

Using WSDL/UDDI and DAML-S in Web Service Discovery

Thomi Pilioura¹, Aphrodite Tsalgatidou¹, Alexandros Batsakis²

¹ University of Athens, Dept. of Informatics, Panepistimiopolis, TYPA Buildings
Ilisia, 157 84, Athens, Greece
{thomi, afrodite}@di.uoa.gr

² Johns Hopkins University, Computer Science Department
3400 N. Charles Street, Baltimore, MD. 21218
abat@cs.jhu.edu

Abstract. Web services enhance current web functionality by altering its nature from document to service oriented. These services are self-describing, self-contained, modular applications accessible over Internet. They can be used by humans or programs in order to accomplish a particular task. To benefit from them, an efficient discovery mechanism for locating and selecting the appropriate web services is required. In contrast to common search engines, this mechanism should be based on required/offered service capabilities rather than on mere keywords. This paper presents the basic requirements for such a mechanism and evaluates current web service technology (WSDL, UDDI, DAML-S) that is used to address these requirements.

1 Introduction

The web service³ paradigm is transforming the Web from a provider of static pages to a provider of interactive, automated and intelligent services that interact via the Internet. Multiple web services will interoperate to perform tasks, provide information, transact business and generally take action for users, dynamically and on demand. The web service paradigm brings a number of advantages to application developers and end-users. The web service model simplifies business application development and interoperation, as it entails code reuse and loose coupling between services thanks to the adoption of widely accepted standards. Additionally, it may serve end-user needs by providing an intuitive, browser-based interface that enables users to choose, configure and assemble their own web services.

However, in order to employ its full potential, the web service paradigm must be supported by an appropriate discovery mechanism. The keyword-based techniques used in common search engines are not suitable as they are prone to low precision and recall. Many irrelevant services may include in their description the query keywords, leading to low precision. Also, the query keywords may be semantically equivalent but syntactically different from the words in the offered services, leading to reduced recall. The key underlying problem is that keywords are a poor way to capture the semantics of a service request or service advertisement. Thus a different mechanism is needed, one that entails locating web services on the basis of the capabilities they

³ The terms “web service”, “service” and “WS” are used interchangeably in the text.

provide. This consists of finding the services that provide the capabilities described in the service request.

The discovery mechanism should enable the location of services by both humans and machines. Humans can be either end-users looking for a service to use it as it is or developers who want to find a service at design-time and to incorporate it in their program. Machine-understandable services will be able to locate each other and interoperate.

There are several initiatives in the web service discovery area. The most prevalent is a combination of Web Services Description Language (WSDL) [2] and Universal Description, Discovery and Integration (UDDI) [3] standards. These are supported by major industry companies and are already implemented in many tools. Another initiative called DAML-S [9] comes from the research community and is based on the semantic web initiative [10].

The goal of this paper is to identify requirements related to service discovery and to assess WSDL/UDDI as well as DAML-S against these requirements. WSDL and UDDI can be easily tested as they are currently supported by many software tools. On the other hand, DAML-S is still immature and not supported by current tools. Thus, we decided to build a DAML-S matchmaker that is used as a case study for identifying problems regarding the adequacy and maturity of DAML-S in Web Service Discovery. The rest of the paper is structured as follows. In section 2, we present the requirements for service description and discovery. In section 3 we briefly describe and assess WSDL and UDDI against these requirements. In section 4, we present and assess DAML-S against the requirements presented in section 2. Finally, we give our conclusions and we outline our future work plans. A brief description of our DAML-S aware Matchmaker is given in the Appendix of the paper.

2 Requirements for Web Service Discovery

Discovery is one of the major challenges of the web service technology [11]. An effective and automated search and selection of relevant services is essential both for human users (developers or non-technical persons) and programs (such as software agents).

Service discovery is the process of finding an appropriate service provider for a service requestor through a service matchmaker. The basic steps of this process are (Fig. 1):

1. Providers describe their services (*Service Description*)
2. Brokers⁴ classify and publish these service descriptions (*Service Publishing*),
3. Requestors ask some broker if there are providers offering services with desired capabilities (*Description of Requestor's Needs*),

⁴ The terms “matchmaker” and “broker” are used interchangeably in the text

4. The broker matches the request against the stored service descriptions and returns a result, which is a subset of the stored descriptions (*Service Matchmaking*).

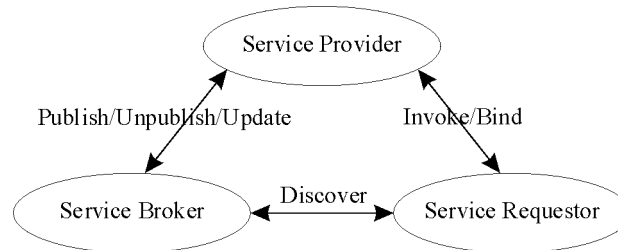


Fig. 1.The WS Interaction pattern

Then, service requestors can invoke services based upon the discovered service descriptions. This process must adhere to a number of requirements in order to be effective and efficient. In the following we identify a number of basic requirements for each of the four aforementioned steps of the service discovery process. Basic requirements need to be supported by any discovery mechanism that promises efficiency. There are also some value-added requirements that bring better performance to a discovery mechanism such as availability, scalability, billing, security and monitoring, but these are out of the scope of this paper.

Service Description⁵ Requirements:

The description of web service capabilities is essential for classifying, discovering and using a service. Some of the main desiderata of service description are presented in the following paragraphs. Thus, web service description:

- needs to contain functional (e.g. what a service does, sequencing of messages) as well as non-functional attributes of the service (e.g. service taxonomy, security, authentication and privacy issues related to the information exchange). An analytical list of non-functional service properties and their uses is given in [7]. Behavioural information (e.g. with what entities the service is interacting in order to produce the desired result, what the service states are etc.) is also required.
- needs to be understandable by humans as well as by machines. This means that each service attribute must be described at both syntactic and semantic level. Syntactic information is concerned with the implementation aspects of a service and thus tailored towards the programmers' requirements. Semantic information is concerned with the conceptual aspects of a service aiming to facilitate end-users by shielding off the lower level technical details, as well as to facilitate developers to find services that best match their needs and to enable automatic service selection and composition. Let us consider a stock quote service, which takes as input a string denoting the stock symbol and returns the stock quote as a number. The syntactic information denotes that the input parameter is a string and the output is a number, whereas semantic information conveys the real world meaning of the

⁵ The terms "service description" and "service advertisement" are used interchangeably in the text

string and the number in the context of stock quote markets. Depending on whether the service requestor is an end-user, a developer or a machine, different kinds of service description are required. For the end-user, only semantic description is needed whereas developers or machines need both semantic and syntactic information.

Thus, the language used for service description needs to provide constructs that enable the description of functional, non-functional and behavioural information in a semantic as well as in a syntactic form of representation. The description language should also support inferences on descriptions. This means that automated reasoning and comparison on descriptions should be possible and efficient. For example, when we require a Booking service we expect our request to be matched against anyone providing Hotel Booking services. One way to achieve this is by using ontologies [12]. Ontologies include computer-usable definitions of basic concepts in a domain and the relationships among them. Ontologies are used by people, databases, and applications that need to share domain information, where a domain is just a specific subject area, like medicine, real estate and financial management. There are several languages for representing ontologies such as Resource Description Framework (RDF) [13], DAML+OIL [14] and the newly proposed Web Ontology Language (OWL) [4]. These languages are very expressive and are based on a class system much like many object-oriented programming and modelling systems. Classes are organized in a hierarchy and offer extensibility through subclass refinement. This enables automated reasoning on taxonomies of concepts.

Service Publishing Requirements:

Publishing is one of the basic activities as it makes a service known and available to be used. The published service may be either a user-facing service targeting the business user or a technical service targeting the developer or a program.

The publishing of services needs to be performed either via a programmatic interface or via a web interface. Furthermore, given the increasing number of services that require to be published, automated publishing mechanisms are needed. These mechanisms will facilitate the building of a crawler to pull Web Service advertisements off the Web, without people having to push service advertisements to the matchmaker.

It is clear that, the effective publishing of services depends on effective categorization that in turn depends on effective information provided in service descriptions and on appropriate taxonomies built by matchmakers. Services may be registered in multiple categories, provided that they function according to the requirements of each of these categories. Categorization of services is not an easy task and depends on both the service provider and the matchmaker. The matchmaker is mainly responsible for the offered taxonomies, while the service provider is responsible for classifying the service into the appropriate taxonomy, unless the provider prefers to assign this task to the service matchmaker. Service matchmakers may compete on the merits of their choice of taxonomy, on the up-to-date accuracy of their listings and on auxiliary information, such as quality-of-service data, statistical information for the use of the service and comments/evaluation results by service users. The latter can be an important factor for building trust on a specific service.

Requirements for the Description of Requestor's Needs:

An important aspect of the discovery process is the description of requestor's needs. The language used for this description must adhere to the same requirements as the one describing the capabilities of the service.

Service Matchmaking Requirements:

The matchmaking process matches existing web service descriptions with requestor's needs. As it is aforementioned, the matching should not be based on keyword search only. Instead, semantic and syntactic information about each attribute in the service request and advertisement must be taken into consideration. This is essential, as equality of concept names (i.e. syntactic information) does not necessarily mean the equality of their semantics. Depending on whether the service requestor is an end-user, a developer or a machine, different information is used to describe requestor needs and different kinds of service description are taken into account. An end-user searches for the service by specifying only semantic information and once s/he finds the appropriate service s/he will use it as it is, without further elaboration. On the other hand a developer or a program are interested in both syntactic and semantic information. The matchmaking process should first examine semantic compatibility between service capabilities and requestor's needs and then syntactic compatibility. Semantic matching has to precede syntactic matching, as it is necessary to assure that both request and advertisement address the same subject area. Then an optimisation must take place in order to return the most highly rated matches and present them in an appropriate way depending on the user.

Matchmaking has to support both early (design time) and late (runtime) binding to web services. This means that both a programming and web interface are needed. In case of early binding, the matchmaker can be queried at design time in order to locate the appropriate service and the located service is then statically bound with the application being developed. In case of late binding, there is no "hard-wired" function call in the program code but instead a "syntactic and semantic description" of what kind of operation to use, which will be dissolved just in time before execution.

The matchmaking process should support both volatile and persistent queries. In case of a volatile query the matchmaker immediately returns matching advertisements that are currently present in the repository. On the other hand, the persistent query is a query that will remain valid for a predefined period. Within the validity period of the query, whenever an advertisement that matches the query is added or updated, the matchmaker will notify the requestor.

The matchmaking process must also support service composition. Service composition is an important issue since it is very likely that the offered services will not satisfy user needs and therefore the combination of basic web services (possibly offered by different companies) into value-added services is needed. In order to automate service composition, the service description must provide declarative specifications of the preconditions and effects of service use.

In the following section we assess the combined use of UDDI and WSDL in web service discovery. This is followed by a short description and assessment of DAML-

S (in section 4) that is proposed by many researchers [20] [21] [22] [23] as an approach that could help overcome the problems of WSDL and UDDI.

3 Using WSDL and UDDI in WS Discovery

In this approach the service is described using WSDL. WSDL is an XML grammar for specifying properties of a Web Service such as what it does, where it is located and how it is invoked, i.e., it describes only the functional and syntactic aspects of a service. WSDL does not support non-functional information of services. For example, it is not possible to indicate the geographic region that a weather service is provided for or the charge associated with the service. Furthermore, it does not provide behavioural information for a service. For this, another language such as Web Services Flow Language (WSFL) [17], XLANG [16] or Business Process Modelling Language (BPML) [15] must be used.

In this approach there is no formal way of expressing requestor's needs. Instead, a service requestor retrieves advertisements out of the UDDI registry based on keyword search (exact pattern matching) on some fields such as `name`, `taxonomy`, `tModel` or `identifier`.

The service publishing process is based on UDDI, which defines a directory for the publication and discovery of businesses and services. The UDDI XML schema defines six data structures: `businessEntity`, `businessService`, `bindingTemplate`, `tModel`, `publisherAssertion` and `operationalInfo`. The `businessEntity` structure describes information about businesses, including their name, description, services offered and contact information. The `businessService` structure provides more detail on each service being offered. Any kind of service can be registered in the UDDI, such as user-facing services and technical services. Each service can have multiple `bindingTemplates`, each describing a technical entry point for a service (e.g., `mailto`, `http`, `ftp`, etc.). `tModels` describe what particular specifications or standards a service uses. A `publisherAssertion` structure allows the declaration of relationships between business entities. Finally, the `operationalInfo` structure is used to convey the operational information of the other data structures. Such operational information includes the date and time that the data structure was created and modified, the identifier of the UDDI node at which the publish operation took place and the identity of the publisher. UDDI also provides identifiers and categories to mark businesses, services and service types using various standard taxonomies (related industry, products or services offered and geographical region).

The service publishing can be performed either through the web interface or via the UDDI Publishing API. Furthermore, UDDI can be complemented by the Web Services Inspection Language (WSIL) [18]. A WSIL document is essentially an aggregation of pointers to service description documents. This means that it defines the locations on the service provider's web site where one could look for web service descriptions. Thus WS-Inspection specification could facilitate in the future the

building of a crawler (analogous to those currently used by keyword-based search engines) to pull Web Service advertisements off the Web, without people having to push advertisements for their services to the UDDI registry.

As already mentioned, the matchmaking process is based on keyword search on some fields such as `name`, `identifier` or `taxonomy`. The latter is the only field conveying semantic information as it enables users to search the registry by industry, product category or geographic location. However, this is not enough for achieving automated discovery as two identical service registrations in the UDDI could mean totally different things, depending on the context in which they are used. Thus UDDI only provides a first level filter in the discovery process. Further discrimination is done by manual inspection of the service descriptions.

An inquiry for a service can be performed either at design time via a web-based user interface or at runtime via the UDDI Inquiry API. At design time, the UDDI registry can be searched by a programmer for suitable services and can be used to locate the appropriate WSDL file. After the programmer has studied the specifications for the Web Service described in the retrieved WSDL documents, s/he generates client proxies so that the application can access the service. Alternatively development tools can be used to generate the required client proxies. At run time, the application does not have to use UDDI if the target web service is always available and always the same. However, if the web service becomes unavailable, the application can query the UDDI registry to determine if the service has been moved to another location. This capability is useful when service providers have to re-host their services. Furthermore, UDDI enables the selection of the appropriate service at run time. For example, in the case of a financial portal providing a “News” service among others, all stock market agencies offering a “News” Web Service could be located and queried by the portal application at runtime to find the one that provides the latest financial news.

UDDI supports both volatile and persistent queries. Persistent queries are possible through the subscription API introduced in UDDI version 3. This API enables users to establish a subscription based on a specific query or on a set of entities (businesses, services, etc.) that the user is interested in. In the case of a query-based subscription, if the result set changes within a given time span, the user is notified. In the case of entity-based subscription, the user is notified whenever the contents of one of those entities change.

Some of the use cases enabled by subscription include notification of new businesses or services that are registered; monitoring of existing businesses or services; obtaining registry data for use in a private registry; and obtaining data for use in a marketplace or portal registry.

As far as service composition is concerned, there is a Technical Note [6] that describes how UDDI, WSDL, and Web Services Conversation Language (WSCL) [19] can be used to create an environment in which services can spontaneously discover each other and then engage in complicated interactions.

4 Using DAML-S in WS Discovery

DAML-S is a DAML+OIL ontology for describing the aim and usage of a web service. DAML-S describes what a service can do and not just how it does it. It provides three essential types of knowledge about a service:

ServiceProfile: it defines "what the service does"; that is, it gives the type of information needed by a service requestor to determine whether the service meets its needs. Service profiles consist of three types of information:

- the provider information that consists of contact information about the entity which provides or requests a service.
- the functional description of the service that is expressed in terms of the information and state transformation produced by the service. The information transformation is represented by input and output properties. The input property specifies the information that the service requires to proceed with the computation. The output property specifies what is the result of the operation of the service. For example a stock quote service would advertise itself as a service that, given a stock symbol, will return the stock quote. The state transformation produced by the execution of the service is specified through the precondition and effect properties of the profile. Precondition presents logical conditions that should be satisfied prior to the service being requested. Effects are the results of the successful service execution.
- a number of features that specify non-functional characteristics of the service (such as what guarantees of response time or accuracy it provides, or the cost of the service). These features assist when reasoning about several services with similar capabilities.

ServiceModel: it defines "how the service works"; that is, it describes the workflow and possible execution paths of the service. For non-trivial services (those composed of several steps), this description may be used by a service-seeking agent in at least four different ways: (1) to perform a more in-depth analysis of whether the service meets its needs; (2) to compose service descriptions from multiple services to perform a specific task; (3) during the course of the service enactment, to coordinate the activities of the different participants; (4) to monitor the execution of the service.

ServiceGrounding: it specifies the details of how to access a service. Typically service grounding will specify a communications protocol (e.g., RPC, HTTP GET/POST, CORBA IDL, SOAP, Java RMI), and service-specific details such as port numbers used in contacting the service.

In the remaining of this section, we focus on assessing DAML-S against the requirements presented in section 2. This assessment was partially based on a DAML-S aware Matchmaker which we have implemented for this purpose. This matchmaker is presented briefly in the appendix of this paper. A number of challenges that affected both the usefulness and the efficiency of the matching

algorithm were encountered. The sources of these technical issues are mainly located in the specification of the language and emerge in the following paragraphs.

It is obvious that the DAML-S approach supports discovery requirements at a higher level than WSDL/UDDI. Through the tight connection with DAML+OIL, DAML-S supports the need for syntactic and semantic representation of services. DAML-S classes may draw properties from other DAML-S classes through inheritance and other relationships. Thus, DAML-S provides a richer representation of an individual service and of the relationships between services. However, there is a problem that derives from the generality of the descriptions. Service providers are allowed to describe their services too vaguely in order to improve their relative position among the search results. A service, advertising itself as capable of doing almost everything (e.g. takes as input an object and returns an object), without being accurate and honest, acts as a source of mislead and degrades the usefulness of the search by spamming the results with junk false positive hits. Thus, special attention must be given to the design of the matchmaking algorithm used by the DAML-S aware matchmaker. The matchmaking algorithm we have implemented partially addresses this problem by calculating the matching degree between inputs (outputs) on the basis of the number of hierarchy levels that intervene between the input (output) of the advertisement and the input (output) of the request.

The behaviour of a service can be represented in DAML-S by using the `ServiceProfile` and the `ServiceModel`. Both of them describe the inputs, outputs, preconditions and effects of a service although from a different perspective: the `ServiceProfile` for enabling the discovery of the service and the `ServiceModel` for mainly controlling the interaction with the service. The two chief components of the `ServiceModel` are the Process Ontology and the Process Control Ontology. The Process Ontology describes a service in terms of its inputs, outputs, preconditions, effects, and, where appropriate, its component subprocesses. The Process Control Ontology describes each process in terms of its state, including initial activation, execution, and completion. A version of the Process Ontology is released in the current version of DAML-S (version 0.7) and can be used to support automated Web Service invocation, composition and interoperation. The Process Control Ontology, which is useful for automated execution monitoring, has not yet been released. DAML-S' strength of defining a service as a process is very important for web service composition because it enables higher-level reasoning about how services may be aggregated to achieve a particular goal. However, the `ServiceModel` is still very immature as only the Process Ontology is defined.

Besides the maturity problem, a very important security issue originates from the possible inconsistency between the `ServiceProfile` and the `ServiceModel`. DAML-S allows the `ServiceProfile` to describe the service in a totally different way than the `ServiceModel` and consequently than the actual service behaviour. As a result, a malicious provider could advertise its service insidiously in order to take advantage of prospecting users. Imagine, a `ServiceProfile` document claiming to take as input a credit card number and return the cardholders name. There is absolutely no guarantee that the service will use the credit card number in the described way, without e.g. charging the credit card. This lack of consistency control

requires that the service requestor disposes a security mechanism which guarantees consistency between `ServiceModel` and `ServiceProfile`.

The `ServiceProfile` is used by both service providers and service requestors to describe respectively their services and their needs. For instance, a provider might advertise a service that provides quotes for a given ticker symbol, whereas a requestor may look for a service that reports current market prices and stock quotes. Thus, DAML-S enables the description of requestor's needs. However, as it is difficult to compose the DAML-S description of a requested service, special assistance provided by tools is needed for both developers and non-technical persons. These tools will enable the definition of all three aspects of a DAML-S description, i.e. `ServiceProfile`, `ServiceModel` and `ServiceGrounding`. Tools should also help requestors to express needs for composite services.

When the discovery is performed by a developer or a program that also needs technical information (contained in the `ServiceGrounding`) and behavioural information about the service (expressed in the `ServiceModel`), then it is clear that the information contained in the `ServiceProfile` is not enough. As already stated, a DAML-S `ServiceProfile` describes a service as a function of three basic types of information: provider information, functional description and non-functional service attributes. The functional description describes what the functions of the service are, e.g. what the input arguments are and what the service returns as output. It is merely a summary of the grounding and process model, divided in four segments: input parameters, output parameters, preconditions and effects.

Input and output parameters are used in our matchmaker for semantically comparing the service request and the advertisement. The search results do not depend on the parameter name, which is only a symbolic name hopefully helping the reader to guess its actual use. Instead they are based on subsumption reasoning and this is the great value of DAML-S. However, this necessitates that the inputs/outputs are accompanied with semantic information. This information is described by a resource pointing to an element of ontology. Unfortunately, the current state of ontologies is in total lawlessness, by having much different ontologies to describe the exact same thing.

The comparison of preconditions and effects fields is not applicable. Both of those fields are totally dependant on the rule representation. However, the current release of DAML-S does not support rule representation. Therefore, the representation of the preconditions and effects fields is carried by using a generic DAML-S construct (`daml:thing` resource). This means that both preconditions and effects could represent anything, vigorously refusing to follow any formalism. Consequently, the matching engine is not able to comprehend and compare these values.

The third part of the `ServiceProfile` contains a number of non-functional service attributes. These attributes are described using three classes: `ServiceCategory`, `QualityRating` and `ServiceParameter`.

- `ServiceCategory`: it is used to specify how the service is classified within a taxonomic scheme. This field can be used as a first filter in the matchmaker for

determining whether the service request and advertisement refer to the same service category. However, such a comparison may not produce safe conclusions, as it is difficult to constrain a service to belong to a single service category. And even if this is achieved, it must also be possible to define queries using query mutation operators in order to potentially retrieve services belonging to different service categories.

- **QualityRating:** this is a very important issue in the selection of services. Although the comparison of the corresponding field in the requested/offered services appears easy, it is nevertheless tricky. The first problem stems from the difficulty of establishing some kind of authority, which will objectively rate the vast, and constantly changing set of services according to their quality. The second problem derives from the fact that this field can take one of the available values such as Excellent, Poor etc. but there is no place where the relation among the various values is denoted. It is therefore impossible for the matchmaker to understand that the `qualityRating_Excellent` is better than `qualityRating_Poor`.
- **ServiceParameter:** there are numerous properties that can be associated with a service. This class provides an extension mechanism for defining new parameters. These parameters are arbitrary at the moment but hopefully ontologies of parameters will be developed in the future.

From what we analysed above, it becomes obvious that it is really difficult for a matching engine to make safe conclusions about the compatibility of services based on these non-functional attributes. The lack of formalism, the intense dependency on the ontology type and content and the non-deterministic behaviour exclude inevitably this part of the `ServiceProfile` from the matchmaking process. The functional attributes might be examined in a higher level by a human inspector or by an automated agent with loose semantics.

In all, this paper ranks the DAML-S open issues into two categories:

- *Issues related to the languages specification:*

The lack of rules and the failing to preserve consistency among different DAML-S sections are some examples in this category. This could be addressed by coupling DAML-S with RuleML [8]. RuleML can describe constraints related to input and output, and also preconditions and effects for planning. Currently, a DAML-S working group is trying to specify rules in DAML, but no proposal has been put forward.

- *Issues inherited from the DAML-S mental ancestor, namely the semantic web:*

It is extremely important for every reference to web resources or objects to use the same ontology. If every service description used its own private ontology, probably it would lead to a more precise description of the service, but the aggregation and composition of more than one service would be practically impossible, since every member would use an “incompatible” private language. One obvious solution is to limit the allowed ontologies, but this would generate a series of critical cascading problems, questioning the value of the whole idea of the semantic web. Who (and

how) would decide and guarantee the correctness of ontology? Seeing that human knowledge is infinite, it seems almost impossible to publish all of this knowledge to a set of documents.

As far as the requirements related to publishing and matchmaking are concerned, it depends on the implementation of each matchmaker. Concluding we can say that DAML-S enables the creation of an efficient discovery mechanism but there is considerable work to be done to enjoy its full potential.

5 Conclusions and Future Work

Web service discovery is an important aspect in web service oriented technology. The discovery mechanism must adhere to a number of requirements in order to be efficient. These requirements are not supported by currently available industry standards such as UDDI and WSDL. The semantic web initiative at W3C is gaining momentum and generating technologies (such as DAML-S) and tools that may help bridge the gap between the current standard solutions and the requirements for advanced web service discovery. However, DAML-S is still in its infancy and a lot of work has to be done in order to overcome its limitations and problems.

The work reported here outlines our first attempt to deal with issues related to web service discovery. An important part of our follow-up work is the investigation of ways for overcoming the problems identified in the previous section. We would also like to extend our matchmaker in order to address all the requirements presented in section 2 as well as several value-added requirements such as availability and scalability.

The merging of the UDDI/WSDL and DAML-S activities is another important issue. First attempts towards this direction are manifested by the use of WSDL in *ServiceGrounding* and the effort described in [20]. Our future work will focus on this issue too.

Acknowledgement

This work has been partially supported by the Special Account of Research Grants of the National and Kapodistrian University of Athens (ELKE) under contract 70/3/5362

References

1. Bernstein, A., Klein, M.: Discovering Services: Towards High Precision Service Retrieval. In: Proceedings of the CaiSE workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications. Toronto, Canada, 2002

2. Web Services Description Language. <http://www.w3.org/TR/2002/WD-wsdl12-20020709/>
3. Universal Description, Discovery and Integration. <http://www.uddi.org>.
4. Web Ontology Language (OWL) Guide. <http://www.w3.org/TR/2002/WD-owl-guide-20021104/>
5. DAML-S 0.7, <http://www.daml.org/services/DAML-S/0.7/>
6. Beringer, D., Kuno, H., Lemon, M.: Using WSCL in an UDDI Registry 1.02. http://uddi.org/pubs/wscl_TN_forUDDI_5_16_011.pdf
7. O'Sullivan, J., Edmond, D. ter Hofstede, A: What's in a service? Towards accurate description of non-functional service properties. In: International Journal of Distributed and Parallel Databases, Special Issue on E-Services, 12(2), Kluwer, Sep 2002
8. The Rule Markup Initiative. <http://www.dfki.uni-kl.de/ruleml/>
9. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K.: DAML-S: Web Service Description for the Semantic Web. In: Proceedings of the 1st International Semantic Web Conference (ISWC), 2002
10. Berners-Lee, T., Hendler J., Lassila, O. The Semantic Web. Scientific American, 284(5), 2001, 34-43
11. Tsalgatidou, A., Pilioura, T.: An Overview of Standards and Related Technology in Web Services. In: International Journal of Distributed and Parallel Databases, Special Issue on E-Services, 12(2), Kluwer, Sep 2002, 135-162
12. Uschold, M., Gruninger, M.: Ontologies: Principles, Methods and Applications. In: The Knowledge Engineering Review, 11(2), 1996, 93-136
13. Resource Description Framework. <http://www.w3.org/RDF/>
14. DAML+OIL. <http://www.daml.org/2000/12/daml+oil-index>
15. Business Process Modeling Language. <http://www.bpmi.org>
16. XLANG. <http://xml.coverpages.org/XLANG-C-200106.html>
17. Web Services Flow Language. <http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>
18. Web Services Inspection Language. <http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html>
19. Web Services Conversation Language. <http://www.w3.org/TR/wscl10/>
20. Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: Importing the Semantic Web in UDDI. In: Proceedings of Web Services, E-business and Semantic Web Workshop (WES), 2002
21. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Proceedings of the 1st International Semantic Web Conference (ISWC), 2002
22. Sollazzo, T., Handschuh, S., Staab, S., Frank, M., Stojanovic, N.: Semantic Web Service Architecture - Evolving Web Service Standards towards the Semantic Web. In:

Proceedings of the 15th International FLAIRS Conference. Pensacola, Florida, May 16-18, 2002 AAAI Press

23. Dogac, A., Cingil, I., Laleci, G., Kabak, Y.: Improving the Functionality of UDDI Registries through Web Service Semantics. In Proceedings of Technologies for E-Services, Third International Workshop, Hong Kong, China, August 23-24, 9-18
24. Batsakis, A.: Semantic Description and Discovery of Web Services, Diploma Dissertation, University of Athens, July 2002

APPENDIX: A DAML-S aware Matchmaker

DAML-S does not include the role of a matchmaker into its schemes. Therefore, we have implemented a `DAML-S aware Matchmaker` that is used as a case study for evaluating DAML-S in Web Service Discovery. The CMU group of the DAML-S Coalition has also developed such a matchmaker [21]. Our goal is not to develop a more sophisticated matchmaker, but to use our matchmaker for assessing DAML-S.

Ideally all three parts of DAML-S must be used for effective service discovery by developers and programs. However, when the discovery is performed by end-users the information of the `ServiceProfile` is enough. In any case, a matchmaking algorithm that takes into account all three parts of DAML-S is infeasible at the moment due to the incompleteness of DAML-S specification. Therefore, our `Matchmaker` uses only the `DAML-S ServiceProfile`. Service providers use the `ServiceProfile` to advertise their services, while service requestors use the profile to specify what service they need and what they expect from such a service.

The `Matchmaker` is a web service itself that performs two basic activities: service publishing and service matchmaking (implemented as two different web services). The service matchmaking web service implements a matchmaking algorithm that takes as parameters the web service advertisement and the request, it parses and compares the DAML-S documents, and returns the matching degree. The service supports two types of queries: simple queries and persistent queries. Persistent queries remain valid for a predefined period; within the validity period of the query, the `Matchmaker` notifies the requestor whenever an advertisement that matches the query is added or updated. The service matchmaking web service is composed by two other web services:

- The `Parser` that is responsible for representing a DAML-S profile document in the computer memory. It parses the DAML-S documents and creates objects that represent the corresponding parts of the DAML-S document.
- The `Comparer`, which takes as parameters two `ProfileDocument` objects (one representing the request and the other the advertisement) and returns the degree of matching.

Our matchmaker, although at preliminary stages, provided useful feedback for an initial assessment of DAML-S. More details about the design and implementation of the `Matchmaker` may be found in [24].