ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 1 Issue 1 Oct-Dec 2013
International Journal of Advanced Research in Computer Science & Technology (IJARCST)

Semantic Web Service Composition with Quality of Service

[I]Adepu Sridhar, [II]S Shiva Prasad, [III]Mahesh Ubale
[I, II, III]Vignan University, Guntur, India
[I]sridharuce@gmail.com

Abstract
The concept of Web Services provides some kind of middleware for the implementation of distributed applications, independent from platforms and programming languages. When developing new software systems, it could be of use in terms of time and costs to reuse the functionality of existing services. This process of reuse is called Web Service composition, but there are still some open topics currently under research. Beneath the matching between semantic description of offered and requested services, also the capabilities of the composed service are of importance. The current Web Service standards are not equipped to consider non-functional requirements, i.e. QoS aspects, of a user to a composed service. An approach is presented how to enhance the Web Services architecture to consider QoS aspects in the composition of a complex service

Keywords
Web Services, QoS, Composition, Semantic, Platforms, Languages, Re-use, Non-functional, Architecture

I. Introduction

The Web Services concept provides a method which enables Web Services to communicate with other services regardless of platforms and programming languages, using such standards like XML, WSDL, SOAP, and UDDI. By enabling applications to share data across different platforms and operating systems, it becomes easy for a business to connect to partners and to extend existing Web Services for new customers. Each Web Service is independent of all others.

However, Web Services can be combined to a collaborating group that performs a particular task. This is called Web Service composition. Composition enables application programmers to create a new Web Service using existing services.

A. Web Service Standards

Web Services are registered and announced using the following services and protocols. Many of these and other standards are being worked out by the UDDI project, a group of industry leaders that is spearheading the early creation and design efforts.

1. Universal Description, Discovery, and Integration (UDDI)

It is a protocol for describing available Web Services components. This standard allows businesses to register with an Internet directory that will help them advertise their services, so companies can find one another and conduct transactions over the Web. This registration and lookup task is done using XML and HTTP(S)-based mechanisms.

2. Simple Object Access Protocol (SOAP)

It is a protocol for initiating conversations with a UDDI Service. SOAP makes object access simple by allowing applications to invoke object methods or functions, residing on remote servers. A SOAP application creates a request block in XML, supplying the data needed by the remote method as well as the location of the remote object itself.

3. Web Service Description Language (WSDL)

The proposed standard for how a Web Service is described is an XML-based service IDL (Interface Definition Language) that defines the service interface and its implementation characteristics. WSDL is referenced by UDDI entries and describes the SOAP messages that define a particular Web Service.

(ii). Web Service Architecture

The Web Service architecture places into relationship among various components and technologies. It comprises a Web Services "stack" or completely functional implementation. The basic architecture includes Web Services technologies capable of: Exchanging messages. Describing Web Services. Publishing and discovering Web Services descriptions.

The basic Web Services architecture defines an interaction between software agents as an exchange of messages among services requesters and service providers. Requesters are software agents that request the execution of service. Providers are software agents that provide a service. Agent can be both service requesters and providers. Providers are responsible for the publishing a description of the service(s) they provide. Requesters must be able to find the description(s) of the service. The Web Services architecture is based on the interactions between three roles: service provider, service registry and service requester. The interactions involve publish, find and bind operations. Together, these roles and operations act upon the Web Services artifacts: the Web Service software module and its description. In a typical scenario, a service provider hosts a network - accessible software module an implementation of a Web Service. The service provider defines a service description for the Web Service and publishes it to a service requester or service registry. The service requester uses a find operation to retrieve the service description locally or from the service registry and uses the service description to bind with the service provider and invoke or interact with the Web Service implementation. Service provider and service requester roles are logical constructs and a service can exhibit characteristics of both, is shown in Figure 1.

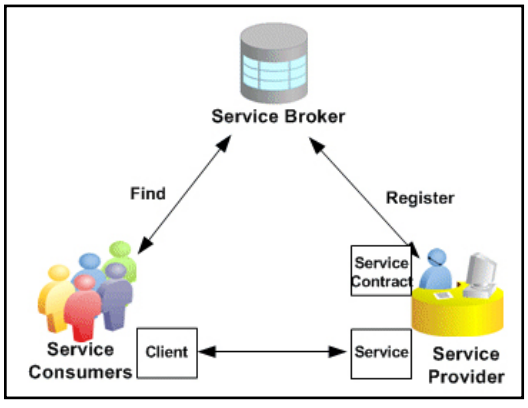

Figure 1: Web Service Architecture

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Service Provider from a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that hosts access to the service.
Service Requester from a business perspective, this is the business that requires certain functions to be satisfied. From an architectural perspective, this is the application that is looking for and invoking or initiating an interaction with a service. The service requester role can be played by a browser driven by a person or a program without a user interface, for example another Web Service.
Service Registry this is a searchable registry of service descriptions where service providers publish their service descriptions. Service requesters find services and obtain binding information (in the service descriptions) for services during development for static binding or during execution for dynamic binding. For statically bound service requesters, the service registry is an optional role in the architecture, because a service provider can send the description directly to service requesters. Likewise, service requesters can obtain a service description from other sources besides a service registry, such as a local file, FTP site, Web site, Advertisement and Discovery of Services (ADS) or Discovery of Web Services.

C. Operations in Web Service Architecture

For an application to take advantage of Web Services, three behaviours must take place: publication of service descriptions, lookup or finding of service descriptions, and binding or invoking of services based on the service description. These behaviours can occur singly or iteratively. In detail, these operations are:
Publish to be accessible; a service description needs to be published so that the service requester can find it. Where it is published can vary depending upon the requirements of the application.
Find in the find operation, the service requester retrieves a service description directly or queries the service registry for the type of service required. The find operation can be involved in the two different lifecycle phases for the service requester: at design time to retrieve the services interface description for program development, and at runtime to retrieve the services binding and location description for invocation.
Bind eventually, a service needs to be invoked. In the bind Operation the service requester invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact and invoke the service.
Service where a Web Service is an interface described by a Service description, its implementation is the service. A service is a software module deployed on network accessible platforms provided by the service provider. It exists to be invoked by or to interact with a service requester. It can also function as a requester, using other Web Services in its implementation.
Service Description the service description contains the details of the interface and implementation of the service. This includes its data types, operations, binding information and network location. It could also include categorization and other metadata to facilitate discovery and utilization by service requesters. The service description might be published to a service requester or to a service registry.

D. Quality of Service for Web Service

With standards like SOAP, UDDI, and WSDL being adopted by all major Web Service players, a whole range of Web Services covering the financial services, high-tech, and media and entertainment are being currently developed. As most of the Web Services are going to need to establish and adhere to standards, QoS will become an important selling and differentiating point of these services. QoS covers a whole range of techniques that match the needs of service requestors with those of the service provider's based on the network resources available. By QoS, we refer to nonfunctional properties of Web Services such as performance, reliability, availability, and security.

1. Web Service QoS Requirements

The major requirements for supporting QoS in Web Services are as follows:

(i). Availability

Availability is the quality aspect of whether the Web Service is present or ready for immediate use. Availability represents the probability that a service is available. Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time. Also associated with availability is time to repair (TTR). TTR represents the time it takes to repair a service that has failed. Ideally smaller values of TTR are desirable.

(ii). Accessibility

Accessibility is the quality aspect of a service that represents the degree it is capable of serving a Web Service request. It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time. There could be situations when a Web Service is available but not accessible. High accessibility of Web Services can be achieved by building highly scalable systems. Scalability refers to the ability to consistently serve the requests despite variations in the volume of requests.

(iii). Integrity

Integrity is the quality aspect of how the Web Service maintains the correctness of the interaction in respect to the source. Proper execution of Web Service transactions will provide the correctness of interaction. A transaction refers to a sequence of activities to be treated as a single unit of work. All the activities have to be completed to make the transaction successful. When a transaction does not complete, all the changes made are rolled back.

(iv). Performance

Performance is the quality aspect of Web Service, which is measured in terms of throughput and latency. Higher throughput and lower latency values represent good performance of a Web Service. Throughput represents the number of Web Service requests served at a given time period. Latency is the round-trip time between sending a request and receiving the response.

(v) Reliability

Reliability is the quality aspect of a Web Service that represents the degree of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web Service. In another sense, reliability refers to the assured and ordered delivery for messages being sent and received by service requestors and service providers.

(vi). Regulatory

Regulatory is the quality aspect of the Web Service in conformance with the rules, the law, compliance with standards, and the established service level agreement. Web Services use a lot of

standards such as SOAP, UDDI, and WSDL. Strict adherence to correct versions of standards (for example, SOAP version 1.2) by service providers is necessary for proper invocation of Web Services by service requestors.

(vii). Security

Security is the quality aspect of the Web Service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control. Security has added importance because Web Service invocation occurs over the public Internet. The service provider can have different approaches and levels of providing security depending on the service requestor.

E. Ontology

Ontology defines a common vocabulary to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. Some of the reasons to develop Ontology are:

To share common understanding of the structure of information among people or software agents, To enable reuse of domain knowledge, To analyze domain knowledge.

Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies. For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications. Enabling reuse of domain knowledge was one of the driving forces behind recent surge in ontology research. For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such an ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain. Analyzing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them.

II. Literature Review

This section discusses work related to Web Services discovery, and Web Services discovery with QoS. And a brief introduction to Web Service discovery. some previous work in the area of Semantic Web and ontology. Describes QoS of Web Services and related research on Web Services discovery with QoS, finally examines the issue of Web Services discovery with QoS.

A. Web Services Discovery

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 Web Service Provider (WSP), the Web Service Discovery Agency/Middle-ware interacting with the Service Registry and the Web Service Consumer (WSC) forming a Web Services Triad [8]. 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 Web Service implementation, is invoked and rendered to the Consumer. A point to note here is that the Web Service Provider and Web Service Consumer roles are interchangeable, meaning a Consumer could be Provider for a different Service.

B. Problems in Finding the Right Services

1. UDDI Problem

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, UDDI allows search on limited attributes of a service, namely, Service Name (selected by the Service Provider), key Reference (unique for a service), so This problem makes it difficult to store within the UDDI, the runtime 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.

2. WSDL Problem

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 nonfunctional aspects Because WSDL is not designed to take the “semantic descriptions” of the service. This makes it difficult to store the nonfunctional aspects of the service such as its Quality of Service (QoS). Parameters such as reliability, availability, response time, throughput, mean time before failure, price, etc. The crucial point to note here is the criteria to search and locate the Web Services. One stream of thought focuses on finding the Web Services based on its functional requirements. 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. However, the Web Service Functional Specifications are not enough to handle the Service Discovery Process. This is because: Web Service 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 Web Service.

With the abundance of Web Services created by many service providers, often a number of Web Services satisfies the functional

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

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. This led to the second stream of thought. Discover Web Services based its non-functional requirements. The predominant factor being "Quality of Service" (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.

C. QoS and Web Services Discovery

Quality of Service, is "a combination of several qualities or properties of a service". It is a set of non-functional attributes that may influence the quality of the service provided by a Web Service. The QoS requirements for Web Services are more important for both service providers and consumers since the number of Web Services providing similar functionalities is increasing. Current Web Service technologies such as WSDL and UDDI, which are for publishing and discovering Web Services, consider only customer functionality requirements and support design time, or static service discovery. Non-functional requirements, such as QoS, are not supported by current UDDI registries. 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.

D. The Semantic Web and Ontology

The current World Wide Web represents information using natural languages. The information is intended for human readers but not machines. The Semantic Web is "an extension of the current Web in which information is given well-defined meaning, enabling computers and people to work in better cooperation". It is a mesh of information that can be automatically processed and understood by machines.

The Resource Description Framework (RDF) is a basic semantic mark-up language for representing information about resources on the Web. It is used in situations where the information needs to be processed by applications rather than humans. RDF Schema (RDF-S) is a language for describing RDF vocabulary. It is used to describe the properties and classes of RDF resources. The Web Ontology Language (OWL) is used to "publish and share sets of terms called ontologies, supporting advanced Web search, software agents and knowledge management". It provides more vocabulary for describing properties and classes of RDF resources than RDF-S. OWL-S, the Web Ontology Language for Services , is an OWL based Web Service ontology. It provides a language to describe the properties and capabilities of Web Services. OWL-S can be used to automate Web Service discovery, execution, composition and interoperation.

WSDL is a XML document used to describe Web Services. It describes the location of a Web Service and the operations the service provides. It defines a protocol and encoding independent way to describe interactions with Web Services. One shortcoming of the current technologies of Web Services, such as WSDL, SOAP and UDDI, is that they describe only the syntax but not semantics of services. By taking advantage of the strengths of both OWL-S and WSDL, Web Service providers can describe their services in an unambiguous form that can be understood by computers.

Ontology is defined as "a specification of a conceptualization" . It uses a formal language, such as OWL, to describe the concepts and relationships in a domain. The main reason to use ontologies in computing is that they facilitate interoperability and machine reasoning. A common ontology for Web Services is the DAML-S ontology, which aims to facilitate automatic Web Service discovery, invocation and composition. It describes properties an capabilities of Web Services but does not provide details about how to represent QoS descriptions.

E. Outcome of Literature Survey

In the early years of Service Oriented Computing, the number of Web Services was few. Finding the relevant services was primarily done by checking for the published services within the UDDI. As the number of available Web Services increase finding appropriate Web Services to fulfill a given request becomes an important task. Most of the current solutions and approaches in Web Service discovery are limited in the sense that they are strictly defined, and they do not use the full power of semantic and ontological representation. Service matchmaking, which deals with similarity between service definitions, is highly important for an effective discovery. Today, WSDLs, are abundant, scattered across the WWW. The count of Web Services already deployed with similar functionality is large 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. So the functional requirements are not sufficient to discover the right services. The predominant nonfunctional WSD approach adopted is to model the Quality of Service (QoS).

F. Problem Statement

Implementing the Semantic Web Service Composition with non-functional properties(Quality of Service).

G. Objectives

- To setting up the environment to Web Service composition.
- To compose simple Web Service based on functional properties.
- To develop the ontology for semantic discovery of services.
- To validate the QoS information.
- To composing Web Service with QoS parameters.
- To test and improve the Composed Services

III. Proposed Work

A. Semantic Web Services Composition Based on QoS

The proposed framework consists of following components as shown in Figure 2

1. Service Registration

It is the process of specification of Web Services to the system. New services are registered in registry through service registration process. There are several registries that different companies (service providers) maintain and all of them are synchronized after regular interval.

2. Service Request

Clients that need a particular service send request through service

request module.

3. Query Information Processor

The purpose of Query Information Processor is to translate request/ response from one form to another. We use translator so that all of the services are registered from external form to a form used by system and vice versa.

4. Composer

The composer composes the selected component services in order to make a single desired Web Service.

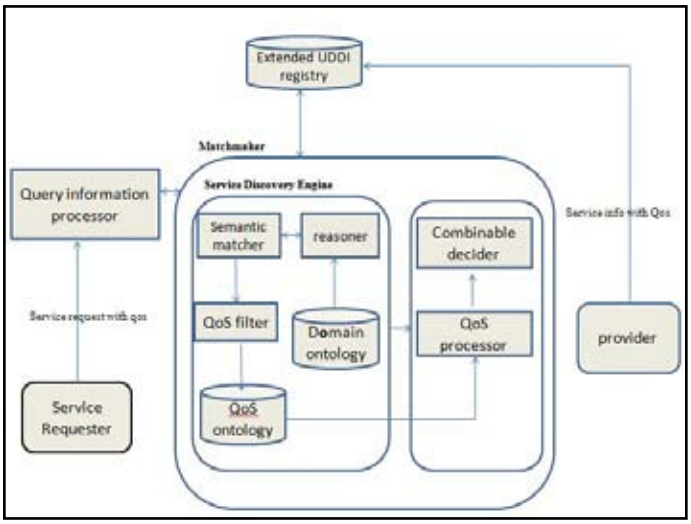

Fig. 2: Semantic Web Services Composition Framework Based on QoS

5. Matchmaker

The purpose of matchmaker is to match the users request from the Web Services database. If match is found, it returns results back to web server. If not then select the Web Services from web, store/update them in database and then return results back to requested composer.

6. Service Registry

The service registries are used to register the Web Services by Web Service providers. Also they are used to request the users desired Web Services. Each registry has the references of Web Services that are actually hosted on service repositories First of all, extend the UDDI to support publishing and querying Web Service with QoS. Web Service providers publish the Web Service with QoS to UDDI registry.

Web Service requestor sends service request with Qos to UDDI registry, before that we validate and format the request information by query information processor. When receiving a request, the service discovery engine selects the advertisements from the UDDI that are relevant with the current request. Then it uses the Reasoner to compute the level that matches. Thus, the reasoner uses the domain Ontology Database as data to use and compute the matching process. So we can get some candidate services that satisfy the requesters needs. According to the QoS constraints in every service description, the QoS filter will extract the QoS data, and select the best suitable services from the candidate services which are used to the composition of services. Finally, we compose the services using the combinable decider. The UDDI returns the composite services to the Web Services requestor. The full process can be described as in Figure 2.

B. Environment setup

To install and configure Microsoft UDDI services the fallowing tasks have to do

- Install Windows Server 2008

Install Internet Information Services
Install SQL Server
Install UDDI Services Components
Web service creation.
3.2.1 Register services by using the UDDI Interface
Before publishing a Web Service, first create its provider.

- Publish a Service Provider, Add Contact Information, Publish a Service, Publish a Binding, Publish a t-Model

C. Extension of UDDI

Search for Businesses in UDDI using any combination of Keyword, Identifier, Locator, Service Type, and Discovery URL. Only businesses that match ALL of the criteria specified are returned. At least one of the search criteria should be mentioned.

1. Search by Business Name

Specify the name of the business you are looking for in the "BusinessName" tag. The businesses with names that start with the characters you entered will be returned.

2. Search for Service

Search for Service using combination of ServiceName and Category. Only services that match ALL of the criteria specified are returned. At least one of the search criteria should be given. Since business services depend on business entities, it is virtually impossible to search business services without specifying business names. If searching for business services without business names is necessary, there is a need to retrieve ALL business entities registered with a public UDDI or private UDDI, a task taking too much time. Based on the above consideration, users must also specify the business names for business service search.

3. Searching by Service

Name Specify the name of the service you are looking for in the "ServiceName" tag. The business services with names that start with the characters you entered will be returned.

4. Searching by Service Type(tModel)

Specify the name of the service tmodel you are looking for. The services with tmodel names that start with the characters you entered will be returned.

The growing number of Web Services available within an organization and on the Web raises a new and challenging search problem: locating de-sired Web Services. As one considers search for Web Services in more detail, it becomes apparent that the keyword search paradigm is insufficient for two reasons. First, keywords do not capture the underlying semantics of Web Services. Current Web Service search engines return a particular service if its functionality description contains the keywords in the query; such search may miss results. For example, when searching zip- code, the Web Services whose descriptions contain term zip or postal code but not zipcode will not be returned. Second, keywords do not suffice for accurately specifying users' information needs. Since a Web Service operation is going to be used as part of an application, users would like to specify their search criteria more precisely than by keywords. Current web- service search engines often enable a user to explore the details of a particular web-service

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

operation, and in some cases to try it out by entering an input value. Nevertheless, investigating a single Web Service operation often requires several browsing steps. Once users drill down all the way and find the operation inappropriate for some reason, they want to be able to find similar operations to the ones just considered, as opposed to laboriously following parallel browsing patterns. Similarly, users may want to find operations that take similar inputs (respectively, outputs), or that can compose with the current operation being browsed. Current UDDI does not support for Quality of Service. There are several alternatives methods to store such information in UDDI registries. Already customers are demanding that WSM vendors store this information in UDDI.
There is an opportunity to allow customers to see all quality of service information in standard locations in UDDI with a common vocabulary for referring to it. This will allow developers and users to make better decisions about what Web Services to use given the QoS information. It also magnifies the value of WSM vendors‟ solutions, as a wider set of people beyond their direct Web Services administrator

D. Methods to Incorporate QoS Into UDDI

There are three basic methods of reflecting this additional information in UDDI:

1. TModel for QoS Information Pointing to External Resource

Define a tModel called QoS Information that references an external quality of service Web Service. A resource at the tModel‟ s overviewURL (such as an a generated XML file) would contain all of the performance and reliability data. Each UDDI binding Template contain a tModel reference to the QoS Information tModel, added to its tModelInstanceDetails collection.

2. Multiple Categories for QoS Attributes

Create multiple additional tModels to represent this QoS information, one for each type of QoS information. Add these categories to UDDI bindingTemplates. In effect, each categoryBag will have multiple keyedReferences, each of which represents a different QoS metric. The values of the metrics would be stored right in the category values. This method has the potential disadvantage of creating unmanageably large categoryBags (i.e. too many categories) on each bindingTemplate. It also requires UDDI V3 to allow usage of categories on bindingTemplates.

3. Extend the UDDI Data Structures

Extend the UDDI information explicitly to have the QoS information using UDDI v3 extensibility mechanism to extend the UDDI data structures (specifically businessService and bindingTemplate) via XML schema derivation. This method has the disadvantage of requiring UDDI v3 support registries. In the future however this option warrants further evaluation, possibly to supplement the approach we recommend. While publishing Web Services in UDDI provider will provide basic service profile which includes input and outputs of Web Service with QoS information this profile location will be given in tmodel overview document Service requester can query for the services using inputs and outputs with QoS information.

E. Domain Ontology Development

Main purpose of Ontology is describes the elements(terms and Vocabulary), their relationship and the rules opf the given domain and to describe the knowledge in generic way. The Ontology(Domain Ontology of Stock Market) will provide the semantic description for the Web Service capabilities this way Web Services are semantically described and discovered composed and invoked by other services automatically thus transforming Web Services into Semantic Web Services.

1. Concepts in the Stock Market Ontology

Stock

Stock represents one fraction of one specific company ownership Instance attributes of Stock are:

- Company: Company is Society owned by several individuals
- Continuous: Reflects whether or not stock can be buy/sell in continuous market
- Face Value: Face Value is the amount expressed in currency representing ownership of the company
- Price Value: Price Value is the price of the last transaction for the stock in an specific market
- Stock Type: Stock type specifies the type of rights in the company
- Best Sell Price: The best sell price that can be found in the market at this point of time
- Maximum: The highest price in specific period of time
- Minimum: The lowest price in specific period of time
- Dividend: The amount of last profit distribution that correspond to single stock
- Relationships of Stock are hasStockMarket, hasLastPrice.

StockMarket

- The Market where the buying or selling of shares takes place
- Instance attributes of Stock Market are
- Name: The name of the market/index(example NASDAQ, Dow Jones)
- Currency: The currency in which stock prices are expressed
- Country: The country where the market is established
- Relationships of Stock Market are hasStocks, hasBrockers, has Sessions hasIndexes, hasCompany. Buy Sell Order
- A client order to buy or sell an amount of shares in the market Instance attributes of Buy Sell Order are
- Buy Sell Date: The date and time in which operation has been performed
- Relationships of Buy Sell Order are hasDepository, hasPortfolio, hasSession, hasStockMarket, hasUserId.

Conditioned Buy Sell Order

- Instance attributes of conditioned buy sell order are
- Expiration Date: the date in which order is cancelled if not performed before
- Price: The price of stock in an specific market.

Last Price

- The value of a last transaction in a session of singe share Instance attributes of Last Price are
- Date: The date when the transaction performed
- Value: Historical values.

Session
- Time unit from opening to closing of the market usually one working day Instance attributes of Session are
- Opening Time: The time at what market is open
- Closing time: the time at what market is closed
- Date: the date and time of historical values/or date of the session
- Index Value: Weighted average of stock process in the session
- Relationships of session are hasVolume, hasStocks.

Index
- The weighted average of stock prices Instance attributes of the Index are
- Name: The name of the Index
- IndexId: The id of the index for index identification
- Initial Value: The Value of index when a session starts
- Actual Value: The latest Value of the index
- Relationships of the index are hasStockWeight, hasMarket

Broker
- Company or person authorized to operate in the stock market
- Instance attributes of Broker are
- CIF: Customer identification number
- Non official taxes: the commission the specific broker applies
- Relationships of broker are hasStockMarket Stock Volume is measurement of the number of stocks that have been exchanged or traded within specific period of time Instance attributes of Stock Volume are Amount Relationships of stock Volume are hasVolumeStock, hasVolumeSession.

F. Protege-ontology Development Tool
With the continuous development of the Web ontology language, there are a lot of ontology development tool. Protege is an integrated software tool used by system developers and domain experts to Develop knowledge base system. It is open, easy to expand, and supports the standards well, including OWL. In addition, its interface is very simple and friendly. Figure 3 is the screen shot of the Stock Market ontology by using Protege.

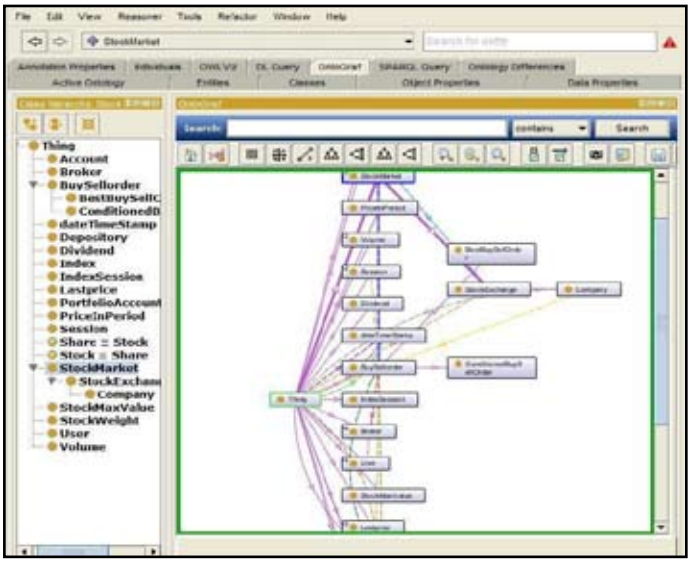

Fig. 3: Snapshot of StockMarket Ontology Developed in Protégé

Figure 3 Shows Stock Market Ontology Developed in protégé tool left panel shows the class hierarchy of Stock Market Ontology, right panel shows the ontology graph which is the relationship among the concepts in the Stock Market Ontology.

G. Calculating the QoS Values by Using WAPT Tool
WAPT stands for Web Application Performance Tool. It performs test by emulating activity of many virtual users. Each virtual user can have its own profile settings. It is possible to have thousands of virtual users acting simultaneously on your web application performing any activity like reading or writing with web server. Once set number of virtual users to act on web application then runs tests for specified time or specified user sessions.
Here are the important parameters to be monitored on the test report:

1. Error Rate
Failure rate against total number of tests run. The error may be due to the high load on server or due to the network problems and timeouts.

2. Response Time
A great parameter to check when you run tests for website performance. This response time indicates time required by server to provide correct reply to the request.

3. Number of Pages Per Second
Number of page requests successfully completed by server per second. After running the test WAPT generate test report for the test and saves as HTML pages The below figure shows the performance test results for Stock Quotes Web Service.
Figure 4 Shows the Performance test results of Stock Quote Web service by using the WAPT tool. Response time with in time intervals of one minute test Duration shown in the Response time table, and number of kilo bits sent to/received from Web service in time intervals of one minute test duration in kilo bits sent, kilo bits received tables respectively.

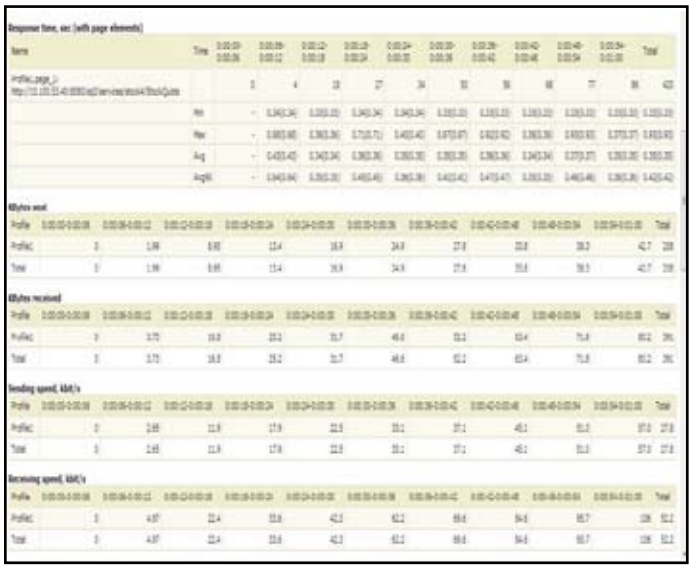

Fig. 4: Snapshot of Performance Test Results of Stock Quote Web Service

The Figure 5 shows the response time graph for Stock Quotes Web Service, X-axis test duration time in seconds,Y-axis Response Time in seconds.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

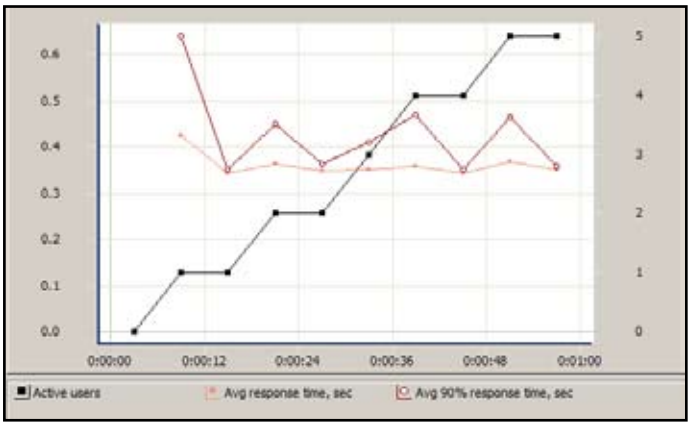

Figure 5: Response Time Graph for Stock Quotes Web Service

Figure 5 Shows the Average response time graph for Stock Quote Web service of test duration one minute where X-axis shows the test duration time in seconds and Y- axis shows the Response Time in seconds. The figure 6 shows the Bandwidth graph for Stock Quotes Web Service, X-axis test duration time in seconds, Y-axis Bandwidth in kilo bits/second.

Figure 6 Shows the Bandwidth Graph for Stock Quotes Web Service for the test duration of one minute where X-axis shows the test duration time in seconds and Y- axis shows Bandwidth in kilo bits/second.

H. Composition

The first step before the composition is to filter all those services from the repository that are relevant to the request, discarding the rest.

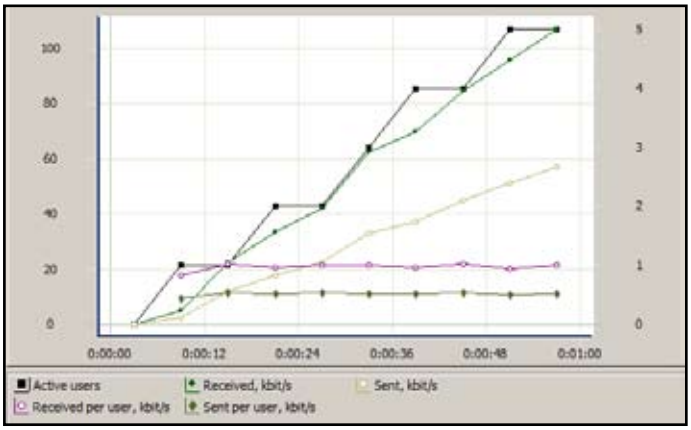

Figure 6: Bandwidth Graph for Stock Quotes Web Service

A Web Service defined by a 3-Tuple S = $\{I_S, O_S, Q_S\}$ where $I_S = \{I^1_S, I^2_S, ...\}$ is the set of inputs consumed by the service, $O_S = \{O^1_S, O^2_S, ...\}$ is the set of outputs retrieved when the service is invoked and $Q_S = \{Q^1_S, Q^2_S, ..., Q^n\}$ is the set of quality attributes service from selected candidate service this output as input if any add that to queue repeate this process till query output matches. Get the WSDL files of the services. After getting WSDL file parse the WSDL file and get the methods and input and output parameters, call the Web Services in the queue in their order of insertion Then sends the valid request in proper format the requested Web Service will gives response in soap message requester receives this soap message. For statically bound service requesters, the service registry is an optional role, because a service provider can send the description directly to service requesters. Likewise, service requesters can obtain a service description from other sources besides a service registry.

IV. Results and Discussion

The composed Web Service Stock Quotes which will give Stock quotes of specified company in specified currency. This Service is composed from three Web Services

- First Web Service which will gives the Stock Symbol of NASDAQ Stock Exchange on taking Company name as input.
- Second Web Service which takes the Stock Symbol as input and gives Stock Quotes in US dollars.
- Third one is Currency Converter which will gives the currency exchange rates on taking country currency Symbols

First search Domain Ontology based on given inputs and find the related Web Services from UDDI get the inputs and outputs and QoS information filter the services based on QoS parameters form these filtered services compose the complex service based on composition algorithm, get the WSDL location and get the service description then send the request to Web Services and get response.of the service. In match making process Based on user inputs and outputs Domain Ontology will be searched if any concepts in domain ontology match for inputs and outputs and get the all related keywords from domain ontology. After getting related keywords search the UDDI using these keyword and get relavent services.

Inputs and outputs of a service are semantically annotated by concepts that are defined in an ontology. Although concepts from an ontology can be related to each other by different types of relations such as the subclass/superclass relationship, so consider that an output of a service O_S^1 matches the input of other service I_S^2 when O_S^1 is equal or a subclass of I_S^2 ($O_S^1 \subseteq I_S^2$). For all these services get the inputs outputs and QoS then if the Advertisement inputs and outputs match query inputs and output then those are exact matches if input or output only matches then partial match then filter the candidate services using QoS requirements. Filter the services based on QoS values which meets the user QoS requirements Group the candidate services if any similar services in functional aspects based on input and output values. Sort the grouped services with same functional properties based on QoS values.

Search the candidate services weather their inputs matching with query inputs if matching done then add this service to queue and check for the weather output is matching with query output then return the service. If output is not matched then check for any The Figure 7 shows the Request for Stock Quotes of Microsoft Company.

In the request query providing the input (Company Name) Microsoft and output Stock quotedata and QoS Parameter Response Time less than 1.5 seconds and Currency in Indian Rupees (INR).

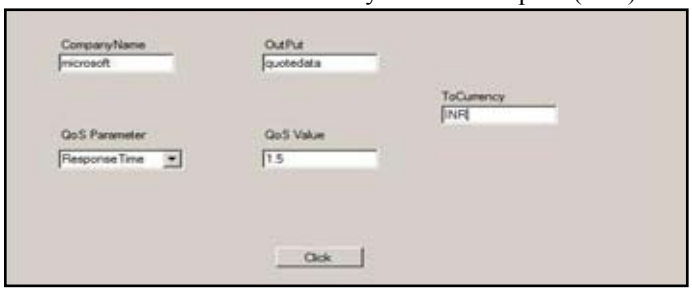

Fig. 7: Snapshot Request for Stock Quotes of Microsoft Company

The Figure 8 shows Stock Quotes of company Microsoft in Indian Currency In the response of composed service

- Company Name is the name of the company Microsoft
- Company Symbol in NASDAQ Stock Market is MSFT Last Trade Amount is the price at which the stock last changed hands from seller to buyer.
- Date and Time is the date and time at which the last share was traded for the company.
- Stock Change is the difference between the last closing price and the current price (Last Price).
- Open Amount is the price at which the first share was traded for the current trading day.
- Day High, Day Low is the highest, lowest stock value in the current day respectively.
- Stock Volume is how many shares have changed hands during the current trading day.
- Previous Close is the stock price of the last transaction of the previous day trading.
- Earn Per Share The portion of a company's profit allocated to each outstanding share of common stock.
- PE ratio is ratio of a company's current share price compared to its per-share earnings.

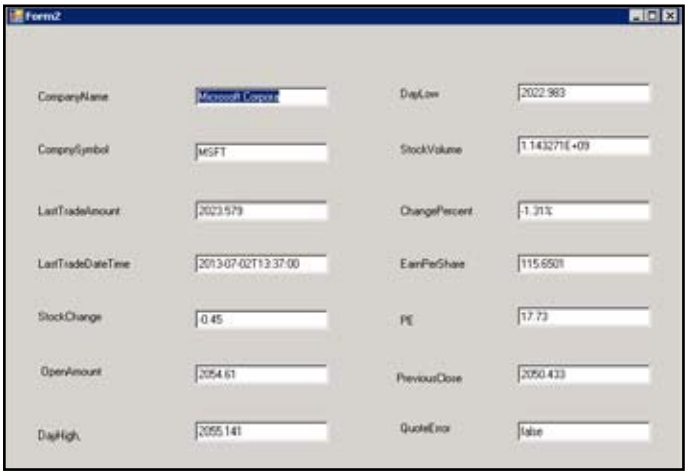

Fig. 8 Snapshot of Response of Stock Quote service for Microsoft in Indian Currency

V. Conclusion and Future Work

With an increasing number of Web Services providing similar functionalities, more emphasis is being placed on how to find the service that best fits the consumer‟ s requirements.

In order to find services that best meet their QoS requirements, the service consumers and/or discovery agents need to know both the QoS information for the services and the reliability of this information. The problem, however is that the current UDDI registries do not provide a method for service providers to publish the QoS information of their services, and the advertised QoS information of Web Services is not always trustworthy. Advertised QoS information is expressed in XML style format and is stored using tModels in a UDDI registry. Services that meet a customer‟ s functionality and QoS requirements are selected. So the by using proposed method with QoS can discover and compose the best services for customer needs. Quality of Service specification for each Web Service provider might be different.

Representing the same concept in different vocabulary and QoS metrics may differ for all Web Service providers. By using the QoS Ontology mapping can be done in effective way.

References

[1] *Ter Beek, Maurice, Antonio Bucchiarone, and Stefania Gnesi. "Web service composition approaches: From industrial standards to formal methods." Internet and Web Applications and Services, 2007. ICIW'07. Second International Conference on. IEEE, 2007.*

[2] *Zhang, Jidong, and Weipeng Huo. "Research on the composition mechanism of Semantic web services." Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. Vol. 3. IEEE, 2010.*

[3] *Nair, M., and V. Gopalakrishna. "Look before you leap: a survey of web service discovery." Int J Comput Appl 7.5 (2010): 5-11.*

[4] *Khan, Farhan Hassan, et al. "QoS Based Dynamic Web Services Composition & Execution." arXiv preprint arXiv:1003.1502 (2010).*

[5] *Ying-Hui, Zhang, Fu Feng-Rui, and Yu Chao. "Semantic Web Service selection based on QoS Ontology." Information Networking and Automation (ICINA), 2010 International Conference on. Vol. 2. IEEE, 2010.*

[6] *Rodriguez-Mier, Pablo, Manuel Mucientes, and Manuel Lama. "A dynamic qos-aware semantic web service composition algorithm." Service-Oriented Computing. Springer Berlin Heidelberg, 2012. 623-630.*

[7] *Zhou, Chen, Liang-Tien Chia, and Bu-Sung Lee. "Semantics in service discovery and QoS measurement." IT professional 7.2 (2005): 29-34.*

[8] *Pathak, Jyotishman, et al. "A framework for semantic web services discovery." Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM, 2005.*

[9] *Lee, Kangchan, et al. "Qos for web services: Requirements and possible approaches." W3C working group note 25 (2003): 1-9*

[10] *Chen Zhou, Liang-Tien Chia, Bu-Sung Lee, "DAMLQoS Ontology for Web Services." International Conference on Web Services (ICWS), Washington. DC: IEEE computer society press, pp. 42-49.*

[11] *Yang, Huirong, Xiaobo Chen, and Shanshan Liu. "Research and Implementation on QoS Ontology of Web Service-Oriented Composition." Information Engineering and Electronic Commerce (IEEC), 2010 2nd International Symposium on. IEEE, 2010.*