Taxonomy and State of the Art of Service Discovery Mechanisms and their relation to the Cloud Computing Stack

George Kousiouris, Dimosthenis Kyriazis, Theodora Varvarigou
National Technical University of Athens, Greece

Eduardo Oliveros
Telefónica Investigación y Desarrollo SAU, Spain

Patrick Mandic
Rechenzentrum Universität Stuttgart, Germany

ABSTRACT

Service Discovery mechanisms are gaining interest in the last years due to the growing bulk of information available, especially to distributed computing infrastructures like Grids and Clouds. However a vast number of characteristics of these implementations exist, each one suitable for a number of purposes. The aim of this chapter is to extract a taxonomy of these characteristics found in modern Service Discovery systems and produce a categorization of existing implementations in a grouped and comparative way, based on these features. Furthermore, the mapping of these characteristics to the Cloud Business model is produced, in order to assist in selecting the suitable solutions for each provider based on his/her location in the value chain or identify gaps in the existing implementations.

INTRODUCTION

Locating resources in a large scale, heterogeneous network is a non-trivial task. There may be several providers, each one with different characteristics, thus offering a variety of different levels of user-requirement fitness. In order to allow awareness about the status and the availability of these resources, a Service Discovery (SD) mechanism must be installed. All Service Providers will require registering in such a mechanism, along with all the necessary information such as URL, rates, compatibility and interface, so that their services are advertised to the end user. The information service then makes all of these available to potential clients, by matchmaking the request with the available resources and returns back the results. A high-level overview of SD's generic architecture is illustrated in the following figure (Figure 1).

In the current model of distributed infrastructures, a number of different artifacts may be addressed. What needs to be discovered is services, their interfaces, resources and their interfaces and their matchmaking in order to produce added value.

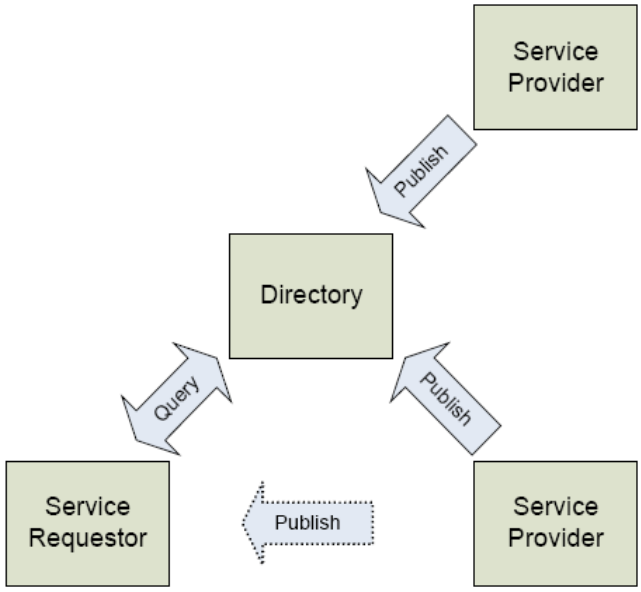

Figure 1 – SD generic architecture

Figure 1: Service Discovery Generic Architecture

During recent years, specific focus has been given to adding value to such mechanisms, through adding functionality on top of this registry. This may refer to the way the registry mechanism is built, deployed or enriched, to more advanced protocol usage that enhances interoperability and management or the use of semantics and processing for increased functionality.

The basic aim of this chapter is to provide an in-depth presentation of the State of the Art with regard to Service Discovery systems and methodologies in distributed computing. This is combined with an identification of the most important characteristics of such systems, taxonomy of them based on the latter and comparison. Finally an identification of future trends will be pursued, in order to suggest which of the presented solutions show the higher perspective. The structure of this work is heavily based on the structure of Yu and Buyya (2005), which to the view of the authors is a model example of how to write taxonomies and surveys. In comparison to this, what is pursued is a specific focus on Service Discovery systems and not in general of workflow systems, an update on the state of the art and the enrichment and rearrangement of parts of the taxonomy that is presented in the former in order to include new advances and features.

TAXONOMY OF DIFFERENT DISCOVERY MECHANISM CHARACTERISTICS

Service Discovery mechanisms can be categorized according to a vast number of characteristics. These features may include their functionality, offered features or implementation differentiations.
The major categories that have been identified are:

Mechanism Structure

This feature regards the design and implementation of the entire service discovery mechanism. This has an effect on the locality of the components as well as the protocols of communication.

The different SD mechanisms can be divided in two groups:

- *Adhoc SD:* This group comprises those mechanisms with the ability to look for services in ad-hoc networks with no fixed infrastructure. Such SD mechanisms do not usually have a need for a central repository but are limited to the concrete network they are located in and are not able to interact with nodes located in other networks. The most representative protocols of this type are: Service Location Protocol (SLP), Zeroconf (based on multicast DNS), UPnP, Jini and Bluetooth's SDP.
- *Wide Area SD or Infrastructure SD:* This group is used in infrastructure networks and involves a central repository where:
    - The descriptions of the available services are usually published by service providers, and
    - The service requestors perform their queries.

The scope for infrastructure SD mechanisms extends to more than one IP network comprising an administrative area network or a site. Implementations incorporate centralized solutions as well as distributed. Furthermore, with regard to the latter, further categorization may apply with regard to the specific nature of the implementation. Characteristic solutions include a hierarchical implementation or P2P implementations.

Information Retrieval

This feature regards the way the information is relayed to the SD mechanism. This can be either static, in the beginning of the advertisement, or it can be refreshed in a given interval. It can also incorporate interfaces and specifications for enabling support for mobile devices, like the SIP protocol, or based on specific events.

Knowledge Modelling

In order to represent knowledge with regard to the context of the registered services or resources in general, a number of ways exist in order to semantically represent this information. This may include semantic annotations and tagging for example through suitable keywords, or the simplest case, attributes in the form of key value pairs.

Another approach is to use structured semantics such as ontologies in order to store and correlate the information from the various sources. This helps in determining not only the important keywords but also hierarchically classify the numerous concepts that may be used to describe a service or resource. This also aids in avoiding different levels of semantic descriptions for the latter.

Knowledge Processing

Regardless of the way the semantic information is stored or represented, the processing of this information in order to perform matchmaking between requests and offers is of extreme importance. This matchmaking refers to the following cases

- Matching client requests to available services
- Matching service requests to available and suitable hardware resources
- Automated composition between different services in order to create adhoc applications with added value that meet the client's requirements in case a service is not found that satisfies the latter. This is mainly referred to as automated composition.

Specifically, this composition may be automated or user-aided, in the form of final approval of the composed workflow of services. Furthermore, the matchmaking may be based on partial or full success rates, use of techniques such as reasoning or based on the class hierarchy that is extracted from the ontology structures.

Specifications

A number of specifications and protocols exist for a variety of needs that arise in implementing an SD mechanism. The needs may vary from the form the registry is implemented, the way information is relayed in an ad hoc network (for the according cases), the way interfaces are going to be designed or the use of semantics in the mechanism.
(WSIL, 2007) is a specification (and XML schema) for aggregating pointers to service descriptions that already exist in repositories, in order not to duplicate them. These descriptions may be written in different formats. Furthermore, a set of rules for accessing this information is defined.
The basic technology standard for implementing service registries is based on the UDDI Specification (2004). The UDDI catalogue consists of three kinds of pages:

- Yellow pages that contain the taxonomy of services in order to locate them based on the category
- White pages that contain contact information and
- Green pages containing the technological information that describes the behaviors and support functions of a Web Service.

The Representational State Transfer protocol (Fielding, 2000) is based on a representation of entities and functionalities as resources. The management of these resources is based on specific standardized interfaces like the GET, PUT, POST and DELETE operations of HTTP. In contrast to the competitive SOAP protocol

(http://www.w3.org/TR/soap/), it does not allow users to specify their own interfaces in the style of inputs, outputs and operations. This leads to improved interoperability. An extensive description of the differences between SOAP and REST can be found in Zhao and Dossi (2009).
SIP is a protocol designed by the IETF (http://datatracker.ietf.org/wg/sip/charter/) for session control on IP-based networks. Although SIP is independent from session specific issues, it is commonly related with Voice over IP (VoIP) scenarios. A session in this context is understood as an exchange of data between associations of participants. This concept can be very useful for Internet applications that rely on a relationship between two or more participants, as it allows taking session handling logic out of specific application logic. SIP provides mechanisms to handle, i.e. create, modify and terminate, these sessions, independent of the type of session being handled. SIP was designed with modularity in mind.
(WS-DD, 2009) is an OASIS specification for allowing dynamic discovery of services by clients in ad-hoc or managed networks. The clients send a probe request while searching for a suitable service, which in turn replies to the sender. WS-DD supports also the use of a Discovery proxy for limiting multicast traffic in a managed network. Furthermore, a new service may send a notification to the same multicast group, to which the clients listen, in order to minimize the polling needs of the latter.
The Web Service Modeling Ontology (WSMO, 2005) is a conceptual framework and a formal language for describing the semantic aspects of Web Services, so that automated discovery and composition of Web Services can be achieved. Through its four elements, ontologies, WS descriptions, goals and mediators, the functional and behavioral aspects of a Web service, the user requirements and the automatic handling of interoperability issues may be addressed.
(WSDL, 2001) is “an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint”. It is the most adopted specification for expressing a service’s functionality to the external world.
(SAWSDL, 2007) is the W3C recommendation of the Semantic Annotations for WSDL and XML Schema specification. Its main goal is not to define a language for semantically representing concepts but to propose mechanisms for connecting external semantic model concepts to internal WSDL or XML components.

Type of Advertised Information

With regard to the type of advertised information, various levels of detail can be followed. These may be complementary to one another. For example, the services or resources in general may advertise their state (available/unavailable) or status (current workload etc.). Furthermore, they may advertise non functional parameters (like QoS characteristics) that are supported by the specific service instance. This of course has an immediate correlation with the semantic representations that must be sufficiently rich in order to depict this. Finally, their functionality must be advertised, which mostly relates to their interfaces and their provided operations. Especially for services, this is mainly performed through the Web Services Description Language.

In the following figure (Figure 2), the taxonomy of Service Discovery in relation to the identified parameters and characteristics is portrayed.

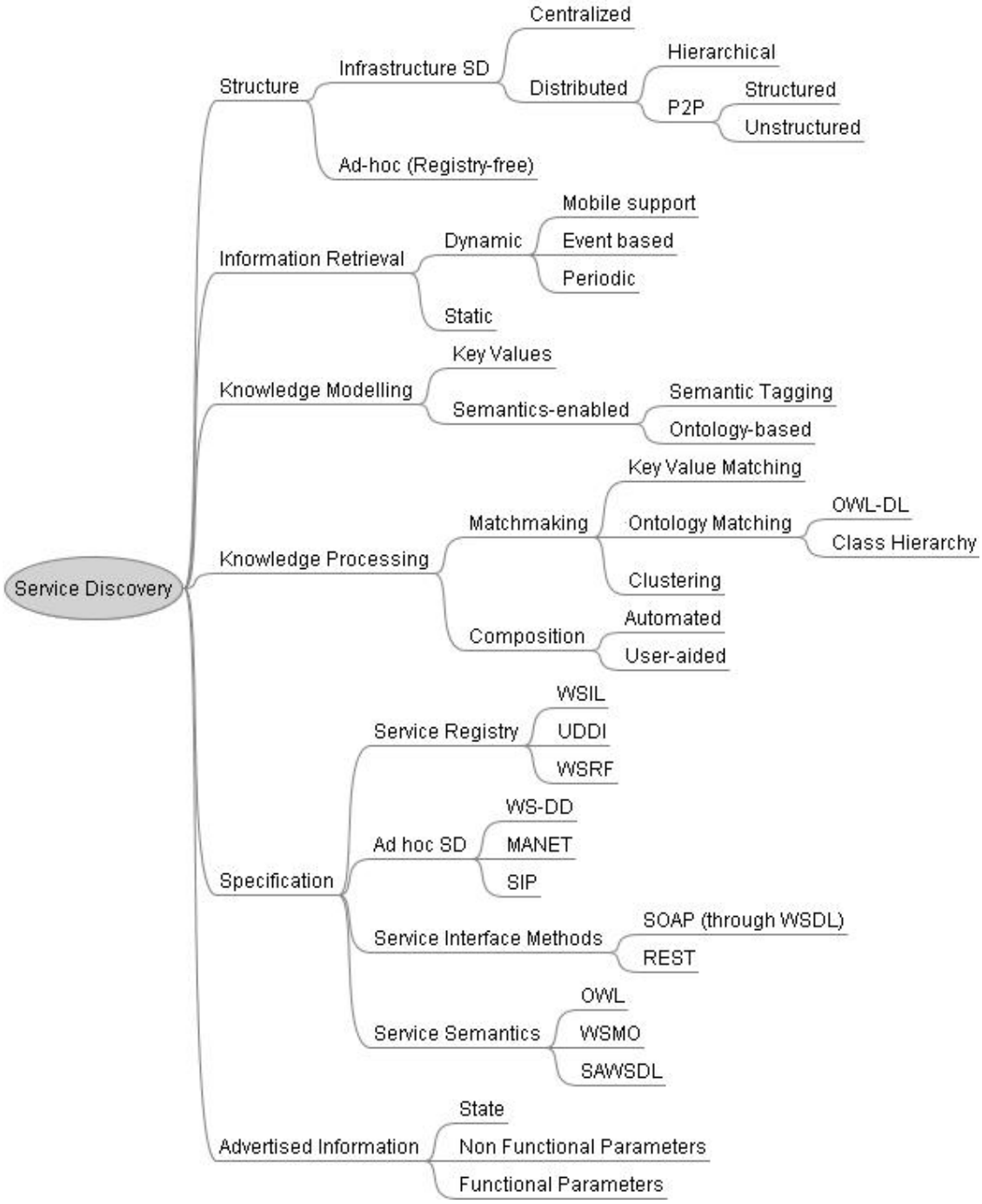

Figure 2: Taxonomy of Service Discovery Features

STATE OF THE ART MECHANISMS- APPROACHES, IMPLEMENTATION AND COMPARISONS

In this section, an analysis of currently used Service Discovery mechanisms is given. This includes sample implementations from all of the above categories, however it must be stressed that due to the extension of the research topic in recent years, these solutions in many cases are limited to only parts of an SD functionality or characteristics as these were described above.. The latter are compared with regard to the aforementioned taxonomy, in order to extract their characteristics.

UDDI

As seen in Section 2, UDDI is a protocol for Service registries. Implementations based on UDDI have gained popularity during recent years. It is based on XML for message exchange but while it is widely adopted and implemented, it is extremely static. Services are registered initially but there is no dynamic monitoring of their state. If a registered service becomes unavailable then there is no notification to the user. Furthermore there is a major manual intervention, either for administrative purposes or for the user in order to make the required selection. The several UDDI processes and mechanisms cover solely operational aspects of the UDDI cloud, data management as well as replication aspects. They are designed and are suitable for dealing with explicitly published changes to the registry data. Such changes are typically performed by operators or publishers. While these processes can be regarded as an approach to automatically handle changes in the registry, they do not represent a solution for the problem of dynamic service invocation or fault tolerance.

ebXML

Another approach to this kind of service is ebXML(http://www.ebxml.org). Its main goal is to make business processes easier and automated. It allows businesses to find another, define trading-partner agreements and exchange XML messages in support of business operations. Its messaging specification is based on SOAP with attachments, but does not use WSDL, a feature which can prove to be limiting. It is mainly focused for business exchange so its flexibility is also limited.

Sun Service Registry

As far as implementations are concerned, for UDDI there is a vast number of them but the one that prevails is the Sun Service Registry (2005), an open-source, Java based utility that implements UDDI and ebXML standards. It supports Web Service registry functions, organization, storage, control of any kind of service metadata or artefact, federated Web Service asset management across multiple repositories, core infrastructural support for service discovery, lifecycle management and SOA governance, security, APIs as well as database abstraction layer. It is tightly integrated into the Java Enterprise System but there is no report about monitoring capabilities or the support of a distributed architecture.

IBM WebSphere Registry

Another implementation is the IBM WebSphere Registry (http://www-01.ibm.com/software/integration/wsrr/). It is a proprietary solution that uses metadata repository for service interaction endpoint descriptions, supports both traditional Web Services (SOAP/HTTP) and SOA services (WSDL, XSD). It is more reliable than open-source packages and optimizes the use of services in SOA by exchanging rich service information with runtime monitoring tools and operational data stores. One question rises about the fact that it does not seem to manage service metadata across the whole SOA life cycle and of course the matter of cost and extensibility in comparison with open source solutions.

MDS

Another popular scheme is MDS, the Monitoring and Discovery Service of the Globus Toolkit 4 (http://www.globus.org/mds). This service also monitors the state of each resource and provides a multi-level hierarchy organization and multiple indexes. It also features WSRF implementation, VO (Virtual Organization- the dynamic concentration of resources for mutual use) support (good for collaborative applications), plug-in specialized functionality, common VO-level service functionality, monitoring and updating of the states of services and resources. Finally, WSRF implements large-scale, reliable distributed monitoring and systems. The main limiting feature of MDS is the query based information service for matchmaking between requests and resources. It is primarily focused on keyword matching

In order to deal with these inadequacies, various schemes have been proposed in order to combine the power of MDS with semantics. (RDF, 2004) is an XML specification/format for describing resources and their properties. Attributes of resources correspond to traditional attribute-value pairs but there are also properties that represent relationships between resources. In a way it is similar to object-oriented programming languages and their concepts and it is extensible. An extension of RDF is the Web Ontology Language (OWL0). OWL uses the power of semantics in order to cope with the restrictions posed by keyword matching. A number of Web Services can be retrieved that may have similar functionality which is hidden when simply viewing their WSDL. With OWL, web services are modelled with ontologies and the semantic representation of concepts and their relations is exploited in order to achieve semantic matching. The basic languages to obtain semantic description from web services are DAML-S and OWL-S (2002). OWL can be added as another layer in existing implementations such as UDDI or MDS, allowing the creation of a hierarchy of concepts in manner that when a query is made the possible returned result is not necessarily syntactically the same with the requested but its relation with the wanted one is derived from the hierarchy. The relations between concepts are depicted in a knowledge tree.

gLite

A useful combination of MDS and semantics is the gLite system (Burke et al, 2007). It integrates MDS with the Relational Grid Monitoring Architecture (http://www.r-gma.org/) in an attempt to improve the query and matchmaking capabilities of the former. The GLUE information model (http://forge.gridforum.org/sf/projects/glue-wg), a

standardized description of a Grid computing system to enable resources and services to be presented to users and external services in a uniform way, is used. RGMA supports relational database-like SQL queries and there are also a CLI, an API, and a Web browser interface available, so that users could publish their own data. The Producers (offered services or infrastructures) feed their information in this relational database, where Consumers (searching for services or infrastructures) subscribe to perform queries based on their constraints. Multiple rows with primary key and a timestamp for information freshness is used for the Producers, so that RGMA queries can be made concerning different time periods (latest, history, streaming results) according to the update requirements. Latest will give more accurate information but history means that there is a larger set of results. Servers collect data from both local clients (primary producers) and other servers (secondary producers).

GridBus

A very useful combination is the one in GridBus (Somasundaram et al., 2006) where MDS is used for the monitoring and management part and OWL is used for the registration and the matchmaking process. This system uses ontology templates, meaning domain specific ontologies that provide hierarchy of concepts along with properties to define their characteristics. Any resource can be modeled as an instance of a specific concept provided that the resource can be described using the properties defined in that concept. Upon resource registry, an instance of an appropriate resource concept in the ontology template is created. If an exact match is not found, then a higher level representation of what is requested by the user can be found. It is based on the knowledge tree and allows the avoidance of a total miss. Specifically, if the user provides a label that does not exist then instead of a failure the user inserts the level where the new label should be, in order to enrich the knowledge tree. A Protégé-OWL editor is used to create an extensible ontology template and Protégé-OWL libraries to dynamically create knowledge base of grid resources. An Algernon (http://algernon-j.sourceforge.net/doc/overview.html) query is used between users and OWL. Algernon is used in order to perform inference on data in knowledge bases and is compatible with the Protégé knowledge base.

S-MDS

In a similar architecture, the S-MDS implementation (http://wiki.dbgrid.org/index.php?smds) describes the resource properties (RP) using ontologies through a mapping procedure. Resource properties describe the resource/service state but they deal only with the document structures and not with its contained semantics/concept. This way querying problems appear. With S-MDS, through the mapping process these properties are transformed to ontologies and are stored and maintained in a WS or RDF repository so that it can be semantically queried. In the beginning there is a fully automatic correlation between RP and Domain Specific Ontologies (DSO), then a creation of a DSO instance based on values defined in the RP and afterwards an enrichment of the DSO by importing other ontologies (DSOE). Finally, a DSOE instance is created based on user inputs. SPARQL or RDF query language is used and a GUI is provided for easy SPARQL creation.

Decentralized Resource Discovery

Regarding other approaches, a decentralized Grid resource discovery system based on a spatial publish/subscribe index is presented in Ranjav et al.(2007). It utilizes a Distributed Hash Table (DHT) routing substrate for delegation of d-dimensional service messages. The Grid brokers create a Chord overlay, which collectively maintains the logical publish/subscribe index to facilitate a decentralized resource discovery process. *G*rid resource query rate directly affects the performance of the decentralized resource discovery system, and at higher rates the queries can experience considerable latencies. Furthermore, system size does not have a significant impact on the performance of the system, in particular the query latency. The basic dependency is the use of Chord and the main advantage the increased fault tolerance of this scheme. The same features exist in MDS and gLite, where also the refresh rate of information can be configured according to the administrator's needs and network performance. A P2P implementation of GT4 appears in Bharathi and Chervenak (2007).

NWS

A distributed system that monitors and predicts the performance of various resources of the Grid is the Network Weather Service (http://nws.cs.ucsb.edu/ewiki/). It has sensors that sample a number of parameters of the resources and then uses a range of algorithms and numerical models in order to predict the near-time performance of these resources. The best fitting algorithm is chosen for every resource according to the prediction success. This is a very attractive scheme that enables a very dynamic view of the network leading to a very high efficiency level. One question is about the speed of selection due to the fact that a number of algorithms has to be tested. Another point is the level of prediction success, especially in accordance with real-time attributes.

Collaborative Tagging

Fernandez et al (2008) present a method based on Collaborative Tagging for matchmaking services to requests. In SWS models, like OWL-S, DAML-S etc., the service provider has the responsibility to handle ontologies and semantic descriptions. However his/her approach may be different than the one the end user that is going to search for a service is following. In this paper, a combination of the ontologies in addition to free text tagging of services by end users is performed. Each service is semantically annotated by end users through tags. These tags constitute the tag-cloud of the service. Each tag has a different weight depending on whether it has been selected by many users. Different clouds exist for describing a service's input, output and operation. At the next level, a hierarchical dendrogram of tag descriptions is created based on clustering techniques that process the clouds from step 1. The basic advantage of this method with regard to semantic descriptions is the merge of two worlds, the service as this is described through its interfaces, and the user's notion regarding its context and usage in the real world.

EASY

EASY (Mohtar et al, 2008) is a framework that can be used on existing SD mechanisms in order to enrich the use of semantics for both functional and non functional (like QoS) characteristics. It is comprised of two main components. The EASY-L is a language for semantic service description that is extensible and contains required information for semantic service matching. EASY-M is a matchmaking mechanism for both functional and non functional parameters that enables both complete and partial matching. Furthermore, due to its rich descriptions, services can be matched with resources or other services. EASY is an add-on on existing SD mechanisms, however it must be deployed on every node participating to the discovery process. These mechanisms can be either centralized or distributed. If the request is not matched with the repository in the vicinity of the client, it is forwarded to other registries.

The major processing workload is during the service advertisement. Based on a DAG approach, the new service is entered on a existing ontology according to the semantics of its description. Thus the online matching is performed much faster. Furthermore, the partial matching of requests could potentially lead to the usage of this implementation for service composition also.

RESTful WSs

In (Zhao and Doshi, 2009), an approach using RESTful web services is presented. In this work, a formal description of the RESTful Web service composition problem is presented along with a formal model for classifying and describing RESTful WSs. The model is based on the description of RESTful WSs as ontology resources and “state transfer” of ontology resources. By using this formulation, a situation calculus based state transition system can be used for automating composition of services.

Publish/Subscribe Broker

In (Hu et al, 2008), an approach based on the publish/subscribe messaging paradigm is presented. In this case, a distributed system is used for the service composition part. This helps in scaling the performance of such a system, that in centralized service repositories is unfeasible. The authors exploit the fact that interaction relationships among the data senders and receivers in a pub/sub system is almost identical to the problems that are faced during service composition. Thus by modeling services using content-based pub/sub messages, existing work on matching of these messages is exploited. In the pub/sub case, each publisher produces data that are forwarded to a channel, to which subscribers have registered in order to receive them. Constraints can be set as to what kind of information a subscriber may get. The use of semantically rich ways in order to express these constraints is not dealt with in this work, however this constraint matching is the basis of the composition functionality. A broker network is used, where service and request agents as well as brokers can join or leave at any time, thus resulting in a decentralized, highly dynamic infrastructure. The process search algorithm finds for a suitable DAG of services that meet the request made in the advertisement. Overall, it is a very promising approach. The semantic capabilities can be enhanced while the lack of any specification used for describing services may be a limiting factor.

Group-based Service Discovey

Group-based Service Discovey (Chakraboty et al, 2006) is an SD mechanism based on the P2P paradigm. Its main goal is to exhibit a high rate of flexibility in order to include pervasive computing environments such as mobile agents in addition to standard distributed infrastructures. No central repository exists for the services or resources to register their existence but the approach is based on dynamic caching of advertisements. Due to the fact that syntactic matching is not efficient in pervasive autonomic environments due to heterogeneity of implementations and interfaces, the authors have used OWL. Through this implementation, semantic capabilities are exploited in order to describe capabilities of services and resources in this environment. The services are also classified into groups based on the class-subclass hierarchy of OWL. This group information is used in order to route the requests towards the most suitable available services. Due to this differentiation, this implementation is efficiently utilizing resources without the need for expensive, in terms of bandwidth and delay, solutions like broadcasting.
In general, the SD lies on three steps:

- Advertisement of services according to vicinity
- P2P dynamic caching of advertisements
- Forwarding of discovery requests based on the service group defined through the ontologies

More information regarding numerous P2P discovery systems in Grid computing can be found in Trunfio et al (2007).

FUSION

The FUSION Semantic Registry (Kourtesis et al, 2008) is built upon a UDDI registry, exploiting the fact that in the latter, references can be given towards the specification with which a description of the functional and non-functional characteristics of a service can be found (like in a WSDL document). In order to support however more fine-grained matchmaking between the requests and responses of the SD mechanism, the authors follow also SAWSDL semantically annotated descriptions of service interfaces in addition to OWL-DL for performing fine-grained matchmaking through DL reasoning.
Regarding the advertised information, FUSION supports both functional and non-functional properties (like QoS) and extensions can be performed in order to enrich the level of advertisements.
In terms of architecture, the FUSION Semantic Registry is independent from the UDDI server implementation and is external to the latter, with the main objective to increase the syntactic and matchmaking abilities. It exposes two interfaces for publication and discovery functions. Operations such as SAWSDL parsing, OWL ontology processing, and DL reasoning are performed internally.

Akogrimo Service Discovery (SIP and MDS)

The Session Initiation Protocol Service Discovery (SIP SD) is a mechanism implemented using the SIP technology (RFC3261, IETF). This particular solution was developed

within the frame of the Akogrimo IST project (http://www.mobilegrids.org/). This mechanism was developed to discover devices connected to the network, such as printers, webcams, proxies, time servers, etc. More information can be found regarding this implementation in Olmedo et al (2009). In general, the approach connects the Akogrimo EMS (Execution and Management System), a distributed system that utilizes the Globus MDS,, with a SIP enabled framework for discovery of mobile agents and resources. Due to the fact that the MDS has been thoroughly studied in the previous sections, and that SIP-enabled Discovery mechanisms, to the best of our knowledge, are very rare in the bibliography, the main focus of the analysis lies on the description of the SIP SD mechanism that follows.

The core of the SIP SD is based on two SIP extensions, namely "SIP – Specific Event Notification" (RFC 3265) and "SIP extension for Event State Publication" (RFC 3903). SIP (Specific Event Notification) creates a framework for communication of events between SIP nodes in an infrastructure. Event packages are used in order to define the events that can be communicated. The SIP extension for Event State Publication is used by a SIP node in order to forward the information to a corresponding agent This extension is used in the SIP SD mechanism to provide the agents with all the necessary data.

The SIP SD mechanism of the Akogrimo project provides some interesting characteristics:

- The ability to alert in case of service changes or subscription to services before they go online.
- Reduced network traffic, due to its compliance with the subscription- notification concept.
- A description language (XML) with more expressivity compared to common attribute-value pair approaches. This can be extended with enriched, semantic, XML based languages such as Ontologies.

Through these features, the mechanism enables higher levels of expressivity and functionality and enhances the classical SD concept with a more flexible and powerful implementation, that can also support mobile resources to be incorporated in the framework.

Publish/Subscribe Event-based mechanism

In Yan et al (2009)0, a resource discovery mechanism is presented that focuses mostly on the depiction of the state of the resources throughout time. The major contribution of this work lies on the fact that the status of the resources can be monitored in real time. Furthermore, the mechanism is event-based in order to notify the users, based on a publish/subscribe mechanism. The intermediate brokers deal with subscription for advertisements and routing of requests to the proper resources. The high level of dynamicity and scalability in this implementation are very attractive features. Four classes are implemented, regarding the level and flow of information, based on whether this information is static or dynamic. One disadvantage is the fact that the matchmaking

is based on key value pairs, which has limited semantic capabilities. This solution is mostly aiming at resources and not services.

MUSIC

MUSIC (Rouvoy et al, 2009) is a component-based planning middleware and framework that has the goal of enabling discovery in ubiquitous environments. It is mostly focused on exchanging current services with replacement ones for self adaptation purposes, however the way this is performed can prove to be helpful for automated composition cases. It focuses also on QoS features. The application is modeled as a series of components with given functionalities. The latter can be dynamically configured with conforming component implementations. So the purpose of the framework is to search for alternative services that meet these requirements. Components (e.g. services) can be plugged in for replacing disappeared components with similar functionality. The planning capabilities of the platform are based on a meta-model. In this model, the composition is described as a set of Roles that are collaborating through Ports. The latter represent functionality that is provided or required from connected components. Properties are used to describe the association between QoS levels and the resource needs. Each port has a Type that represents its needs in terms of interfaces and protocol. So during an automated composition, the comparison of these can aid in determining whether the interconnection of two components is feasible. This feasibility involves also QoS features. This planning activity is key for selecting the fittest services that provide the end-to-end desirable output to the requester.

The MUSIC platform supports a number of SD protocols like SLP and UPnP. A service can register to these mechanisms and advertise its presence, along with a static description of its offered QoS, with regard to its non functional capabilities. For the dynamic QoS offered by the service, this is dependent on the node that hosts the former. So it is adapted later on, depending on the load. Discovery listeners may register to different MUSIC platforms in order to be aware of new services that register. It extends OSGI in order to be able to support distributed services and to be able to handle different communication and discovery protocols. The main disadvantage of this platform seems to be the fact that the services are bound to specific nodes of execution, a fact that is contrary to the modern Cloud Computing Stack, as this is described in the next Section. If this limitation is lifted then this solution is one of the most promising ones. Another hindering factor is the limited use of semantics. An addition of an ontology on top of the meta-model described earlier could help in the enhancement of this framework.

A thorough survey and classification especially for service composition frameworks can be found in (Ibrahim et al, 2009).

Goal-based Service Ontology

Goal-based Service Ontology (Bonino da Silva Santos et al, 2009) is a framework that has the aim of assisting non-technology experts in service discovery and composition. It uses the concept of goal for representing abstract service requests. The approach is based on goal modeling and has as a prerequisite the need for shared domain specific ontologies between the involved parties. Through ontological representations, the tasks that are valid are correlated with the specific goals that domain users may have. Furthermore, the context-aware platform is based on the aforementioned semantic descriptions and uses

contextual information from the user in order to select the appropriate tasks to aid in the fulfillment of the goal. The providers register their services and semantically annotate them by using the domain specific ontologies. The main focus is to link services with tasks and tasks with goals. Then, whenever a client inserts a new goal, the sequence of tasks and subtasks is defined through the semantic representations, allowing for the framework to deploy the necessary services.

Following, a comparative table (Table 1) of the most advanced solutions with relation to the aforementioned taxonomy features is presented.

Tool	Structure	Information Retrieval	Knowledge Modeling	Knowledge Processing	Automated Composition	Specification	Advertised Information
RESTful WSs (Zhao et al, 2009)	-	-	Ontologies	Situation Calculus	Yes	REST, OWL	Functional
EASY (Mokhtar et al, 2008)	Dependent on the underlying SD	Dynamic	Ontologies	Ontology matching (+partial+subsumption)	-	OWL	Functional and Non-Functional (QoS)
Pub/Sub Broker (Hu et al, 2008)	Distributed, Ad hoc	Dynamic	Key value pairs	Key value matching	Yes, DAG based	-	Functional
Akogrimo SIP SD (Olmedo et al,2009)	Distributed, Hierarchical	Dynamic, Mobile	Key value pairs	Key value matching	-	SOAP, WS-*, SIP	State, Non Functional
GSD (Chakraborty et al, 2006)	Distributed, P2P	Dynamic, Mobile	Ontologies	Ontology matching (+partial)	Yes	OWL	Functionality (in terms of OWL service group)
FUSION	Centralized	Dynamic,	Ontologies,	OWL-DL	-	OWL, SAWSDL,	Functional,

Semantic Registry (Kourtesis et al, 2008)		service-driven	Semantic Annotations			UDDI	Non Functional
Event-based (Yan et al, 2009)	Distributed	Dynamic, event-based	Key value pairs	Key value matching	-	Publish/Subscribe	State (or status) in real-time
MUSIC (Rouvoy et al, 2009	Distributed	Dynamic, event-based	Meta-model	Comparison of Meta-model characteristics	Yes, Meta-model based	OSGI, SLP, UPnP	Functional and Non-Functional (QoS)
GSO (Santos et al, 2009)	Centralized	-	Ontologies	Ontology matching (service-task-goal)	Yes, Goal oriented	OWL and WSMO (service descriptions in any language)	Functional and Non-Functional (QoS)

Table 1: Implementations in relation to Service Discovery Taxonomy characteristics

What is shown from the above analysis is that there are numerous efforts, each with advantages and disadvantages. There is no clear solution that is fittest in comparison to all the others. Furthermore, there does not seem to be one solution that covers all aspects of the taxonomy. However, as it will be described in the following chapter, this may not be a limiting factor in the current computing environment.

Especially with regard to service composition, while considerable focus has been given on satisfying input and output criteria, the semantic functionality of services is not considered. Having the same I/O (or according descriptions) and protocol of communication, which are the basic criteria for composition approaches, may make two services compatible but it does not necessarily provide the functionality that is needed by the end user. For example, a service may have as input (and output) an image file, but the internal operations may be completely different (gray scale in one case or color enhancement in the other). So a gap that is worth pursuing is the ability to give services a functional semantic aspect regarding their internal operation on data.

FUTURE TRENDS AND REQUIREMENTS

In the future the most important feature will be the dynamic nature of networks and providers. Multiple Service or Infrastructure Providers will federate in order to achieve a better overall result and to increase their offered services with limiting cost by adding value from each contributor (The Future Of Cloud Computing, 2010)0. In order to manage such kinds of organizations the most necessary feature will be not just to discover resources but to also combine and monitor them and their availability. Furthermore, semantic matchmaking is another hot topic, due to the fact that an advanced solution in this field will allow the full automation of the selecting and negotiating procedure with minimal user intervention. Furthermore, the automatic comparison from multiple compatible sources will contribute to the discovery of more cost effective resources.

However, one issue that arises from the current business models, towards which modern Cloud solutions evolve, is the fact that with the creation of different roles for SaaS, PaaS and IaaS that are undertaken by separate entities, the flow of information from these layers to the Service Discovery mechanisms, typically residing in the PaaS layer, may be restricted. For example, an IaaS provider will certainly advertise information related to the supported types of infrastructures but will almost never do the same for information regarding the workload of these resources.

So, what is critical for an SD mechanism on the PaaS layer is to be able to express service and resource requirements in a semantically enriched way, including features such as functional and non-functional characteristics. In this level, three operations may be performed:

- Matchmaking of client requests to available services
- Composition of services, thus combining two or more services in an integrated workflow level in case the first step is not successful
- Filtering of IaaS providers that can accommodate all or part of the workflow

With regard to the second point, this was not the case in the past. In most cases, the service provider was also the one that provided the infrastructures for execution. With the advent of Cloud computing however, additional matchmaking must be performed, between the offered SaaS and the final platform of execution.

In addition, regarding the last point, what is seen as the most recent trend in Cloud computing is the federation of different Cloud providers through a number of scenarios such as Cloud bursting or multi-cloud execution cases. Typical examples of this kind are the projects RESERVOIR (http://www.reservoir-fp7.eu/) and OPTIMIS (http://www.optimis-project.eu/). The part that will be critical in such a distributed, multi-role environments, combining services from different providers with resources from potentially different providers, is the service composition and matchmaking part with semantic enhancements. This matchmaking may involve composing different services in order to add value and create more complicated and advanced combinations and

matching these components to specific IaaS providers (from a functional point of view) that according to their advertisements are able to accommodate the former. The matchmaking capabilities are in general critical, as also identified in Carrascosa et al (2009).

So in the PaaS level that orchestrates this process, what is crucial is to be able to accommodate SaaS to IaaS resources. Furthermore, what is needed is to be able to combine SaaS in order to create new products, based on mash-ups of services. Another point of importance is that the semantic descriptions of the service must be able to address also functional requirements in terms of platform of execution. This will aid the PaaS provider to find the best service that meets the client's requirements, match it with a suitable hardware resource from the IaaS layer and possibly extend it to a workflow of services through automated composition. State information is not critical for this layer, due to the SaaS paradigm. Through this, the software services that are offered by the PaaS provider are deployed in the IaaS layer as many times as needed.

In the following figure (Figure 3), the various layers of the Cloud Computing Stack in relation to the different SD mechanisms are presented, with an analysis of their functionalities.

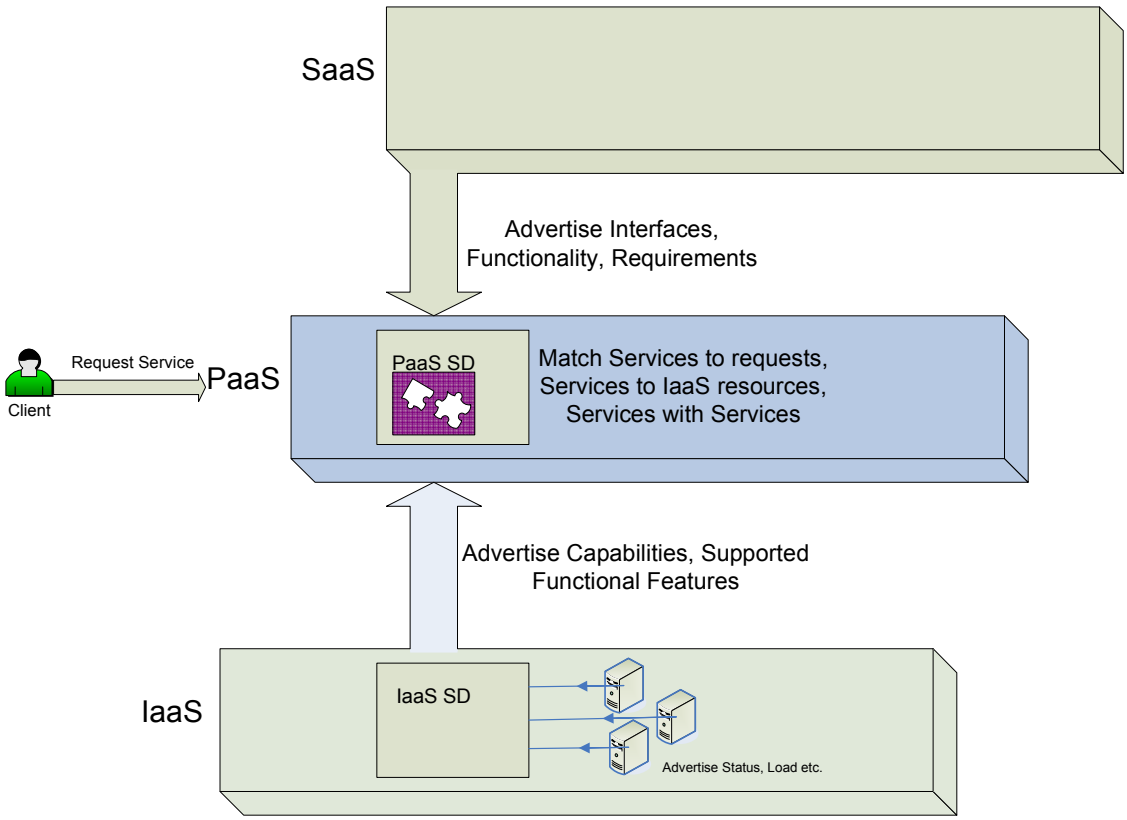

Figure 3: Cloud Computing Stack in relation to SD mechanisms

On the other hand, an SD mechanism that resides internally to the IaaS provider will be used in order to obtain information regarding the status of the resources. As seen in the previous paragraphs, the IaaS provider will almost never advertise, for competitive reasons, the workload or status of its resources. However, such an internal mechanism is critical in order to manage the infrastructure and proceed to intelligent and flexible decisions regarding allocation of tasks, for example with regard to scheduling decisions. For this reason, a highly flexible and dynamic mechanism is required, that will be able to include information such as state and QoS characteristics. This information must be always up to date, with a very high refresh rate. Furthermore, a distributed mechanism is necessary because these resources may not be located in the same geographical area or infrastructure.

Due to the fact that information will be passed to the PaaS layer in order for the latter to select the most appropriate provider, what is also critical for an IaaS provider is to give this data with a semantically rich description. For this reason, an SD mechanism with specific semantic capabilities towards external PaaS providers is necessary. This will aid the PaaS provider to perform better matches with the according infrastructures.

So as a conclusion for the optimal selection of which SD mechanism to implement and which characteristics this must address, the answer is that this depends on the level of use of this component. In the following table (Table 2), a concentrated list of needed characteristics for each layer of the Cloud Computing Stack in relation to the taxonomy identified in Section 3 is presented. The SaaS layer is not considered in this approach due to the fact that in general software will be transformed in a service oriented implementation and be adapted to an existing platform. So the latter is the one that decides which information must be provided by the Application Developer in order for this adaptation to take place. For example, with what semantic representation this software service will be described in order to be included in the platform is a choice of the PaaS provider. Furthermore, automated composition is considered separately due to the fact that its importance is expected to be critical in the upcoming years.

Cloud Layer	Structure	Information Retrieval	Knowledge Modeling	Knowledge Processing	Automated Composition	Advertised Information
PaaS Layer	Distributed	Dynamic	Semantics-enabled	Semantic Matchmaking	Yes	Functional, Non Functional (with regard to service capabilities)
IaaS Layer	Centralized	Dynamic	Semantics-enabled	Needed in order to	No	Functional, Non

External			(Ontologies)	classify internal resources and advertise their type		Functional (with regard to service capabilities)
IaaS Layer Internal	Distributed	Extremely dynamic	Semantics-enabled	No	No	Non Functional and State

Table 2: Requirements for the SD mechanisms for each layer of the Cloud Computing Stack

Based on this analysis, there are solutions that can handle SD mechanisms, at least in the IaaS layer as seen from Table 1 of Section 3. What seems to be the most obvious gap in research up to date is the fact that service composition in many cases is user directed and actually semi-automatic. Furthermore, one issue that is critical for the future and is not addressed today is the need to merge ontologies from potentially different providers, either at the SaaS or PaaS layer. This merge should be performed automatically, however this ability does not exist in modern SD mechanisms.

CONCLUSIONS

This chapter has performed a taxonomy of the characteristics that a service/resource discovery mechanism may address. This includes a variety of features that can be a part of these mechanisms.

A number of indicative solutions that implement part of these features have been presented in detail and with regard to their position in the taxonomy.

The investigation of current Cloud-based business models (in the form of the Cloud Computing Stack) and their specific per layer requirements has also been performed, in an effort to identify what are the most attractive characteristics (and therefore appropriate mechanisms). Gaps have been identified with respect to these approaches and the current business models.

As a conclusion, in recent years a large number of efforts have been implemented in the Service Discovery and Information field. Solutions exist with various features, with regard to dynamicity, level of information stored, method of processing, semantic annotation. Discovery mechanisms may be used for both hardware resources and software services, each one with a specific range of requirements for an efficient implementation.

There is no clear dominance of one solution over the others. Furthermore, the plug-in fashion of a number of these mechanisms is enabling their use with a number of different underlying technologies. Based on this and the analysis that has been made regarding each solution's characteristics and each layer's requirements, an actor may choose a suitable combination according to his/her needs and the level of complexity that is needed or wanted in each case. This selection may be further enhanced by what is

foreseen to be the most critical characteristic for each provider, depending on the latter's nature and objectives.

REFERENCES

Yu, J. and Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record.* 34, 3, 44-49. DOI= http://doi.acm.org/10.1145/1084805.1084814

Fernandez, A., Hayes, C., Loutas, N., Peristeras, V., Polleres, A., Tarabanis, K. (2008): Closing the Service Discovery Gap by Collaborative Tagging and Clustering Techniques. In *Proceedings* of ISCW 2008, Workshop on Service Discovery and Resource Retrieval in the Semantic Web

SAWSDL. (2007).*Semantic Annotations for WSDL* Retrieved from http://www.w3.org/TR/sawsdl/

WS-Discovery Specification (2009). Retrieved from http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html#wsdiscovery

WSIL specification (2007), *Web Services Inspection Language Specification.* Retrieved from http://www.ibm.com/developerworks/library/specification/ws-wsilspec/

UDDI Specification (2004). Retrieved from http://www.uddi.org/pubs/uddi_v3.htm

R. T. Fielding (2000). Architectural Styles and the Design of Network-based Software Architecture. PhD thesis, University of California, Irvine,.

Zhao, H. and Doshi, P. (2009): Toward Automated RESTful Web Service Compositions. In *Proceedings* of the 2009 IEEE International Conference on Web Services (ICWS 2009), 189-196, IEEE Computer Society.

S. Hu, V. Muthusamy, G. Li and H.-A. Jacobsen (2008). Distributed Automatic Service Composition in Large-Scale Systems. In *Proceedings* of DEBS 2008, pages 233-244.

M. Handley, H. Schulzrinne (1999), "SIP: Session Initiation Protocol", RFC3261, IETF,.

A. B. Roach (2002). "Session Initiation Protocol (SIP) – Specific Event Notification", RFC 3265, IETF,

A. Niemi,. (2004) "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, IETF.

V. Olmedo, V.A. Villagrá, K. Konstanteli, J.E. Burgosc, J. Berrocal (2009), Network mobility support for web service-based grids through the session initiation protocol. In Future Generation Computer Systems, doi:10.1016/ j.future.2008.11.007.
Chakraborty, D., Joshi, A., Yesha, Y., Finin, T. (2006): Toward distributed service discovery in pervasive computing environments. In *IEEE Transactions on Mobile Computing* 5(2), 97–112
Kourtesis, D., Paraskakis I. (2008): Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In *Proceedings* of the 5th European Semantic Web Conference (ESWC 2008), Lecture Notes in Computer Science (LNCS), vol. 5021, Springer-Verlag Berlin Heidelberg, pp. 614628,

WSMO. (2005). *Web Service Modeling Ontology. Retrieved from* http://www.w3.org/Submission/WSMO/

Mokhtar, S.B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y. (2008): Easy: Efficient semantic service discovery in pervasive computing environments with QoS and context support. In Journal of Systems and Software 81(5)

Thamarai Selvi Somasundaram, R.A.Balachandar, Vijayakumar Kandasamy, Rajkumar Buyya, Rajagopalan Raman, N.Mohanram and S.Varun (2006). Semantic-based Grid Resource Discovery and its Integration with the Grid Service Broker. In *Proceedings* of the 14th International Conference on Advanced Computing and Communications (ADCOM 2006, IEEE Press, Piscataway, New Jersey, USA, ISBN: 1-4244-0715-X, 84-89pp), Dec. 20 - 23, 2006

Stephen Burke, Simone Campana , Antonio Delgado Peris, Flavia Donno, Patricia M´endez Lorenzo, Roberto Santinelli, Andrea Sciab`a (2007).“gLite-3-UserGuide”, v.1.1, CERN-LCG 2007

RDF. (2004). *RDF/XML syntax specification.* Retrieved from http://www.w3.org/TR/REC-rdf-syntax

DAML-S. (2002). *Describing Web Services using DAML-S and WSDL* .Retrieved from http://www.daml.org/services/daml-s/0.7/daml-s-wsdl.html

Sun Service Registry for SOA. (2005). Retrieved from http://xml.coverpages.org/ni2005-06-15-a.html

Rajiv Ranjan, Lipo Chan, Aaron Harwood, Shanika Karunasekera, Rajkumar Buyya (2007). “Decentralised Resource Discovery Service for Large Scale Federated Grids”. In *Proceedings* of the Third IEEE International Conference on e-Science and Grid Computing 2007 pp 379-387

Shishir Bharathi, Ann Chervenak, (2007). “Design of a Scalable Peer-to-Peer Information System Using the GT4 Index Service”. In *Proceedings* of the Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)

P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, S. Haridi (2007). Peer-to-Peer resource discovery in Grids: Models and systems. In *Future Generation Computer Systems* 23 (7) (2007) 864–878.

Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., Botti, V. (2009). "Service Oriented MAS: An open architecture". In *Proceedings* of the AAMAS 09.

Yan, W., Hu, S., Muthusamy, V., Jacobsen, H.-A. & Zha, L, (2009). Efficient event-based resource discovery. In *Proceedings* of the 2009 inaugural international conference on Distributed event-based systems, Nashville, TN.

Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S., Lorenzo, J., Mamelli, A., Scholz, U. (2009). MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In: Cheng, B.H.C., et al. (eds.) *Software Engineering for Self-Adaptive Systems*. LNCS, vol. 5525. Springer, Heidelberg Survey composition

Noha Ibrahim1, Frédéric Le Mouël (2009). A Survey on Service Composition Middleware in Pervasive Environments. In *International Journal of Computer Science Issues*, Vol. 1, 2009.

L. O. Bonino da Silva Santos, G. Guizzardi, R. Silva Souza Guizzardi, E. Gonc¸alves da Silva, L. Ferreira Pires, and M. J. van Sinderen (2009). "Gso: Designing a well-founded service ontology to support dynamic service discovery and composition," In *Proceedings* of the 2nd International Workshop on Dynamic and Declarative Business Process (DDBP 2009).

WSDL. (2001). Web Service Description Language. Retrieved from http://www.w3.org/TR/wsdl

OWL. (2009).*Ontology Web Language version 2*. Retrieved from http://www.w3.org/TR/owl2-overview/

The Future of Cloud Computing. (2010). *Expert Group Report.* Retrieved from http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf