Look Before You Leap: A Survey of Web Service Discovery

Mydhili K Nair
Assistant Professor
M S Ramaiah Institute of Technology
Bangalore, India

Dr. V.Gopalakrishna
Director
Integra Micro Systems
Bangalore, India

ABSTRACT
This paper is an exhaustive review summarizing the results, observations and findings of the renowned researchers in the domain of Web Service Discovery. After extensive scavenging on the Internet, we felt that there is a paucity of good quality survey papers, which can help, provide directions to a researcher looking for a fertile area to explore in the Web Services arena, especially in *Discovery of Web Services*. We highlight here the list of problems, which need to be looked into and investigated. This comprehensive listing of the open-ended unresolved issues is presented in a novel way, by providing its *Cause-Effect Analysis*. In this paper, we hand-hold you to into a literary journey that provides a glimpse of the huge spectrum of work investigated by researchers globally in the field of *Discovering the Right Services*, based on the Requirements provided by the Consumer.

Categories and Subject Descriptions
Web Services Discovery Survey, QoS Typology, Quality Models, Service Discovery Architecture and Frameworks.

General Terms
Ranking and Selection Algorithms, Quality of Service (QoS), Quality Metrics and Attributes, Broker Matchmaking, Reputation and Trust, Endorsement, Service Provider and Consumer.

Keywords
Web Service Discovery Survey, QoS, Ranking Algorithms, Matchmaking Algorithms, WSMO, OWL-S, DAML-S, Broker, Ontology, UDDI, Service Discovery Frameworks.

1. INTRODUCTION
Four Doctrines for every aspiring Doctorate:

- Download Papers, Categorize, Read. Do a thorough literature survey of the area of research
- Keep a lookout of the problems highlighted by fellow researchers and how they tried to solve it.
- Avoid the ***"Analysis Paralysis"***. Thoroughly analyze these problems and check their validity, feasibility, relevance. Sieve and filter the problem(s) you want to work on.
- Best place to start. A good survey paper!

This paper aims to be the last doctrine! We give a concise view of our survey into the realm of Web Services Research. Fig 1 depicts the entire gamut of domains involved in this research domain, namely, Discovery, Composition, Security, Choreography, Orchestration, Governance and Management[1,4,7,8].We would like to draw attention to the scope and focus of this paper, Services Discovery, specifically ***Discovery of Web Services***. Other spheres are beyond the scope.

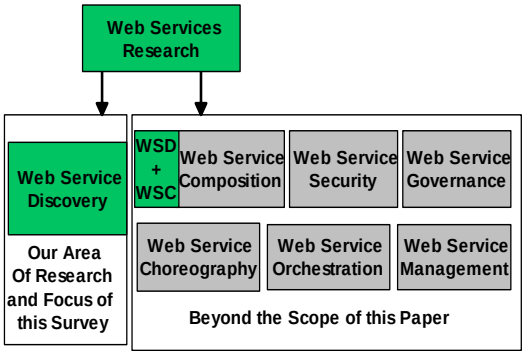

Fig 1. Scope of Our Survey

The realm of ***Web Service Discovery (WSD)*** is a fundamental area of research under distributed and ubiquitous computing. It has been a fertile area of research for less than a decade now. Nevertheless, there has been a significant amount of work carried out by industries, universities and individual researchers. These works cover a wide range of aspects related to the techniques and approaches involved in Discovering and Selecting Services.

In this paper, we provide comprehensive results of our survey into these reputed works, **focusing** on the ***Requirements*** given by the Consumer or User of the Web Service. We investigate into the various types of *Requirements* that a Consumer is likely to have. We review the various techniques adopted by researchers to model these requirements and effectively search for the correct Web Services that match the requirements specified.

A precise summary is provided in this paper, of our investigation into the works of researchers in *modeling* the Consumer Requirements, especially the ***Quality of Service (QoS)***, a complex non-functional aspect to model. The ***Pros and Cons*** of these various modeling approaches is provided, taking care *to spotlight the open-ended unsolved issues*, so that researchers reading this survey can take these problems and work on them!

In this paper, we **do not** delve into the other spheres in Web Service Discovery such as *Broker Architectures, Frameworks, Match-making, Selection* and *Ranking Algorithms* as well as techniques to *store, organize* and *publish* the Web Service Profiles effectively for them to be discovered easily. To understand these, please refer to an allied work of ours [12]. What **we do provide** in this paper, is an exhaustive list of challenges that exist in the broad domain of Web Services, relevant to Web Service Discovery. Apart from listing the

problems, we provide a ***Cause-Effect Analysis*** of these problems thrown up which are yet to be solved.

The remaining parts of the paper are organized as follows. Section 2 covers basic concepts in the Web Services domain, while Section 3 focuses on the problem in hand, namely ***Web Service Discovery (WSD)***. Section 4 describes the work done by various researchers in WSD. Section 5 attempts to provide a comparative analysis. Section 6 provides the conclusion.

2. Background and Concepts

2.1 Foundation of the Services Domain

The **umbrella domain** of our survey is ***Service Oriented Architecture (SOA)*** and ***Computing (SOC)*** .*Many seek* clarity, on the exact difference between the two terminologies. We attempt to give the precise distinction here.

In ***SOA***, software resources are packaged as '***Services'***. They are well-defined, self-contained modules, which provide business functionality. The *Services* are independent of the state or context of other services. They communicate with each other requesting execution of their operations to collectively support a common business task or process. They have a published interface and are described in a standard definition language.

Basic services, their descriptions, and operations *(publication, discovery, selection, and binding)* that produce or utilize such descriptions constitute the foundation layer of the ***SOA Pyramid*** [1, 2, 4]. The higher layers provide additional support required for service composition and service management. ***SOC,*** is the computing paradigm, which utilizes *Services* as the fundamental element of developing applications. To build this Service model, SOC relies on SOA, to define the layers, functionality and roles of the various services and stakeholders involved [3, 5, 11].

Services come in ***two flavors***: *Simple* and *Complex*. **Simple Services** focus on doing specific business tasks, while **Composite Services** involve *assembling* or *composing* existing services. Whatever the flavor, Services are offered by ***Service Providers***. They are organizations that procure or develop the service implementations, supply their service descriptions and provide related technical and business support. For example, there could be three Service Providers providing three distinct *Simple Services* such as Order Tracking, Order Billing and Customer Relationship Management. A Service Provider Enterprise could offer a *Composite Service* that ***composes*** these services together to create a distributed E-Business application, which provides customized ordering, customer support, and billing for a specialized product line (e.g., telecommunication equipment, medical insurance, etc).

Service Composition[6][3][5] is an emerging and fertile area of research, having aspects such as ***Business to Business(B2B)*** protocols, service conversation messaging formats, service integration, composition algorithms etc. In this paper, we **do not** touch upon these aspects, which are inherent to the principles of Service Composition. What we do cover, is the analysis of the work of researchers who have attempted to ***discover*** these complex, composed service, based on their functional and non-functional requirements. Fig. depicts the scope of our study.

SOA is a vast and complex subject, embracing a gamut of technologies innately integrated. SOC is built on inherently related themes ranging from Service Management, Composition, Security, Choreography, Orchestration and Discovery[2].

Our realm of research is **Web Services(WS),*which are specific kind of services*** identified by a URI. WSs are 'end-points' of a Service where they use Internet as the communication medium and adhere to open Internet-based standards. We summarize here the results of our meticulous survey in *Web Services*, particularly ***discovering the right services,*** as depicted in Fig. 1.

2.2 The Web Services Nuts and Bolts

Fig 2. Three Deployable Web Services and a WSDL Interface

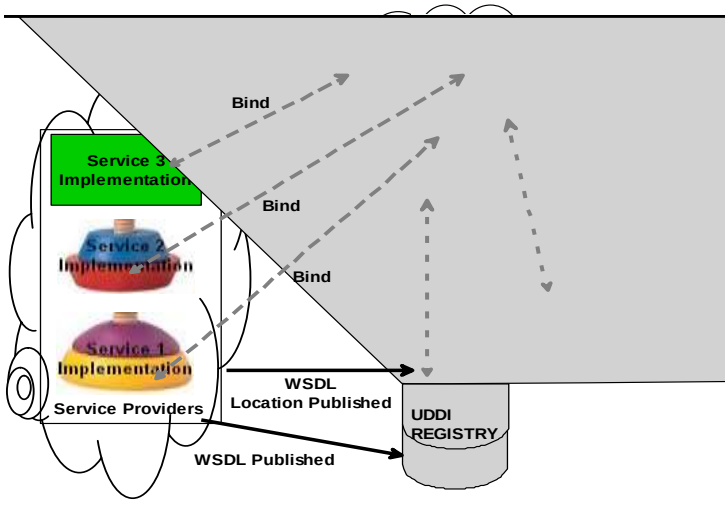

Fig 3. Web Services Basics

The ***World Wide Web Consortium (W3C)*** defines a *"Web Service"* as a software system designed to support interoperable machine-to-machine interaction over a network[7,8]. The buzzword here is '*interoperable*', which is achieved through a platform independent, standard meta-data for message exchange, which is XML(EXtensible ***M***arkup ***L***anguage). Interactions of Web-Services occur as ***Simple Object Access Protocol(SOAP)*** calls carrying **XML**(E***X***tensible ***M***arkup ***L***anguage) data content. The service description of the WS is expressed using ***Web Service Definition Language***(***WSDL***)an XML-based standard. As shown in Fig 2., **WSDL** is used to ***Publish*** a WS in terms of its ***ports*** (addresses implementing this

service), ***port types*** (the abstract definition of operations and exchanges of messages), and ***bindings*** (the concrete definition of the packaging and transportation protocols used to inter-connect two conversing endpoints).

The ***U**niversal **D**iscovery **D**escription and **I**ntegration*(***UDDI***)standard is a *directory service* that contains service publications and enables ***W**eb **S**ervice **C**onsumers*(***WSC***) to locate candidate services and discover their details. Thus, WSs are defined as a set of standards, SOAP, UDDI, WSDL, which enable a flexible way for applications to interact with each other over networks. All these standards are XML based allowing applications to interact with each other across networks, no matter what languages and platforms they use [7,9].Self description and platform independence are two features which distinguish web services from other distributed computing technologies such as *CORBA*(***C***omponent ***O***bject ***R***equest ***B***roker) and *DCOM*(***D***istributed ***C***omponent ***O***bject ***M***odel)[14].

In true distributed computing sense, either the WSDL could be published as such in the UDDI Registry or the repository could hold the location of the WSDL file, as depicted in Fig 3. Search engines such as Google, AlltheWeb, Baidu, Yahoo have become new sources for finding web services[47].Web Crawler WSDL Search Engines can be created to fish-out the required WSDLs of Web Services sought after. This is covered in Section 4.2, Table 2.

3. Web Service Discovery: Principle

The Web Services developed, deployed and published by the Service Providers mean nothing unless the Service Consumers can search, locate and bind to them. This fundamental need forms a relationship between three kinds of participants: the ***W**eb **S**ervice **P**rovider(**WSP**)*, the *Web Service Discovery Agency/Middle-ware interacting with the Service Registry* and the ***W**eb **S**ervice **C**onsumer(**WSC**)*, forming a Web Services Triad[8].

The typical interactions involve the ***publish***, ***find*** and ***bind*** operations [8][2] as shown in Fig. For example, a Provider hosts an internet accessible module, which is the actual implementation of a given service. A *WSDL* of the WS is defined by the Provider, which is the description of the service and an interface to access it. This WSDL could be provided to the Consumer directly so that it they can bind to the service. However, this is not a feasible approach, as it is impossible for the Provider to know who the potential Consumers of his service are. Therefore, the WSDL is provided to a well-known Service Discovery Agency, who **publishes** it, thus making the service *'discoverable'*. The Discovery Agency is associated with a *UDDI*, which is a registry maintain the details of all the services published with it.

Thus, when a Consumer wants a Service with a particular functionality (e.g. Hotel Booking), he initiates the **find** operation, to retrieve the service description(WSDL), from the Discovery Agency. Using this WSDL, the Consumer binds with the Service Provider, after which the internet accessible module, which is the actual WS implementation is invoked and rendered to the Consumer. A point to note here is that the WSP and WSC roles are interchangeable, meaning, a Consumer could be Provider for a different Service.

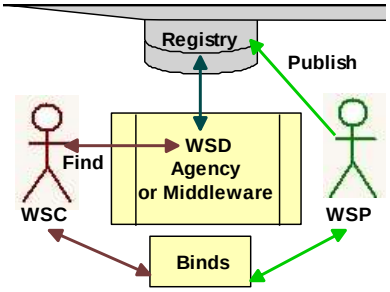

Fig 4. Web Services Triad plus Discovery Agency /Middleware

4. WSD: Problems, Solutions by Researchers

4.1 Problems in finding the 'Right' Service

Although the future of Web Services looks very promising, there are a lot of challenging problems associated with it. In this paper, we highlight the problems in ***W**eb **S**ervices **D**iscovery*. Simply put this is hunt for a solution or technique to find the *'right service'* for the Consumer. We start by listing down the well-known problems worked and analyzed by researchers around the world.

As conveyed in Section 2.2, the key artifact in discovering a service is through its WSDL and searching for its published description in the UDDI. The third player in WSD apart from the Providers and Consumers is the WSD Agency / Middleware.

Table 1 summarizes the known and important problems associated with Web Service Discovery, tackled by various organizations and individual researchers all over the world.

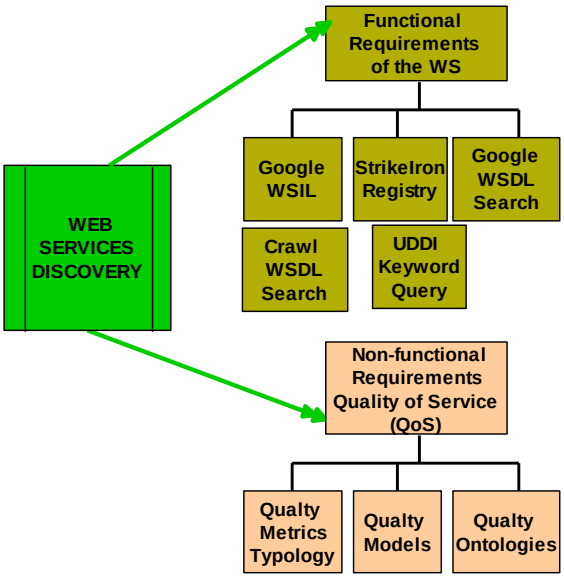

Fig 5. Two Streams of Thought in Web Service Discovery (Functional and Non-Functional Requirements)

Table 1. Cause Effect Analysis of Some Important Problems Associated with Web Service Discovery

Sl#	Problem	Cause / Reason of the Problem	Effect of the Problem
	WSDL Problem		
1	**WSDL** is inherently designed to give descriptions detailing its functional aspects like Service Type, Implementation interface details such as the port to bind to, the type of parameters etc. It is ***not designed*** to publish the non-functional aspects[2,4,7,8,9].	WSDL is ***not designed*** to take the ***"semantic descriptions"*** of the service. It is used to *Publish* a WS in terms of its ***ports***, ***port types*** and ***bindings***[2,4,7,8,9].	This makes it difficult to store the non-functional aspects of the service such as its ***Q**uality **o**f **S**ervice* (***QoS***). Parameters such as reliability, availability, response time, throughput, mean time before failure, price, etc. ***Several techniques*** have been formulated to solve this problem[23,25,32,33,14,35,34]
	UDDI Problem		
2	Current **UDDI** implementations are limited in scope. It is ***not innately designed*** to publish and store the QoS requirements and other non-functional requirements of a service[31,18,19].	UDDI allows search on ***limited attributes*** of a service, namely, ***Service Name*** (selected by the Service Provider), ***keyReference*** (unique for a service), or on a ***categoryBag*** (listing all the business categories[31,18,19].	This problem makes it difficult to store within the UDDI, the ***run-time performance parameters*** of the service capturing its QoS parameters. It is also difficult to capture the ***Customer Feedback*** about the service and store it to analyze and improve on these valuable metrics[31,18,19,20,15,16].
3	Public UDDI registries, that were run by IBM, Microsoft, SAP and NTT Com. have been ***shut down*** in the beginning of 2006[15].	There was ***no consensus*** regarding ownership of the root UDDI rsegistries. UBRs used to contain listings of businesses that ***no longer existed*** and sites that were ***no longer active*** [19,15].	There is ***no Universal Registry*** where all Web Services are published. This makes it difficult to check the performance, scalability and statistical gathering of data. An earlier work carried out by Su Myeon Kim and Marcel-Catalin Rosu[19] reports that only one-third of the 1200 registered services referenced a valid WSDL.
	Web Service Discovery Middle Agent / Engine Challenges		
4	The WSes published are tagged with a lot of information, making ***narrowing down / filtering out the attributes*** difficult. [18,19]	This is because repositories store information gathered about the service provider apart from the service profile information. [18,19]	Web Service Discovery Engine must be able to ***process this vast data*** about the service provider[23,18,19].
5	There would be ***multiple versions*** of WSes in the repositories. This means that Web Service Discovery Algorithms must be able to handle both *'design time'* and *'run time phases'* of discovery. The matchmaking algorithm must be able to bind to the correct WS version[3,2,27,16,17].	Web Services are an emerging technology. The development methodology of WS Projects usually adopts an ***incremental*** model[3]. Thus, the basic WSDL structure depicting the port, port types and bindings will not change between incremental changes. The underlying web method implementations imbibing the incremental changes, will differ across versions[2][21].	The ***'design time'*** matches would be done against the Web Service Interface, WSDL, which is the same between versions. [26***]'Run time'*** matches must be able to pull out the correct Version, which has the service description and profile that match the user needs. [28]The latest version need not always be what the user wants!
6	Majority of current approaches proposed by researchers, ***lack*** a reliable, stable and trust-worthy ***dynamic discovery and binding architecture***.[22,13].	In many approaches, apart from the basic match-making, there is the method of ranking the Web Services based on its '***Reputation'***, which is a factor calculated in addition to its Service Profile[26,27,28]	The pertinent question here is the ***integrity, reliability*** and ***trust-worthiness***[25,29,29] of the Agencies rolling out these 'Reputations' / 'Endorsements'. They could be doctored to suit the business needs of a Service Provider.
7	Service Consumers should be able to automatically select and bind to the desired services ***without manual intervention***[26,27,28].	Fully automated complete binding involves resolving multiple versioning between the WSes, ***complicated*** matchmaking, ranking and selection algorithms etc[2,4,7,8,9].	The inability to have a trust-worthy framework for dynamically binding caused the ***mushrooming*** of a lot of ***'Broker'*** Architectures that does this job[23,24,25]
8	Current approaches do not have ***memory*** of past service bindings and interactions[25,26,27].	There is no standardized means to imbibe ***learning*** from past experience[22,13].	***Redundancy*** and ***repetition*** in advertising the Service Profiles, matching them against the Customers requirements[13,14,24,25].
9	The biggest hurdle is the ***heterogeneity*** between services. For example, they may have different formats for exchanging data [15,16,17,21,27,28].	Domain specific terms and concepts differ between vendors, causing ***non-uniformity*** in the way data is published. Technical differences exist between WSs with similar functionality implemented on different platforms[15,16,17,21,27,28].	This causes immense problems while scavenging for WSs based on its functional requirements, which are innately domain specific. [15,16,17,21,27,28]

Table 2. Summary of the Search Methods to extract the WSDL: Functional Requirements Based WSD

	Google WSDL	**Google WSIL**	**StrikeIron Registry**	**UDDI Registry**	**Crawl WSDL Search**
Approach	Use Google WS API & Developer Kit and extract the WSDL references using the Google Search Engine.	Here, the Google WS API & Developer Kit is used to extract ***W**eb **S**ervice **I**nspection **L**anguage* Document WSIL, which can be parsed for WSDL locations.	Search a registry not associated with a public Universal Business Registry (UBR) like Microsoft. Instead, StrikeIron Registry is used.	Search is implemented using Sun's Java WS Toolkit(JAXR) to query the registry. Public UBR like Microsoft needs to be used.	Uses web crawling to try and locate a WSDL from a domain. 'Crawling' starts at home page and goes to a specified number of levels.
Specifics	Uses *Four Options* namely, **a)**Using Stemmed terms from the Home Page. **b)**Using Unstemmed terms from Home Page. **c)**Using "**inurl:**" Keyword search. E.g. URL is www.amazon.com For Google **WSDL** Approach: Software extracts "**inurl:wsdl**" from "**inurl:domain**" which here means extract WSDL from "amazon.com" For Google **WSIL** Approach: Software extracts "**inurl:wsil**" from "**inurl:domain**" which here means extract WSIL from "amazon.com" **d)**Using "**site:**" Keyword Search For Google **WSDL** Approach: Software extracts "**inurl:wsdl**" from "**site:domain**". Means extract WSDL from "amazon.com" For Google **WSIL** Approach: Software extracts "**inurl:wsil**" from "**site:domain**". Means extract WSIL from "amazon.com"		StrikeIron Registry is not automatically maintained, therefore WSs are not dynamically discovered. WSP register their services just as they would do in a UBR. StrikeIron provides its own software as well as API to search in its Registry. The format of the URL is http://www.strikeiron.com/search?amazon, where 'amazon' is the domain name to search for. The HTML results page is retrieved and parsed to pull out the WSDL locations.	tModels define the category. Uses two approaches: **Query by:** **a)**Name **b)C**ategory **a)Name** of the WS is searched. It tries to find a matching business name, from where the tModel entry can be traced to extract the WSDL. **b)Category** Search involves searching all tModel entries with classification **"wsdlspec"** and searching for the domain name in WSDL list.	The most well known crawler is **WebSPHINX.** It is a complex crawler configurable with a GUI front end. *Configurable* to specify crawl levels. For E.g., one can choose to crawl within a domain or move to other URLs, which have the key term of the domain. Google **"site:"** parameter needs to be activated to facilitate this crawling.

4.2 Solutions to the WSD Riddle

Table 1 precisely captures the ***Cause-Effect Analysis*** of the Problems associated with this fertile research area package called *'Discovering the Right Web Services'*. We now proceed to highlight the solutions to some of these problems proposed by Researchers across the globe.

The crucial point to note here is the criteria to search and locate the Web Services. As depicted in Fig 3., **one stream of thought** focuses on finding the WSs based on its ***functional requirements***[12][14][15][16]. For example, a Consumer could be looking for a Service Description(WSDL) that combines a set of related services for the travel domain giving an overall plan including airfare, hotel, and car rental. As is evident, the Consumer wants the *functional or operational aspects* of the service. He may be interested in selecting the Web Service from among a list of competitive Services, based on the amount of intricate details put forth by the overall rental plan.

Holger Lawsen and *Thomas Haselwanter[14], Daniel Bachlechner et al [15]Hicks et al [16]* have done elaborate and meticulous work on scavenging for Web Services based on their functional requirements. The summary of their overall approaches and specific techniques used to extract the WSDL is shown in Table 2. In all these approaches we see a predominantly keyword based technique, some searches mined to multiple filtered levels. A point to note here is that the techniques used by *Google* are applicable for other search engines such as *Baidu*, *AlltheWeb* and *Yahoo.*

However, the **WS Functional Specifications are not enough** to handle the Service Discovery Process. This is because:

- WS need to be automatically and dynamically discovered and selected at runtime. A mechanism needs to be in place to ensure that this automatic discovery happens and the best services are chosen. This needs specifications in the service profile beyond the mere functional aspects of a WS.

- With the abundance of WS created by many service providers, often a number of WSes satisfies the functional requirements of a service request. Methods were evolved to rank and select the best Web services for a request among a list of candidate Web services, which can provide similar functionality.

Table 3: Summary of Investigation into Various QoS Ontology Languages

Sl #	Language & Contributors	Feature	Approach Specifics	Pitfalls
1	**DAML-QoS Ontology** ***C. Zhou et al, 2005[42]***	•WS Domain specific ontology •Developed to supplement DAML-S QoS ontology	Has ***three*** Layers: •QoS Profile Layer: Designed for match-making purpose •QoS Property Definition Layer: To elaborate the property's domain and range constraint •QoS Metric Layer: To define and measure QoS metrics	•The Value Type range to hold QoS Metrics is limited. There are only a few value types such as string, numeric, Boolean etc. •The Impact(Positive, Negative, Exact, Close, None)of the QoS property value cannot be defined.
2	**OWL-Q Ontology** ***K. Kritikos et al, 2006[39]***	•Upper Level Ontology that extends OWL-S to describe WS QoS.	Have ***multiple facets***, each of which can be developed and extended independently. •Connecting Facet supports linking OWL-Q ontology with OWL-S ontology. •Basic Facet associates a service profile with several QoS offers (given by providers) or with only one QoS request (given by requesters). •QoS Metric Facet describes classes and properties used for defining QoS metrics. •Measurement Directive Facet is used for measuring simple metrics •Function and Schedule Facets are used for computing and measuring complex and/or dynamic metrics. •Unit Facet describes the unit of a QoS metric. •QoSValueType Facet describes the value types a QoS metric can take.	•The Impact(Positive, Negative, Exact, Close, None)of the QoS property value cannot be defined. •Rudimentary facility to do QoS Grouping of properties that share the same characteristics. QoS Grouping helps evaluate the QoS Value of a WS. •No support for WSD participant roles other than Service Provider and Requestor. Important third parties such as Certifying Authorities, Endorsement Agencies cannot be modeled.
3	**WSMO-QoS Ontology** ***X. Wang, I.Toma et al[43,44] 2006,2004***	Extended WSMO model[44] to specify quality metrics, value attributes and their correspondi ng measureme nts.	Defines a new ***class QoS*** that can be attached to the *class webService* or *class Goal* in the WSMO model. This class has the following attributes: •*hasMetricName* (string) •*hasValueType* (linguistic numeric, boolean…) •*hasMetricValue* (corresponding value which has value type specified in hasValueType) •*hasMeasurementUnit* (including conversion functions for different measurement units), •*hasValueDefinition* (logical expression for computing QoS value) •*isDynamic* (boolean) and *isOptional* (boolean) •*hasTendency* (low/small, high/large, given, for representing the tendency of the value) •*isGroup* (specifying that the property is defined by a group of other properties or not) •*hasWeight* (depicting the property's importance).	•No definitions of concrete QoS properties. These are a group of common and domain independent properties. •Very weak support to specify relationships between QoS properties such as independence or correlation (inversion, opposite, parallel). •Few support for the usage of QoS information, except for QoS priority, mandatory, and QoS grouping.
4	**QoS-Ont Ontology** ***G.Robson, I. Sommerville et al, 2005[45]***	Based on existing QoS taxonomies and models.	Has several ontologies organized as three layers •Base Layer: Has a base QoS ontology representing a minimal set of generic QoS concepts. •Attribute Layer: Contains ontologies defining particular QoS attributes and metrics. •Domain Specific Layer: Links the lower layers to specific types of systems. This layer provides concepts for connecting QoS concepts in lower layers with OWL-S service profiles. For example, network systems or Web service systems.	•This approach is similar to OWL-Q[39] mentioned above. •Does not support specifying a QoS profile from a set of QoS characteristics •No support for QoS relationships. •The conversion mechanism is used for units of QoS metrics but not for mapping different QoS parameters. •The usage support of the ontology is also very limited.

This led to the **second stream of thought**. Discover WSs based its ***non-functional requirements***. The predominant factor being *'**Q**uality **o**f **S**ervice'(**QoS**).* A Consumer may want a Service that offers the fastest response time while for another reliability or constant availability could be the criteria and a third Consumer may treat security as his most important parameter.

All these, namely, security, response time, reliability, availability come under the non-functional QoS requirements of a service. Therefore, the body of work we surveyed and referenced here is based on QoS. First, we investigated into the various Quality Models, Metrics, Attributes and Typology associated with QoS.

The hurdles to cross in order to model the quality parameters are:

- The Service Providers and Requesters use different languages and models for QoS advertisements and requirements.
- It is necessary to find a way to evolve a system, which understands different QoS concepts in QoS descriptions.
- Different domains and applications may require different QoS properties; therefore we need a more efficient and flexible method to express QoS information. [32,35,36,37,38]

Semantic technology, especially **ontology**, can be used for achieving QoS interoperability. The next section focuses on this.

4.3 QoS Ontology for Semantic Modeling

In this section, we review the approaches adopted by researchers to develop QoS Ontologies. The pros and cons of those approaches and open issues yet not addressed are discussed here.

***Ontology Definition*[41]:** *It is the study of categories of things that exist or may exist in some domain. It is a catalog of the types of things that are assumed to exist in a domain of interest **D**, from the perspective of a person who uses a language **L** for the purpose of talking about **D**.*

The most common language **L** used is predicate calculus, conceptual graphs or ***K**nowledge **I**nterchange **F**ormat(**KIF**)* all in realm of un-interpreted logic. By itself, logic says nothing about conceptual graphs or ***K**nowledge **I**nterchange **F**ormat(**KIF**)* all in anything unless combined with something. The *'something'* here is Ontology, specifically *QoS Ontology*, and the Domain of interest, **D**, is *Web Service Discovery*. This combination of logic and ontology has given us a range of languages that can express relationships about the entities in the domain of interest.

Table 3 gives a summary of our survey and exhaustive study into a few selected QoS Ontology Languages, namely **DAML-QoS, OWL-Q , WSMO-QoS** and **QoS-Ont**,. Some other works such as ***C**ontext **O**ntology **L**anguage(**CoOL**)* by Thomas Strang et al, *QoS-Onto Language* by I. V. Papaioannou et al, *QoS-MO* by G. F. Tondello et al, *onQoS Language* by E. Giallonardo et al. are not covered in this paper due to lack of space to include them. Our allied work [12] elaborates these.

5. Conclusion

In the early years of Service Oriented Computing, the number of Web Services were few. Finding the relevant services was primarily done by checking for the published services within the UDDI(UBR). By 2006, UBR closed and other alternate approaches bravely pioneered by StrikeIron could not make much impact. Today, WSDLs, are abundant, scattered across the WWW. The count of Web Services already deployed with similar functionality is mammoth in number. There is an increasing need to evolve Service Discovery Methods that help the Consumer to find the right kind of services for his requirements.

In this paper, we have put forth our survey results of the work conducted by researchers across the globe on the WS Discovery techniques based on User Requirements as their input. We conclude that the functional requirements of the WS are more or less handled by the WSDL. In this paper, we have provided an analysis of the various techniques used by search engines such as Google, Yahoo, Baidu, AlltheWeb and Web Crawlers such as WebSPHINX to fish-out the relevant WSDLs.

We have also established that the functional requirements are not sufficient to discover the right services. The predominant non-functional WSD approach adopted is to model the *Quality of Service(QoS).* In this paper, we give a concise analysis of the QoS Ontology Modeling using various semantic languages evolved by researchers. We provided the specifics of the languages, the advantages and the yet to be solved open-issues.

We hope this meticulous survey would help researchers get a strong foothold on the realm of Services Discovery and the nitty-gritty associated with it. We hope it helps them *'Look Before they Leap'* to know the challenges, analyzing them and finally narrowing down of the specific problem they would spend a good many years working on during their research tenure!

6. REFERENCES

[1] Papazoglou M.P, Willem-Jan van den Heuvel, 2007, "Service Oriented Architectures:Approaches, Technologies and Research Issues", VLDB Journal 2007, Springer-Verlag, Vol 16, Issue 3.

[2] Thomas Erl, 2008, "Service-Oriented Architecture: Concepts, Technology and Design", Pearson Education, 2nd Impression.

[3] Papazoglou M.P, 2003, "Service–Oriented Computing: Concepts, Characterisitcs and Directions", In the Proceedings of Web Information Systems Engg. (WISE '03)

[4] Eric Newcomer, Greg Lomov, "Understanding SOA with Web Services", Pearson Education, First Indian Reprint 2005.

[5] Papazoglou M.P, Georgakopoulos D, 2003, "Service Oriented Computing", Communications of the ACM, Vol.46, No.10.

[6] Jian Yang, M.P.Papazoglou, 2004, "Service Components for Managing the Life-Cycle of Service Compositions", In the Proceedings of CAiSE'03, pp 97-125.

[7] Frank P Coyle, XML, Web Services and the Data Revolution, Pearson Education, Eighth Impression, 2010.

[8] Sandeep Chatterjee, James Webber, Developing enterprise Web Services: An architect's Guide, 1st Indian Reprint, 2004.

[9] Steve Graham et al, Building Web Services with Java: Making sense of XML, SOAP, WSDL and UDDI, Pearson Education.

[10] Mydhili K.Nair, Chandan Bhosle, V. Gopalakrishna, 2009, "Net Mobile-Cop: A Hybrid 'Intelli-Agent' Framework to Manage Networks", In the Proceedings of IEEE International Conference on Intelligent and Multi-Agents(IAMA).

[11] Mydhili K Nair, Shishir M Kakaraddi, Keerthi M Ramnarayan, V Gopalakrishna, 2009,"Agent with Rule Engine: The 'Glue' for Web Service Oriented Computing applied to Network Management System",In Proceedings of IEEE International Conference on Service Computing(SCC)

[12] Mydhili K. Nair, V.Gopalakrishna, 2010, "Standing on the Shoulders of Giants: A Survey into Web Services Discovery", (In Press).

[13] Yao Wang, Julita Vassileva, 2007,"Towards Trust and Reputation Based Web Service Selection", In Multi-Agent and Grid Systems(MAGS) Journal.

[14] Ioan Taoma, Thomas Strang, et al, 2007, "Discovery in Grid and Web Services Environments: A Survey and Evaluation", In Multi-Agent and Grid Systems(MAGS) Journal, Vol 3, No.3

[15] Holger Lausen and Thomas Haselwanter,2007, "Finding Web Services", In the Proceedings of European Semantic Technology Conference(ESTC 07)

[16] Daniel Bachlechner et al, 2006, "Web Service Discovery – A Reality Check", In the 3rd European Semantic Web Conference.

[17] Janette Hicks, Madhusudhan Govindaraju, Weiyi Meng, 2007, "Enhancing Discovery of Web Services through Optimized Algorithms", In the Proceedings of IEEE International Confernece on Granular Computing(GRC 07)

[18] Ali ShaikhAli, Rashid Al-Ali et al, 2003, "UDDIe: An Extended Registry for Web Services", IEEE Workshop on Service Oriented Computing: Models, Architectures and Applications

[19] Adrian Mello, 2002 "Breathing new life into UDDI", Tech Update, ZDNET.com.

[20] Su Myeon Kim and Marcel-Catalin Rosu, 2004, "A survey of public web services", In the Proceedings of Proceedings of the 13th International Conference on the World Wide Web.

[21] John Garofalakis, Yannis Panagis, Evangelos Sakkopoulos and Athanasios Tsakalidis, 2004, "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?", In the International Workshop on Web Engineering.

[22] Ziqiang Xu, 2007, "Reputation-Enhanced Web Services Discovery with QoS", In the Proceedings of ICWS 07

[23] T. Rajendran, P. Balasubramanie,2010,"An Optimal Broker-Based Architecture for Web Service Discovery with QoS Characteristics", IJWSP, Vol. 5, No. 1

[24] Keith Decker, Katia Sycara, Mike Williamson, 1997, "Middle-Agents for the Internet", In the Proceedings of IJCAI 1997

[25] Tao Yu and Kwei-Jay Lin, 2004, "The Design of QoS Broker Algorithms For QoS-Capable Web Services", In IJWSR

[26] E. M Maximilien, Munindar P. Singh, 2004 "Toward Autonomic Web Services Trust and Selection", In the Proceedings of 2nd International Conference on Service Oriented Computing.

[27] E. M Maximilien, Munindar P. Singh, 2003, "Agent-based Architecture for Autonomic Web Service Selection, In the 1st International Workshop on Web Services and Agent Based Engg

[28] E. Michael Maximilien, Munindar P. Singh, 2005, "Multiagent System for Dynamic Web Services Selection", 1st Workshop on Service-Oriented Computing and Agent Based Engineering.

[29] E. M Maximilien, Munindar P. Singh, 2005, "Self-Adjusting Trust and Selection for Web Services", In the Proceedings of 2nd International Conference on Automatic Computing.

[30] E. Michael Maximilien, Munindar P. Singh, 2001, "Reputation and Endorsement for Web Services", In ACMSIGeCom Exchanges, Vol 3, Issue 1.

[31] A.Blum, "UDDI as an Extended Web Services Registry: Versioning, quality of service, and more". White paper, SOA World magazine, Vol. 4(6), 2004.

[32] D. Fensel and C. Bussler, 2002 "The Web Service Modeling Framework WSMF", In Electroni Commerce Research and Applicatins Journal Vol 1, Issue 2.

[33] Shuping Ran, 2003 "A Model for Web Services Discovery With QoS", In ACMSIGeCom Exchanges, Vol 4, Issue 1.

[34] Sravanthi Kalepu, Shonali Krishnaswamy, Seng Wai Loke, 2003 "Verity: A QoS Metric for Selecting Web Services and Providers", In the Proceedings of 4th International Conference on Web Information Systems Engineering Workshop(WiSE 03)

[35] Yannis Makripoulias et al, 2005, "Towards Ubiquitous Computing with Quality of Web Service Support", In UPGRADE, The European Journal for the Informatics Professional", Vol 6. No. 5

[36] Vuong Xuan Tran, Hidekazu Tsuji, Ryosuke Masuda, 2009, "A new QoS ontology and its QoS-based ranking algorithm for Web services", In Elsevier Journal: Simulation Modelling Practice and Thoery.

[37] S. Frolund and J. Koisten. 1998, "QML: A Language for Quality of Service Specification", Hewlett-Packard, http://www.hpl.hp.com/techreports/98/HPL-98-10.html

[38] A. Dan et al. 2002,"Web Service Level Agreement (WSLA) Specification", http://www.research.ibm.com/wsla.

[39] A. Sahai, A. Durante, and V. Machiraju.,2002,"Towards Automated SLA Management for Web Services", HP, www.hpl.hp.com/techreports/2001/ HPL-2001-310R1.pdf.

[40] K. Kritikos and D. Plexousakis. 2006, "Semantic QoS Metric Matching", In Proc. of the European Conference on Web Services (ECWS2006), IEEE Computer Society.

[41] I. V. Papaioannou, D. T. Tsesmetzis, I. G. Roussaki, and M. E. Anagnostou, 2006,"A QoS Ontology Language for Web Services". In Proc. of the 20th International Conference on Advanced Information Networking and Applications (AINA2006), IEEE Computer Society.

[42] http://www.jfsowa.com/ontology/ (accessed on 12th Aug 2010, 1.28 am)

[43] C. Zhou, L. Chia, and B. Lee, 2005, "Web Services Discovery on DAML-QoS Ontology". In the International Journal of Web Services Research(IJWSR)

[44] G. Dobson, R. Lock, and I. Sommerville,2005, "QoSOnt: a QoS Ontology for Service-Centric Systems". In Proc. of the 2005 Euromicro SEAA.

[45] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, 2006, "A QoS-Aware Selection Model for Semantic Web Services".In Proc. of the 4th International Conference Service Oriented Computing, LNCS, Springer Verlag, Volume 4294.

[46] D. Roman, H. Lausen, and U. Keller, 2004,"Web Service Modeling Ontology (WSMO)", http://www.wsmo.org/ 2004/d2/v1.0/20040920/