

QoS AWARE FORMALIZED MODEL FOR SEMANTIC WEB SERVICE SELECTION

Divya Sachan¹, Saurabh Kumar Dixit¹, and Sandeep Kumar¹

Department of Electronics and Computer Engineering, Indian Institute of Technology
Roorkee, Roorkee, India

Abstract

Selecting the most relevant Web Service according to a client requirement is an onerous task, as innumerable number of functionally same Web Services(WS) are listed in UDDI registry. WS are functionally same but their Quality and performance varies as per service providers. A web Service Selection Process involves two major points: Recommending the pertinent Web Service and avoiding unjustifiable web service. The deficiency in keyword based searching is that it doesn't handle the client request accurately as keyword may have ambiguous meaning on different scenarios. UDDI and search engines all are based on keyword search, which are lagging behind on pertinent Web service selection. So the search mechanism must be incorporated with the Semantic behavior of Web Services. In order to strengthen this approach, the proposed model is incorporated with Quality of Services (QoS) based Ranking of semantic web services.

This paper focuses on various concepts of Quality of Service associated with web services. Various QoS parameters like performance, availability, reliability and stability etc. are formalized in order to enhance the pertinence of web service selection. A QoS mediator agent based Web Service Selection Model is proposed where QoS Consultant acts as a Mediator Agent between clients and service providers. Model suggests user's preferences on QoS parameter selection. The proposed model helps to select pertinent Web Service as per user's requirement and reduce the human effort.. Further process of adding ontology with semantic web services is also illustrated here.

Keywords: *QoS, SOAP, Web Service Selection (WSS), Ontology.*

1 Introduction

Web Services [1] assists in providing solutions for distributed business processes and applications which are accessible via the Internet. In case a single WS doesn't meet the complex requirements, several web services combine together to provide a composite solution. In such cases selecting several Web Services for Web Service Composition becomes a major step in the overall process.

WS are nothing more than software ingredients that interact with one another by sending XML messages wrapped in SOAP envelops. WS communication is built on SOAP. SOAP is XML based information packaging definition. It provides a structured way for information exchange between peers in a distributed environment.

Web Services are defined as “self-contained, self-describing, modular application that can be published, located and invoked across the Web” [1]. These web services are described by using standards like WSDL and then service descriptions are published in some UDDI [2] registries. Whenever a service request is invoked, a search is performed between service request description and available web service description which can satisfy the functional requirement of request.

As we know Service Oriented Architecture (SOA) is not only the service’s architecture as per technology basis but it also renders the policies, practices and frameworks to assure that pertinent services are provided and consumed by users. Goals of SOA are, firstly service provider offers several services and secondly prospective users of the services dynamically choose the best service from the set of services offered. In reference to current Web when we put a query in a search engine, we find a long list of WS as per the similar keywords. Now we have to ad-hoc decision to choose a WS. Now it is just a matter of chance that we select a relevant WS to perform our work on Web. So we can say that above mentioned goals of SOA, are partially executed as WS are described and listed in public registries but there is no means to choose the best from the set of services offering the same functionality. A consumer is thus forced to make an ad -hoc decision of choosing a service from multiple services offered for the same functionality.

In such scenario Quality of Services (QoS) assist in ranking the WS and selecting the best WS from a list of candidate web service having similar functionality based on their QoS descriptions, in response to a service request made by the user. The QoS information is used for categorizing web services in regards to a request of QoS demands [5]. Such QoS information comprises of performance (in terms of response time, latency etc.), accessibility, availability, throughput, security etc. which are expressed as a set of QoS properties. These QoS information have considerable impact on expectation of a user and the experience of using a Web Service. Hence it can be used as a main factor to distinguish and rank Web Services. The service which gets the highest QoS value is selected first. However it should be clear that this ranking step is performed only after the functional matching with the user’s request has been done.

The remaining paper is structured as follows: Section 2 deals with the related work in the field of modeling of QoS parameters. Section 3 gives an overview of Quality of Service parameters. Section 4 gives an overview of the proposed model. Section 5 is focused on formalization of different QoS parameters. Section 6 gives the implementation of QoS based WSS model. Section 7 gives an overview of the Simulation Environment and the Implementation aspect of the proposed model. It also provides Simulation results and evaluation. Section 8 deals with conclusions and future prospects of this work.

2 Related Work

Web Service Delegation model [9] provides safety and privacy. This work shows how a Delegation Web Service increases the security for Web Services, but doesn’t consider other QoS based parameter for selection. Web Service discovery based on QoS [10] suggests QoS enhanced UDDI architecture and discusses different QoS parameters, but doesn’t provide methods to calculate them and computing all the QoS parameters for service selection approach may lead to miss the relevant Web Service with low QoS parameters. [11] Introduces a model to calculate QoS parameters of different Web Services and advocates the use of the Web Service Broker as selector architecture.

Combining QoS-based Service Selection with Performance Prediction [12] selects Web Service based on performance prediction (availability, reliability, bandwidth, request time) using artificial neural network. Performance prediction model lags in other QoS parameters like security, correctness, failure masking etc. Performance criteria might be different with respect to functionality of Web Services and user's interest. Model of Pareto principle based QoS Web Service Selection [13] uses 80-20 rule to compute QoS rank of Web Services. Model reduces computation complexity of service selection as only 20 % of Web Services are ranked according to QoS parameter.

3-Way Satisfaction [14] for Web Service Selection Preliminary Investigation uses selection in community of similar Web Services. Master Web Service calculates SCORE of other slave Web Services based on capacity, execution time and availability. This approach solves the problem of selection Web Service within a community.

[19] Has proposed a novel modeling approach using associative classification. A CBA [19] algorithm is used to classify the candidate WS to different QoS levels. They classified the WS within each class, with respect to their distances from the user's demand for the QoS criteria. In nutshell approach uses the classification data mining algorithm to select the most eligible services respect to the user demand. Further approach uses semantic similarity between WS by semantic links it increases the accuracy of proposed modeling.

Benaboud and Maamri [20] presented a framework for WS discovery and selection based on intelligent mediator agents. In order to add dynamism to WSS model, they have applied OWL-S and domain ontology concepts. Agent based framework is implemented using JADE [21], which was implemented using JENA API. Modeling approach is based on matching and domain ontology of WS, it does not consider parameter based selection.

Guo and Le [22] proposed that discovery of WS should be based on the semantic match between WS providers and consumer query. It contributes by providing procedures to represent WS by the OWL-S profile and OWL based language for service description. A description of the design and implementation of a WS matchmaking mediator which acts on OWL-S ontology is made. It also uses an OWL reasoner to compare ontology based WS descriptions.

A SWSS model based on QoS attribute is presented in [23]. Framework is modeled by adding semantics of QoS attributes with web service profiles. It describes the design and implementation of a WS matchmaking agent. Agent uses an OWL-S based ontology and an OWL reasoner to compare ontology based service descriptions. [11,12] provide a sketch for framework implementation, but how to exactly formalize and retrieve QoS values from WS profiles, still requires a novel work.

Different models are suggested in the field of web service selection, but the proposed model in this paper additionally support security, reputation, availability, correctness and reliability for efficient service selection. In addition to QoS based WS selection, our approach takes user's preference of QoS for service selection. As Request is about journals and research work there is no need to calculate rank of WS using security, availability and performance. Similarly if user requests for online purchasing then cost, security are important to consider rather than correctness. So in nutshell model selects most relevant service among the functionally similar

Web Services, as per user's preference. User can specify any QoS parameter which should get the preference but in the absence of user's input, over all RankQoS based on weighted sum of specified QoS parameters is considered.

3 Quality of Service (QoS)

The Quality of Services Ranking describes the quality of web services. It is an important consideration when the consumer makes decision on service selection. Normally, the QoS attributes can be classified in two categories: dynamic and static - as described in [17]. Li et al. explain in [17] that dynamic attributes could be changed in the execution time, for example response time and throughput; static attributes are defined by service providers before service executions and are usually not updated during the execution. Table 3.1 presents some example attributes by this classification.

Attributes	QoS parameters
Dynamic	Availability, response time, throughput, reputation, stability etc.
Static	Scalability, capacity, accuracy, security, price,

Table 3.1: Behavior of QoS Parameter.

- QoS based selection translates user's vision into business processes more efficiently, since a Web Service can be designed according to QoS metrics.
- QoS allows for the evaluation of alternative strategies when adaptation becomes necessary. The unpredictable nature of the surrounding environment has an important impact on the strategies, methodologies, and structure of WPs. Thus, in order to complete a WP according to initial QoS requirements, it is necessary to expect to adapt and reschedule a WP in response to unexpected progress, delays or technical conditions [3].
- It allows for the selection and execution of WPs based on their QoS, to better fulfill customer expectations.
- This approach help to fulfill the service oriented architecture's goal. Now users are not forced to make Ad-hoc decisions to select pertinent service among the set of services which are functionally equivalent.
- It makes possible the monitoring of WPs based on QoS. WPs must be rigorously and constantly monitored throughout their life cycles to assure compliance both with initial QoS requirements and targeted objectives.

QoS parameters have different behavior on QoS ranking of Web Services. QoS contribution of Ranking Web services depends on its tendency towards better performance of Web Service . "Higher the better" and "Lower the better" tendencies of various QoS parameters are shown in table 3.2. In order to formulate "Lower the better" QoS, we need to consider inverse of its exact QoS value in normalized range.

QoS Parameters	Higher the better	Lower the better
Security	✓	✗
Integrity	✓	✗
Capacity	✓	✗
Scalability	✓	✗
Response Time	✗	✓
Mean Time Between Failure	✗	✓
Exception handling	✓	✗
Failure masking	✓	✗
Accountability	✓	✗
Failure semantics	✓	✗
Latency	✗	✓
Incomplete Transactions	✗	✓

Table 3.2: Tendency of QoS parameters for web service Ranking

4 Proposed Model

A *QoS mediator agent based Web Service selection* and Web Service registry model (shown in Fig. 1) is proposed in this paper. Whenever a search is performed, the mediator agent selects the list of matched services from the service pool and provides to the client. Client set the preference to QoS parameter and highest ranked service will be provided to the client.

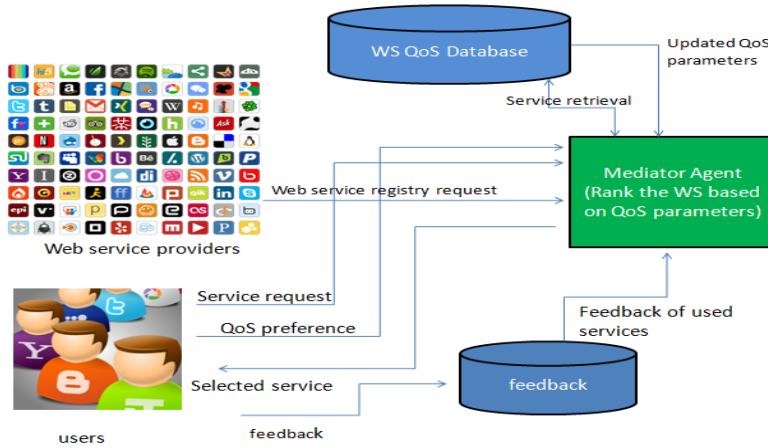


Fig. 1 QoS mediator agent based Web Service Selection Model.

Locating the desired Web Service to a client requirement is a difficult task as similar Web Services are readily available to satisfy a request. If there are many equivalent results returned by the QoS database, then the service_selection method is called that takes input as matched services on user request and then depending on the linearity of the constraints and the user's preferences, it executes the appropriate WS selection algorithm and returns back the results to user. Whenever a service registry is requested by service providers, a new Web Service is entered in database and respective Web Service is joined in given web-service set.

A. Algorithm for Service Registry:

Step 1: Web Services provided by different providers are stored in QoS database as their functional type and QoS values.

*Step 2: Normalization is performed using below formulae.
For negative parameters ("lower the better")*

$$Q_i = \begin{cases} \frac{Q_i^{\max} - Q_i}{Q_i^{\max} - Q_i^{\min}}, & \text{if } Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1, & \text{else} \end{cases} \quad [4.1]$$

For positive attributes ("higher the better")

$$Q_i = \begin{cases} \frac{Q_i - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}}, & \text{if } Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1, & \text{else} \end{cases} \quad [4.2]$$

Step 3: Calculate Initial_Rank_{QoS}.

Step 4: For each Web Service if initial QoS parameters are satisfying minimal criteria, assign them AverageRank_{QoS}, to participate in Service Selection procedure.

$$\begin{aligned} &\text{checkif}(Initial\ Rank_{QoS} \geq ThresholdRank_{QoS}) \\ &\text{then } Rank_{QoS} \leftarrow AverageRank_{QoS} \end{aligned}$$

Step 5: According to performance and feedback, Rank_{QoS} will be keep updating, on every selection of Respective Web Service.

If a WS has published good range of QoS parameters during its registry with Mediator agent, we have assigned it AverageRank_{QoS}. As every new WS should get an equal chance to be selected in QoS based WSS system, further on the basis of user's feedback and other WS availability, its QoS Rating will be updated.

B. Algorithm for Service Selection:

Step 1: From each user's query the semantically matched WS are extracted from database.

Step 2: Depending on the query the QoS parameters provided by the user is retrieved.

Step 3: Rank_{QoS} is calculated. A listing of the QoS parameter based on weighing schemes and the user's preference is also made used of.

Step 4: If user's required QoS parameters are not specified then rank the Web Services on standard weighing schemes and calculate Rank_{QoS}.

*Step 5: The best Rank_{QoS} Web Service is selected.
Where*

$$Rank_{QoS} = \sum Q_i \times W_i \quad [4.3]$$

$$AverageRank_{QoS} = \frac{\sum Q_i \times W_i}{N} \quad [4.4]$$

Where N is number of Web Services of given type and

Q_i = {performance, availability, reliability, feedback, execution time, throughput, security, scalability} is normalized value of considered QoS parameters and W_i is weighted contribution of selected QoS parameter.

5 Formalization of QoS Parameters

- **Performance:** It is a measure of how well a WS performs during its execution [5]. This is a prominent ingredient in overall having “higher the better” tendency. Performance is a composite attribute comprises of weighted sum of latency, throughput, and response time.

$$Q_{performance} = \sum_{i=1}^{i=3} W_i * Q_i \quad [5.1]$$

Where

- $Q_{performance}$: is a performance based rating of WS ranging in between 0 to 1.
- $Q_i = \{Q_{latency}, Q_{throughput}, Q_{response\ time}\}$
- *Latency* and *response time* have “lower the better” tendency for $QoS_{performance}$ rating.
- On the other hand *Throughput* has “higher the better” contribution on $QoS_{performance}$ ranking

- **Availability:** Availability of WS play a vital role in QoS rating with the behavior “higher the better”. It is measured as the probability of WS that the service will be up after selection. As the name shows it have complementary behaviour with unavailability. In this WSS modelling we have considered a WS with low $overallRank_{QoS}$ is considered as unavailable. Low $overallRank_{QoS}$ is assigned to a WS if WS $overallRank_{QoS}$ is less than $thresholdRank_{QoS}$.

$$QoS_{availability} = 1 - QoS_{unavailability} \quad [5.2]$$

Where

- $QoS_{unavailability}$: shows QoS value of unavailability of a WS, varying between 0 and 1.
- $QoS_{unavailability}$ has “lower the better”
- **Experience:** It is a dynamic parameter that increases as more number of times service gets selected in QoS based WSS system. Number of times service has been selected $QoS_{experience}$ value gets incremented by 1. It is also a positive parameter, shows “higher the better” contribution on QoS rating of a WS. With prior simulation we concluded that including *Experience* directly to QoS rating will biased the system towards earlier registered WS. So instead of directly adding $QoS_{Experience}$, Experience is used to calculate other QoS parameters like *Reputation*, *throughput* etc.
- **Reputation:** Reputation of WS shows satisfaction of users. It is collective built over the time as per user’s feedback. Feedback may be positive or negative.

$$Q_{Reputation} = \frac{N_{pos} - N_{neg}}{Experience} \quad [5.3]$$

Where

- N_{pos} : Number of times service is ranked with positive feedback.
 - $Q_{reputation}$: is a reputation based ranking of WS ranging -1 to +1.
 - N_{neg} : Number of times service is ranked with negative feedback
 - *Experience*: is a QoS parameter defined above.
- **Incomplete:** Number of times service was not successfully completed. Incompleteness is considered as a number of counts respective WS was being selected but not completed its execution. It is a negative parameter as it has “lower the better” contribution in QoS ranking.
- **Reliability:** Reliability is the ability of a WS to perform well over a given time span. Reliability[15] of a WS is measured in terms of *MTBF(mean time between failure)*, *recoverability*, *performance* and *availability*.

$$Q_{reliability} = \sum_{i=1}^{i=4} Q_i * W_i [5.4]$$

Where

- $Q_i = \{Q_{MTBF}, Q_{recoverability}, Q_{performance}, Q_{availability}\}$
 - Q_{MTBF} is the quality attribute of WS measured as mean of the time spans , when WS was in failure condition. It has “lower the better” contribution in QoS rating.
 - W_i is the weight associated with respective parameter.
 - $Q_{recoverability}$ Is the ability of WS from failover and disaster [15]
- **Security:** Security is the measure of how much it is secure to use a WS regarding different security threats [16]. Different security mechanisms used by Web Services are ranked Initially by mediator agent [16]. To calculate Risk of WS.

$$Q_{security} = F(\text{security protocol, encryption methods, auditability, Risk}) [5.5]$$

Where

- Risk should be minimum for good $Q_{security}$.
 - Risk is measured in terms of inbound and outbound attacks [16].
- **Scalability:** scalability of WS is measured in the term of maximum number of simultaneous transactions on WS without decreasing its performance. It is taken as a core parameter at the time of service registry and used at the time of load balancing. It is a static parameter as it is computed by mediator agent at the time of WS registry.

$$\text{Scalability} = \{\max(N_s) \mid \text{Performance is constant}\} [5.6]$$

Where

- N_s : number of simultaneous transactions on WS

- Performance is QoS value of WS.
- **Throughput:** $Q_{throughput}$ is the vital contributor in QoS rating with “higher the better” behavior. [5] Throughput is defined as total number of completed transactions by a Web service over a time period.

$$Q_{throughput} = 1 - \frac{Q_{incomplete}}{Q_{experience}}$$

Where

- $Q_{throughput}$: is a throughput based ranking of WS ranging between 0 to 1.
- $Q_{incomplete}$ and $Q_{experience}$ are the QoS parameter we have formalized earlier
- **Dependability:** It refers to the service delivering capability of a service that can be trusted justifiably. It is calculated from the complete transaction of its successor and predecessor services at dynamic composition.
- **ExecutionTime:** Initially published by Web Service provider. Execution Time of a service should be updated at the time of web service registry at QoS Agent side. and further updated as per user’s feedback of execution time {exact, delayed}.
- **ResponseTime:** Response time is the overall time required to complete a service request. It is a composite quality attribute comprises of latency and network delay with “lower the better” tendency towards QoS Rating of a WS. Equation 4.1.9 is used for computation of response time.

$$Q_{responsetime} = 100 * \frac{(e^{-(r^2)} * e^{(-r+\beta r)})}{(1 + e^{(-r+\beta r)})} \quad [5.8]$$

Where

- r is the response time, which is measured by running the service.
- βr is the Service Level Agreement (SLA) value for response time.

The first component deals with the contribution of the present response time in the quality rating. The second component deals with the contribution of the overall difference from the published value. It is the contribution of the overall deviation of response time from the published SLA. Mei and Meeuwissen in [18] explained that the SLA is a concept to get QoS guarantees between service providers and consumers at the network level. We can regard the SLA as a measure standard, which can be used to evaluate the service quality.

- **Stability:** As we know stability refers constancy of WS[5]. A good WS should have lesser variation in its QoS rating. Stability concerns whether service is dependable or not. As much variation in QoS attributes shows dynamic behavior of WS. A less stable WS cannot be predicted for its performance. It may be the scenario that after selection WS is not executing according to modeling prediction or its performance is not on a par. It leads to diminish the

efficiency of SWSS model. In nutshell, Stability is a considerable attribute for QoS based SWSS modeling.

Stability is a positive attribute of a WS, it show dependability on Web Services. However Li-Li and Yan [25] has defined Stability as rate change of web services parameters. But they did not discuss how to measure it or retrieve from web service profile. Stability is inversely proportional to rate change of dynamic QoS attributes like performance, response time. It depends on deviated value of dynamic QoS attributes. Equation 5.10 is used for stability computation in this SWSS model

$$Q_{\text{stability}} = 1 - \sum_{k=0}^3 \Delta Q_i * W_i \quad [5.9]$$

Where

- $Q_{\text{stability}}$ is QoS rating of WS lies in between 0 to 1.
- ΔQ_i is deviated value of QoS parameters $\{Q_{\text{performance}}, Q_{\text{responsetime}}, Q_{\text{reputation}}\}$.
- W_i : is weighted contribution of respective QoS parameters in order to calculate QoS value of stability.

While computation of ΔQ_i is done on the basis of its previous QoS value and current value of respective QoS attribute. Previous QoS value is stored in web service profile's ontology as the mean of previous value.

$$\Delta Q_i = |\text{mean } Q_i - \text{new } Q_i|$$

Mean value of QoS attributes is keep updating after every selection, using equation.

Where

$$\text{Mean QoS}_{\text{new}} = \frac{(\text{Mean QoS}_{\text{previous}} * \text{Experience} + \text{QoS}_{\text{new}})}{\text{Experience} + 1} \quad [5.10]$$

- Mean $\text{QoS}_{\text{previous}}$: denotes previously stored mean value of a particular QoS parameter.
- Mean QoS_{new} : Refers to the updated mean value of a QoS parameter.
- Experience : is an attribute associated with WS profile which refers to the number of times WS gets selected.

Up to this level we have formalized different QoS attributes regarding a WS. Every QoS attribute has its own range of rating. So to calculate cumulative QoS Rank, we need to normalize them in a common range. In our system we normalized different QoS attributes in the range of 0 to 1 using equation. Now all the attributes are of same range so we can directly use them for their weighted contribution on QoS rating of WS.

$$Rank_{QoS} = \sum Q_i \times W_i$$

$$AverageRank_{QoS} = \frac{\sum Q_i \times W_i}{N}$$

Where

- N is number of Web Services of given type.
- $Q_i = \{performance, availability, reliability, feedback, execution time, throughput, security, dependability, scalability\}$: is normalized value of considered QoS parameters and W_i is weighted contribution of selected QoS parameter.

6 Implementation of QoS Based Semantic WSS Model

6.1 Modeling Architecture for QoS based Semantic WSS

In current web some core Quality of service (QoS) parameters of a web service are registered in the UDDI entry. These QoS information can be retrieved by the consumers or the brokering systems. So there is a requirement of database to store all the Quality of service information regarding web services (as we have used for Net-logo simulation of SWSS). Maintaining such a large database for QoS for innumerable web service is an onerous task. Li and Zhou elaborated in [17] that such mechanisms based on the UDDI registry, are less efficient due to the fact that their selection results always contain irrelevant and unjustified values. There is no mean to associate QoS Parameters with service profiles. This problem is caused due to lack of semantic support. To address this problem, QoS based semantic WSS approach is proposed where services are built as OWL-S profile and QoS parameters ontology is attached with profile itself. In order to add semantics to QoS based WSS model, semantically enriched *Railway reservation* web service profiles are built using OWL-S [28,29] plug in *protégé 3.1*. OWL-S: is a tool that provides ontology for web service concepts. OWL-S is enriched with core set of markup language constructs for describing the properties and capabilities of a Web service in unambiguous, computer-interpretable form. OWL-S Editor is provided as a Protégé (3.x) plug in. OWL-S describes web service in following three areas (fig:6.1).

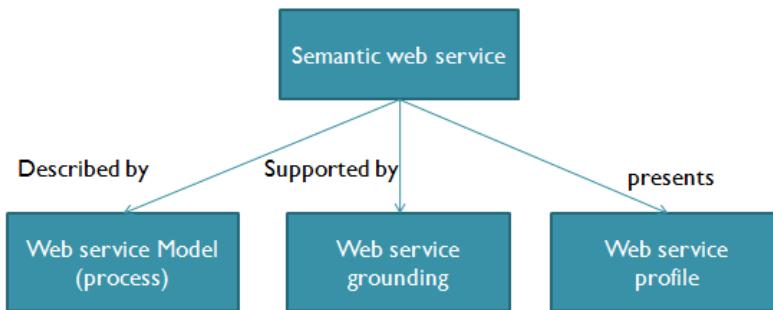


Figure 6.1: OWL-S classes relationship for developing web service profiles

Process Model: A process model describes web service performs its works. Here we specify IOPR information. IOPR stands inputs, outputs (specific conditions for various outputs, preconditions (circumstances that must hold before a WS can be invoked), and results (changes brought after process execution).

OWL-S *process model* provides following three different types of processes: Atomic process, Simple process, Composite process.

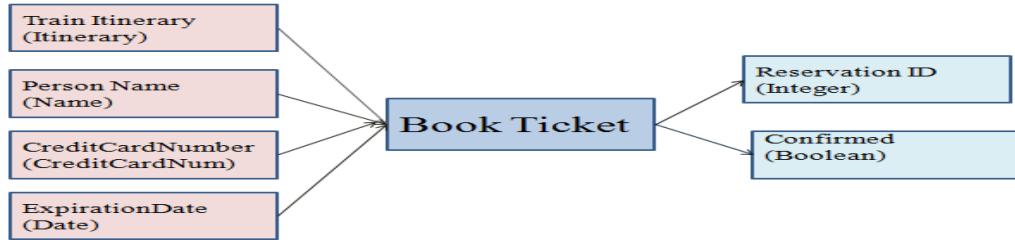


Figure 6.2: IOPR of Book ticket (atomic process)

Atomic processes are basic of WS functionality where we define IOPR of process. Representation of considered *Railway reservation* Web Service *Book ticket, search train and select train* are the atomic processes, which shows basic tasks of service. IOPR of atomic processes(Book Ticket) used in rail reservation WS are shown in (figure 6.2). *Simple processes* are abstract process descriptions. Simple process can relate to other composite or atomic processes. *Composite processes* are the composition of atomic processes. This process model breakdowns a composite task into simpler components. Here we need to define flow of control and data between atomic processes as shown for *RailwayReservation* (composite process) in figure 6.3.

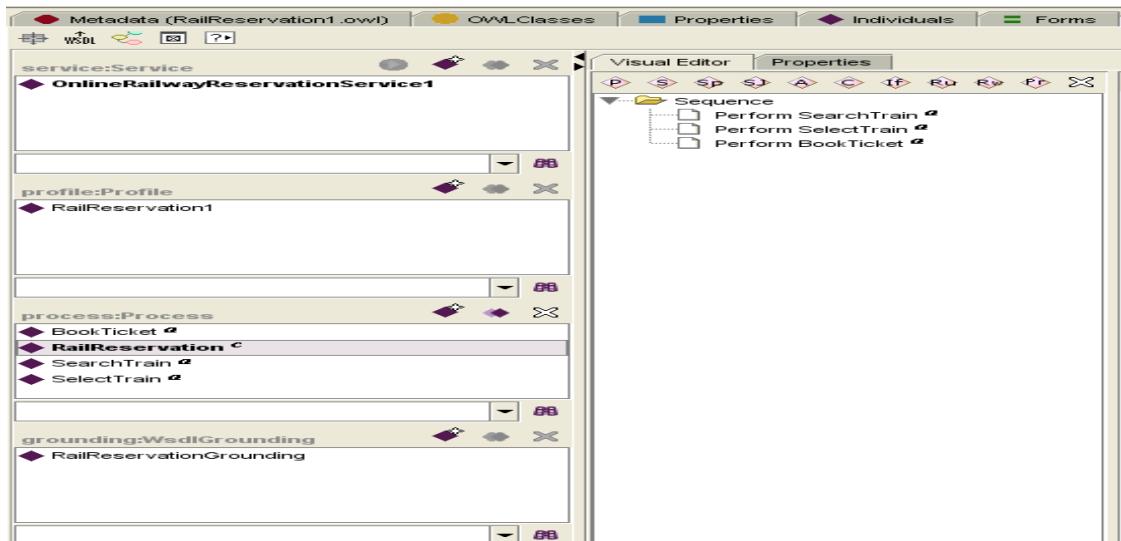


Figure 6.3 process graph of Railway Reservation (composite process)

Grounding: A grounding [29] of WS describes the invocation of web service (figure 6.4) by detailing the way in which atomic processes in a service's process model map onto a concrete messaging protocol.

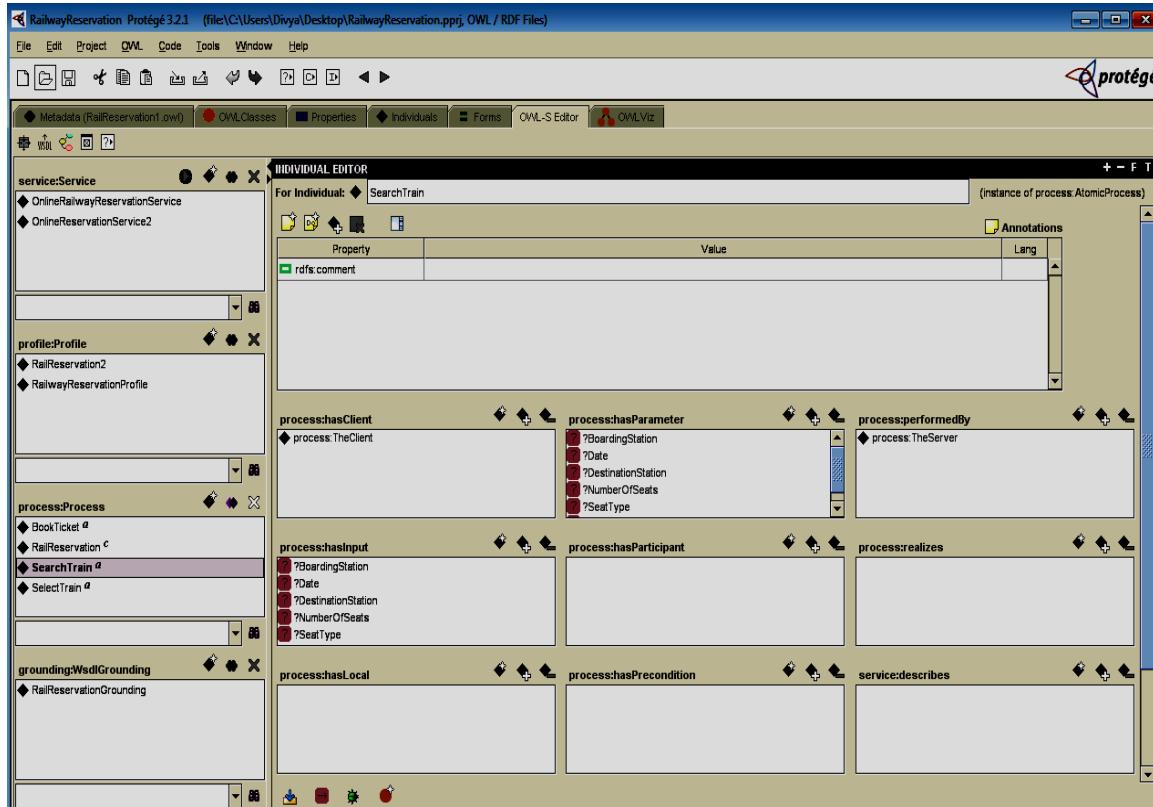


Figure 6.4 Web services grounding - “Railway reservation”

In order to implement QoS based SWSS model, ontology of QoS based Semantic WSS modeling (figure 6.1) and ontology of QoS parameters (as shown in figure 6.5) was developed using protégé [26]. Figure 6.2 shows a generic ontology of the QoS parameters that has been used in the proposed model design.

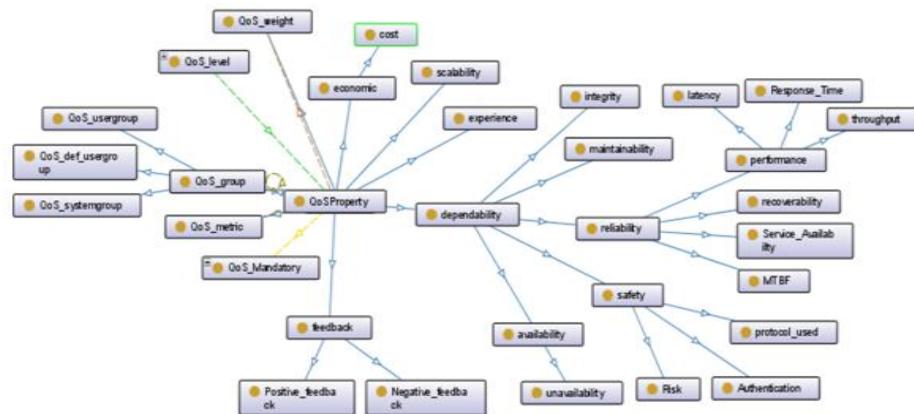
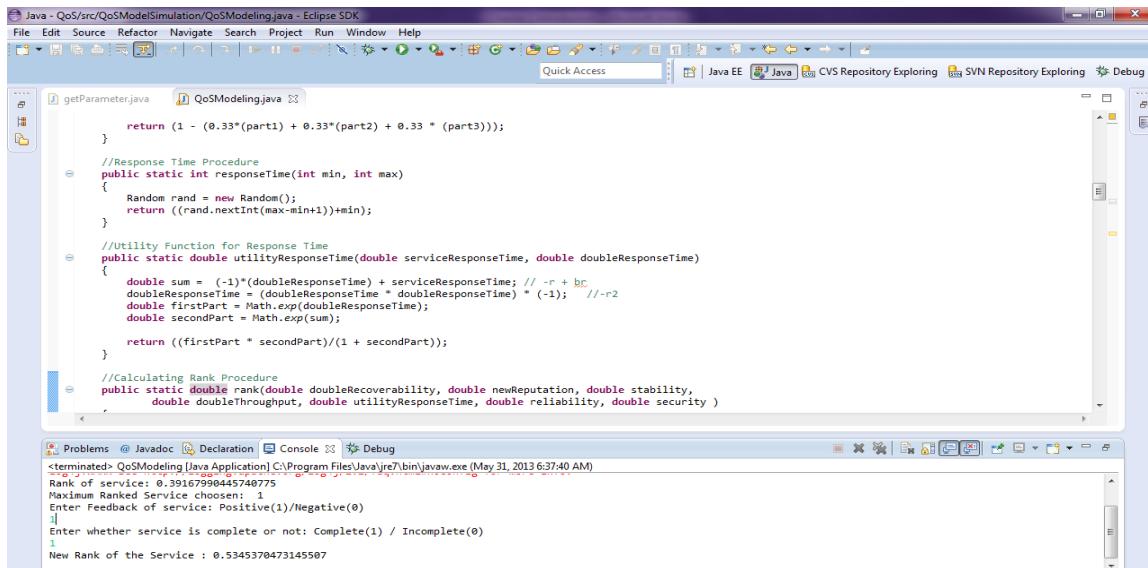


Figure 6.5 Generic Ontology of the QoS parameters

QoS ontology developed for proposed semantic WSS modeling depicted in Figs- 5.2. Ontology shown in figure 6.5 relates to various roles in defining QoS information like QoS description, QoS mandatory, QoS level, QoS weight, QoS formula, QoS priority, and QoS group. While ontology shown in fig 5.2 shows a set of core QoS properties.

6.2 Integration of QoS Modeling With Semantic Service Profile

The QoS model architecture shown above is integrated with the Semantic Service Profile. Figure 6.6 shows the results which were obtained after incorporating the service profiles with our model design. The programming was done on Eclipse [27] platform with Java being the programming language. Jena library [28] was used for interaction with the Service profiles which were prepared using Protégé 3.2.1. The formalizations which were made earlier in the report were incorporated in the model prepared here.



The screenshot shows the Eclipse IDE interface with two tabs open: `getParameter.java` and `QoSModeling.java`. The `QoSModeling.java` tab contains the following Java code:

```

Java - QoS/src/QoSModelSimulation/QoSModeling.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Java CVS Repository Exploring SVN Repository Exploring Debug
getParameter.java QoSModeling.java
    return (1 - (0.33*(part1) + 0.33*(part2) + 0.33 * (part3)));
}

//Response Time Procedure
public static int responseTime(int min, int max)
{
    Random rand = new Random();
    return ((rand.nextInt(max-min+1))+min);
}

//Utility Function for Response Time
public static double utilityResponseTime(double serviceResponseTime, double doubleResponseTime)
{
    double sum = (-1)*(doubleResponseTime) + serviceResponseTime; // -r + br
    double doubleResponseTime = (doubleResponseTime * doubleResponseTime) * (-1); // -r^2
    double firstPart = Math.exp(doubleResponseTime);
    double secondPart = Math.exp(sum);

    return ((firstPart * secondPart)/(1 + secondPart));
}

//Calculating Rank Procedure
public static double rank(double doubleRecoverability, double newReputation, double stability,
    double doubleThroughput, double utilityResponseTime, double reliability, double security )

```

The `Console` tab at the bottom shows the output of the Java application:

```

<terminated> QoSModeling [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 31, 2013 6:37:40 AM)
Rank of service: 0.39167998445740775
Maximum Ranked Service chosen: 1
Enter Feedback of service: Positive(1)/Negative(0)
1
Enter whether service is complete or not: Complete(1) / Incomplete(0)
1
New Rank of the Service : 0.5345370473145507

```

Figure 6.6 Screen shot of programming interface

The result clearly validates the proposed model design. The code prepared is modular with separate functionalities assigned to separate modules. `getRank()` is one such function which gets the previous rank of the service from the service profiles and is used to find the best available service in the market. Similarly many other functions like `rank()`, `utilityResponseTime()`, etc. has been made which work together to assign new rank to the service depending on the feedback provided by the user.

7 Simulation Environment, Implementation, Results and Comparative Analysis

The system is developed using NETLOGO [7] and MYSQL [8] database. Initially service given by providers is assumed in text format of particular functional types, then *core* QoS parameters published by service provider are normalized and store in database. This QoS database is being processed by NETLOGO MYSQL extension.

Every service registry request will generate a new Web Service in system linked with *Service Agent 0*. Respective entry for Web Service is done in database and initial rank is calculated and *AverageRankQOS* is assigned. For further selection, service's updated Rank is assigned to *RankQOS*.

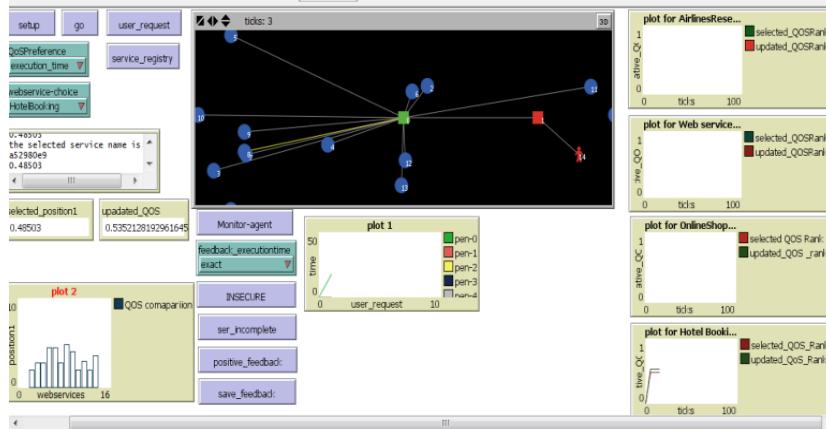


Figure 7.1 Best Service Selection of type “hotel booking” on User’s request.

To quantify the mediator agent based Web Service selection model, we have made a graphical comparison (Fig. 7.2) between *RankQOS* and successful number of services (*experience – incomplete number of services*) provided by Web Services at a given time period. As both are varying in similar way, means highest QoS rank Web Service always have high performance as per user’s request. High valued *RankQOS* service should be selected maximum time. As experience and *RankQOS* are following the same graph pattern, so we can say that proposed model is selecting the best service among the same functionality Web Services. As in the above graph experience is not NULL for low valued *RankQOS*, this infers that every service is getting chance to select.

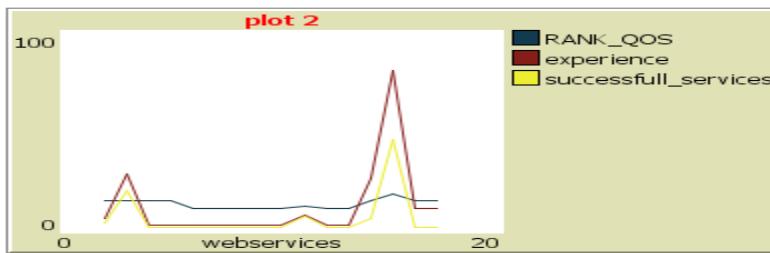


Fig. 7.2 Service Selection comparison for a type of Web Service

After comparison with the Selection model using Pareto Principle [13] we found that this Pareto model do not rank the web services according to user’s requirements and feedback. On the other hand the model proposed in this paper takes user’s requirements and satisfaction into consideration for ranking the web services. 3-Way Satisfaction for Web Service Selection [14] gives selection method with in a community and uses *Score* (*availability, correctness, execution time*).

A comparative analysis based on QoS parameters is done in order to justify the usability of proposed modeling. Selecting most pertinent Web Service and not missing the appropriate Web Services are the two complimentary issues in Web Service selection process. Selecting Web Services according to user's requirement will accomplish both the issues. Feedback is also collected for further selection of same Web Service and to rank web services as per user's requirement.

Table 7.1 shows that Web service ranking with the proposed model uses additional parameters like security, reliability, throughput, reputation etc. for efficient and effective selection as shown in table.

Table 7.1 Comparison of SWSS modeling based on QoS parameters.

QoS parameter used in various models	AHP [15]	MM [3]	Pareto Prin. SM [13]	3-way Satisfaction Model [14]	Req. based Broker Arch. [5]	WSS based on Naïve Bayes [4]	Proposed QoS based WSS model
Availability	Yes	Yes	No	Yes	Yes	No	Yes
Reliability	Yes	Yes	No	No	Yes	No	Yes
Depend- ability	No	Yes	No	No	No	No	Yes
Reputation	Yes	Yes	Yes	Yes	Yes	No	Yes
Feedback	No	No	No	No	Yes	No	Yes
Failure Semantic	No	No	No	No	No	No	No
Flexibility Scalability	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Security	No	No	No	No	No	No	Yes
Response Time	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Throughput	Yes	Yes	No	No	Yes	Yes	Yes
Integrity	Yes	Yes	No	No	Yes	No	No
Stability	Yes	Yes	No	No	Yes	No	Yes

8 Conclusions and Future Work

In order to enhance efficiency of Web Service selection based on QoS Ranking, this model presents mediator Agent based approach for QoS based Web Service selection. It presents a uniform and lightweight solution for mining QoS parameters. The model selects most relevant service among the functionally similar Web Services, as per user's preference. User can specify any QoS parameter which should get the preference but in the absence of user's input, over all RankQoS based on weighted sum of specified QoS parameters is considered.

A modeling approach is proposed for QoS based semantic WSS model (SWSS) and formalization for various QoS attributes (like reliability, stability, availability, incompleteness etc.) are presented. Mediator agent based semantic web service model is proposed in the modeling approach. Further service registry and service selection algorithms are listed for pertinent WSS based on QoS parameters. Later on different QoS attribute associated with web services are discussed with proper formalization.

The proposed model is simulated on Net-Logo and the results were used to check the efficiency of the proposed model. A generic ontology for the QoS parameters was prepared and the model was integrated with the semantic service profile. In order to read the ontology of the website Jena library is used. The results obtained show the validity of our proposed model. A comparative study of other research work on QoS parameters is also made to emphasize upon the large number of QoS parameters taken into consideration in the proposed model.

In our future work, we would like to explore more QoS parameters and continue our research in the field of service composition in semantic environment. Parameters like integrity, compliance, etc. can be modeled and formulated accordingly to improve the existing service selection models available in this research area. Further QoS based SWSS model can be collaborated with context attribute based parameters to increase the effectiveness and efficiency of the models.

Reference

- [1] W. Junhao, G. Jianan , J. Zhuo. Semantic Web Service Selection based on QoS . International Joint Conference on Service Sciences, 2011: 163-169
- [2] UDDI .http://www.uddi.org/pubs/uddi_v3.html.
- [3] G. Guo, Fei Yu and Dong Xie. A Four level Model For Web Service Selection Based on QoS Ontology .Third International Symposium on Information Science and Engineering, 2010: 630-634.
- [4] S.Chitra , K. Vidhya and G. Aghila .A Web Service Selection based on Ranking of QoS using Naïve Bayes .ICCCT,2010: 782-789
- [5] K. Kritikos, D. Plexousakis. Requirements for QoS-Based Web Service Description and Discovery, IEEE Transactions on ServiceComputing ,2009: 320-337.
- [6] GUO De-keREN, YanCHEN Hong-huiXUE, Qun-weILUO, Xue-shan. A Web Services Selection and Ranking Model with QoS Constraints. journal of Shanghai Jiaotong University,2007, 41(6):870-875.
- [7] NETLOGO. http://www.netlogo.org/pubs/netlogo_v5.html.
- [8] J. Blom, R. Quakkelaar M.Rotteveel “NetLogo SQL Wrapper User manual”ccl.northwestern.edu/netlogo
- [9] H. S. Hwang, H. J. Ko, K. I. Kim, U. M. Kim. Agent-Based Delegation Model for the Secure Web Service in Ubiquitous Computing Environments. In Proceedings of the 2006 International Conference on Hybrid Information Technology. Volume 1,pp.51–57. Nov. 2006.
- [10] S. Ran. A Model for Web Services DiscoveryWith QoS. ACM SIGecom Exchanges 4(1):1–10, 2003.
- [11] D. A. DMello, V. S. Ananthanarayana. Quality Driven Web Service Selection and Ranking. In Proceedings of the Fifth International Conference on Information Technology:New Generations. Volume 5, pp. 1175–1176. Chicago, Apr. 2008.
- [12] Zhengdong Gao, Gengfeng Wu . combining QoS based service selection with performance prediction. In proceeding IEEE International Conference on e-Business Engineering 2005.
- [13] Lican Huang. Pareto Principle to Improve Efficiency for Selection of QoSWeb Services, 7th IEEE Consumer Communications and Networking Conference (CCNC), 2010 :1-2.

- [14] Erbin Lim, Philippe Thiran , Zakaria Maamar, Jamal Bentahar. 3-Way Satisfaction For WebService Selection Preliminary Investigation, IEEE International Conference on Services Computing (SCC), 2011: 731-732.
- [15] V. X. Tran, H. Tsuji and R. Masuda, A new QoS ontology and its QoS-based ranking algorithm for web services. ELSEVIER, Simulation modellingpractice and theory (17) 2009: 1378-1398.
- [16] V. Prasath, Modeling the Evaluation Criteria for Security Patterns in Web Service Discovery, International Journal of Computer Applications 2010: 0975 – 8887.
- [17] S. Li and J. Zhou, "The WSMO-QoS Semantic Web Service Discovery framework", International Conference on Computational Intelligence and Software Engineering, 2009: 1-5.
- [18] D.A. Menasce and V. Dubey, "Utility-based QoS Brokering in Service Oriented Architectures", In Proceedings of IEEE International Conference on Web Service (ICWS 2007), pp. 422 - 430, Salt Lake City, UT, 2007.
- [19] R.D. van der Mei and H.B. Meeuwissen "Modelling End-to-end Quality-of-Service for Transaction-Based Services in Multi-Domain Environments", In proceedings of IEEEInternational Conference on Web Services, pp.3 – 462, Washington, DC, USA, 2006.
- [20] M. Makhlughian, S. M. Hashemi, Y. Rastegari and E. Pejman, Web Service Selection Based On Ranking Of Qos Using Associative Classification, International Journal On Web Service Computing (IJWSC), Vol.3, No.1, March 2012: 1-14.
- [21] R. Benaboud, R. Maamri, and Z. Sahnoun, Semantic Web Service Discovery Based on Agents and Ontologies, International Journal of Innovation, Management and Technology, Vol. 3, No. 4, August 2012: 467-472.
- [22] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, (2003). Jade Programmer's Guide. [Online]. Available:<http://sharon.cselt.it/projects/jade/>.
- [23] R. Guo^{1,2}, J. Le and X. Ling Capability Matching of Web Services Based on OWL-S Xia3 Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05) IEEE 2005.
- [24] S. CHAARI, Y. BADR and F. BIENNIE, Enhancing Web Service Selection by QoS-Based Ontology and WS-Policy 23rd Annual ACM Symposium on Applied Computing SAC'08, March 16-20, 2008: 2426-2432.
- [25] QU Li-li and C. Yan, "QoS Ontology Based Efficient Web Services Selection",International Conference on Management Science & Engineering (16th), pp. 45-50, 2009.
- [26] Protégé, Ontology Editor, 2007.<http://protege.stanford.edu/>. Cited on 3rd june2013.
- [27] Eclipse", <http://www.eclipse.org/>, last visited on June 2013.
- [28] "Jena Library",<http://jena.apache.org/>, [last visited on June 2013](#).
- [29] S. Saadati and G. Denker. An OWL-S Editor Tutorial Version 1.1, SRI International,Menlo Park, CA 94025.