Journal of Web Engineering, Vol. 5, No.3 (2006) 265-290

CONTEMPORARY WEB SERVICE DISCOVERY MECHANISMS

JOHN GAROFALAKIS[1,2], YANNIS PANAGIS[1,2], EVANGELOS SAKKOPOULOS[1,2] AND ATHANASIOS TSAKALIDIS[1,2]

[1]*Department of Computer Engineering & Informatics*
School of Engineering, University of Patras
Rio Campus, 26500 Patras, Greece
[2]*Research Academic Computer Technology Institute*
Internet and Multimedia Technologies Research Unit
61 Riga Feraiou Str. 26110 Patras, Greece

{garofala, panagis, sakkopul, tsak}@ceid.upatras.gr

Received April 20, 2005
Revised October 20, 2005

The introduction of software development via Web Services has been the most significant web engineering paradigm, in the last years. The widely acknowledged importance of the Web Services' concept lies in the fact that they provide a platform independent answer to the software component development question. Equally important are the mechanisms that allow for Web Service discovery, especially as the latter has turn to an arduous task. This paper critically presents the latest methods, architectures, models and concerns that have arisen in the Web Service Discovery area.

Key words: Web Services, WS Searching, WS Discovery
Communicated by: S. Christodoulou

1 Introduction

Web Services (abbr. WS) have emerged as a dominating set of recommendations and standards (W3C, OASIS). They have marked current web engineering methodologies and are ubiquitously supported by IT vendors and users. In short they are interoperable software components that can be used in application integration and component based application development. As the demand for WS consumption is rising, a series of questions arise concerning the methods and procedures to discover the most suitable to use. In fact there is much hiding behind the discovery of a Web Service. This work aims to present, examine and analyze the different technologies, techniques and mechanisms in the area.

Initially a definition outline should be attempted of what discovery mechanisms stand for. A first description of discovery mechanisms for service providers appears in [56] as the match-making process. It is the process of finding an appropriate service provider for a service requester through a middle agent [11]. It includes the following general steps: a) Service providers advertise their capabilities to middle agents, b) middle agents store this information, c) a service requester asks a middle agent whether it knows of service providers best matching requested capabilities and d) the middle agent, in order to reply, tries to match the request against the stored advertisements and returns a subset of stored service providers' advertisements.

A more up-to-date approach [4] defines the WS Discovery mechanism in a broader sense as "the act of locating a machine-processable description of a WS that may have been previously unknown and that meets certain functional criteria". It is a service responsible for the process of performing discovery, a logical role, which could be performed by the requester agent, the provider agent or some other agent.

The main use of WSs comprises the remote invocation of services, by sending and in most cases receiving messages. While the WS technology is further adopted by IT practitioners and developers, the number of available WS is continuously rising. As a consequence, the usual WS discovery process needs that is based on UDDI record lists requires more and more time and patience by the developer/ consumer. This practice however is not efficient in many circumstances as they need to be able to choose between an abundance of provided Web Services. In order to better elucidate the situation we present a few typical scenarios of potential Web Service applications:

- Data providers, providing data such as a stock quote
- Business-to-business process integrations, such as those that send a purchase order from one company to another.
- Integration with multiple partners, and even with competitors
- Enterprise application integration, for example, integration of a company's e-mail database with its human resources (HR) database
- Help desk applications, enabling help desk operators to query/modify internal customer data.

Since WSs found in repositories can be tagged with a wealth of information, methods to narrow the discovery down to those matching a particular technical fingerprint can be quite complicated. Web Service Discovery mechanisms allow access to service repositories that can warehouse information about businesses, services and further details. In that sense, such mechanisms should be capable to retrieve a wide spectrum of information concerning the service providers themselves beside their advertised services.

Moreover, there is a need for dynamic discovery structures that will be up-to-date in an "online" fashion, providing efficient and available Web Service choices. The discovery mechanism should offer a number of capabilities, recognizable at both development and execution time. During development, one may search a web service repository for information about available services. At execution, client applications may use this repository to discover all instances of a web service that match a given interface.

The main obstacle affecting the Web Service Discovery mechanisms is heterogeneity between services. A high level approach is considered by the emerging Web service architecture [1]. Each of the examined solutions in this work tries to overcome different aspects of this heterogeneity in order to match the best Web Service available. The identification of different kinds of heterogeneity gives an impression on what has to be considered in order to avoid or mitigate them [36]:

- Technological heterogeneities (different platforms or different data formats).
- Ontological heterogeneities (domain-specific terms and concepts within services that can differ from one another, especially when developed by different vendors).
- Pragmatic heterogeneities (different development of domain-specific processes and different conceptions regarding the support of domain-specific tasks).

Having in mind the above, this work tries to critically present the existing solutions in Web Service Discovery and set future goals. In section 2 the main players in the discovery game are outlined. The architectural aspects of the WS discovery mechanisms are examined in section 3. The data models facilitating discovery are discussed in section 4. Quality of Web Service provisioning appears in section 5. Section 6 is devoted to Web Service Sequencing for workflow procedures. Finally, an overall assessment along with future steps and conclusions is presented in section 7.

2 Roles in the Discovery Game: Description of Players

WS Discovery mechanisms include a series of registries, indexes, catalogues, agent-based and Peer to Peer (P2P) solutions. The most dominating among them is the Universal Description Discovery and Integration (UDDI) standard that is currently in version 3 [33]. It can be considered relatively mature as little has changed in depth since the first edition of the standard. The different main players are presented in the following subsections in order to have a first taxonomy among the available solutions.

2.1 Pre-Web Service Solutions

The Web Service Architecture is a, relatively new, solution to a couple of old problems those of sharing data and remote method invocation. Before proceeding in the WS technology details, related older, though well-known technologies are discussed in this section.

Since the mentioned above problems are fairly old, several methodologies have been already presented causing the development of what is called *Distributed Application Development*. Distributed application development is the art and engineering of getting data from one machine to another. The data exchanged and the types of calculations performed can be virtually of any kind: purchase orders, customer data, mathematic computations, and so on. There have been many technologies for building applications that can send data back and forth including CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation), and DCOM (Distributed Component Object Model).

All of the aforementioned approaches presented significant difficulties to catch up and gain wide adoption from the development community. A typical example is DCOM, the distributed counterpart of Microsoft COM (Component Object Model). DCOM usage is essentially also limited to Windows platforms, although it is feasible to develop cross-platform solutions. Common Object Request Broker Architecture, or CORBA, is also a distributed technology designed to facilitate development over heterogeneous environments. Nevertheless, CORBA application development is particularly involved, which is a reason for CORBA not to become really popular.

The intrinsic difficulties to adopt any of the above drove the requirements for a distributed architecture, which would be based on simple, platform-independent and widespread existing protocols. Therefore, the development community was led to devise the Web Services architecture.

2.2 Web Service Discovery Players Overview

As in most of the previous distributed application development technologies, also in the WS technology there is the concept of a *remote method registry*. There are catalogue based registries, simplistic registries based on web indexes or more complex but promising P2P based registries. An overview of the main WS discovery categories are initially presented in this section, together with an outline of the different categorizations that are discussed in the following.

Web Service catalogue-based registries are the dominating technological basis for WS Discovery mechanisms. They are specialized repositories which implement a specification framework as metaschema. In particular, prior to the UDDI standard, organizations lacked a common approach to publish information about their products and web services for their customers and partners. UDDI established the first uniform method that included details for integration of already existing systems and processes between business partners.

UDDI allows the enterprises to discover and share information with regard to the web services and other electronic and non-electronic services that are registered in a kind of registry. A UDDI registry service is itself a WS that manages information about service providers, service implementations, and service metadata. In order to find a web service using the UDDI, much information regarding the required service is needed. The requirements include key words, part of the service's name and users' patience, in order to select the suitable service through the results of the registry. The available search tools are very simple and do not take into consideration any cross-correlations between web services and the qualitative characteristics of each web service, forcing the user to repeat the search from the beginning using new keywords. Realization of a UDDI registry can have different end-user purposes (Table 1).

Type	Description
Public	Querying and matching information is public to all web service consumers. In that sense public UDDI appears to be a WS itself. Publishing information into the registry is supported through secure channels (e.g. https), but this does not spoil its public character. Data communication with other registries is supported.
Protected	The notion of trust between collaborators characterizes this kind of registries. Such registries are implemented within a closed-group environment with monitored access to third parties. Administrative features may be delegated to trusted parties. Data communication with other registries is allowed only if explicitly specified.
Private	It is about fully secured isolated registries. They are usually domain specific registries in an internal network. Data communication with other registries is not allowed.

Table 1 UDDI Registry Instances

The UDDI specifications include:

a) SOAP APIs that allow querying and publishing of information,

b) XML representation for the registry data model and the SOAP message formats,

c) WSDL interface definitions of the SOAP and

d) APIs Definitions of various technical models that facilitate category systems for identification and categorization of UDDI registrations.

Using the above specifications, it is commonly recognized, though not specifically mentioned, that there are three types of information supported by the catalogue. These types included registration of white, yellow and green pages.

- *White pages* include basic contact information and identifiers such as organization name, address, contact information, and other unique ids.
- *Yellow pages* describe a web service using different categorizations (taxonomies). This way it is possible to discover a Web Service based upon its category.

– *Green pages* include the technological information that describes the behaviors and support functions of a Web Service.

Before proceeding to the P2P solution logic of WS Discovery, we should also include the simplified implementation of an index, in the catalogue type. In short, it is a list of references for Web Services. Such compilations are neither authoritative nor validated. They are usually harvested collections of published information by the service providers (usually using web spiders).

Peer to Peer (P2P) platforms provide a good arena for the Web Service Discovery mechanisms implementation. A P2P overlay network provides the infrastructure for routing and data location in a decentralized, self-organized environment in which each peer acts not only as a node providing routing and data location service, but also as a server providing service access. P2P can be considered a complete distributed computing model. Recently proposed P2P systems include Pastry [46], Chord [53] and NIPPERS [28]. Chord arranges the network of peers to a ring. A similar approach is CAN [44], which utilizes a torus network topology. In all systems, nodes are assigned IDs drawn from a global address space. Peers are also assigned a range of keys from the global address space that they are responsible for. Each peer also stores auxiliary information in order to appropriately route key lookups. Usually, a key lookup is initiated at a peer. In this case the peer consults its look-up table in order to successfully route the query to the peer that stores the queried key. In the case of Chord routing with the aid of look-up table, simulates binary search on the address space of all peers, thus a request in an N peer network can be routed in $O(\log N)$ messages.

Chord mainly, has been adopted as the overlay P2P network for distributed web service architectures. The hosts in the P2P network publish their service descriptions to the overlay, and the users access the up-to-date Web Services. Architectural aspects are briefly discussed in this section and the interested reader may want to continue in section 4.1 for more details on data models.

An example of Chord-based solution is presented in [22]. It is an architecture called P2P-based Web service Discovery (PWSD) that uses a Chord P2P protocol as overlay, consisting of Service Peers (SP). Each SP is mapped to several Logical Machines (different machines corresponding to the same hardware). Each Logical Machine maintains the necessary interfaces to map and search WSs in the P2P network. Service Descriptions as well as queries are hashed and routed in the Chord network.

One more interesting P2P system is called Speed-R. It has been presented in [52] and it is a WS storage and retrieval system that uses ontologies and a P2P infrastructure. Some nodes in the P2P subsystem are assigned registries, which are in turn partitioned according to their specific domain. An ontology is assigned to each domain. The P2P system is based on JXTA [17] implementation. Its architecture is based on role assignment to peers (for example some nodes have undertaken the role of controlling updates and propagating them), thus their system may suffer from single point failure.

Closing this section, catalogues and P2P solutions are considered the major players in Web Service Discovery. Realization of these mechanisms includes several different flavors which are presented analytically in the following sections. An overall taxonomization graph is presented below in Figure 1. All categories introduced are depicted in rounded rectangles, while the most efficient solutions in each category are included in normal rectangles as leaf nodes. The figure is provided as a high level map of the available methods and only highlights our attempt for categorization. The reader may use it as a helpful roadmap for the most appropriate properties and matching capabilities to choose among, before implementing a WS discovery solution. Moreover, the community may find for the first time, to the authors' best knowledge, an overview of the WS searching and matching

techniques. Each category includes representative description of corresponding solutions and at the end of each section a summarizing techniques table-like list is presented.

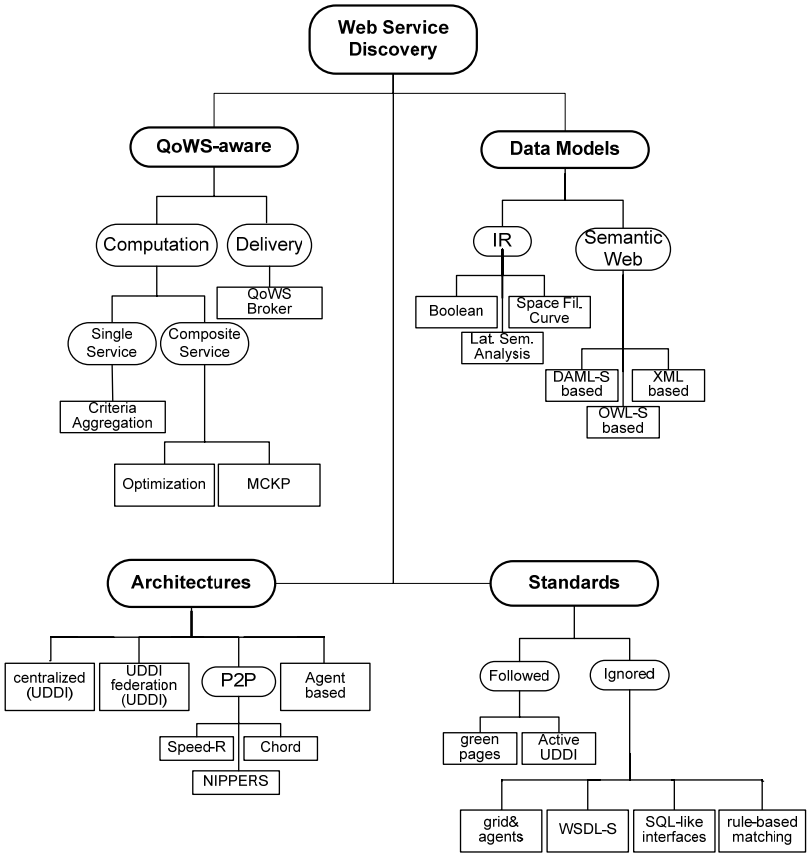

Figure 1 A categorization of approaches to Web Service Discovery

3 Architectural Aspects, Standards and Platforms

In this section, we investigate and present the different WS discovery approaches based on the different architectural perception that they follow. In particular, we distinguish the WS search and selection handling mechanisms according to:

1. the *level of automation* they involve,
2. *network topology* issues,
3. compliance with *standards and recommendations* and
4. the *technology platforms* available.

The approaches of this section correspond in the lower half of Figure 1 and specifically at the subtrees entitled *architectures* and *standards*.

3.1 Manual Procedures vs. Intelligent Automation

The *level of automation* in WS discovery can range from the standard developers' manual searching, up to quite automated techniques. An early approach on this categorization perspective appears already in the W3C WS architecture [4]. Web service discovery, carried out manually (e.g. by implementers during built-time) but also automatically (e.g. by self-assembling applications during run-time), is a three-phase process consisting of *Web service search*, *Web service assessment* and *selection of Web services for the configuration process*.

In the *manual discovery* process, a requester human uses a discovery service (typically at design time) to locate and select a service description that meets the desired functional and other criteria. Under intelligent automated discovery, a requester agent performs and evaluates this task, either at design time or run time.

To trick standard manual selection, a "hacker" approach, as proposed in [24], is to postpone the decision of which service to bind to until execution time by querying UDDI for the access points of services that are known to implement this WSDL. However, this approach can merely be categorized as being non automated.

It is important to notice that UDDI by design follows the manual approach. The several UDDI processes and mechanisms cover solely operational aspects of the UDDI cloud, data management, and replication aspects. They are designed and are suitable for dealing with explicitly published changes to the registry data, which are typically done 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.

A step towards automation can be considered to be the mechanism presented in the work [62] that introduces Web portal services' capability. This mechanism can be used for the detection and consumption of Web portal services. It enables the precise location as well as the automated discovery and invocation of Web portal services, though not extensive experimental support is provided.

A more automated technique is found in the area of WS selection based on different non functional parameters. There is a promising online WS discovery QoS solution in [27] where best case matching is performed using combined historical and online QoS data. The approach deals mainly with non functional parameter matching but it provides the ground to work towards a generalized automatic WS matching.

3.2 Centralized vs. Decentralized Solutions

Network topology issues are also one more distinctive factor among the different WS discovery solutions. There are single point of operation approaches, more distributed solutions and full P2P based ones. We present them in two categories based on their centralization rank.

Centralized Services: In the conceptual network level, a registry is conceived as an authoritative, centrally controlled store of information. The standardized representative of this category is the UDDI registry [33], which is usually implemented in single node architecture (physical network level). A lightweight version of a registry is the centralized service of indexes. An index is a compilation or guide to information that exists elsewhere. It is not authoritative and does not centrally control the information that it references. The key difference between the two approaches is not just the difference between a registry itself and an index. Indeed, UDDI could be used as a means to

implement an individual index: just spider the Web, and put the results into a UDDI registry. Rather, the key difference is one of control: Who controls what and how service descriptions get discovered? In the registry model, it is the owner of the registry who controls this. In the index model, since anyone can create an index, market forces determine which indexes become popular. Hence, it is effectively the market that controls what and how service descriptions get discovered [4].

Before proceeding to the next solution, a solution that is a conceptually centralized approach, though in the physical level it has a decentralized existence has to be included. When leaving from the strict "single point" of operation, there is one primitive, though well-known and widespread, network decentralization-like approach, which follows the following concept: Different UDDI nodes are connected together and form a single service that, while appearing to be virtually a single component, is composed of an arbitrary number of operator nodes. This solution is called the *UDDI cloud* or *federation* [45]. An operator node is responsible for the data published at this node: in UDDI terms, it is the *custodian* of that part of the data. Data consistency issues are resolved by the invocation of data replication procedures, inherent in the UDDI. Re-querying the registry faces invocation failures caused by static service caching

Decentralized Solutions: More elaborated decentralized solutions have also been proposed, which are usually Peer-to-Peer based network solutions. Such systems as proposed in [49], [52], usually build on Peer-to-Peer (P2P) technologies and ontologies to publish and search for Web Service descriptions. An additional Peer-to-Peer solution (P2P) is also proposed in [25]. They present a Peer-to-Peer (P2P) indexing system and associated P2P storage that supports large-scale, decentralized, real-time search capabilities. One more decentralized approach which is agent-based is presented in [30]. This approach aims to describe an environment called DASD (DAML Agents for Service Discovery) where WS requesters and providers can discover each other with the intermediary action of a Matchmaking service.

On the other hand, a novel approach using solid theoretical support and introducing improved performance in searching and managing WS in overlay networks is NIPPERS [28] which has been proved to be better in the complexity of the discovery procedure than the popular current DHT overlay network Chord. One more relative peer based approach is included in the work [65] that presented the BDT-Grid WS discovery approach.

Distant ancestors of the distributed lookup registries are the Whois++ [61] and rWhois [47] look-up protocols. Both protocols provide online look-up of individuals, network organizations, key host machines, etc. Their key attribute is their hierarchical and distributed architecture, in a similar vein but different context with the contemporary decentralized lookup protocols.

3.3 Complying with Recommendations vs Overriding them

In this subsection, we categorize the available solutions in order to distinguish the ones that adhere to the existing W3C/OASIS standards and recommendations from the pioneering ones that demand extensive alterations. The intuition behind this categorization is that overriding standards policies can possible result to promising and intelligent solutions but also hard to implement and to be widely adopted ones.

Following the UDDI standard: In this category, we present solutions proposed that build upon the established concepts without any demand for new infrastructure. In order to enrich the UDDI specification, information that is required for the procurement of WS is added to the administrative

information sections, namely white and yellow pages [24]. When performance is a requisite, data are included on the quality of service (e.g. expected meantime between failures, maximum response time, maximum data throughput, and so on), which are crucial to assess its applicability. In order for discovery to be able to query about security of a Web service, security details have to be included as well (see also Section 5.1).

One further extension is delivering improvements to the green pages [36]. The behavior of Web services (strictly speaking of its particular methods) has to be documented by specifying the pre- and post-conditions of the methods that are being published to their interfaces. Thereby, designing by contract [44] is supported. A pre-condition expresses the constraints under which an invoked method returns correct results. A post-condition describes the state resulting from a method's execution and thus guarantees that it will satisfy certain conditions. Constraints regarding the ordered invocation of Web service methods can occur between Web services. They are called coordination constraints and help configuring a Web service on the basis of application processes.

Moreover to support the popular distributed development technique of dynamic service invocation within a network of distributed services, UDDI has to be further extended. To achieve such functionality, the so-called Active UDDI [21] has been proposed. It is an extension of the existing UDDI infrastructure without requiring changes to the data structures or the APIs themselves but using a totally new web service that plays the role of a man-in-the-middle. This solution provides a proxy based approach in order to dynamically provide registry updates.

Overriding recommendations - Introduction of other approaches: On the other hand, there are a number of approaches that introduce a totally new architecture to support WS discovery. Such approaches cannot easily and straightforwardly be incorporated into existing UDDI-based WS discovery mechanisms.

In particular, the WS discovery mechanism proposed in [31] is an open, large-scale and inter-operable, multi-agent system in the context of Grid computing. It is an attempt to integrate agent technologies with Web Services. The Grid problem is defined as flexible, secure, coordinated resource sharing, among dynamic collections of individuals, institutions and resources [15]. An extension of UDDI registry is used, with additional information (meta-data) about agents and ontology-based pattern-matching in order to accommodate the kind of searching that is required to locate an agent service according to the performative it supports. The proposed extension of UDDI contains WSDL descriptions of all agents that have been registered. In this way dynamic discovery and invocation of services by software through common terminology and shared meaning is enabled.

Another non-standardized approach is given in [53], where the WSDL-S language is presented, which has been created by extending WSDL 2.0. This attempt can be categorized in both centralized and decentralized catalogues. At the same time, it provides an interesting extension of standards by allowing backwards compatibility. In particular, this language provides space for semantic annotation of the web services and their functionality. The annotation can be either produced automatically from respective annotation in the source code of a web service or inserted afterwards in the WSDL-based description file.

One more approach that deals with WSDL alterations is presented in [63]. The authors proposed the alternative idea to WSDL based Web Services, called WS-Net. WS-Net is an executable architectural description language incorporating the semantics of Colored Petri-net with the style and understandability of object-oriented concepts. WS-Net describes each web services component in three

layers: *interface net* declares the services that the component provides to other components; *interconnection net* specifies the services that the component acquires to accomplish its mission; and *interoperation net* describes the internal operational behaviors of the component. This approach helps enhance the reliability of web-services oriented applications. However, manually transferring the WSDL specifications into the WS-Net specifications is not a trivial job.

Towards a different track of altering the standards, an SQL-like language for WS retrieval purposes is proposed in [64] called the *flexible Web Service retrieval* approach. It is an orthogonal service space and it utilizes multi-valued specialization relationships between services. Service similarity degree is measured based on the correlations between operations offered by services. The authors provide a query language and an application that implements a Web Service, the "flexible registry". The provided visualized service space helps users to focus their intention. On the other hand, users can flexibly describe their retrieval destination by choosing the specialization relationships and determining the similarity degree between the source and the target.

Finally in this category, a technique not following the UDDI standard using rule-based decision matching technique is proposed in [14]. The technique provides more precise matching results, when compared to existing matching syntactic techniques, as it is based on the functional semantics of web services. Scenario networks are used to partially describe the external behavior of web services. A many-to-many semantic-based matching approach for scenario networks is proposed adopting the concept of functional substitutability.

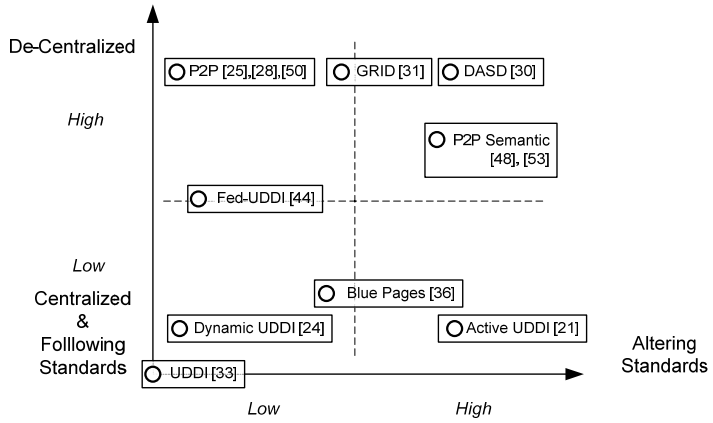

Figure 2 Categorization based on the architectural approach

A comparative outline of the different techniques is outlined in Figure 2. To visualize the architectural differences and similarities, we have placed the above mentioned approaches on a 2-dimesional space. In the axes origin, standard UDDI [33] searching capabilities are found. While moving across the horizontal axis, techniques tend to provide wider alternations to the current discovery standards. In the vertical direction and upwards, decentralization is increased, hence P2P- and Grid-like solutions appear. In the upper right quarter the most "anti-catalogue" solutions are placed. In descending order, the most dominating of the solutions presented in this section, besides the classic UDDI, are

a) the P2P techniques

b) the promising though not widely implemented grid based discovery mechanisms

c) and for the more distant future, semantic p2p-based techniques.

Table 2, presents a table-like list with summary of the algorithms that are included in this section.

3.4 Industrial Platforms

Following the surge for WS Discovery, major industrial implementation platforms support also natively, a WS search functionality. For instance, Windows 2003 has a UDDI server preinstalled with the OS, whereas many J2EE vendors build UDDI instances into their application servers (see [24]). There is a number of different solutions available to adopt when implementing a vendor based WS discovery architecture. Most of the solutions provided allow keyword based searching and do not incorporate any of the more efficient approaches already presented. However, it is important to shortly discuss about the available platform WS discovery solutions in order to have a full view of the different architectural possibilities.

Proposed Solution	**Manual Procedures vs. Intelligent Automation**	**Centralized vs. Decentralized Solutions**	**Following vs Overriding Recommendations**
Dynamic UDDI/ at execution time "hacker" approach [24]	Manual	Both	Following
online WS discovery QoS solution in [27]	Semi-Automatic	Both	Overriding
automated discovery and invocation of Web portal services, [62]	Semi-Automatic	Centralized	Overriding
UDDI registry [33]	Manual	Centralized	Following
UDDI cloud or federation [45]	Manual	Centralized (with replication)	Following
DAML Agents for Service Discovery) [30]	Manual	Agent based	Overriding
NIPPERS [28]	Manual	Peer based	Following
BDT-Grid [65]	Manual	Peer based	Following
Whois++ [61] and rWhois [47]	Manual	Not Applicable	Obsolete
Green pages - pre- and post-conditions of the methods [36]	Manual	Centralized	Following
Designing by contract [44]	Manual	Centralized	Overriding
Active UDDI [21]	Automatic	Centralized	Following
Grid based [31]	Manual	Decentralized	Overring
WS-Net [63]			
WSDL-S with semantics of Colored Petri-net [53]	Manual	Both	Overring
Flexible Web Service retrieval with SQL-like language [64]	Manual	Centralized	Overriding
Rule based decision [14]	Semi-Automatic	Centralized	Overriding

Table 2 The QoWS parameters and the respective units of measurement.

Java 2 Enterprise Edition (J2EE) (http://java.sun.com/j2ee/index.jsp) Sun Microsystems is positioning its Java API for XML Registries (JAXR) as a single general purpose API for interoperating

with multiple registry types. JAXR allows its clients to access the Web Services provided by a Web Services implementer exposing Web Services built upon an implementation of the JAXR specification.

Microsoft .NET (http://www.microsoft.com/net/) At first, Microsoft had the discovery of Web Services with DISCO in the form of a discovery (DISCO) file. A published DISCO file is an XML document with links to other resources describing the Web Service. Since the wide spread adoption of UDDI, however, Microsoft has supported it in order to maximize interoperability between solutions in what is, after all, a set of specifications for interoperability. In addition to providing a .NET UDDI server, the UDDI SDK provides support for Visual Studio .NET and depends on the .NET framework. Products such as Microsoft Office XP offer support for service discovery through UDDI.

Java-Based APIs The UDDI specifications do not directly define a Java-based API for accessing a UDDI registry. The Programmer's API specification only defines a series of SOAP messages that a UDDI registry can accept. Thus, a Java developer who wishes to access a UDDI registry can do so in a number of ways:

a) using Java-based SOAP API,

b) using a custom Java-based UDDI client API or

c) using JAXR.

Simple Web Services API (SWSAPI) (http://aspn.activestate.com/ASPN/WebServices/SWSAPI) Various leaders from the Open Source world (PHP, Perl, Python) have collaborated to develop an API that is simple enough to be used by even beginning users and yet sophisticated enough to handle even complex web services. At present, the API concentrates on the problems of:

a) reading and writing WSDL files and URLs,

b) caching remote WSDL files locally,

c) invoking services described by the WSDL (invocation) and

d) allowing programmatic access to the WSDL's internal components (introspection). Over time there will be associated specifications for dealing with server side issues, XML schema issues and so forth.

J2EE and .NET seem to hold the majority of the current Web Service based solutions mainly due to the integration and support of powerful development platforms such as JBuilder and Visual Studio .NET which have wide acceptance among developers.

4 Data Models and Searching for Web Services

In this section, the important aspect of the Web Services discovery of services' *modelling* will be discussed. The term *model* in this context refers to the representation of Web Services, a process that takes place before their discovery. In this section we present alternative viewpoints: the *Information Retrieval* approach, the *Semantic Web technology* based solutions and the *similarity matching* mechanisms.

4.1 The Information Retrieval approach

There are a number of IR approaches in the literature that have been adopted for WS discovery techniques. We will discuss techniques utilizing keyword searching, document vectors transformation, latent semantic indexing and more.

The simplest Data Model is the *Catalogue/Keyword Based*. This model is followed by the legacy UDDI standard and the discovery mechanism it supports. In a nutshell, the textual description that accompanies each Web Service is stored in the UDDI catalogue along with the tModel that provides the Service functionality. The retrieval stage comprises a user or a search program, entering a query to the catalogue. The query consists of keywords, which are matched against stored descriptions. The matched Web Services are then returned as a candidate answer set and the user browses them in order to find which one of them really suits her needs or, what tends to be a frequent case, resubmits another query.

The above approach is followed by the current UDDI registries and it resembles the classic Boolean Information Retrieval model [2]. Despite its simplicity and ease of implementation, it suffers from either lots of returned results or very few returned ones.

A mainly architectural drawback of the approach is the fact that usually a centralized registry hosts the majority of descriptions and thus receives millions of requests being thus a bottleneck point. Decentralized proposals are discussed in section 3.2

An elegant approach to tackle the inadequacy of keyword-based Web Service Discovery was proposed in [48]. The key concept in their approach is to represent service descriptions as *document vectors*, a dominant approach in the IR field (see [2]). A description text, thus, corresponds to a vector $\vec{d}$ in the vector space spanned by all the terms used in all Service description texts. They go one step further by representing all the document vectors, as columns in a term-document matrix A.

Another IR technique is afterwards applied, which transforms the matrix A achieving a representation of the document collection by its more significant semantic concepts, or what is called *Latent Semantic Indexing* (LSI) [3]. This method is observed and proved to be able to return documents of the modeled text collection, which are more closely related to the semantics of the expressed query, regardless of exact matching or not with the query terms.

When applying LSI to the discovery of Web Services they observed that description vectors resulting from the transformation of the original matrix were mapped more closely to the vector space representation of the query, than the respective representations of plain keyword-based descriptions.

A Web Service modeled as d-dimensional vector, can also be thought of as a point in d-dimensions. In that respect a geometric data structure for indexing multidimensional data can be deployed in indexing and querying Web Services. Instead of transferring the problem to high-dimensions, Schmidt and Parashar [50], use a transformation, which injectively maps points in higher dimensions, to numbers. This transformation is called *space-filling curve*. Many space-filling curves have been proposed (see [16]), but among those, the Hilbert curve has the important property that adjacent intervals are mapped to nearby regions in d-dimensions.

In the system of Schmidt and Parashar [50], a unique ID is generated for each Web Service, through the Hilbert curve mapping. The IDs are then stored in a Chord [54] of Web Service Peers. Thus, the storage and retrieval of WS inherits the load balancing capability and the dynamic nature of Chord. The main advantage of the model followed by [50] is that it can efficiently support partial

match queries. These queries are realized efficiently mainly due to the clustering properties of Hilbert curve. The querying procedure is further enhanced, by some query optimization heuristics.

An interesting technique that combines keyword matching with P2P storage is proposed by Li et. al. [25]. It presents a system that also maps the XML Service Descriptions over a P2P Network, using distributed hashing. In that sense peers act both as service providers and request generators. The XML Service Descriptions are parsed in order to extract service keywords and keywords are hashed with the MD5 hash function. The underlying P2P Network Protocol is again a Chord [54] and the modified system is called XChord. Chord's distribution policy is enforced to route the generated hash descriptions to nodes. A Web Service query starting at a peer node, is also decomposed into keywords which are subsequently sought for using the Chord searching principle. XChord is proved more stable, load balanced and less space consuming according to the conducted experiments.

4.2 Semantic Web technology based approach

Apart from the classic IR techniques adopted, WS discovery literature has much work to present focused on performing semantic matching for Web Services discovery. This development is increasingly significant since it seems to be able to tackle some of the UDDI catalogue inadequacies. The predominant problem is the restrictions posed by keyword matching that do not allow retrieval of WS with similar functionality; two WSDL descriptions can be used to describe the same Service but with different words. However, when modeling web services with ontologies the semantic representation of concepts and their relations can be exploited and thus semantic matching to be performed. Semantic descriptions of Web Services can be obtained with the use of DAML-S [10] or OWL-S [37] languages.

Early acknowledgement of the importance for Semantic web service description is found in Dogac et al. [12]. They discuss the advantages of describing service semantics through ontology languages and describe how to relate the semantics defined with the services advertised in service registries like UDDI and ebXML.

Paolucci et al. [38] present a framework to allow WSDL and UDDI perform semantic matching. Web Services are modeled as ontologies, or Service Profiles as they are called, with the use of the DAML-S [10]. Typically, a Service Profile contains information for the Actor (provider), Functional Attributes like Geographic Location and Functional Descriptions such as inputs/outputs of the service. By maintaining ontology hierarchies, it is possible to perform semantic matching, which is subsequently performed by exploiting the subsumption capabilities of DAML. In order to combine ontologies with the UDDI Registry, the authors define a separate layer, the DAML-S Matchmaker. The matchmaker does not extend any of the UDDI page categories, but it is treated as an add-on, which undertakes semantic matching and the mapping of ontologies to UDDI Descriptions. Semantic matching especially when using DAML-S has several advantages. First of all it provides matching flexibility, because results are returned that can differ syntactically with the input query. It also provides accuracy since no matching is performed unless this is derived from the hierarchy and finally the concept of matching degree can be supported.

A recent development was the introduction of a new language, OWL-S, to combine semantic annotation of Web Services with their discovery and invocation with WSDL and SOAP (see [55]). This approach has led to the OWL-S Matchmaker module.

The approach of adding DAML extensions to UDDI descriptions is also adopted by [52]. Instead of providing a separate layer, they perform a simpler construct by enabling both WSDL descriptions and UDDI registrations contain semantic information. This information is simply a mapping between WSDL entries and DAML+OIL ontologies. In the case of UDDI, different tModels are provided to represent functionality, input/output, etc. A matching procedure that uses templates and exploits semantic descriptions to provide semantic matching is performed.

Moreau et al. [31] also perform some kind of semantic matching but in a different context. They describe agents performing Grid computations, as WSs. Hence, they transform agent ontologies into XML and semantic matching is performed by validating structurally expressed queries against agent description schemas. In the approach of [30] a combination of semantic annotation of web services and agent-based publishing and discovery is followed.

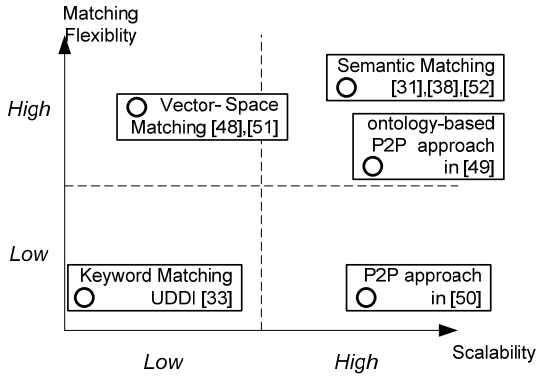

Figure 3 Taxonomy of discovery models with respect to Scalability and Matching Flexibility

Figure 3 presents the already mentioned approaches of this section according to the representation and modeling of the Web Services discovery process. At the axes origin standard keyword-based searching [33] capabilities are positioned. While moving across the horizontal axis, techniques tend to provide better scalability to the current discovery standards. An extremely valuable technique is Schmidt and Parashar's, technique which is a P2P based solution proposed in [50]. In the vertical direction the matching flexibility is increasing and vector-space based mechanisms appear. In the upper right quarter the most flexible and scalable solutions appear, though semantic matching techniques are a promising but rather distant future strategies. In descending order, the most dominating of the solutions presented in this section, besides the classic UDDI, are:

a) Multiple criteria searching is the immediate next step in this category. Vector space based searching can be particularly efficient and it would incorporate traditional IR policies.

b) The P2P technique which is really scalable. NIPPERS system may also be included in the same category.

c) Also in this case promising though not robust semantic oriented solutions are included.

Interesting implementation frameworks for semantic matching are proposed in [36] and [19]. The first paper provides a framework that uses ontologies to discover ("bind" in that case) the web services that best match a specific operation domain (desired set of operations). The available data are represented with domain ontologies and the available operations, with operation ontologies. Generalization relationships on both models are encoded in XML formats and so are binding

relationships. Binding can thus be performed by a binding ontology that decides what fits where according to binding relations. Overhage in [36] proposes the implementation of semantic descriptions as an extension of the UDDI protocol, termed as E-UDDI. E-UDDI introduces "blue pages", sections that contain semantic descriptions of Web Services; the latter are implemented in DAML-S. The model described in [36], also provides extensions to the green page section of the UDDI, by adding the capability to define constraints in the WS execution sequence. However E-UDDI seems to be more a vision than an implementable system. Figure 3, presents a graphical taxonomy of the Discovery Models.

Analytic approach for semantic annotation and publication of Web Services is dealt in [40], [42], [53], which is about the integrated METEOR-S Semantic Web Service Annotation Framework (MWSAF). Several methods and tools are discussed in order to allow software developers to incorporate semantic descriptions of Web services during code development. This approach leverages the annotation mechanism provided by the Java programming language but it can also be implemented with the corresponding annotation mechanisms in C# .NET. The authors provide verification by implementing a Semantic Web Service designer for source code annotation and Semantic Description generator for generation of rich descriptions of Web services.

Another cross-disciplinary application of evolutions of intelligent WS selection is found in [20]. It is argued that Semantic Web Services can serve as a key enabling technology to achieve the goals of Universal Multimedia Access (UMA), where users can consume any multimedia resource anywhere at any time, is the driving vision of ongoing ISO/IEC Moving Picture Experts Group (MPEG) standardization efforts.

4.3 Similarity based Web Service retrieval

In many cases both in the classic Information Retrieval and in the Web Service Discovery, search by similarity is necessary. The reason is that unless similarity based matching is performed the request agent is required to know the exact service descriptions. Similarity search is met under two kinds. Either a user submits a query and retrieves similar results or once a Web Service is retrieved a user might want to seek for Web Service with similar operations or similar input/outputs.

Sajjanhar et al. [48] perform similarity matching by representing Service description texts as described in Section 4.1 and thus they are able to use LSI-based algorithms to compute similarity between services at the description level. This is achieved since LSI is observed to be efficient in mapping vectors of conceptually similar documents, in neighboring points of a low-dimensional vector space, which is spanned from the underlying concepts (see [3]).

In the implementation of Schmidt and Parashar [50], it is allowed to specify web service queries using wildcard characters. For example, using the conventions in [50], suppose that the keyword space of web service descriptions is divided into two dimensions, business sector and service functionality. Thus queries of the form *(travel, booking), (stock*, *)*, are allowed. These queries are allowed due to the mapping of data points with the Hilbert Curve, (see Section 4.1). With this mapping, areas that are candidate answers for the wildcard queries, are mapped in consecutive regions, the *clusters*, and furthermore in consecutive one-dimensional intervals. Once the clusters are identified, queries are routed to the peer nodes in the overlay, storing those clusters. A heuristic is also developed to prune some nodes, reducing thus the message overhead for cluster notification.

Woogle [13] faces approximate retrieval in a different context, yet the authors employ clustering techniques. Clustering is formed on the basis of association rules'[a] formation among the terms constituting operation names. Hence, clusters are formed between associated parameters with care to ensure high intra-cluster and low inter-cluster similarity. Intuitively, when all the parameters inside a set, frequently co-occur, then they may correspond to similar functionalities.

The authors compute four kinds of similarity, namely: input/output parameter name similarity, input/output concept similarity, operation description similarity and WS description similarity. Let's consider I/O parameter name similarity first. In this kind of similarity, operations are characterized by a bag of words, holding parameter names. Similarities are computed with the use of the TF/IDF metric, see [2]. Analogous procedures are used for similarities in operations description and WS description. For I/O concept similarity words sin bags are substituted by their clusters prior to applying TF/IDF metrics. These four similarity types are linearly combined in a single similarity score. Table 3 presents a brief summary of the reviewed methods.

	Method	**Comment**	**Partial Matching**	**Scalability**
IR-based	Standard UDDI	Keyword-based discovery	Not Supported	Low
	Sajjanhar et al. [48]	Vector-space discovery	Supported	Low
	Space filling curves [50]	Space filling-curves over P2P	Supported	High
	XChord [25]	Hashed descriptions over P2P overlays	Not Supported	High
	Woogle [13]	Approximate retrieval search engine with clustering	Supported	Low
Semantic	Semantic matching [19, 36, 38, 49, 52]	Semantic annotations and matchmaking	Supported	High

Table 3 Summarization of systems and characteristics

5 Quality of Web Service Provisioning

Web Service discovery in all previous sections referred in the process of locating WS and narrowing the range of WS selection, which match some certain functional specifications. However, there is a number of non functional parameters that play an important role at the final WS consumption choice. In this section, we discuss Quality of WS Provisioning (abbr. QoWS), which has been more or less set aside in most the work in WS area. In this section, we discuss at first the QoWS parameters affecting WS provisioning and in the following subsections concerns of QoWS searching and delivery techniques. Therefore, neither concrete definition nor globally accepted notions of QoWS exist. An early QoS definition work [43] has attempted to highlight the issues associated with non-functional service properties. It is argued that an increased level of service property information would facilitate

[a] An association rule of the form x→y, describes frequent co-occurrence of two data items x and y and the degree to which the existence of x implies that of y.

more thorough decision-making by a service requestor. Some recent work however, presents an attempt to define more strictly the QoWS parameters and methods of delivery. System designs are also presented in [57],[58] which deliver QoWS to the end user application.

5.1 QoWS parameters

Ran [43], Ouzzani [35], Ouzzani and Bouguettaya [34] and Liu et al. [26] highlight the predominant parameters that define the Quality of Web Service. Although there is some dispute on the establishment of quality criteria, we present here the quality parameters that are common to the majority of the related work:

- **Computational Criteria.** We are interested in parameters such as,
  - *Latency*, the average time for an operation to return results after its invocation. It includes network and execution delays.
  - *Availability*, the probability to get the service running
  - *Throughput,* the percentage of completed web services over a time period.
- **Business Criteria.** They refer to criteria related to business aspects of the service invocation
  - *Execution Cost,* measures the cost of service invocation
  - *Reputation,* measures the mean degree of satisfaction, assigned by served users.
- **Security Criteria.** They refer to the safety constraints that have to be satisfied during the execution period. These include:
  - *Authentication*, indicates whether user identification is provided
  - *Encryption,* indicates whether message encryption is supported
  - *Access Control,* indicates whether access is restricted to qualified user groups.

Ran [43] presents a modification framework of the tModel for the integration of QoWS characteristics. Not all of the above parameters are measurable. However, if a parameter is not directly measurable, a Boolean value can represent the absence (false) or presence (true) of the specific quality requirement. Table 4 synopsizes quality parameters and gives the appropriate measurement units for each one of them.

QoWS Parameter	**Unit**
Latency	Usually milliseconds
Availability	Real value in [0,1]
Throughput	Real value in [0,1]
Execution Cost	Real value
Reputation	Real value[b] in [0, r]
Authentication	{TRUE, FALSE}
Encryption	{TRUE, FALSE}
Access Control	{TRUE, FALSE}

Table 4 The QoWS parameters and the respective units of measurement.

[b] The value of *r*, is application defined. Ouzzani [35] sets it to 10 while in [26] it is set to 5

5.2 QoWS aware Discovery

The need for QoWS aware Discovery arises both in provisioning of a single Web Service and in complex Web Service applications. As far the complex WS applications case is concerned a typical scenario entails a user executing a complex query which is transparently translated to a set of WSs, which may have to be executed in a specific order. This execution sequence is also referred to as *execution plan*. This section presents mechanisms to support WS execution, compatible to the QoWS criteria set in the preceding section.

A single WS can possibly be provided by more than one distinct access points. Therefore, in both execution sequence and the provision of the single web service, a first step includes the construction of a set of all possible instances of the required Web Service(s), the *selection candidates*. Next, in the case of the execution sequence, a *dependency graph* must be constructed [35], to model the particular order of WSs inside the execution plan, i.e. each WS corresponds to a unique node and a directed edge from WS *i* to WS *j*, denotes that *i* must be executed before *j* in the particular execution plan. Apparently, it is also imperative to define a ranking function on the selection candidates, a function to measure the provided QoWS.

The challenging task is to devise a scoring function that combines the QoWS criteria in a righteous manner in order to produce a single score for each Web Service or execution plan, respectively. The main methodologies are summarized. Liu et al. [26] represent each WS as an *m*-dimensional vector, *m* the number of parameters taken into account. Then the selection candidates are represented as a matrix, Q: $n \times m$, where *n* is the number of WSs with similar functional. This matrix goes through successive normalizations which aim to bring the different criteria to the same level and to cluster criteria into groups. Thus, after the necessary normalizations, the QoWS criteria are grouped and normalized, and a single QoWS score for Web Service ws_i results from the relation

$$\mathrm{QoWS}(ws_i) = q_i \cdot w^{\mathrm{T}} \tag{1}$$

where the above relation indicates the dot product between the quality vector q_i for service ws_i and a user supplied vector *w*, of importance weights for each quality category. Obviously, the WS achieving the highest QoWS score, will be selected.

When it comes down to selecting the optimum execution plan for a service request the situation is as follows: we have multiple execution plans, each accompanied with a QoWS score, the latter being a linear combination of ratings for individual WSs. What is asked is to select the execution plan that will maximize the total quality of service. It is almost obvious that the above problem boils down to linear optimization. This is the case in [34, 35]. For each execution plan *p*, we distinguish between negative QoWS parameters and denote them by *neg*, and positive QoWS parameters denoted by *pos*. Then for each plan *p*, the relation in (2) provides the total provided QoWS

$$F(p) = \left(\sum_{Q_i \in neg} \frac{Q_i^{\max} - Q_i}{Q_i^{\max} - Q_i^{\min}} + \sum_{Q_i \in pos} \frac{Q_i - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}} \right) \tag{2}$$

where Q_i denotes the measured value for a parameter and $Q_i^{\max}$ ($Q_i^{\min}$) the maximum (minimum) value over all available choices for service *i*. We have to note that the value for each Q_i is properly

aggregated over all the participating services in *p*. The score in (2) resembles the normalization logic of [26]. The overall objective function is defined in (3).

$$C = \left(\sum_{Q_i \in neg} W_i \cdot \frac{Q_i^{\max} - Q_i}{Q_i^{\max} - Q_i^{\min}} + \sum_{Q_i \in pos} W_i \cdot \frac{Q_i - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}} \right) \quad (3)$$

where W_i is a user provided weight that shows bias over specific QoWS parameters. The cost function can be further tuned to include *matching degree*, the degree to which an execution plan satisfies user query, and *QoWS rating*, a score computed for each WS operation to measure the operation's deviation from the advertised quality.

The author in [35], notes that if one sought the entire solution space for the optimal solution, he would require time exponential to the number of participating operations. Therefore, several heuristics are presented that can compute the optimal solution in an efficient way.

Tao and Lin [58] present a different algorithm to select the optimum execution plan. Suppose that we are interested in an execution plan comprising of a sequence of *k* consecutively executed Web Services $\{S_1, S_2, ..., S_k\}$, or "pipeline" as called in [58]. For each service a set of candidate points of access is defined, and a utility function for each. The utility function encompasses benefit factors and cost for each selection candidate. The higher the utility function the highest the QoS that is delivered. If it is desirable the selected execution sequence to have total execution time below a threshold *R*, the whole problem can be formulated as *Multiple Choice Knapsack* (MCK). In the MCK we have several classes, each class with a weight and a profit, and a knapsack with limited capacity. The goal is to collect an item from each class, such that the total profit is maximized, however the total weight does not exceed knapsack's capacity. The reduction of execution sequence selection to MCK can be done as follows:

- Each executed step corresponds to a class in MCK
- The selection candidates for each service are the items in the MCK classes with the response time of each candidate to be the weight of this item and the utility function to be the profit of the item selection
- The objective is to maximize the total utility while keeping the total response time below *R*, the knapsack capacity.

Tao and Lin [58], give a dynamic programming algorithm and some variations to solve MCK, and consequently service sequence selection, efficiently. The extension of the pipeline selection to non-linear Web Service compositions, i.e. Directed Acyclic Graph organizations, is carried out after enumerating all distinct paths of length *k*, and running a shortest path algorithm to decide the optimal execution path.

5.3 QoWS delivery

An orthogonal aspect with respect to QoWS discovery is the QoWS delivery. A series of related publications [9], [57] present what is called a *QoS Broker*. The QoS Broker is implemented as a separate module, between the UDDI registry and the end user(s). QoS Broker consists of a database that records service QoS information, a QoS Information Manager that handles all the stored information, a QoS Negotiation Manager that performs all the tasks to find qualified Web Services, to

ensure that sufficient resources are allocated, to confirm and to allow binding, and the QoS Analyzer that interacts with the DB to compute and store QoS scores for each the provided WSs. An overview of the QoS broker is shown in Figure 4.

The QoS broker needs to use two kinds of algorithms; one for discovering QoS qualified services and one to ensure resource allocation. The algorithms of the first kind fall in the category of algorithms presented in the previous section, however the particular choices in [9], [57], are not disclosed. The authors in [57] have developed several allocation strategies for homogeneous resource allocation (HQ) and non-homogeneous allocation (RQ), i.e. favoring some requests against others. Table 5 summarizes the main characteristics of the surveyed methods for QoWS support.

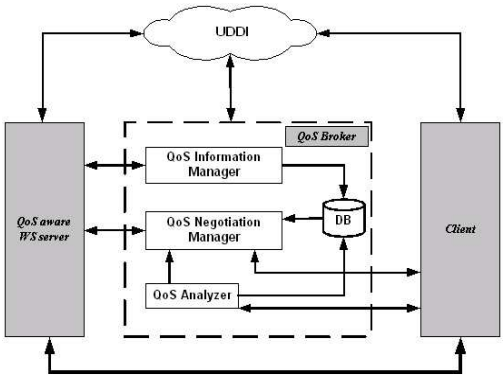

Figure 4 The QoWS broker architecture, adapted from Chen et al. [9]

	Method	**Comment**
Discovery	Ouzzani [35], Ouzzani and Bouguettaya [34]	Quality scores for each web service, heuristics to discover min. cost execution plan
	Liu et al. [26]	Method to derive an aggregate score for a web service and select accordingly
	Tao and Lin [58]	Model execution plan as MCK, solve with dynamic programming
Delivery	QoWS Broker [9], [57]	An intermediate to register WS profiles and provide quality assurances
	Menascé [66]	Multiattribute utility theory is used for the combination of different QoS attributes

Table 5 Attempts for QoS support

6 Sequencing Web Services into business workflows

Web services have been utilized to facilitate end-to-end business processes as they provide cross platform support and transparent integration among service providers. The nature of these processes becomes increasingly complex and as a result languages like BPEL4WS [5] provide a framework to represent and control them. In [6], the concept of *workflow* quality of service is introduced. The authors outline research directions in order to develop efficient workflows using web services towards

the following areas: specification, prediction algorithms and methods, monitoring tools, and mechanisms to control the quality of service. In this section, we present mechanisms that deliver sequencing WS intelligence.

The work of [60] introduces the persistence of services via commitments. Commitments represent agreements between service requesters and providers, which ensure successful business transactions when fulfilled. Like service level agreements, commitments are supposed to last longer than individual executions of services. The management of commitments is performed through a defined set of operations, which furthermore assist in the design of service models and workflows

WebQ [40] is proposed as WS workflow Framework with QoS support. WebQ implements adaptive selection process and simultaneously provides corresponding binding and execution of Web Services. The specific framework provides a QoS model for WS discovery and execution, as well as a system that utilizes a series of proposed rule-based algorithms for the computation of the QoS parameters.

In [19] it is recognized that quality in WS workflows can be assured through efficient and dynamic handling of exceptions. The authors introduce a number of quality parameters and implement their specifications in Web-Flow. Again in this case, rules are utilized to monitor and to handle exception automatically. The proposed system ensures the process workflow in several cases as the violation of quality constraints or service faults.

A systematic and analytical approach for efficient workflow access is given by Ouzzani and Bouguettaya in [34] (extensive details are presented in Ouzzani's Phd thesis [35]). They propose a novel infrastructure that offers complex and optimized query facilities for Web services. To facilitate the interactions between users and Web services, they define a set of domain-specific operations called virtual operations — so-called because they don't belong to any actual Web service. In this way, the space of Web services within a given application domain is represented in a generic way. At completion of the query, different matching modes match the virtual operations against concrete operations from actual Web services. The proposed model selects and combines appropriate operations based on relevance, QoWS, matching degrees, ratings, and feasibility.

Driven by the research in cross-enterprise workflow and AI planning, in [24] an overview of automatic Web Service composition is presented. The work limits its focus only to efforts concerning automatic Web service composition. It discusses a categorization in methods deriving from the workflow area and in those that come from AI planning research community. In fact, the authors set a distinction between methods that can generate the process model automatically and the methods that can locate the correct services if an abstract process model is given. The former usually come from the workflow area and they consider a composite service to include a set of atomic services together with the control and data flow among the services. On the other hand, dynamic composition methods need automatic plan generation. Most methods in such category are related to AI planning and deductive theorem proving. Interesting approaches discussed include: EFlow [7], composite service definition language (CSDL) [8], Polymorphic Process Model (PPM) [51] and Web service composition method based on PDDL [29], SWORD [41] and theorem proving approaches [59].

Another aspect of QoS assurance while sequencing WS workflow is the need to utilize multiple Web Services for a single remote process invocation. In this case multiple WS invocation is called *WS composition*. Interesting approaches have been presented by Menascé in [66] and [67] to facilitate dynamic composition of WS taking into consideration QoS parameters. The author supports

elaborately that multiattribute utility theory could provide an answer to the combination of different QoS attributes. As a result the best WS composition is the one with the highest combined score.

7 Conclusions

The Web Service discovery mechanisms discussed, strive to achieve a set of goals to enhance efficiency at the matching and finally at the binding procedure. Among others, discovery mechanisms should enable search and assessment solely based on a Web Service's outer view. Assessment is based on multi-criteria decision-making [23].

Furthermore, focus should be given on defining QoWS metrics. This is important since it will enable the refinement of WS Discovery mechanisms in order to reach minimum standards in performance, security and availability in matching and binding results. Support of load balancing in the WS delivery starting at the moment of choosing one will then be possible. This can boost performance especially at situations with excessively increased workload is met.

The introduction of Blue Pages in [36] can be considered as a next step towards that direction having in mind the rising need for the Semantic Web realization attempt. When directing upwards the de-centralized solutions axis in Figure 2, P2P solutions [25], [28], [50] have shown the best performance up to now. Additionally, in Figure 3 the cases of [28] and [50] provide wide scalability and options for network load balancing and service stability which are clearly a competitive advantage to the simplistic keyword based searching of UDDI, even in its federated edition. Overall, in categories of Figure 2, GRID [31] solution provides the maximum potentials for efficient discovery especially for cross-enterprise oriented solutions. Furthermore, in terms of flexibility the vector space model provides a solid ground for approaches better than Summarizing, a service oriented grid like WS discovery solution can be supported by the necessary community mechanisms for the next generation effective searching for WSs.

Case/ Problem Description	**Discovery Technique**
Single point failure resistance – Load balancing	[45] UDDI Federation
Low availability or short appearance of nodes providing the service	[25], [50] Peer to Peer, [34] flexible WS search
Programmer based Searching using Simple Interface	[33] UDDI, [64] Flexible retrieval
Dynamic – Program based discovery	[27] Dynamic selection
Grid support	[31] GRID, [28] NIPPERS
Semantic capabilities integration	[49] P2P
Full industrial support during installation, deployment and support	[33] UDDI
QoS support	[9] & [57] QoS broker, [26] QoS framework
Information Retrieval well known techniques incorporation	[48] & [51] Vector Space Matching
QoS aware execution planning	[35], Linear Optimization

Table 6 Cases that arise in Web Service Discovery and their corresponding solution techniques.

A concluding highlighted presentation of the approaches that would attempt to provide an overall ranking of the mechanisms discussed, would perhaps mislead the user towards potentially wrong assumptions. However, a last categorization is presented in Table 6, which groups techniques

according to particular cases and problems that practitioners frequently deal with. Table 6 presents a Case/Problem vs. Technique correspondence table.

The work on Discovery mechanisms should reach resulting structures not only applicable to WSs, but web-based or other software components in general. This would require introducing some additional specifications about the platform, the system requirements, the type of reuse, the type of code, and the scope of supply [36]. These specifications could be added within the administrative information. Consequently, even a unified specification of software components could eventually be achieved (which could be the basis for future component catalogues and CASE tools).

Concluding this discussion, several approaches have been discussed using different view-points. Beside UDDI, emerging decentralized aspects such as P2P solutions are promising. Enhancement of data models is also possible by elaborating both IR techniques and ontologies, especially when taking into account that research about the Semantic Web is particularly popular at the moment. WS Discovery mechanisms have a role even more important than Web searching, because they facilitate the need for collaboration among business processes and consumers over widely accepted Web standards.

Acknowledgements

Part of this research has been supported by the Karatheodoris Research Program (B.391) at the University of Patras, Greece.

References

1. Austin, D., Barbir, A., Ferris, C. and Garg, S. (eds.): Web Service Architecture Requirements. W3C Working Group Notes (2004) http://www.w3.org/TR/wsa-reqs
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval, Addison-Wesley, (1999)
3. Berry, M. W., Dumais, S. T. and O'Brien, G. W.: Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4), 573-595, (1995).
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion M., Ferris, C. and Orchard, D. (eds.): Web Services Architecture. W3C WG Note. http://www.w3.org/TR/ws-arch/
5. BPEL4WS Specification: Business Process Execution Language for Web Services Version 1.1. *http://www-106.ibm.com/developerworks/library/ws-bpel/*
6. Cardoso, J., Sheth, A. and Miller, J.: Workflow Quality of Service. *Proceedings of the International Conf. on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference*, 2002.
7. Casati, F., Ilnicki, S., and Jin. L.: Adaptive and dynamic service composition in EFlow. *Proceedings of 12th International Conference on Advanced Information Systems Engineering*, 2000.
8. Casati, F., Sayal, M., and Shan M.-C.: Developing e-services for composing eservices. *Proceedings of 13th International Conference on Advanced Information Systems Engineering,* 2001.
9. Chen, H., Tao, Y. and Lin, K.-J.: QCWS: An Implementation of QoS-Capable Multimedia Web Services. *Proc. of the IEEE Fifth International Symposium on Multimedia Software Engineering*, 2003.
10. DAML-S Coalition: DAML-S: Web Service Description for the Semantic Web, *Proc. 1st Int'l Semantic Web Conf.,* 2002.
11. Decker, K., Sycara, K., Williamson, M.: Middle-Agents for the Internet. *Proc. 15th IJCAI* (1997) 578-583
12. Dogac, A., Laleci, G., Kabak, Y. and Cingil, I.: ExploitingWeb Service Semantics: Taxonomies vs. Ontologies. In *IEEE Data Engineering Bulletin*, Vol. 25, No. 4, December 2002.
13. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J., Similarity Search for Web Services. *Proc. of the 30th International Conference on Very Large Databases,* pp.372-383, 2004.
14. Elgedawy, I., Tari, Z. and Winikoff, M.: Scenario Matching Using Functional Substitutability in Web Services. *WISE 2004, LNCS 3306*, pp.59–65, 2004.
15. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid. Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*. (2001)

16. Gaede, V., Gunther, O.: Multidimensional Access Methods. *ACM Computing Surveys*, 30(2). (1998)
17. Gong L., JXTA: A Network Programming Environment. *IEEE Internet Computing*, (5)3:88-95, 2001.
18. Greiner, U. and Rahm, E.: Quality-Oriented Handling of Exceptions in Web-Service-Based Cooperative Processes, *Proc. of EAI-Workshop 2004 - Enterprise Application Integration*, pp.11-18, 2004.
19. Hu, Z.: Using Ontology to Bind Web Services to the Data Model of Automation Systems, Revised from the *NODe Workshops on Web, Web-Services, and Database Systems*. (2002) 154 – 168.
20. Jannach, D., Leopold, K., Timmerer, C. and Hellwagner, H.: Toward Semantic Web Services for Multimedia Adaptation. *WISE 2004, LNCS 3306*, pp. 641–652, 2004.
21. Jeckle, M., Zengler, B.: Active UDDI - an Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation, Revised from *the NODe Workshops on Web, Web-Services and Database Systems*, (2003) 91-99
22. Jinghai, R. and Xiaomeng, S.: A Survey of Automated Web Service Composition Methods. *SWSWPC 2004, LNCS 3387*, pp. 43–54.
23. Konito, J.: A Case Study in Applying a Systematic Method for COTS Selection. *Proc. 18th Int. Conf. on Soft. Eng.* (1996) 201–209
24. Lacey, P.: Uddi & Dynamic Web Service Discovery. http://www.ddj.com/articles/2004/0402
25. Li, Y., Zou, F., Wu, Z., Ma, F.: PWSD: A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network, *Proc. APWeb04, LNCS 3007* (2004) 291-300
26. Liu, Y., Ngu, A., and Zeng, L. QoS Computation and Policing in Dynamic Web Service Selection. *Proc. World Wide Web Conference*, pp. 66 – 73, 2004.
27. Makris, C., Panagis, Y., Sakkopoulos, E., and Tsakalidis, A., Efficient and adaptive discovery techniques of web services handling large data sets, *J. Systems and Software*, in press.
28. Makris, Ch., Sakkopoulos, E., Sioutas, S., Triantafillou, P., Tsakalidis, A., Vassiliadis, B.: NIPPERS: Network of InterPolated PeERS for Web Service Discovery. *Proc of the 2005 IEEE International Conference on Information Technology: Coding & Computing*, 2005.
29. McDermott, D.: Estimated-regression planning for interactions with Web services. *Proceedings of the 6th International Conference on AI Planning and Scheduling*, 2002.
30. Montebello, M., C. Abela, C.: DAML Enabled Web Services and Agents in the Semantic Web. Revised from the NODe Workshops on Web, Web-Services and Database Systems (2003) 46 – 58
31. Moreau, L., Avila-Rosas, A., Dialani, V., Miles, S. and Liu, X. Agents for the Grid: A Comparison with Web Services (part II). *Proc. of Workshop on Challenges in Open Agent Systems* (2002) 52-56
32. O'Sullivan, J., Edmond, D. and Ter Hofstede, A.: What's in a Service? Towards Accurate Description of Non-Functional Service Properties. *Distributed and Parallel Databases*, 12, pp.117–133, 2002.
33. OASIS UDDI Specifications TC - Committee Specifications http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm
34. Ouzzani M., Bouguettaya A.: Efficient Access to Web Services. *IEEE Internet Computing* (2004) 34-44
35. Ouzzani, M., Efficient Delivery of Web Services, PhD Thesis, Virginia Tech University. (2004)
36. Overhage, S.: On Specifying Web Services Using UDDI Improvements. In *Proc. NetObjectdays,* 2002.
37. OWL-S Specifications, http://www.daml.org/services/owl-s/1.0/
38. Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: Semantic Matching of Web Services Capabilities. *Proceedings of the 1st Int. Semantic Web Conference,* 2002.
39. Patel, Ch., Supekar, K. and Lee, Y.: A QoS Oriented Framework for Adaptive Management of Web Service based Workflows. *Proc. of Database and Expert Systems 2003 conference,, LNCS*, 2003, pp.826 – 835.
40. Patil, A., Oundhakar, S., Sheth, A. and Verma, K.: METEOR-S Web service Annotation Framework. In the *Proc. of the World Wide Web Conference*, 2004.
41. Ponnekanti S. R. and Fox. A. SWORD: A developer toolkit for Web service composition. *Proceedings of the 11th World Wide Web Conference*, 2002.
42. Rajasekaran, P., Miller, J., Verma, K. and Sheth, A.: Enhancing Web Services Description and Discovery to Facilitate Composition. *SWSWPC 2004, LNCS 3387*, 2004.
43. Ran S.: A Model for Web Services Discovery with QoS, *ACM SIGecom Exchanges*, 4 (1) 1-10 (2003).
44. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. *Proceedings of ACM SIGCOMM`01* (2001).
45. Rompothong, P., Senivongse, Tw.: A Query Federation of UDDI Registries, *ISICT* (2003)
46. Rowstron, A., Druschel, P.: Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Proc. of the 18th IFIP/ACM Middleware* (2001).
47. rWhois RFC, http://rfc.net/rfc2167.html
48. Sajjanhar, A., Hou, J., Zhang, Y.: Algorithm for Web Services Matching, *Proc. APWeb 2004, Lecture notes in Computer Science 3007* (2004) 665-670.

49. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: A scalable and ontology-based P2P infrastructure for semantic web services. *Proc. 2nd Int. Conf. P2P'02*, 104–111.
50. Schmidt, C., Parashar, M. A.: Peer-to-Peer Approach to Web Service Discovery. *World Wide Web: Internet and Web Information Systems*, 7. (2004) 211-229
51. Schuster, H., Georgakopoulos, D., Cichocki, A. and Baker, D.: Modeling and composing service-based and reference process-based multi-enterprise processes. *Proc. of 12th Int. Conf. on Advanced Information Systems Engineering*, 2000.
52. Sivashanmugam, K., Verma, K., Mulye, R., Zhong, Z., and Sheth, A.: Speed-R: Semantic P2P environment for diverse Web Service registries, http://webster.cs.uga.edu/~mulye/SemEnt/Speed-R.html. (2004)
53. Sivashanmugam, K., Verma, K., Sheth, A. and Miller, J.: Adding Semantics to Web Services Standards. *Proceedings of the 1st International Conference on Web Services*, pp.395-401, 2003.
54. Stoica, I., Morris, R., Karger, D., Kaashoek M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for Internet applications. *ACM SIGCOMM`01*, 2001.
55. Sycara K., Dynamic Discovery, Invocation and Composition of Semantic Web Services, Third Hellenic Conference on AI, SETN 2004. LNCS 3025 (2004) 3-12
56. Sycara, K., Klusch, M., Widoff, S., Lu, J.: Dynamic service matchmaking among agents in open information environments, ACM SIGMOD Record, v.28 n.1. (1999) 47-53
57. Tao, Y. and Lin, K.-J. The design of of QoS Broker Algorithms for QoS-capable Web Services. *Proc. of IEEE International Conference on e-Technology, e-Commerce and e-Service*, pp.17-24, 2004.
58. Tao, Y., and Lin, K.-J. Service Selection Algorithms for Web Services with end-to-end QoS Constraints. *Proceedings of the IEEE International Conference on E-Commerce Technology,* pp.129-136, 2004.
59. Waldinger. R.: Web agents cooperating deductively. *Proceedings of FAABS 2000*, *volume 1871 of Lecture Notes in Computer Science*, pages 250–262, 2001.
60. Wan, F. and Singh, M.P.: Enabling Persistent Web Services with Commitments. *Information Technology and Management (ITM)*, 6(1), 2005. http://www.csc.ncsu.edu/ faculty/mpsingh/papers/mas/itm-04.pdf
61. Whois++ RFC, http://rfc.net/rfc1835.html
62. Yu, H., Mine, T. and Amamiya, M.: Towards Automatic Discovery of Web Portals – Semantic Description of Web Portal Capabilities. *J. Cardoso and A. Sheth (Eds.): SWSWPC 2004, LNCS 3387*, 2004.
63. Zhang, J., Chung, J.Y., Chang, C.K. and Kim, S.: WS-Net: A Petri-net Based Specification Model for Web Services. *Proceedings of ICWS 2004*, pp. 420-427.
64. Zhuge, H. and Liu, J.: Flexible retrieval of Web Services. *The Journal of Systems and Software*, 70, 2004, pp. 107–116.
65. Drosos, L., Sioutas, S., Sakkopoulos, E., Sirmakessis, S. BDT-Grid: An Efficient Scalable Peer-to-Peer Lookup System for Web Service Discovery, *Proceedings of ACM Hypertext*, *International Workshop on Peer to Peer and Service Oriented Hypermedia:Techniques and Systems*, 2005.
66. Menascé, D.A, Composing Web Services: A QoS View. *IEEE Internet Computing* 8(6), 2004, pp. 88-90
67. Menascé, D.A., Response-Time Analysis of Composite Web Services. *IEEE Internet Computing* 8(1), 2004, pp.90-92