

The necessity of semantic technologies in grid discovery

Serena Pastore

INAF- Astronomical Observatory of Padova, Padova, Italy

Email: serena.pastore@oapd.inaf.it

Abstract— Service discovery and its automation are some of the key features that a large scale, open distributed system must provide so that clients and users may take advantage of shared resources. Most actual distributed systems are built with several middleware applications developed by using grid and web service technologies that allow an infrastructure to be implemented where users work as if they were in a local system. The paper, starting from the work done within grid projects in which the INAF Institute was involved, examines issues encountered and solutions proposed in the optics of sharing and thus using web resources such as web services in a specific grid system. By guaranteeing the efficient use of such resources, different discovery mechanisms, developed within grid and web service areas, have been evaluated. The results show the necessity of an appropriate resources' description in terms of the information data model, protocol and search tools that could integrate semantic technologies required for automating the process. Enabling automatic discovery means both the enriching of description with one of the semantic languages that are in constant development (i.e. OWL-S, WSDL-S) and the availability of a mechanism able to interpret and process such information. This work aims at taking advantage of the improvement in semantic technologies to prove the efficacy of this approach in making use of applications that need a grid environment for their execution.

Index Terms— service discovery, grid technologies, web services, semantic web services, UDDI, SOAP-based web service, REST-style, semantic discovery, OWL-S, WSLD-S.

I. DISCOVERY CHALLENGES IN DISTRIBUTED SYSTEMS

Modern computing systems and architectures make use of distributed environments [1], most of them built with several middleware applications developed by using grid and web service technologies [2] that allow an infrastructure to be created where users work as if they were in a local system. In an open even if controlled network that crosses different organizational boundaries, several types of resources, abstracted as services, are indexed to be collected, monitored and shared by providers to be used by consumers. The resources, as the grid concept explains [3], are heterogeneous in every aspect and comprise the hardware facilities (processing and storage capabilities) that are the basis of computing grids, but also software capabilities and collaboration tools. Specifically today many applications are developed as web and web service applications because of the nature

of data and the availability of network connections and web interfaces. In the complexity of the resultant environment, automatic search and discovery of resources matching specific requirements set by users, client applications or agents is one of the key features that such infrastructure requires. The realization of these capabilities is strictly related to the efficacy of the resources in self-describing functionalities and capacities by means of specific data models, schemas, syntax and protocols adopted to describe both the resources and the exchange of information between the participants. Service discovery in such an environment is difficult because of the need to filter out services suitable to the task. Different approaches to the discovery goal have been studied and applied in grid systems according to the different middleware applications adopted for their building; each one usually implements a data model for resources' description and a specific web services framework. Moreover, even in the web service area, specifications and tools have been developed to exploit the web interface information in terms of the service's capabilities and functionalities that such applications expose by using specific languages and that can be used in discovery, composition and security tasks. In any case automation in providing these operations is a common criterion to be reached to avoid human intervention in finding resources and allowing real exploitation of the environment. This is one of the goals of the semantic web [4], where resources are expressed not only in natural language but also in a computer interpretable language, and may be read and used by software agents, thus permitting them to find, share and integrate more easily. This is the vision of the web as a universal medium for data, information and knowledge exchange, which to be realized, needs of a variety of enabling technologies most of them expressed in formal specifications.

The paper addresses the challenges that have arisen in the efforts of sharing (thus, indexing, discovering, monitoring and securing) software resources such as the web and web service astrophysical applications in a specific grid environment, that is, the EGEE-II (Enabling Grids for E-science) infrastructure (<http://www.eu-egee.org>). This environment, which for European research institutes represents the reference grid platform, is built on the gLite middleware (<http://glite.cern.ch>) as a result of the software developed in previous grid projects.

Working in this infrastructure, within different grid projects in which the INAF (National Institute of Astrophysics, <http://www.inaf.it>) was involved has required analysis of different resources' description languages and the testing of various discovery mechanisms in order to realize effective use and management of web resources. In fact, the abstract information model used in an EGEE-compliant grid that is based on the GLUE (<http://forge.gridforum.org/sf/projects/glue-wg>) model does not provide, especially in the first version of the schema, a sufficient description in dealing with such resources. Moreover, the great availability of description languages but the frequent changes in the specifications' development require study for the best description that could be integrated in the grid schema and take advantage of this information in the automatic searching and discovery of the resources as a tradeoff between existing software and the domain's goals.

This work aims at taking advantage of the improvement in semantic technologies in descriptions and automatic discovery to prove the efficacy of this approach for applications that need a grid environment for their execution. To filter out the most suitable services, a semantic description may qualify all the characteristics of the service, and semantic discovery makes use of this information since it is capable of meaningful interactions. Also, grid developers are studying the extension of the current grid discovery system with semantics through the semantic grid project (<http://www.semanticgrid.org>). However, the project relies on a specific middleware that is already web-service compliant. In this paper, we will analyze languages and solutions that could be integrated into and used with minimal modification in the European infrastructure.

II. MOTIVATION OF THE STUDY

The grid activities taken on by the INAF Institute started with the inter-institutional GRID.IT grid project [5] that aimed at creating and exploiting a grid system for scientific applications. Due to the push of the entire astronomical community gathered around the IVOA (International Virtual Observatory Alliance, <http://www.ivoa.net>), web services and grid technologies are implemented in the development, management and use of astrophysical applications. These kinds of applications [6] process a large amount of data (thus taking advantage of data and computing grid) that are stored worldwide in different systems (thus taking advantage of the web service paradigm for the need of interoperability) and whose access may be restricted (thus benefiting from grid security mechanisms). The INAF Institute collects people and resources involved in the grid project in the INAF Virtual Organization (VO) [3], which is one of the VOs around which the grid system is logically organized. The VO contributes to grid resources with several sites provided by different headquarters, each one with a set of logical grid nodes implemented within physical machines. The gLite toolkit around which

the EGEE-II architecture [7] is built distinguishes grid core services such as workload management (WMS – Workload Manager System and RB/LB- Resource Broker/Logging and Bookkeeping), data management (FTA, FTS – File Transfer Agent/Service), information system (BDII – Berkeley database information index or MON- monitoring system), VO users management (VOMS- Virtual Organization Management Service), and proxy server (PX), from grid resources such as computing resources (CE), storage resources (SE) and processing nodes (WN). The focus is on software resources, since the goal for a resource provider such as the institute in this scenario (Figure 1) is to share not only computing facilities but also applications and services that deal with astronomical data that could be used by grid users. In this option we refer to web and web service applications developed with different programming languages and thus deployed in specific web application containers [8].

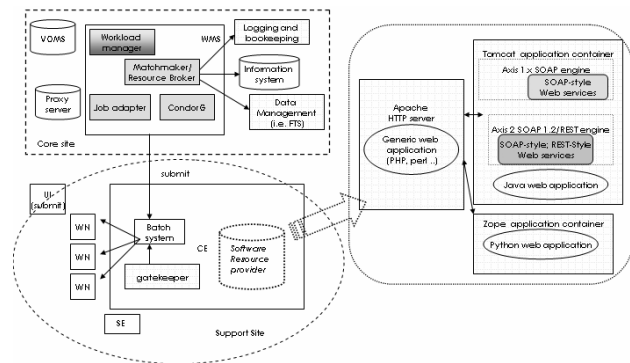


Figure 1. EGEE-II architectural structure and the software resource provider.

Generic web applications are developed, for example, by using Perl or PHP languages and are managed directly by web servers (i.e., the Apache HTTP server, <http://httpd.apache.org>) through specific modules that process such languages. Other applications, however, make use of specific software platforms: Java web applications are managed by a J2EE-based framework [8] (i.e. Tomcat engine, <http://tomcat.apache.org> engine), while Python web applications by a specific container (i.e., such as a Zope framework, <http://www.zope.org>). This software may, however, have the Apache server as the front-end so that every request to the web application is processed by this server. In terms of web service development, the different methods used for their writing have also been adopted in the astronomical community. In this scenario, effective use of such a resource comes from the data model adopted for resources and indexing and the discovery mechanisms used in the environment.

A. Grid resources, the GLUE schema and a web or web service software resource.

Grid services (i.e., information services, data and workload management) provide uniform interfaces to the types of system elements that are realized by the grid resource (i.e., essentially computing and storage resources). EGEE-II resources' data model, based on the

GLUE schema, describes each VO's site with a *site* and *service* elements (Figure 2).

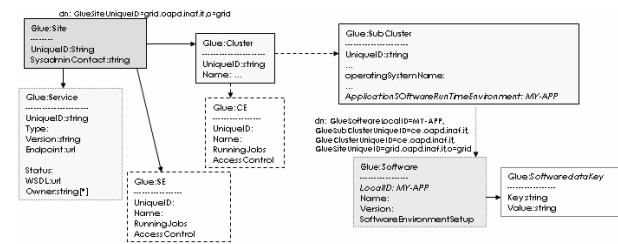


Figure 2: Grid resources according to the GLUE data model for LDAP schema

The schema version (1.x) actually adopted offers an LDAP and XML implementation for indexing resources to be used by the discovery system (called information service). However, the work on the schema (by the OGF Glue Working group, <http://www.ogf.org>) will carry out a model that will be a schema that can operate with a variety of grid modeling approaches. Currently, while a *service* element describes the main functionalities of a site (i.e., discovery, authorization), the *site* element models all the processing and storage capabilities with the definition of *cluster* and *storage* elements. A software resource is thus associated to a *SubCluster* element, which is the basic component of the cluster. The cluster represents an abstraction of a processing environment: the *Subcluster* is associated with a physical host, and its capabilities include all installed software. However, the information is attached to the software that is used for its indexing, and discovery is actually limited to the location (i.e., path) and the version. A web or web service application is thus a *Subcluster's* attribute, while the service element may represent the discovery method.

B. Web services programming styles and platforms

A web application in general, but a web service specifically, is accessed through the network and usually provides an interface that enables humans and machines to keep up to date about protocols and mechanisms used for transmitting messages in the client-web relationship. A web service, for example, is described according to a specific language XML-based such as WSDL (a W3C, <http://www.w3.org> recommendation) through the abstract functionality the web service provides: how and where to access the service are explained in detail. In terms of web services, a distinction is made about the paradigm used to program such an application, which results in SOAP-based [9] and REST-based [10] approaches that should be followed for web service development. The distinction results in a service-/activity-oriented or resource-oriented design and the decision to follow one of the two approaches depends on the developer and his or her goals. A REST-based web is focused on the REST (Representational State Transfer) architectural style, a client-server interaction that it is not a standard, but uses standards (i.e., URI, URL, XML) to describe resources (conceptual entity) as representations (i.e., concrete manifestation of a resource) that place the client in a state, the format of response data (schema-like) and how

the service is invoked. The REST-style web service makes available URLs that the client accesses to receive XML documents (i.e., WSDL document) with all the useful information to interact with it. Then submissions requests are sent as the payload of an HTTP operation: via GET, whether clients just receive a representation, via POST, PUT or DELETE if the client can modify the resource.

The SOAP-style web service is activity-oriented and based on the SOAP specification (i.e., the latest version is 1.2) that defines the XML-based framework for exchanging structured information in a decentralized, distributed environment over a variety of underlying protocols (i.e., Internet protocols such as HTTP for transport), but independent of any programming. Their interface is the WSDL document that defines and describes service-specific operations (i.e., as an exchange of messages). Moreover, the web service architecture on which such an application is built defines a set of specifications (essentially by the W3C or OASIS, <http://www.oasis-open.org> organization) that describe supporting features such as a discovery framework (i.e., UDDI, <http://www.uddi.org>), security standards (i.e., WS-Security [9]) and other WS-* specifications (i.e., WS-Coordination, WS-ReliableMessaging, [9], etc.). A SOAP-based web service may thus be described within the UDDI specification, which defines a standard method for publishing, inquiring and discovering operations based on a registry. According to such a data model (Figure 3) built on objects and their attributes, each service (*businessService* object) that an organization (*businessEntity* element) provides is characterized by functional information such as its entry points (*bindingTemplate*) and is classified or categorized through abstract entities (*tModel*) according to mechanisms based on property-based lookup (*identifierBag*) or categorization (*categoryBag*).

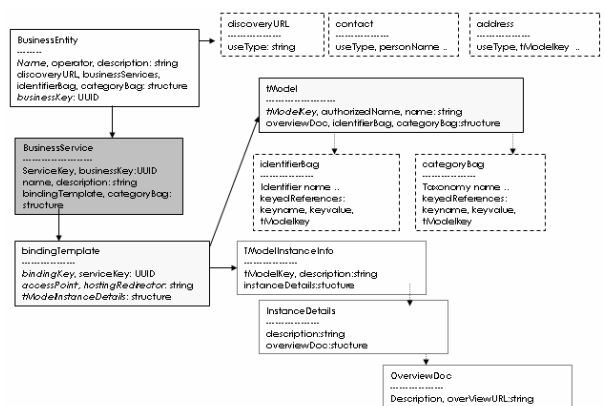


Figure 3 Software/web service resource according to the UDDI data model

Different programming languages may be used to develop web services because of the availability of libraries for specific languages (i.e., PHP and Python libraries) and SOAP engines (such as Apache Axis, <http://ws.apache.org> for Java developers) that are able to deal and process web service transactions. Axis, however, in the last redesigned version 2, supports both SOAP 1.2

and the REST style of web services together the several modules that provide WS-* functionalities (i.e., WS-Security supported by Apache Rampart module). The same business logic implementation can offer a WS-* style interface as well a REST style interface simultaneously. The focus thus is on Java web applications [11] since such language is one that has actually been adopted in the astrophysical area in addition to the grid area; also, Java implementation of different WS-* specifications is available.

III. THE NEED FOR SEMANTICS IN WEB SOFTWARE APPLICATIONS

Apart from the approach chosen to write web service-based applications, a web software resource is fully described by the WSDL document that provides the network interface usable by users and agents. The necessity for the richness of such description has been shown in the grid environment to take advantage of such information in the discovery task. For example, semantic discovery is the process of discovering the services that are capable of meaningful interactions even though the languages with which they are described may be different. A semantic discovery process relies on semantic annotations, which contain high-level abstract descriptions of requirements and behavior and are used to be published in a registry. Around XML, many specifications [12] have been developed to enhance service functionalities. For example, WSDL does not include semantics, and thus services with similar descriptions may have different meanings. Resolving the ambiguity in description is an important step toward automating the discovery and composition of such these applications.

There is a plethora of projects involved in introducing semantic technologies in the web services [13] area by developing models or specific languages. Such standards should provide the capability to discover, extract, model and enhance information. Most of them fit into a set of layered XML specifications that become W3C recommendations and comprise the RDF (Resource Description Framework, <http://www.w3.org/RDF>) core model and schema, and the OWL Web Ontology language (<http://www.w3.org/2004/OWL>). According to the semantic approach, the resource and thus the web service and its interface refer to an agreed-upon vocabulary or ontology that defines the constructs by explicitly specifying the tags that are used to describe the properties and capabilities of services in unambiguous form.

There are a variety of technologies, even if they are by no means exclusive, that results in an ontology for web services and in languages for semantic annotations. OWL-S (<http://www.w3.org/Submission/OWL-S/>) is an upper ontology considered to be a language that represents semantic models and is untied by the WSDL description. WSDL-S specification (<http://www.w3.org/Submission/WSDL-S/>) is a language for adding semantic annotation to WSDL-described services. This method, however, allows semantic domain

models to be maintained outside of the WSDL document and in fact the annotation could be done with any ontology language.

These approaches may in some way be applied to a different version of the WSDL specification with an update from the 1.x to 2.x version (Figure 4).



Figure 4. A comparison between the versions 1.x and 2.0 of the WSDL document for the same service (QueryService) making queries to a database that supports the select operation.

A. A WSDL-based web service application

A WSDL 1.1 document (Figure 4) describes a service as a collection of endpoints (*port*) with each one defined as a combination of a *binding* that specifies a protocol and a data format and a network address. Moreover, a *portType* collect the operations supported by the service that are described in terms of messages exchanges and the data types used.

The WSDL 2.0 document modifies such structure and describes the service as a reference to a single *interface* that the service supports and a list of endpoint locations where the service can be accessed. While the interface defines the supported operations, the *binding* element specifies message format and transmission protocol details, and the *types* element defines the data types used in the messages as in the previous specification.

The first web services developed within the project followed the WSDL 1.x specification, but if there is a need to change to the new version (i.e., to apply languages developed for it), converters are available for the transformation into WSDL 2.0 compliant services (i.e., in the form of an on-line conversion tool <http://www.w3.org/2006/02/WSDLConvert.html>).

A semantic description of the resources helps in providing an automated process for their location and thus their use by matching service capabilities and client-specified or agent-specific constraints. However, the resources should be indexed and published by the provider in a registry so that requesters can find the resource when they query the registry. This means enabling the registry to store and deal with semantics and the discovery mechanism to search for and process this kind of information.

WSMO studio (<http://www.wsmostudio.org/>) is a semantic framework that, among several features, includes a SAWSDL editor, and inside the METEOR-S project, the Radiant plug-in for the Eclipse programming environment allows for annotating existing WSDL documents into WSDL-S or SAWSDL via an OWL ontology.

IV. DISCOVERY APPROACHES IN GRID FOR GRID SERVICES

The enrichment in resource description should necessarily result in taking advantage of such information in searching and finding methods in the grid. Solutions adopted for the indexing and discovery problem in several grid systems depend on the architecture and building toolkits. In a grid environment, it is necessary to have an updated list of all resources and solutions spread from indexes, directories and other approaches differing in the technologies used and the way they ask for the list to the information providers. Several information services and the resultant discovery process have shown similarities and known problems. Built on the previous version (GT2) of the Globus Toolkit (<http://www.globus.org>), gLite middleware was initially based on the LDAP (Lightweight Directory Access Protocol) directory service and protocol by means of the Globus MDS2 [15] (Monitor and Discovery Service) system. This service, now deprecated by the Globus Alliance, was based on a hierarchy of LDAP servers (one for each resource's type as on the storage and computing element and one for a site) and showed scalability and stability problems. It has been substituted by a same hierarchy of distributed BDII (Berkeley Database Information Index) system that indexes resources by querying the sites with a pool model. Discovery is based on the LDAP query model [16] by searching information (entries and their values) collected on the leaves of the DIT (directory information tree) and travels towards the root. The client or agent can query the tree at every level, but the higher the level against which queries are made, the older the obtained information. The technical implementation of this structure is based on the OpenLDAP software (<http://www.openldap.org>) whose tools are used in performing the operation. gLite middleware is going to substitute the BDII indexes with the R-GMA system (<http://www.r-gma.org>), which adopts a relational database model. The model implements a virtual database for a given VO and uses tables and relationships between tables to store information about the resources following the GLUE data model. The resultant software is a set of Java servlets hosted in a servlet engine. The searching protocol makes use of SQL queries: the data are produced and accessed locally by a producer that, however, refers to a central registry. In this way, the searching of a software resource relies on filtering the attributes associated with a specific entry (the element) of the R-GMA tables. However, the database may be used to store the varied information from grid schemas from whence it can be retrieved via SQL queries. Some connection timeouts and too much

load on the registry have been outlined. But the gLite middleware has been developing the Service Discovery [7] package as a set of plugins implemented as client libraries to different discovery mechanisms (up to now to the R-GMA service as well as for the BDII). It is made up of a command line interface, and an API (the Simple API for grid application or SAGA) is currently using this API for service discovery. Also in this perspective, the test of a different discovery mechanism in grid has been done, making it possible to interface with gLite commands and thus the grid, and locate suitable services. Unfortunately, this tool is in development and, for example, the Java API is not available, but the presence of this enhancement encourages the work.

A. Introducing UDDI in grid

The adoption in the grid of an UDDI private registry [17] has initially been a choice for allowing the deployed applications and all its capabilities to be found and searched. Introduced as a complementary approach to the grid mechanism, the private registry has been implemented by using the jUDDI software (<http://ws.apache.org>), a pure Java web application that relies on a database deployed on a grid machine. UDDI is the specification that empowers clients to make queries about services and lets developers publish their services by specifying in the registry any related information. The registry itself is based on multiple standards, and moreover, most semantic web service projects work on enhancing such a registry to also support semantic information. As part of the same layered stack, there is a strict relationship between WSDL and UDDI, which emphasizes the implementation of WSDL 1.1 to UDDI mapping (Figure 7) as reported in an OASIS technical note [18].

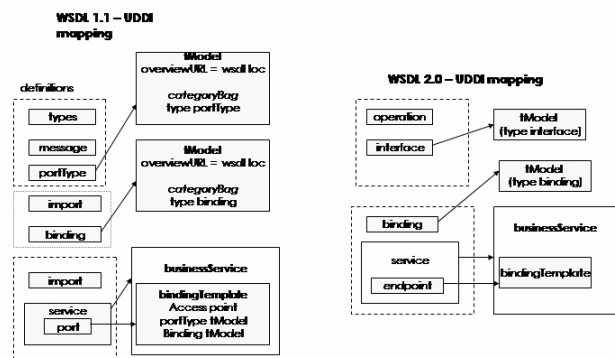


Figure 7 Mapping between WSDL artifacts and UDDI constructs

With the reference to the WSDL decomposition in the abstract definition (i.e., operations with *portType* and *binding*) and the implementation aspects (*service* and *port*) of the service, the abstract definitions are mapped into *tModel* (categorized respectively as *PortType* or *binding*) since this abstract structure is used to represent unique concepts or constructs for classification goals. Then while a *service* is naturally mapped to *businessService*, the *port* that implements a particular *portType* using the protocols defined by a named binding refers to the *bindingTemplate* element, which contains

technical information about a service entry point and construction specification. The service address is thus listed as the value of the *overviewURL* element, while the *tModel* that references the WSDL file has the categorization taxonomy of *uddi-org:types* and a value of *wSDLSpec* applied to it by using a *categoryBag* element. A *bindingTemplate* structure is created for each unique URL access point used by the service, and it refers to the *tModels*. The web resources service may be found with the *inquiryURL* API by searching binding *tModels* that have the required namespace and binding name. However, this is a poor query model based on a keyword match that simply locates a service based on a description of its functionality. Moreover, the search should be done directly to the registry. In a similar way, mapping (a *tModel* for the interface and one for the binding element) could be done within WSDL 2.0 and UDDI (Figure 7) even if this is not validated by OASIS.

B. Discovery tools in grid

In an EGEE-compliant grid, discovery is done by making use of specific tools. Since discovery follows resource indexing, one study [17] has tried to include UDDI and thus WSDL schema in an LDAP implementation by referring to the RFC-4403 (LDAP Schema for UDDIv3, <http://www.ietf.org/rfc/rfc4403.txt>). The typical grid process requires that the client application to access the resource send all its requirements encapsulated in a job and described in the JDL language [19]. This description, which is derived by the Condor (<http://www.cs.wisc.edu/condor>) Classified Advertisements (classads) library, makes use of the mapping (the classad) from attribute names that are used in resources' schemas to describe operations and requirements to expressions. The WMS accepts this job, examines the specified requests and searches for the appropriate node (CE) to send it for execution. The final decision about which resource should be used is the outcome of the matchmaking process between submissions requests and available resources. In this context, any application, considered to be a software package installed on a site, is usually published by the BDII LDAP server or the R-GMA database with a string identifying the name of the software, and the *RunTimeEnvironment* attribute is used inside the JDL file to discover the application (Figure 8).

In the same way, to make use of the UDDI software in the grid discovery process, its name should be searched in order to discover the specific grid nodes where it is installed. Then the query for the service should be also included in the client program that aims to access the web service. For example, a specific querying of the UDDI registry may be done using the Ruddi library (<http://www.ruddi.biz>) that gives features for allowing UDDI access. Using it, the search can be focused on *tModels* types. To make use of the grid, such a Java client application (Figure 8) should be included in a JDL file that also specifies the environment requirements and is sent to the grid workload management system by use of the grid commands (i.e., *glite-job-submit*).

These approaches that are based on keywords (objects are treated as character strings) are of no help to search on the basis of what web services provide. Furthermore, the data information model that is at the core of both solutions does not describe all their functionalities. Probably, the SD APIs will be able to interact with more complex mechanisms such as UDDI registry even if as it is this method lacks automation.

Moreover, the work that middleware developers are doing on the schema involves also queries issues: defining a schema for each service type, identifying which attributes would be considered static and which are dynamic and determining whatever the information returned by the query can be reduced to key-value pairs.

V. UDDI AND DISCOVERY SEMANTIC INFORMATION

While grid discovery modules implemented in the broker do not support semantic description and discovery, the adoption of a UDDI registry as one of the discovery methods in the grid environment could carry on to the realization of a semantic discovery. The validity of such a solution is also confirmed by the fact that other grid projects have introduced enhanced versions of this standard. An example is the Grimoires software (Grid Registry with Metadata Oriented Interface, http://www.omii.ac.uk/mp/mp_grimoires.jsp), the registry solution provided by the University of Southampton (and managed program of the OMII-UK, Open Middleware Infrastructure Institute) deployed as an extension of the UDDI framework.

As has been shown, semantics may be added to the provider/client interaction by annotating the web application and to the broker component that should be able to store and process such information [20]. From the broker perspective, a registry should combine an ontology language with a semantic discovery, and thus the means to store and to interpret the information in the matching. Enhancing an UDDI-based registry for semantic ability includes the storing of a semantic description inside the registry and then the deployment of a semantic matchmaker module able to process the semantic descriptions present in the UDDI advertisements. The solution proposed makes use of an enhanced module for a jUDDI-based registry, the matchmaker software that has been developed by the

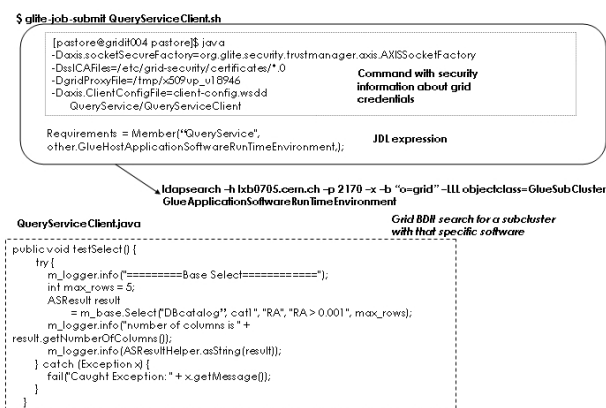


Figure 8: An example of JDL file and of the java client application needed to execute the QueryService web service from the grid

Intelligent Software Agents laboratory of Carnegie Mellon University (<http://www.cs.cmu.edu/~softagents/>).

A. An OWL-S-enabled UDDI registry

The OWL-S matchmaker [21] is client-server software that extends UDDI with a capability port (APIs) to send, receive and process semantic queries. A layer over UDDI is developed to handle semantic data, which is based on the mapping between OWL-S profiles and UDDI records through *tModels* since OWL-S provides a capability-based mechanism and in the same way the mappings that link the model to the required WSDL files. The software includes the jUDDI registry and an OWL reasoner (such as RACER, <http://www.racer-systems.com>) in order to process the OWL-S description present in the UDDI advertisement. A snapshot of the different components deployed on the grid node is shown in Figure 9.

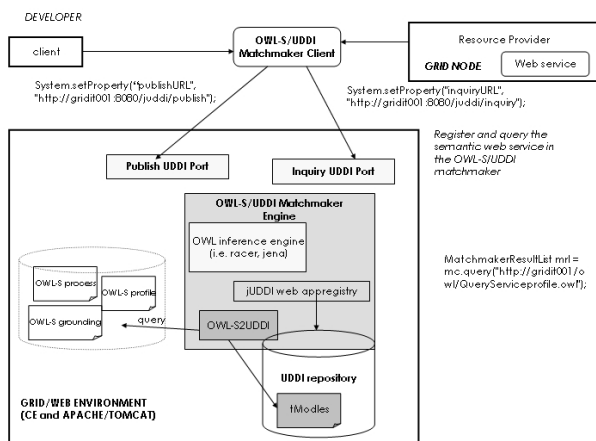


Figure 9: The OWL-S/UDDI client-server matchmaker and its components

From a client perspective, the client package provides methods to interact with the matchmaker by using APIs (inquiryURL and publishURL) that interact with the UDDI registry. The OWL-S profiles are registered in the UDDI, and the client discovers the agreed-upon semantic model by either directly using the Capability search object (since the queries are represented using *CapabilitySearch* class) or using an OWL-S profile URL that it loads over standard HTTP. When querying in the second way, the API maps the OWL-S profile to *CapabilitySearch* based on a mapping similar to the OWL-S to UDDI mapping. The OWL document representing the semantic model is located by finding the appropriate *tModel* and access to a service category.

B. UDDI and SAWSDL

The achievement of a registry able to process SAWSDL descriptions is related to the possibility of an analogous mapping of WSDL 2.0 definitions in UDDI that will enable UDDI queries based on artifacts and metadata. The first projects about this interaction are for example inside the METEOR-S project [22] that with the Lumina component aims at supplying a unified WSDL-S/SAWSDL to UDDI mapping structures, which enables the user to discover the services based on the operation

level. However, the testing of this solution is still in the feasibility testing stage.

VI. CONCLUSIONS AND FURTHER DEVELOPMENT

The need for deploying web and web service applications in an EGEE-II compliant grid system has within different projects, several problems. The main issue has been the type of shared grid resources that are in the same type poorly described by the grid data model and potentially useful for the technologies that surround web services. Experiences with grid information and discovery service have outlined some problems in dealing with such resources, thus limiting their use, indexing and discovery. Moreover, keyword-based techniques used for searching operations are a poor way to capture the semantics of a service request or service advertisement. To make use of such web resources, both clients and software agents need an interpretable description of the service and the means for access. This led to a semantic discovery process to facilitate the use and management of distributed resources and the grid environment. The first steps done to solve this problem have been the introduction in the grid of a UDDI registry as a complementary solution to the existing grid information service since its specialization for web service discovery followed in the first implementation by the mapping of UDDI schema on to LDAP schema. This approach has been revealed successfully with respect to middleware software updates and technologies improvement related to XML-based languages able to enhance service capabilities. The gLite middleware in fact has introduced the Service Discovery package that will be able to act as a front-end to whatever discovery mechanism and thus also UDDI. Moreover, this mechanism not only has been introduced in a grid environment (such as Grimoires) as service information, but there are many other projects working on the enhancement of UDDI-based registry to process semantic information all based on XML standards.

On the other hand, the possibility of implementing different mechanisms in the grid system is proved by the fact that there is an effort to provide a common data model able to integrate the different solutions present in the built grid. In the same way, the grid community is working on a semantic grid even if the actual solutions are not implemented in the EGEE-II system since based on GT4.

Work continues in adopting also in grid environment resources description languages (i.e., such as OWL-S or SAWSDL) able to describe the functionality of services in terms of inputs and precondition keys and output and the effects they produce and in implementing semantic discovery mechanisms. Using these technologies, grid consumers and grid providers through this kind of registry will take advantage of an environment of distributed resources. The focus is on testing available software developed in web service and semantic web services area to reach a greater integration with grid middleware and tools by grid client applications for resource discovery.

REFERENCES

- [1] A.S. Tanenbaum, M. Van Steen, *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.
- [2] I. J. Taylor, *From P2P to web services and grids. Peers in a client/server world*. Springer-Verlang London, 2004
- [3] Foster, I., Kesselman, D., *The Grid 2:Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd ed., USA, 2003
- [4] J. Davies, R. Studer, P. Warren, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, Wiley, 2006.
- [5] A. Baruffolo, J. Alcalá, M. Barbieri, M., Benacchio L., Cancian M., Capasso G., Cascone E., Castelli G., Pasian F., Pastore S., Pavlov M., Pucillo M., Smareglia R., Taffoni G, Volpato A., Vuerli C. "WP10: Astrophysics in GRID.it", In Grid-enable Astrophysics, Polimetrica international scientific publisher, 2007, pp. 13-25.
- [6] A. Volpato, G. Taffoni, S. Pastore, C. Vuerli, A. Baruffolo, R. Smareglia, G. Castelli, F. Pasian, L. Benacchio, E. Ambrosi, A. Ghiselli, Astronomical database related applications in the Grid.it project, *Proceedings of Astronomical Data Analysis Software & System (ADASS XIV)*, ASP Conference Series, Vol. XXX, 2005, pp. 15-13
- [7] EU Deliverable JRA1 Design team, EGEE Middleware Architecture, EGEE-DJRA1.1-594698-v1.0, EGEE, 2005.
- [8] S. Jablonski, I. Petrov, C. Meiler, U. Mayer, *Guide to web application and platform architectures*. Springer-Verlag, 2004.
- [9] O. Zimmermann, M. R. Tomlinson, Stefan Peuser, *Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects*. Springer, 2nd. Ed. 2005.
- [10] L. Richardson, S. Ruby, *RESTful Web Services*. O'Reilly Media, Inc. 2007.
- [11] S. Pastore, Implementing a Web Services Architecture Framework in a Grid Infrastructure, *GESTS International Transaction on Computer Science and Engineering*, Vol. 19, No 1, 2005, pp. 61-72
- [12] B. Daum, U. Metern, *System Architecture with XML*, Morgan Kaufmann, 2002.
- [13] A. Sheth, J. Cardoso, *Semantic Web Services and Web Process Composition*, Springer, 2005.
- [14] E. Shaya, B. Thomas, P. Teuben, Z. Huang, A Science Ontology for Goal Driven Datamining, in *Astronomy American Astronomical Society 207th Meeting*, 2006.
- [15] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, Grid Information Services for Distributed Resource Sharing, *Proceedings of the 10th IEEE Int. Symposium on High-Performance Distributed Computing*, IEEE Press, 2001.
- [16]] Howes, T. A., Smith, M.C., Good, G. S., and Howes, T., *Understanding and Deploying LDAP Directory Services* (2nd Edition), Addison-Wesley Professional, 2003.
- [17] S. Pastore, The service discovery methods issue: A Web Services UDDI specification framework integrated in a grid environment, *Journal of Network and Computer Applications*, Elsevier Science, In press, Corrected proof, 2006.
- [18] J. Colgrave, K. Januszewski, OASIS UDDI Spec TC, Technical Note, Using WSDL in a UDDI Registry, Ver.2.0.2, 2004
- [19] F. Pacini, DataGrid Job Description Language Attributes, release 2.x, DataGrid-01-TEN-00142-0_2, 2003
- [20] Li Kuang, Jian Wu, Shuiguang Deng, Ying Li, Wei Shi, Zhaohui Wu, Exploring semantic technologies in service matchmaking, *IEEE ECOWS 2005 Conference*, 2005.
- [21] Srinivasan, M. Paolucci and K. Sycara, Adding OWL-S to UDDI, implementation and throughput, *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)* San Diego, California, USA, 2004
- [22] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller, METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, *Journal of Information Technology and Management, Special Issue on Universal Global Integration*, Vol. 6, No. 1 (2005) Kluwer Academic Publishers pp. 17-39.

Serena Pastore began to work in 1995 as system and network manager at the Civil Engineering Faculty of the University of Padova, after the degree in Electronic Engineering at the University of Padova, in 1993. In 1999 she started working at the National Institute of Astrophysical (INAF) initially involved in the web area (i.e., web server management, web application design and development). In 2003 she was involved in INAF grid projects and from 2005 she also collaborates with the INAF Communication Office as technical solutions engineer to realize projects regarding education and outreach of Astrophysics. Finally from September 2007, she will begin to teach Computer Science at the University of Ferrara. Member of IEEE, ACM, OGF and of the Italian Association of Engineers, her research interests are internet technologies and web standards, distributed systems and mobile and wireless technologies.