDDI provides a language to specify an unbounded set of features of services that can help the process of service location and selection as well as service invocation.

Because UDDI enjoys the wide support of many prominent software and hardware companies that invested heavily in web-services, it is becoming the de facto standards repository of web services. Nevertheless, as we describe below, UDDI has a number of shortcomings especially in the search mechanisms that prevent the exploitation of its full capacities.

DAML-S provides a way to solve this problem, by allowing a semantic description and matching of services within UDDI. Therefore, if we could translate the DAML-S representations into UDDI representations we combine the best of two worlds: we take advantage of the UDDI popularity and support, while publishing semantically grounded descriptions of services that can be used to perform capability based search for services. In this section we describe the UDDI service representation and how DAML-S representation can be compiled into UDDI representations.

3.1 UDDI Representation

The representation of services in UDDI is shown in figure 2. A business is represented as a *BusinessEntity* object that records information like name of the business, points of contact such as the physical address, telephone number, e-mail and fax number, the URL of the company web site, and so on (for a more complete representation of BusinessEntities see figure 5). A BusinessEntity is associated with one or more *Business Services* that are descriptions of the specific services that a business provides. In turn a Business Service is associated with one or more *BindingTemplates* that specify the service access end point. UDDI supports the specification of a wide range of binding specification that range from HTTP, to mail, fax and others.

The description of businesses and services provided above supports the description of the basic information of services: how they are named, who to contact to gain information, how to invoke them. But it does not provide any indication of what type of service has been defined or additional technical details about the service. In addition to the description of businesses, services and binding templates, UDDI provides a data structure called *TModel* that allows the specification of additional attributes of the entities described in the UDDI repository. An example of TModels is provided in figure 3; the example shows the different fields of a TModel: *Name* is an identifier of the TModel, a unique *Key* assigned to the TModel in form a UUID [1], an text *Description* of a TModel, an *Overview URL* where a more detailed text description of the TModel can be found; finally, the *Technical Model Locators* specify information that is used within UDDI to control the use of the TModel.

TModels define attributes that can be used to specify information about the services. Figure 4 shows how the NAICS TModel may be used to specify the classification of a Commercial Banking service. The *KeyName* in the figure is

```
Technical Model Information

  Name:          ntis-gov:naics:1997
  Key:           UUSID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2
  Technical Model Description(s):
                 Business Taxonomy: NAICS (1997 Release)
                 This tModel defines the NAICS industry taxonomy.
  Overview URL:  http://www.uddi.org/taxonomies/Core_Taxonomy_OverviewDoc.htm#NAICS
  Technical Model Locator(s):
    Code:        categorization
    Description: types
    Type:        UDDITYPE
```

Fig. 3. TModel for the NAICS taxonomy

a name of the attribute, the *KeyValue* is the value is the code of Commercial Banking services within the NAICS classification, and *TModelKey* is the key of the TModel used, in this case the TModel defined in figure 3.

```
KeyedReference
  KeyName=    NAICS code
  KeyValue=   52211
  TModelKey=  UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2
```

Fig. 4. TModel for a Commercial Banking Service

UDDI supports two types of TModels: the first type are TModels that express technical specifications of the services such as the protocols that they adhere to or interchange formats such as the RosettaNet Partner Interface Processes specification [8]. The second type of TModels express abstract specifications about the service within predefined classification and taxonomic schemes, as the example of the NAICS TModel above. In this case, a service can specify its position within the general classification scheme.

UDDI allows a wide range of searches of the registry: services can be searched by name, by location, by business, by bindings or by TModels. For example it is possible to look for all the services that have a WSDL representation, or for all the services that adhere to the specification of the RosettaNet. Unfortunately, the search mechanism supported by UDDI is limited to keyword matches and UDDI does not support any inference based on the taxonomies referred to by the TModels. For example a banking service may describe itself as "Commercial Banking" which is a valid entry in NAICS, but any search based for "Depository Credit Intermediation" services will not identify the banking service despite the

fact that "Commercial Banking" is a subtype of "Depository Credit Intermediation."

3.2 Compiling DAML-S Profiles into UDDI Representations

The mapping of DAML-S profiles into UDDI Representations is shown in figure 5. The figure shows that some information can be mapped directly from DAML-S Profiles to UDDI records. This is the case with provenance information like the name and address of the service provider. DAML-S specific attributes such as *inputs*, *outputs*, *geographicRadius* and so on are instead represented using the TModel mechanisms described above.

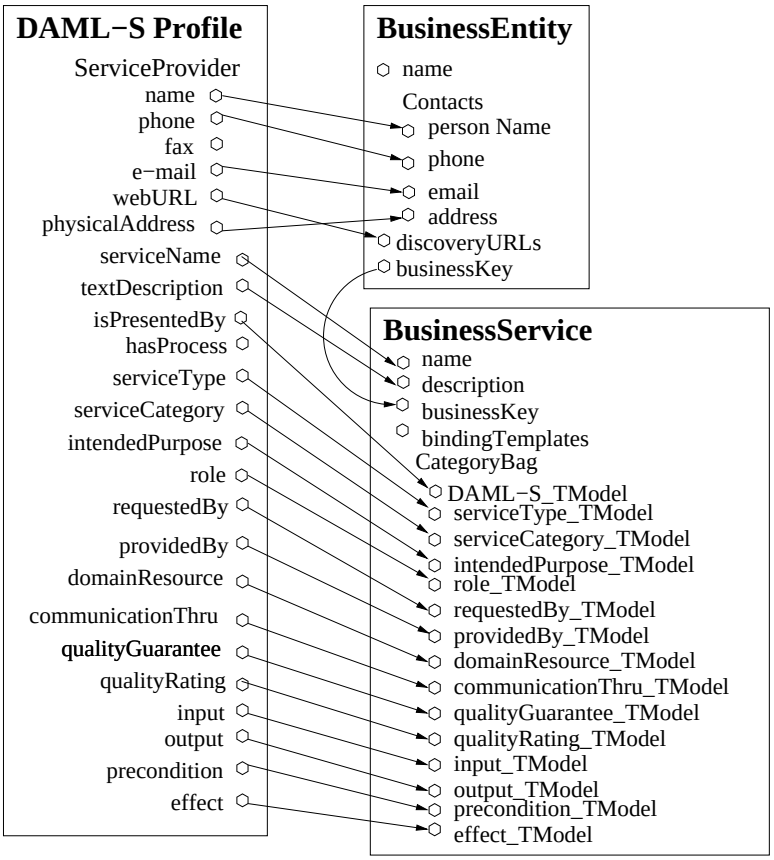

Fig. 5. UDDI Service Representation

Following figure 5, the description of the provider of the service is mapped into an instance of the UDDI BusinessEntity that is used as a representation of

the Business that deliver the service. If a business with the same information is already available in the registry that business is reused and referenced by the business service description, otherwise a new business is created. The latter case is actually the most common, since in order to publish a service with UDDI, a business has to declare itself, therefore all the provenance information is already available to UDDI.

The mapping of the other attributes requires the specification of a set of 15 UDDI TModels, one for each attribute of the DAML-S Profile representation. BusinessService records use these TModels to index the values they store from the DAML-S Profile they intend to represent. As an example consider the case of a stock quote reporting service that takes as input a ticker symbol and returns as output the current quote. The representation of the inputs and outputs of such a service is shown in figure 6. One of the TModels, the *DAML-S TModel*,has a special meaning: it states that the service advertised has a DAML-S service representation, and its value is the URI of the DAML-S service that is presented by the current Profile.

```
CategoryBag
  KeyedReference
    KeyName=    Input
    KeyValue=   financialOntology:ticker
    TModelKey=  ''UUID of the DAML-S Input TModel''
  KeyedReference
    KeyName=    Output
    KeyValue=   financialOntology:quote
    TModelKey=  ''UUID of the DAML-S Output TModel''
```

Fig. 6. TModel for a Commercial Banking Service

Figure 5 also shows that two UDDI features are left unchecked: the first one is the *name* of the business, the second one is the *bindingTemplates* of the service. The business name has already been provided to UDDI at the time of publishing the Profile because, as pointed out above, UDDI forces a business to declare itself in order to publish the services it provides. The bindingTemplates represent information that can be gathered from the Service Grounding module of DAML-S, but the details of that module have not been specified yet.

One advantage of the mapping described here is that it is completely embedded in UDDI. Indeed, the description of the service can be augmented by using additional UDDI specific TModels such as the NAICS TModel described above in figure 3. Furthermore, all the search functionalities provided by UDDI can be used to retrieve information about services that are represented as DAML-S services.

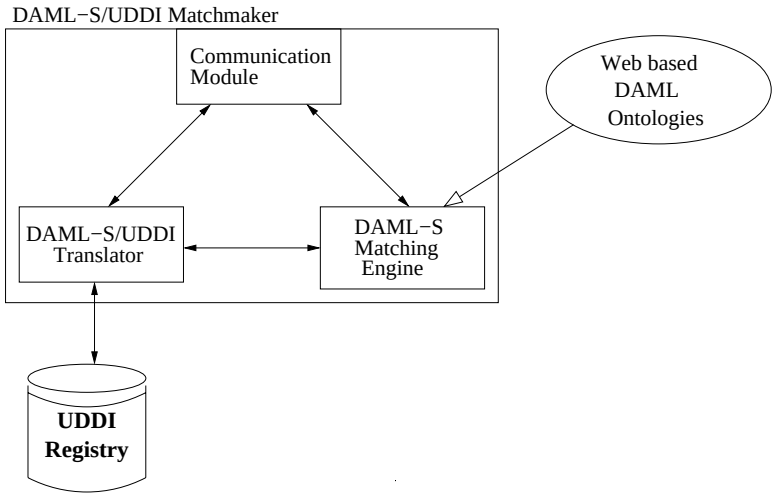

Fig. 7. The architecture of the DAML-S/UDDI Matchmaker

4 Adding semantic matching capabilities to UDDI

We implemented a matching engine that can be used to augment UDDI registries[3] with an additional semantic layer that performs capability matching between service records. The matching engine that we implemented is based on the algorithm described in [7] that uses DAML ontologies published on the web to compare attributes of the Service Profiles, with particular regard to Inputs, Outputs, Preconditions and Effects. The result of this work is that services that advertise using DAML-S are also advertised with the UDDI registry, and therefore they can be found and retrieved by using UDDI keyword search. In addition, they can also be found through our capability matching engine.

The architecture of the combined DAML-S/UDDI Matchmaker is described in figure 7.The Matchmaker receives advertisements of services and requests for service in DAML-S format from outside through the *Communication Module*; upon recognizing that a message is an advertisement, the Communication Module sends it to the *DAML-S/UDDI Translator* that constructs a UDDI service description using information about the service provider, and the service name. The result of the registration with UDDI is a reference ID of the service. This ID combined with the capability description of the advertisement are sent to the DAML-S Matching Engine that stores the advertisement for capability matching. Requests follow the opposite direction: the Communicator Module sends them to the DAML-S Matchmaker that performs the capability matching. The result of the matching is the advertisement of the providers selected and a reference to

[3] We are currently using the IBM test site.

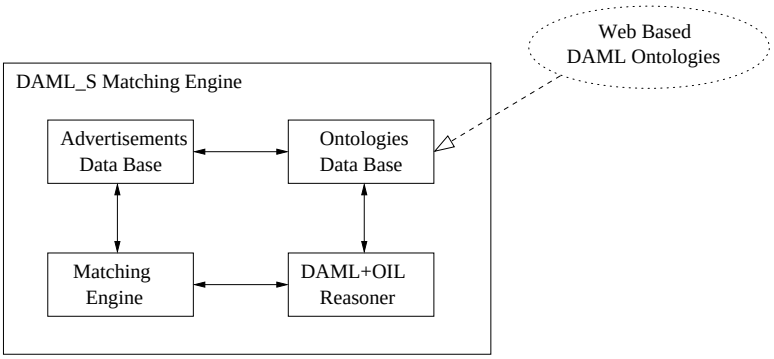

Fig. 8. The architecture of the DAML-S Matching Engine

the UDDI service record. The combination of UDDI records and advertisements is then send to the requester.

The actual DAML-S based matching engine architecture is displayed in figure 8. Upon receiving a request, the *Matching Engine* component selects the advertisements from the *AdvertisementDB* that are relevant for the current request. Then it uses the *DAML+OIL Reasoner* to compute the level of match. In turn the DAML+OIL Reasoner uses the *OntologyDB* to as data to use to compute the matching process. The AdvertisementDB also takes advantage of the *OntologiesDB* to index advertisements for fast retrieval at matching time.

5 Conclusion

This paper show how the limits of UDDI can be overcome by taking advantage of DAML-S and its support for functional descriptions of capabilities of and matching upon those descriptions. In its current form UDDI does not provide any support for finding service on the basis of what tasks they perform. It is impossible to ask UDDI for a "car selling service" because UDDI because such a request cannot even be expressed. On the opposite DAML-S offers the specification of what a service does and the possibility of expressing additional features of services such as the expected response time or constraints in the availability of the service. The mapping described in this paper supports the translation of DAML-S Service Profiles into UDDI descriptions of Business Services allowing therefore UDDI to express capabilities of services and laying the ground for capability matching functionalities in UDDI.

The mapping described in the paper has been fully implemented and it is currently being tested as a module of a DAML-S based capability Matchmaker that uses UDDI as its advertisements storage. Future work involves a complete integration between the Matching Engine and UDDI so that UDDI will be able to take full advantage of DAML and the semantic web.

References

1. ISO/IEC 11578:1996. Information technology – open systems interconnection – remote procedure call. http://www.iso.ch/, 2001.
2. US Census Bureau. North american industry classification system (naics). http://www.census.gov/epcd/www/naics.html, 1997.
3. Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.
4. DAML-S Coalition. Daml-s 0.6 draft release (december 2001). http://www.daml.org/services/daml-s/2001/10/, 2001.
5. DAML Joint Committee. Daml+oil (march 2001) language. http://www.daml.org/2001/03/daml+oil-index.html, 2001.
6. DAML-S Coalition:, A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web. Forthcoming in Proceedings of the First International Semantic Web Conference (ISWC01), 2002.
7. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *ISWC2002*, 2002.
8. RosettaNet. RosettaNet Web Site. http://www.rosettanet.org, 2000.
9. Katia Sycara and Mattheus Klusch. Brokering and matchmaking for coordination of agent societies: A survey. In A et al Omicini, editor, *Coordination of Internet Agents*. Springer, 2001.
10. Katia Sycara, Mattheus Klusch, Seth Widoff, and Janguo Lu. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record (Special Issue on Semantic Interoperability in Global Information Systems)*, 28(1):47–53, 1999.
11. UDDI. The UDDI Technical White Paper. http://www.uddi.org/, 2000.
12. W3C. Extensible markup language (xml) 1.0 (second edition). http://www.w3.org/TR/2000/REC-xml-20001006, 2000.
13. W3C. Soap version 1.2, w3c working draft 17 december 2001. http://www.w3.org/TR/2001/WD-soap12-part0-20011217/, 2001.