Discovery in Grid and Web Services Environments: A Survey and Evaluation

Ioan Toma∗  Kashif Iqbal  Dumitru Roman  Thomas Strang
Dieter Fensel  Brahmananda Sapkota
Matthew Moran

Digital Enterprise Research Institute (DERI),
Galway, Ireland and Innsbruck, Austria.

firstname.lastname@deri.org

July 30, 2006

Abstract

An important step in realizing the vision of both Web services and Grid is the provision of a robust, automatic and reliable solution for finding services or resources in such environments. Different solutions to this problem were already proposed, each with its specific model and realization. Although all solutions address the same problem, it is very difficult for a non-expert and even for an expert in the field to decide if one solution is better than another and why. This paper proposes a systematic set of criteria, a framework, that can help in the evaluation of different discovery approaches. Furthermore some of the most relevant discovery approaches in Web services and Grid environments are surveyed and evaluated according to the proposed evaluation framework.

Keywords: Discovery; Web services; Grid; Matchmaking; Evaluation Framework.

1 Introduction

Service oriented architectures (SOAs) are quickly becoming the de-facto solutions for providing end-to-end enterprise connectivity. They promote a service view on the world in which functionalities provided as services by different enterprises could be assembled and re-used in a standardized manner. The SOA principles are implemented by a set of technologies. Two of them, Web service and Grid computing, have nowadays a great impact in industry and academy. Grid computing aims to provide the computational power and data management infrastructure necessary to support the collaboration of people, together with data, tools and computational resources [17]. The most common problems that Grid addresses are computationally hard and
data intensive problems in science and engineering. Grids offer a solution to these problems by joining geographically distributed computational and data resources. These resources are delivered to heterogeneous user communities. The Grid provides the protocols, services and software development kits needed to enable flexible, controlled resource sharing on a large scale. Sharing in Grid is, necessarily, highly controlled with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.

Web services, on the other hand, try to solve the problem of enterprise application integration and automation of business processes by making use of the Web as a global infrastructure for distributed computation [1]. They offer a new level of automation in eWork and eCommerce, where fully open and flexible cooperation can be achieved, on-the-fly, with low programming costs [15]. To realize this, a suite of standards like UDDI [5], WSDL [10], and SOAP [34] are in common use. In this way software applications can be accessed and executed via the Web based on the idea of Web services.

One important problem common to both domains mentioned above is how to find an item (resource, service) that can fulfill the requested functionality. An automated solution to this problem known as discovery would be a big step forward in Web services and Grid domains.

Recent efforts in Grid computing [11] treat resources as Web services and for this reason discovery of resources becomes discovery of Web services in the latest Grid systems. Resources are basically wrapped by a Web service. Having this model in mind an evaluation of discovery approaches for Web services and Grid environments becomes similar.

The goal of this paper is to provide a set of criteria that can be used to evaluate the current effort in discovery of resources and services. The paper is structured as follows: Section 2 provides the terminology used in this paper. Section 3 defines a framework used in the evaluation of discovery approaches. Section 4 surveys some of the most relevant discovery approaches from both Web services and Grid areas and evaluates them based on the evaluation framework previously defined. Section 5 summarizes the evaluated approaches and abstracts the common best-practices from the various approaches. Finally, Section 6 presents the related work and concludes the paper.

2 Terminology

In order to evaluate a process or a mechanism, one has to have a clear understanding about the settings, the entities manipulated and the functionality of the process or mechanism. Some questions automatically pop-up when trying to define such an evaluation mechanism for service and resource discovery in Grid and Web services environments.

What is Grid? There are many interpretations of the term Grid [20, 19]. This paper does not provide a new definition of the term Grid, but instead it uses one of the most common adopted and used definition provided in Open Grid Services Architecture [18]. According to [18], the Grid provides the protocols, services and software development kits needed to enable flexible, controlled resource sharing on a large scale within a Virtual Organization. A Virtual Organization is a dynamic collection of individuals, institutions and resources bundled together in order to share resources as they tackle common goals.

What is a Service? The term service has been lately semantically overloaded [27]. Communities like: business science, information science and computer science have a different understanding of what a service is [3]. In this paper the notion of service is understood as defined in the conceptual
model architecture for semantic Web services [27]. According to [27] a service is a provision of value in some domain (not necessarily monetary value).

**What is a Web service?** Web services have emerged as a promising technology that offers a standard way to access and integrate functionalities. They are loosely coupled software components published, located and invoked across the Web. For the Web service concept definition is again adopted from [27]. A Web service is defined as a computational entity accessible over the Internet (using Web service Standards and Protocols)

**What is a Resource?** Resources are important concepts in a Grid environment. They form the low level entities that are accessed and used to fulfill a user request. Different resources can have the same functional capabilities but they may have different access policies associated, different time access, etc. As was pointed out before, in the latest approaches in Grid [11] resources are treated as Web services and for this reason discovery of resources becomes discovery of Web services in the latest Grid systems.

**What is Discovery?** Discovery is an important process in both Grid and Web services environments. Very briefly it can be described as the process that takes as input a user request and returns a list of resources or services that can possibly fulfill the given request.

### 3 Evaluation Framework

With the terminology defined previously in mind we propose an evaluation framework for discovery approaches in Grid and Web services environments. This framework is then used in next sections to evaluate some of the most prominent approaches from Grid and Web services areas.

Our framework contains a set of criteria that we believe are the most relevant when performing evaluation of discovery approaches. We list and explain each of the criteria below:

- **Query Language and Advertising Language**
  Language support for query formulation as well as service and resource description is crucial for the discovery process. It is important to evaluate how expressive is the language and how easy it is to formulate queries and advertisements which are used during the discovery process. Special attention should be given to the semantic support that the language provides to express different aspects of advertisements and queries.

- **Scalability**
  Scalability is an important issue when talking about systems that have to accommodate changes in users, resources, etc. It is important to analyze how a system reacts when there is a grow in one or more of its dimensions. Another aspect that must be considered is how the scalability of the subsystems, or related systems, affects the scalability of the whole system. The scalability of a discovery mechanism, is for example heavily influenced by the scalability of the sub-components such as: query processor, storage, etc.

- **Reasoning support**
  The provision of an automatic discovery mechanism can be significantly improved if machine processable descriptions are provided. These descriptions are further checked if they matched against each other (request versus services). New knowledge can be inferred based on existing facts (domain ontologies and domain background knowledge) and this knowledge can be further used during the discovery process. Therefore it is important to
identify the reasoning support provided to enable the discovery process. Of course, this criterion is very closely related to Query Language and Advertising Language mentioned above.

- **Matchmaking vs. Brokering**
  When talking about the interactions between parties during the process of discovery two approaches can be distinguished (See [30]):

  1. **Matchmaking:**
     The entity which performs the matching determines if the request and the advertisements match but does not interfere in the next step which is the interaction between matched request and advertisements.

  2. **Brokering:**
     The entity which performs the matching determines if the request and the advertisements match and furthermore controls the interaction between matched request and advertisements.

Many of Grid approaches for discovery distinguish between these two facets of the discovery process. Therefore it is important to evaluate an approach based on the support that it provides for pure matchmaking or brokering or both.

- **Mediation Support**
  Mediation provides an intermediary services, linking both data resources and application programs. Due to the heterogeneity of the environments (different data models, different protocols, etc.) where discovery has to be performed a mediation support is required. Therefore we add mediation support as another criteria to the evaluation framework.

4 **Survey and Evaluation**

We use the framework defined above to evaluate some of the most prominent approaches in Grid and Web services environments with respect to discovery. Due to space limitations we consider for evaluation only the discovery mechanisms provided by the following initiatives: Unicore [29], Globus [16], Ontology Based Matchmaker [31], Web Service Modeling Ontology [28], [22], OWL-S approaches [20].

**UNICORE** ([Uniform Interface to C)Ommputer RE)sources) [29] is a Java based environment for secure and seamless access to remote supercomputers. The design goals for UNICORE include an open architecture based on the concept of an abstract job, consistent security architecture, minimal interference with local administrative procedures, exploitation of existing and emerging technologies, a zero administration user interface through a standard Web browser and Java applets. The UNICORE client enables the user to create, submit, and control jobs from any workstation or personal computer using the Internet. Third-party components, such as Globus [16], can be integrated into the UNICORE framework to extend its functionality. Resource discovery in UNICORE Grid architecture is accomplished by a GRIP Resource Broker component [8] capable of locating resources on both UNICORE and Globus Toolkit based Grids, developed. The broker take workflows described in the UNICORE AJO framework and brokered the sites on EUROGRID[3]. UNICORE works with a conceptual representation of the job called Abstract Job Object, shortly AJO, which can be carried out on various computational and data services at collaborating sites.

- **Query Language and Advertising Lan-\[1\]http://www.eurogrid.org


The AJO is the basis for the platform and site neutral specification of requests for computational, data, and software resources. The object-oriented structure and syntax of the AJO make specification largely independent of hardware architecture, system software interfaces, and site-specific operational rules.

**Scalability**
There is a centralized broker which can be decentralized with the support of other brokers. The brokers follow hierarchical organization.

**Reasoning support**
No reasoning support for discovery is considered in UNICORE.

**Matchmaking vs. Brokering**
The EUROGRID/GRIP Resource Broker takes the workflows described in the UNICORE AJO framework and brokered the sites on EUROGRID to received offers from those sites which could enact the workflow and provide mechanisms for these sites to return tickets describing the quality of service policy they would offer.

**Mediation Support**
No mediation support for discovery is considered in UNICORE.

GLOBUS [16] provides software infrastructure for resource management of autonomous distributed systems with provisions for policy extensibility and co-allocation. A central element of the Globus system is the Globus Toolkit, which defines the basic services and capabilities required to construct Computational, Data or Service Grids.

The basic services implemented by key components include: Grid Security Infrastructure (GSI), third party resource brokers, Grid Resource Allocation and Management (GRAM), data management (GridFTP, GASS), resource reservation (GARA), and communications. Globus is constructed as a layered architecture in which higher level services can be developed using the lower level core services [12]. Globus offers Grid information services via an LDAP-based network directory called Meta-computing Directory Services (MDS). Initial versions of MDS (version lower than 2.4) are based on LDAP network directory. Later versions of MDS (version higher than 3) and later releases are OGSA based. MDS comes with a default schema and information providers can populate the schema. However, the default schema provided by MDS is not used in practice because it is simplistic (no notion of clusters for providers). Efforts like GLUE [3] try to provide a standard schema.

The MDS3 component of the Globus Toolkit Version 3.2 provides information about Grid resources for use in resource discovery, selection, and optimization. The MDS3 component is therefore a broad framework that includes any part of GT3 that generates, registers, indexes, aggregates, subscribes, monitors, queries, or displays Service Data in some way. The Index Service combines ServiceProviderExecution components with DataAggregation and ServiceGroup components to create a dynamic data-generating and indexing node, similar in concept to a MDS2 hierarchical GIIS. Index Services can be combined in a variety of topologies, useful in building Virtual Organizations. MDS follows both a push and pull protocols for resource dissemination. Higher-level tools such as resource brokers can perform resource discovery by querying MDS using LDAP and XML query protocols. Customers describe required resources through a resource specification language (RSL) that is based

---

2. www.hicb.org/glue/glue-schema/schema.html
on a pre-defined schema of the resources database.
The task of mapping specifications to actual resources is performed by a resource co-allocator, which is responsible for coordinating the allocation and management of resources at multiple sites. RSL allows customers to provide very sophisticated requirements, but there is no possibility for resource providers to specify their constraints on customers. The MDS namespace is organized hierarchically in the form of a tree structure. Globus offers QoS in the form of resource reservation. It allows application level schedulers such as the Nimrod/G resource broker to extend scheduling capabilities. The resource brokers can use heuristics for state estimation while performing scheduling or re-scheduling whenever the status of the Grid changes. Globus is a Grid toolkit and thus does not supply scheduling policies; instead it allows third party resource brokers.

- **Query Language and Advertising Language**
  The Globus Resource Specification Language (RSL) provides a common interchange language to describe resources. The RSL provides the skeletal syntax used to compose complicated resource descriptions, and the various resource management components introduce specific \((attribute, value)\) pairings into this common structure. Each attribute in a resource description serves as a parameter to control the behavior of one or more components in the resource management system.

- **Scalability**
  Globus supports distributed query based service discovery. The schedulers are in hierarchical organization. MDS can be decentralized using federation approach.

- **Reasoning support**
  No reasoning support for discovery is provided in Globus.

- **Matchmaking vs. Brokering**
  Globus does not provide matchmaking or brokering services itself, rather metaschedulers can be used for brokering and matchmaking purposes like Condor-G, CSF, and Nimrod/G. Globus provides a monitoring and discovery service to provide both static and dynamic status information about resource properties.

- **Mediation Support**
  No mediation support for discovery is provided in Globus.

**ONTOLOGY BASED MATCHMAKER (OMM)** propose a flexible and extensible approach for performing Grid resource selection using an ontology-based matchmaking technique and algorithm. Unlike the traditional Grid resource selectors (like Condor-G) that describe resource and request properties based on symmetric flat attributes, separate ontologies (i.e., semantic descriptions of domain models) are created to declaratively describe resources and job requests using an expressive ontology language. Instead of exact syntax matching, ontology-based matchmaker performs semantic matching using terms defined in those ontologies. The loose coupling between resource and request descriptions removes the tight coordination requirement between resource providers and consumers. Matchmaker can be easily extended, by adding vocabularies and inference rules, to include new concepts (e.g., UNIX compatibility) about resources and applications and adapted the resource selection to changing policies. These ontologies can also be distributed and shared with other tools and applications.

Matching between request specifications to resource capabilities is done in terms of rules. Similar to other matching systems, matchmaker provides the ability to describe properties and matching preference. Matchmaker also supports bi-lateral matching and gang-matching. The ontology-based matchmaker consists of three main components: (1) *Ontologies* that capture the domain model and vocabulary for expressing resource advertisements and job requests, (2) *Domain background knowledge*, capturing additional knowledge about the domain and (3) *Matchmaking rules*, defining when a resource matches a job description.

The background knowledge uses the vocabulary from the ontologies to capture background information. Matchmaking rules use both ontologies and background knowledge to match a request to resources. Ontology-based matchmaker is built on top of TRIPLE/XSB deductive database system, which is a centralized information store with no persistence. Resource discovery is through centralized queries composed using the vocabulary in the request ontology. The system aims to use Globus MDS as a persistent information store.

- **Query Language and Advertising Language**
  Resource descriptions, request descriptions, and usage policies are all independently modeled and syntactically and semantically described using a semantic mark-up language: RDF [4], RDF Schema [7].

- **Scalability**
  Resource discovery is through centralized queries to a deductive database composed using the vocabulary in the request ontology.

- **Reasoning support**
  Domain background knowledge captured in terms of rules is added for conducting further deduction. Likewise, matchmaking procedures written in terms of inference rules are used to reason about the characteristics of a request, available resources and usage policies to appropriately find a resource that satisfies the request requirements. Additional rules can also be added to automatically infer resource requirements from the characteristics of domain-specific applications without explicit statements from the user.

- **Matchmaking vs. Brokering**
  OMM is a prototype system that currently provides only matchmaking services. In the next implementations the matchmaker will provide both matchmaking and brokering services. The brokering service will be built on top of matchmaking service.

- **Mediation Support**
  No mediation support for discovery is provided in OMM.

**WEB SERVICE MODELING ONTOLOGY (WSMO)** [28] aims to develop an overall framework for Semantic Web services in order to support automated Web service discovery, selection, composition, mediation, execution, monitoring, etc. Every WSMO component description may include an extensible set of non-functional properties, based on the Dublin Core Metadata Set [35]. Two major design principles, inherited from WSMF [15] are applied in WSMO: (1) *Principle of maximal decoupling*: all WSMO components are specified autonomously, independent of connection or interoperability with other components and (2) *Principle of strong mediation*: the connection and interplay between different components is realized by Mediators that resolve possible occurring heterogeneities between the connected components. WSMO defines four top-level notions related to Semantic Web services:
1. **Ontologies**: are the key to link conceptual real world semantics defined and agreed upon by communities of users. Ontologies define a common agreed upon terminology by providing concepts and relationships among the set of concepts.

2. **Goals**: are descriptions of user’s desires; they represent the information space and state of the world after the execution of the service that would potentially satisfy the users desires.

3. **Web services**: are service descriptions for services that are requested by service requesters, provided by service providers, and agreed between service providers and requesters.

4. **Mediators**: address the handling of heterogeneities occurring between elements that shall interoperate by resolving mismatches between different used terminologies (data level), on communicative behavior between services (protocol level), and on the business process level.

The conceptual model of WSMO Discovery is provided in [22]. In comparison with other frameworks for service discovery, WSMO Discovery provides a complete framework for discovery. Three major steps in discovery are distinguished: Goal Discovery, Web service Discovery and Service Discovery. The first step, **Goal Discovery** is about abstracting goals from user desires. The second step, **Web service Discovery** is about how to match abstracted goal descriptions with semantic annotations of web services. The last step, **Service Discovery** is about finding real services whose abstract descriptions where discovered in the previous step. Different levels of service description can be considered ranging from simple keywords to more complex semantic descriptions. A special attention in WSMO is given to the relation between discovery and mediation. This relation is more than natural when we think about the heterogeneity of the environment with different users and services using different terminologies.

In order to make communication possible between different parties mediation is required. WSMO proposes a discovery mechanism with strong mediation support. In the matchmaking process, WSMO Discovery distinguishes between four types of matchmaking: (1) **Exact Match**: all relevant services and at the same time no irrelevant services are considered; (2) **Plug-in Match**: all relevant services but also services which are considered as irrelevant for the goal can be delivered; (3) **Subsumption Match**: only relevant services but not necessary all of them are considered; and (4) **Intersection Match**: in this case of matching, the service, whose description matches the request description, is able to deliver some relevant objects, but might deliver objects which are considered as irrelevant for the goal too.

To illustrate the WSMO discovery approach we use an example from the travelling domain. Let us consider the following scenario: a user, John Doe, wants to book a hotel room in Innsbruck on the 1st of August 2006 with a price lower than 150 euro. According to WSMO discovery model the user’s desire, which is in most cases express in natural language, is first abstracted to more generic goals. Such a generic goal could be for example the booking of a hotel room in Innsbruck. The process of abstracting from a user desire to generic, reusable goals is called **Goal Discovery** and can be implemented using keyword-matching techniques. Once a set of pre-defined goals has been discovered as a result of the Goal discovery process, one of them will be selected and refined, either automatically from the textual desire or manually using appropriate tools, in order to express the concrete requester goal. Given such an abstract, reusable
goal, the Web service discovery will automatically find the Web services that are matching this goal. More precisely abstract goals and semantically annotated Web services are matched at the ontological level. There are three approaches to realize the matching, namely keyword-based discovery, discovery based on simple semantic descriptions of services or shortly lightweight discovery, and discovery based on complex semantic descriptions of services or shortly heavyweight discovery.

We use the same example from the travelling domain to illustrate how each approach for Web service discovery is working. According to [27] and [22] is very important how services and requests are modelled since this influences the discovery process. Therefore we are going to describe how a hotel booking service and a user goal can be modelled at different levels of semantic descriptions.

In **Keyword-based discovery** a Web service or a goal is basically seen as a bag of keywords. Therefore a hotel booking service (HBWebService) can be modelled as follows:

\[
\text{HBWebService} = \text{Hotel, Innsbruck}.
\]

Listing 1: Keyword-based modelling of HBWebService

The service advertised provides booking functionality for hotels in Innsbruck. A goal is modelled in the same way, e.g. for the previous mentioned user desire a corresponding goal for hotel booking (HBGoal) is modelled as follows:

\[
\text{HBGoal} = \text{Hotel, Innsbruck, not more than 150 euro per room.}
\]

Listing 2: Keyword-based modelling of HBGoal

During the keyword-based discovery the keywords from the goal are matched against the keywords used to describe the Web service by using classical information retrieval techniques. The service descriptions that get the higher scores are returned as possible matches for the given goal. In our case, the service described before is a match because given all keywords from its description are in the goal description as well. However, in many cases, the keyword-based discovery fails due to the fact that is does not use explicit, well-defined semantics. For this reason the other two approaches for Web service discovery, namely lightweight and heavyweight discovery were introduced.

**Lightweight discovery** follows a set-based modelling approach for goals and Web services. A Web service is seen as a computational entity which can be invoked by a client. By invoking a Web service certain information is generated as the outputs of the service and certain changes occurred in the world as the effects of the service. Thus, a service description in this case captures the objects that can be delivered by a service. Web services and goals are therefore modelled as set of objects. The previous hotel booking service can be described as follows:

| :--- |
| ?s{ place hasValue ?H, amount hasValue ?A, currency hasValue ?C, and ?H.locatedIn hasValue Innsbruck and ?H.bookedRooms < ?H.totalRooms. } |

Listing 3: Simple semantic descriptions of HBWebService

The corresponding goal is modelled as follows:

| :--- |
| ?g{ place hasValue ?H, person hasValue ?P, date hasValue ?D, amount hasValue ?A, currency hasValue ?C, and ?H.locatedIn hasValue Innsbruck and ?H.bookedRooms < ?H.totalRooms and ?A < convertCurrency(150, ?C, Euro) and isEuropeanCurrency(?C). } |

Listing 4: Simple semantic descriptions of HBGoal
The semantics of the goal and web service descriptions are represented by a set of objects. The objects advertised by the service provider are all instances \(s\) of the class HotelBooking for which the constraints specified in the web service description have to be satisfied. The set of relevant objects for the requestor are all instances \(g\) of the class HotelBooking for which the constraints specified in the goal description have to be satisfied. Please note that in our example the hotel booking service advertises all the possible hotels in Innsbruck whereas the user John Doe is interested in finding only one hotel that fulfils his needs. The process of matching goals to services during lightweight discovery is resumed to determine the set-theoretic relationship that exist between the two sets representing the goal, respectively the Web service. In our case the set of objects representing the goal \(g\) is entirely subsumed by the set of objects representing the service \(s\) and therefore we have a plug-in match.

Heavyweight discovery or discovery based on complex semantic descriptions of goals and Web services extends the lightweight discovery by considering the relations between the outputs and effects created by a service, on one hand, and the input provided by the requester, on the other hand. Modelling goals and Web services can be done either by considering an extended version of the set-based theoretical model described previously either by adopting logical formalisms like Transaction Logic [6]. In the rest of this section we consider the set-based modelling approach. The previous hotel booking service is modelled as follows:

\[
\text{HBWebService}(\text{?H, ?P, ?R})
\]

\[
\begin{align*}
\text{pre:} & \quad \text{?H.locatedIn hasValue Innsbruck and} \\
& \quad \text{?H.bookedRooms < ?H.totalRooms and isRequestBy(?P) and} \\
& \quad \text{?A < convertCurrency(150, ?C, Euro) and} \\
& \quad \text{isEuropeanCurrency(?C)).} \\
\text{post:} & \quad \text{isReservation (?R) and ?R.holder hasValue ?P}
\end{align*}
\]

Listing 5: Complex semantic descriptions of HBWebService

The corresponding goal is modelled as follows:

\[
\]

\[
\begin{align*}
\text{pre:} & \quad \text{?H.locatedIn hasValue Innsbruck and} \\
& \quad \text{?H.bookedRooms < ?H.totalRooms and isRequestBy(?P) and} \\
& \quad \text{?A < convertCurrency(150, ?C, Euro) and} \\
& \quad \text{isEuropeanCurrency(?C)).} \\
\text{post:} & \quad \text{isReservation (?R) and ?R.holder hasValue ?P}
\end{align*}
\]

Listing 6: Complex semantic descriptions of HBGoal

The matching of goals and Web services descriptions during heavyweight discovery can be implemented as in lightweight discovery based on set theory. However, an important difference is that single and multiple invocations of the Web service are considered.

The last step in the overall discovery process is the service discovery. It takes the Web services discovered during the Web service discovery step and tries to interact with each service provider to determine the subset of the relevant service providers whose Web services actually are able to deliver the requested service. This process can be seen as refinement of the Web services previously discovered and will require a significant mediation support especially at process and protocol level.

- **Query Language and Advertising Language**

In order to express user requests and service descriptions, a family of representation languages, is provided for WSMO. This family of languages, called WSML [14], provides a formal syntax and semantics for WSMO. WSML languages: WSML-Core, WSML-DL, WSML-Flight, WMSL-Rule, WSML-Full are based on different logical formalisms: Description
Logic Programming for WSML-Core, Description Logics for WSML-DL, Logic Programming for WSML-Flight and WSML-Rule, and First-Order Logic for WSML-Full. Description of user requests and service advertisements are done using concepts from ontologies that can be as well formulated in WSML.

• **Scalability**
  The WSMO approach for discovery is a complete solution that addresses a wide spectrum of approaches starting from natural language descriptions of Web services and discovery requests to precise logical definitions of Web services and requests descriptions. The actual implementation of the discovery framework proposed in WSMO is work in progress. Scalability issues will be addressed within this work.

• **Reasoning support**
  The reasoning support required during WSMO discovery process depends on the WSML language that is used to describe requests and services. As mentioned above the WSML family of representation languages is based on a wide set of logical formalisms: Description Logics Programming, Description Logics, Logic Programming and First order Logic. An appropriate reasoner for each of these logical formalisms or even better a native reasoner for WSML languages is required during matchmaking process. The WSML group is currently developing a native WSML reasoner (see [13]).

• **Matchmaking vs. Brokering**
  The discovery framework in WSMO does not consider the notion of brokering as defined in section [3], but this can be very easily plugged-into the framework. For matchmaking different approaches are considered: exact match, plug-in match, subsumption match, intersection match.

• **Mediation Support**
  WSMO provides a strong support for mediation. Actually in WSMO mediators are top conceptual elements that solve heterogeneity problems between different terminologies, protocols or processes. In WSMO, the mediation support for discovery is considered at different levels of this process. At keywords-based level, the mediation support considers methods like: stemming and synonym recognition. At simple semantics level, mediation support considers methods like: controlled vocabularies and shallow ontologies. Finally at the rich semantic level, the mediation support considers methods like: complete logic and heavy-weight ontologies.

**DAML-S/OWL-S APPROACHES**
Many approaches for discovery using DAML-S [32] or it’s successor OWL-S [33] have been proposed [26, 25]. In [26] a DAML-S semantic matching between advertisements and requests is proposed. The matching algorithm is based on subsumption reasoning in DAML+OIL [21]. A service profile and a request are considered to match when all the outputs of the request goal are matched against all, or a subset of service output, and as well all the inputs of the service are matched against all, or a subset of request goal. Different degrees of matching were identified: (1) *Exact Match*: the outputs, respectively the inputs being matched are exactly the same, (2) *Plug-in Match*: the output of the service subsumes the output of the request, (3) *Subsumes Match*: the output of the request subsumes the output of the service and (4) *Fail*: no matching services were found for the request goal.
In [25] a different approach for discovery using DAML-S is proposed. Compared with the previous approach all the entities of the service profile are used, namely: inputs, outputs (like in the previous approach) and as well preconditions and effects. A implemented prototype based on RACER\(^5\) is available. Different degrees of matching are consider as well: (1) **Exact Match**: in this case the advertisement \(A\) and the request \(R\) are equivalent concepts, (2) **Plug-in Match**: in this case the request \(R\) is a sub-concept of advertisement \(A\), (3) **Subsumes Match**: in this case the request \(R\) is a super-concept of advertisement \(A\), (4) **Intersection Match**: in this case the intersection of request \(R\) and advertisement \(A\) is satisfiable and (5) **Disjoint Match**: none of the matches previous presented. The strength of the match is decreasing from the **Exact Match** to **Disjoint Match**. By using a Description Logic reasoning procedure to detect possible matching, this approach inherits the time consuming operation of classifying the profiles in profile hierarchy.

- **Query Language and Advertising Language**
  The DAML-S/OWL-S approaches use the DAML-S/OWL-S specifications to model and the DAML+OIL/OWL associated languages to describe the user requests and services descriptions. All these languages are based on Description Logics logical formalism. Ontologies are the backbone in all these approaches; they offer the common terminology to describe user requests and services descriptions.

- **Scalability**
  The performance and scalability of DAML-S/OWL-S approaches for discovery depend on the architecture adopted and are influence by the performance and scalability of DL reasoners.

- **Reasoning support**
  The reasoning support for the DAML-S/OWL-S discovery approaches is determined by the underling logic mechanism used to describe the knowledge, which is Description Logics. Main reasoning approaches include: subsumption reasoning, instance checking, etc.

- **Matchmaking vs. Brokering**
  In both DAML-S/OWL-S approaches mentioned above only the matchmaking perspective is considered. No future interaction involving the matching entity after the matching is provided (no brokering approach). In both approaches different degree of matching are considered: exact, plug-in, subsumption.

- **Mediation Support**
  DAML-S/OWL-S approaches for discovery do not provide any mediation support.

5 Summary
Table 1 presents the summary of discovery approaches evaluated according to the proposed evaluation framework. / Based on the previous evaluation the most interesting features of one approach, or combination of features from different approaches are selected and proposed as suggestions that might be considered when developing a discovery mechanism.

- **Query Language and Advertising Language**
  Semantic enabled frameworks for discovery in conjunction with rich and expressive query and advertising languages are requirements for an automatic discovery mechanism. From the previous surveyed approaches OMM([31]), DAML-S/OWL-S ([26], [25]) and WSMO([28], [22]) consider this aspect.

\(^{5}\)http://www sts tu-harburg de/ r.f moeller racer/
Asymmetric description of resources, services and requests using an ontological support is one important feature that a good discovery mechanism must consider.

- **Scalability**
  For scalability issues approaches like Globus [16] can be considered. An scalable architecture possible based on hierarchical federated approach or other approaches (eg. P2P systems) is an important feature that a good discovery mechanism must consider.

- **Reasoning support**
  Depending on the advertising and query language that a discovery mechanism adopts the reasoning support will be inferred. Broad reasoning support is very well addressed in approaches like: WSMO([14], [28]). For set based modelling style and reasoning (Descriptions Logics [2]) only, solutions provided by approaches based on DAML or OWL might be enough. Good solutions for reasoning with rules and complex descriptions are provided by WSMO and OMM.

- **Matchmaking vs. Brokering**
  For complete discovery, in Grid environments especially, both matchmaking and brokering functionalities must be considered. From the previous surveyed approaches only OMM considers both matchmaking and brokering.

- **Mediation support**
  Mediation support during matchmaking/discovery might be considered due to the heterogeneity of descriptions, data, etc. From the previous surveyed approaches, WSMO provides a complete solution for this problem.

### 6 Related Work and Conclusions

Although some survey papers about Grid and Web services in general and discovery approaches in particular exist (eg. [23], [24]), none of this work presents a clear set of criteria that can be used when a discovery approach is evaluated.

In [23] a taxonomy of Grid resource management systems is presented, but without a special focus in discovery. Different Grid approaches are surveyed and place in a taxonomy that considers: the type of Grid system, the resource model and the scheduling characterization.
In [24] the problem of automatic discovery and composition of Web services using semantic annotation is address. In this context a summary and evaluation of current approaches in service discovery is presented and the shortcomings of these approaches were identified. Nevertheless there is no framework, or a set of criteria that can guide the evaluation process.

In this paper, we presented an evaluation framework for discovery approaches in Web services and Grid environments. Further we have exemplified the use of our framework by evaluating some of the most relevant discovery approaches in Web services and Grid areas. Based on this evaluation, a set of suggestions for the design of a discovery solution were extracted. We believe that our framework captures most of the relevant evaluation criteria and it can be easily used to identify the strengths and weaknesses of a Grid or Web services discovery approach.

7 Acknowledgements

The work is funded by the European Commission under the project Adaptive Services Grid. (ASG)

References


The OWL Services Coalition. OWL-S 1.1 beta release. Available at http://www.daml.org/services/owl-s/1.1B/, July 2004.


8 Author’s Bios

Ioan Toma is a researcher and a PhD candidate at Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria. His current research areas include Semantic Web Services, Grid Computing and the relation between them. He is involved in several EU and Austrian funded projects (e.g. ASG, DIP, SemanticFred and Grisino). He obtained his graduated engineer of computer science (Dipl. Ing.) and M.S. titles from Technical University of Cluj-Napoca, Romania in 2002, respectively 2003. Before joining DERI as researcher in April 2004 he was a junior lecturer at Technical University of Cluj-Napoca. From 2000 to April 2004 he carried out various software development projects in Romania.

Kashif Iqbal received the first class degree in Information Technologies from National University of Sciences and Technology, Pakistan in 2004. He is a researcher and part time PhD student at Digital Enterprise Research Institute (DERI), National University of Ireland, Galway and University of Portsmouth, UK respectively. He has been working with distributed system technologies including Grid, Web Services, and Multi-agent systems for more than three years in various CERN and EU-funded projects. His main research interests include Semantic based Indexing and Discovery Services. He has been an active member of Semantic Grid research group in Global Grid Forum.

Dumitru Roman is an engineer in computer science; he received his diploma engineer degree from Technical University of Cluj-Napoca, Romania. He currently works as a researcher at the Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria, where he is mainly involved in the area of Semantic Web Service; in this context he represented DERI in several EU projects (e.g. ASG, SWWS, SWING, etc.) and he is currently the WSMO working group (an european research initiative in this area) coordinator.

Thomas Strang studied computer science at the University of Technology (RWTH) in Aachen and received his Dipl.-Inform. degree in 1998. Since July 2000 he is working as a researcher with the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. Here his research
focus is on ubiquitous and pervasive computing, location- and context-awareness, service discovery and execution frameworks, Semantic Web and smart mobile devices. In 2003 he was awarded a Doctor’s degree in natural sciences (Dr. rer. nat.) at the University of Munich, Germany, with a dissertation on service interoperability in ubiquitous computing environments. Since October 2004 he is a professor for computer science at the University of Innsbruck, Austria, and also gives lectures at the University of Munich. He is also Executive Director at the Digital Enterprise Research Institute (DERI) in Innsbruck, Austria, since January 2005.

Dieter Fensel is a full professor of computer science at University of Innsbruck. Prof. Dieter Fensel holds a Doctor’s degree in economic science (Dr. rer. pol.) at the University of Karlsruhe in 1993 and in 1998 he received his habilitation in Applied Computer Science. He is the scientific director of Digital Enterprise Research Institute, one leading institute in the areas of Semantic Web and and Semantic Web Services technology. He has been involved in numerous national and internal research projects. He is the project coordinator of many European projects like: DIP, Knowledge Web, Ontoknowledge, OntoWeb, and SWWS. He published around 150 papers as books and journal, book, conference, and workshop contributions. He co-organised around 150 scientific workshops and conferences and has edited several special issues of scientific journals. He won the Carl-Adam-Petri-Award of the Faculty of Economic Sciences from the University of Karlsruhe (2000).

Brahmananda Sapkota is a PhD researcher with the Digital Enterprise Research Institute (DERI), National University of Ireland, Galway. His research focus is on Semantic Web Services in general and Semantic approach for service discovery in particular. His research interests also include Peer-to-Peer networks, semantic middleware and application integration. Brahmananda obtained his Bachelor’s degree in Software Engineering with Honours from Kathmandu University, Nepal in 2000. Before joining University of Twente, the Netherlands in 2001 to pursue his Masters degree in Telematics, he went on to gain some teaching experience at Kathmandu University, Nepal. Currently Brahmananda is an active participant of the WSMO and WSMX working groups. He is also a voting member of the OASIS SEE technical committee. He has participated and contributed in several nationally and EU funded projects such as DIP, Lion, TSC, and TripCom in the area of his research interests.

Matthew Moran is a researcher with the Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, working in development and system architecture for Semantic Web Service systems. He has exhaustive professional experience in software development, and is involved in several national and international research projects. Matthew Moran is a founding member of the WSMO and WSMX working groups, co-author of the WSMX submission to W3C and is co-editor of the architecture document for the OASIS SEE technical committee.