Journal of Universal Computer Science, vol. 13, no. 9 (2007), 1138-1156
submitted: 15/8/06, accepted: 1/3/07, appeared: 28/9/07 © J.UCS

Web Service Selection based on QoS Knowledge Management

Poulia Adamopoulou, Evangelos Sakkopoulos, Athanasios Tsakalidis
Miltiadis Lytras
Department of Computer Engineering and Informatics,
School of Engineering, University of Patras
Rio, 26500 Patras, Greece
{adamopop, sakkopul, tsak, lytras}@ceid.upatras.gr

Abstract: The Semantic Web vision is among others to allow automatic identification and selection of Web documents and services to meet the requirements of users. In this work we provide a novel solution that supports organizational knowledge flow utilizing non-functional qualitative criteria for web service consumption. A knowledge management web service selection mediator is presented based on the Web Services resource framework (WSRF). It enhances organizational best practices and promotes reusability of successful services during the process of online Web Service selection. Apart from meeting functionality requirements, the mediator utilizes previous domain knowledge to base its decision upon Quality of Web Service (QoWS) ontology knowledge. The proposed solution is open and able to host a number of different selection policies and business logic implementations. We present and experimentally evaluate and compare four such selection policies. The design of the proposed mechanism is analyzed and implementation details are discussed. Evaluation results have shown that the solution is able to satisfy cases and scenarios that have been derived while studying and working on web services selection business processes for different enterprises.

Key Words: Knowledge Management for Quality of Web Service, WS Resources, Web Service Selection Algorithm
Category: C.2.4, H.3.4, H.3.5

1 Introduction

Web Services (WS), which have emerged as a dominating set of recommendations and standards (W3C, OASIS), promise to mark a step toward the new era of Semantic Web [Berners-Lee et al 2001]. Web Service technologies promote re-usability and therefore, they are a powerful vehicle for the implementation of enhanced knowledge management Information Technology practices. The Web increasingly turns to be a repository of interconnected documents and services offered by a number of organizations that strive for advanced IT solutions. However, as the number of different web service providers increases, the user-consumers find it more and more time exhausting to detect the "best" matching web service. In particular, service selection requires more than capabilities descriptions and input/output type matching. Rather, service should include information concerning quality characteristics (brand, reputation, cost, success rate) to facilitate effective knowledge transfer. This additional knowledge is necessary for the acceptance and penetration of web service technology both into cross-organizational IT cooperation [Ordonez De Pablos 2006] and e-service consumption by end-user.

In Service Oriented Architecture [SOA 2004] the WS Discovery Knowledge Management [Lytras and Sicilia 2005] is defined in a broader sense as "the act of locating a machine-processable description of a WS that may have been previously unknown and that meets certain functional criteria." It is a service responsible for the process of performing discovery, a logical role, which could be performed by either the requester agent, the provider agent or some other agent.

In order to minimize negotiation effort and maximize the chance to discover an effective and efficient web service fitting particular contractual and technical profile, QoS metadata [Sicilia 2006] information needs to be made available in an accessible way for machine processing. Such Quality of Web Service knowledge management procedures [Russ and Jones 2006] can be implemented using extensions within the structure of the existing Web Service descriptions, such as the Web Services Description Language (WSDL).

Invoking services from distance, by sending and in most cases receiving messages back, has up to now been the main use of Web Services. A few typical scenarios of potential Web Service applications include business-to-business process integration, cooperation with multiple partners and even with competitors or enterprise application intercommunication, for example, integration of a company's customers database with its sales department application. The fact that WSs found in repositories can be tagged with a plenitude of information complicates the methods that narrow the discovery down to those services that match a particular technical fingerprint. However, as the demand for WS consumption is rising, this situation is proved to be inefficient because of the needs suggested by the complicated applications. These application domains not only require access to composite WSs but also seek the possibility to choose between an abundance of provided Web Services with the same functional and possibly different qualitative characteristics.

In fact, delivering QoS is a critical and significant challenge for most web service solutions because of the dynamic and unpredictable nature of the Web. It is foreseeable that there may be more than one Web services available that can meet the functional requirements with different quality of service attributes. On the other hand, many web applications require a guaranteed level of service quality delivered to them. Therefore the ability of incorporating quality of service into service discovery process becomes very important. However, most of today's Web service implementations do not guarantee the levels of service quality delivered to their users.

At present, UDDI only allows clients to look for Web services based on their functionality but not QoS knowledge management. The lack of adequate QoS metadata support prevents the adoption of performance sensitive web services. Without a careful management of web service QoS, critical business applications may suffer detrimental performance degradation, and result in customer dissatisfaction or financial losses.

However, in this area, to the authors best knowledge, no prior work has been proposed concerning the "on-line" selection of Web Services taking into consideration QoS metadata based on utilizing a standard framework such as the WS resources (WSRF). WSRF specifications were developed in the first place to support and define Web Service conventions that enable the discovery of, introspection on, and interaction with stateful resources in standard and interoperable ways. WSRF does so within the context of established Web Services standards. In this way we present a novel solution that is build upon a widely acceptable framework-standard.

In particular, we propose a Web Service architecture by deploying a WSRF based and QoS enabled Broker between Web Service clients and Web Service providers. The broker is an independent entity (in fact a Web Service itself), which utilizes the QoS metadata about service providers that may offer qualified Web Services to a client and makes selection decisions based on some quality criteria. The proposed solution is accompanied by four (4) different quality of service policies that handle different cases of Web Service consumption requirements. The solution is designed and developed using WSRF.NET, which is an implementation of the WSRF specifications on the Microsoft .NET technology platform [.NET framework 2006]. To evaluate the solution, we have also conducted two (2) sets of experiments for all available policies to measure and compare the performance under different customer requirements (low and high demand).

The rest of the paper is organized as follows. Section 2 presents related work in the area. Section 3 provides two sub-sections with prerequisites concerning a) WS background information and b) a brief overview of the WS-Resource framework. Section 4 presents the QoS-broker architecture. Our prototype implementation is discussed in Section 5. Section 6 shows some system performance data. The paper is concluded in Section 7, where future steps are also presented.

2 Related work and Motivation

Though the Web Services Knowledge Management research area is relatively new, there are several promising attempts. There are some proposal of ontologies supporting QoS for Web Services. In particular, QoSOnt solution is designed for service-centric systems and it comprises three layers: base QoS, quality attribute layer, and domain specific layer [Dobson et al 2005]. The base QoS layer contains generic concepts relevant to QoS. The second layer contains ontologies defining particular QoS attributes and their metrics. The domain specific layer links to specific types of information systems. Furthermore, WSQoS ontology [Tian et al 2004] is designed for Web service discovery with QoS support. Another QoS ontology [Maximilien and Singh 2004] lets service agents match advertised quality levels with specific Qos preferences. Moreover, supporting Quality of Service metadata for Web Services (QoWS) has found considerable interest in [Liu et al 2004], [Yu and Lin 2004]. In our case, we have introduced QoWS knowledge ontology into the mediator service.

Several studies focus on the dynamic selection of the best provider for a particular Web Service, others on dynamic resource management (e.g., load balancing) to support sufficiently fast Web Service execution. Furthermore, metadata enabled - semantic discovery of web services that manage and deliver web media content approach is presented in [Sakkopoulos et al 2006].

In this case, we mainly focus on the incorporation of QoS metadata in the selection process. An extensive Web Service Discovery overview can be found in [Garofalakis et al 2006]. In general the WS matching procedure does not usually take into account QoS concerns such as the response time. However, in a real-world invocation environment, aspects such as response time are paramount. In the work of [Ran 2003], several desirable characteristics that define Quality of Web Service (QoWS) are presented. Furthermore, in [Ouzzani and Bouguettaya 2004] a first attempt was made to address the issue of Web Service Discovery setting certain QoS constraints.

However, our work extends the previous concepts as it supports multiple QoS policies depending on the operational preferences of the mediator owner to choose from. The different policies are compatible to work all in parallel allowing their application to specific Web Services' categories such corresponding category-annotation details exist. Multiple policies are particularly useful in cases of corporate based WS catalogues where an enterprise may need to apply a specific QoS policy to the Web Service resources available through its network in order to maximize throughput or to minimize the probability of network overload under specific business intelligence. Moreover, the proposed work is build upon a standardized framework for e-Services supporting greater scalability and applicability. It allows application even to WS based Grid infrastructure beside standard business collaboration and workflow environments.

Additional alternate approaches include:

– an P2P technique in [Makris et al 2005a]

– an efficient and adaptive "online" discovery technique that is discussed in [Makris et at 2005b]

– the impact of Web Services in combining Hypermedia Services that is presented in [Karousos et al 2003]

3 Prerequisites

In this section, Web Services and WS-Resources overviews are presented to facilitate the readership for the rest of this work. First, the WS framework is discussed and following the WSRF technology concepts are introduced.

3.1 Web Services background

Some introductory information on Web Services technology are presented in this section. The key elements and related standards are outlined shortly. The most important concept perhaps in modern software development is the Service Oriented Architecture or SOA [SOA 2004]. SOA is an architecture that provides specifications for the implemented distributed system such as:

1. Operation transparency,
2. Message driven communication,
3. Services are accompanied by service descriptions, and
4. Platform independence.

The Web Service alternative to distributed software development has been adopted by the community due to its flexibility, ease of development and platform independence, characteristics of the core protocols of WSs, namely XML and HTTP and related WWW protocols.

The typical WS operational model [Ran 2003] is the “publish-find-bind” model. This model involves three entities: The provider, the consumer and the registry. The provider registers its services to the UDDI registry using SOAP/WSDL and waits for service requests. A consumer process, queries the registry for a Web Service, using SOAP messages. Note that the queries provided by the standardized UDDI querying APIs address only functional requirements. Our efforts are focused at this particular point in order to increase the intelligence of the selection process. Once, selection is done, the consumer process determines the binding procedure and from this point and on it requests, with SOAP messages, service provision from the provider. The provider ideally responds, executes and returns results in SOAP messages.

3.2 The WS-Resource framework

The WS-Resource construct has been proposed as a means of expressing the relationship between stateful resources and Web Services [WSRF 2004]. More specifically, while Web Service implementations do not explicitly maintain state information during their interactions, their interfaces must frequently allow the manipulation of state, that is, data values that persist across and evolve as a result of Web Service interactions. The messages that the services exchange imply the existence of an associated stateful resource type.

For example, an online bidding system must maintain state concerning ongoing auctions status, information about the current bid for that auction, information about the bidder who currently has the highest bid and information about the individual offering the item up for auction. Web Service interfaces that allow requesters to make bids, and manage the bidding system must necessarily provide access to this state. Therefore, it is desirable to define Web Service conventions to enable the discovery of, introspection on, and interaction with stateful resources in standard and interoperable ways. These observations motivated the Web Services Resource Framework (WSRF) which is a set of specifications that define a "stateful resource" as an abstraction for the state with which a Web Service interacts, and the way it can be discovered, queried and manipulated via Web Services. This set of specifications describes the relationship between "stateful resources" and Web Services in terms of WS-Resources, an abstraction for modeling and discovering the state manipulated by Web Services. A WS-Resource is "the combination of a Web Service and a stateful resource", that is, an abstraction for a collection of state that is manipulated by a particular Web Service. WSRF consists of a set of specifications that define how WS-Resources are named, discovered, queried, indexed, altered, and how their lifetimes are managed. These technical specifications are [WSRF 2004]:

- WS-ResourceLifetime: describes mechanisms for WS-Resource destruction, either synchronous or asynchronous to a destroy request
- WS-ResourceProperties: describes a definition of a WS-Resource, and mechanisms for retrieving, changing, and deleting WSResource properties.
- WSRenewableReferences: describes a conventional decoration of an endpoint reference with policy information needed to retrieve an updated version of an endpoint reference when it becomes invalid.
- WS-ServiceGroup: describes an interface to heterogeneous by-reference collections of Web Services.
- WS-BaseFaults: describes a base fault XML type for use when returning faults in a Web Services message exchange.

WSRF.NET is an implementation of the WSRF specifications on the Microsoft .NET platform [WSRF.NET 2004]. WSRF.NET consists of a set of libraries and tools that allows Web Services to be transformed into WSRF-compliant Web Services. Service authors annotate their Web Service code with metadata via .NET attributes and the WSRF.NET tools convert the original Web Service into a WSRF-compliant service. Authoring a service in WSRF.NET means annotating the service logic with meta-data through attributes, and using the WSRF.NET tools to create and deploy the WSRF-compliant Web Service.

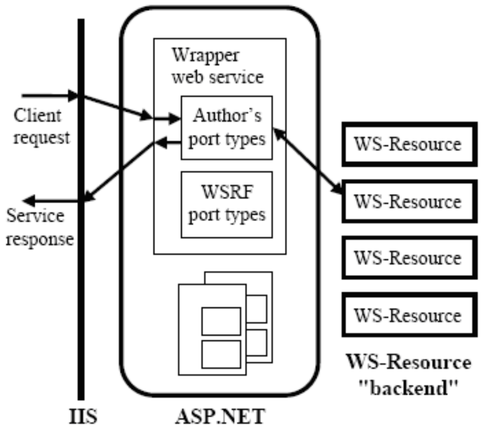

Figure 1: WSRF.NET architecture

4 Architectural approach

In this paper, we propose a novel WS discovery approach to facilitate knowledge reuse based upon qualitative characteristics and previous good practices. The followed architecture includes a service broker that acts as a mediator between providers and consumers. The broker is designed for QoS-capable servers that may provide information for different service quality levels to different clients depending on their requirements. The QoS broker manages a set of Web Services whose endpoints information and qualitative description stores. In specific, the QoS Broker searches the service registry to locate the Web Services available. Once the services are discovered, information such as endpoint, URI, function name and description are stored in QoS Broker's database based on their categorization.

As the end users of Web Services, consumers, send their service request (functional and non-functional (i.e. QoS) requirements) to the broker which selects the most suitable WS provider. The broker collects QoS information on the providers and when a client request arrives, identifies the service providers that can satisfy not only the operational but also the QoS needs based on a specific

QoS policy. The broker searches on a Web Service class in its database to find a qualified service and a service provider (server) that can allocate sufficient resources to achieve the desired quality. If more than one candidate is found, a decision algorithm is used to select the most suitable one. The QoS information will be used to make the decision.

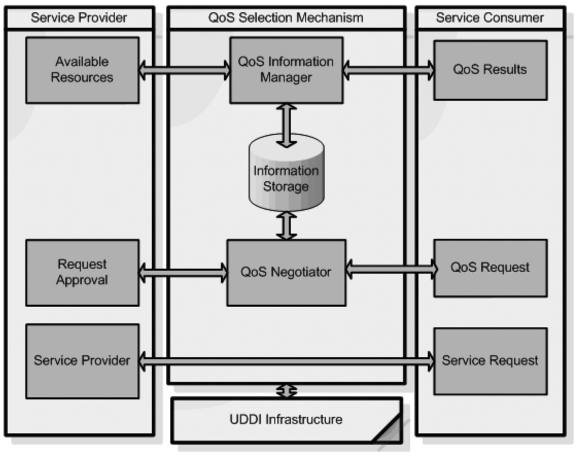

Figure 2: Selection Mechanism Overview

Web Service class is a collection of independent Web Services with common functionality and different QoS. Each service has maximum capacity Cmax, which is the maximum server resources that can be used to serve clients and the current capacity Ccur which is the currently available server resources. The service selection algorithm used by the broker in order to determine the most suitable service (among the services in of a WS class) decides based on the providers' server load. Other selection criteria could be cost, latency, response time etc.

The service providers have dynamic server load as clients come and go. The clients' arrival pattern and their resource requirements are unpredictable. If a server assigns excessive system resources to clients, new requests might not be accepted. Another possibility is the server to reduce the QoS provided to one or more clients in order to accept new requests. This reduction of QoS disturbs the clients' stability and is named QoS instability. Our system is modeled as a set of clients that invoke Web Services, enter the system in order to use resources and then leave, releasing the resources that had been allocated to them. An essential criterion in the selection of a Web Service is a new demand not to influence the activities of the existing clients, so that reconfiguration of resources is not required. The service selected by the broker should have available resources, that is Cmax - Ccur > 0. By sending a new service request to a server with light load, a client can be serviced faster than by a busy server. Moreover, this way the server load is distributed to more servers and the system is more balanced. This way, the selection algorithm returns the service with the highest availability that has max(Ccur). When the selection procedure is completed, the broker sends to the client the reservation ID, a Web Service endpoint, a URI, and a function. Then, clients can consume their reserved share of server resources.

In order to provide guaranteed service qualities, QoS information is retrieved using properly injected metadata into web service descriptions. Furthermore, feedback provided by the clients will help the broker to make even better decisions. Clients' feedback reports are used to update the broker's database, which is initialized using the WSDL QoS metadata. The QoS information is collected and send back to QoS Broker after each successful service invocation. The QoS data are stored in the broker's database as historical QoS information and will be used as selection criterion for future clients' requests. This feedback is important as it records the client's satisfaction.

In conclusion, the broker stores in it's database information on the Web Services such as maximum resources (Cmax), available resources (Ccur) and grades. Each grade represents the result of a linear combination of the clients' quality evaluations. When a client invokes a service, the broker sends the Web Service endpoint that is selected using the selection algorithm. Server resources are assigned to the client and when he is serviced he releases these resources and informs the broker for the current available resources.

Since services communicate by sending and receiving messages and data through the network, the transport of results from one service to other includes some delay. In this application we considered that the delay of transport and the cost between two services are predetermined and constant.

4.1 Selection Policies

In order to facilitate different possible business scenarios, we include in the architecture support for a number of different functional/operational modes. We followed a design and development strategy that allows the incorporation of different selection policies (logic) based on QoS information. The main aim was to provide an open infrastructure of a WS selection broker that could follow a separate selection approach depending on the class/category of the Web Service requested. For example, a corporate based WS broker instances may follow a specific x for WS that belong to corporate partners (where more accurate and exhaustive results are expected) and some different y selection policy for WS exposed by 3rd parties (where extra security precautions have to be met and reservation has perhaps to be followed).

We have designed, implemented and experimented with four (4) service selection policies, while more may be introduced without substantial architectural changes. A short presentation of the policies follows:

Selection Policy 1: Selection based on the availability

The most suitable Web Service is selected based on the availability of the Web Services managed by the broker; the Web Service with the highest current server capacity. In case of a request failure, no retries of the selection procedure are attempted and no service is selected, due to the high load of the available servers or the high consumer's demands.

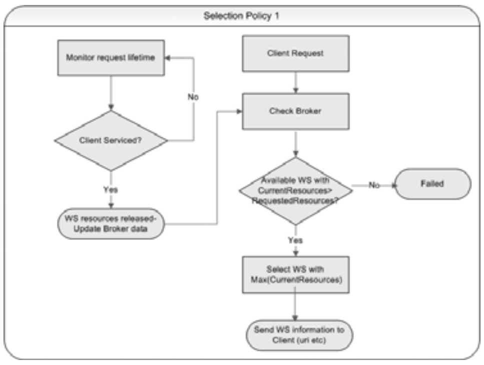
Selection Policy 1
Monitor request lifetime
Client Request
No
Check Broker
Client Serviced?
Available WS with CurrentResources> RequestedResources?
No
Failed
Yes
WS resources released-Update Broker data
Yes
Select WS with Max(CurrentResources)
Send WS information to Client (uri etc)

Figure 3: Selection Policy 1 Logical Diagram

Selection Policy 2: Selection based on the availability and QoS information

In the second selection policy, we take into account the QoS historical information based on the client's feedback. The broker searches among the available Web Services that have sufficient resources for the service with the higher grade. In case there is no available Web Service with sufficient resources to satisfy the client, the request is marked as failed.

Selection Policy 3: Provide resources below client's capacity request

The third policy is an extension of the second one that handles exceptions by reducing the resources provided to 75% of the resources requested. This demands QoS enabled providers. In this case, the quality of service ratings received by the end-user are reduced.

Selection Policy 4: Reservation based Selection using QoS information.

The forth selection policy provides an extended solution using reservations. When a client request fails, there is no available WS that can serve him. Then the WS with the highest grade and available resources is selected (among those whose total available resources are sufficient to serve the client). This WS resource is reserved and can not be assigned to any other client until it has enough additional

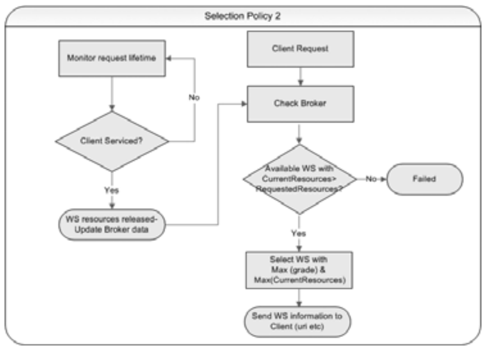
Selection Policy 2
Monitor request lifetime
Client Request
No
Check Broker
Client Serviced?
Available WS with CurrentResources> RequestedResources?
No
Failed
Yes
WS resources released-Update Broker data
Yes
Select WS with Max (grade) & Max(CurrentResources)
Send WS information to Client (uri etc)

Figure 4: Selection Policy 2 Logical Diagram

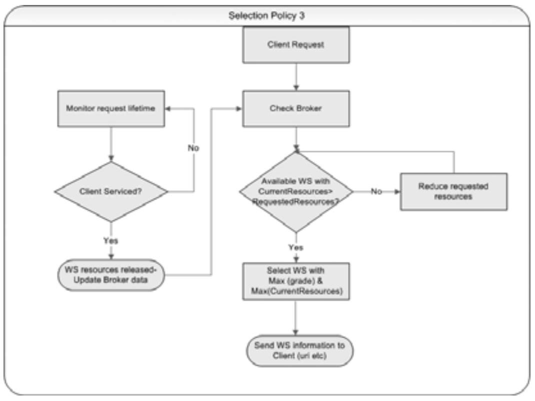
Selection Policy 3
Client Request
Monitor request lifetime
Check Broker
No
Client Serviced?
Available WS with CurrentResources> RequestedResources?
No
Reduce requested resources
Yes
Yes
WS resources released-Update Broker data
Select WS with Max (grade) & Max(CurrentResources)
Send WS information to Client (uri etc)

Figure 5: Selection Policy 3 Logical Diagram

available resources.

5 Implementation details

In our implementation, we use WSRF.NET to create and deploy the QoS broker in order to achieve maximum scalability and compatibility for the infrastructure. WSRF.NET is the first public release that leverages the WSRF specifications. It is an implementation of the WSRF specifications on the Microsoft .NET platform that consists of a set of libraries and tools that allow Web Services to be transformed into WSRF-compliant Web Services. WSRF.NET uses the ASP.NET architecture for Web Services. The choice of the .NET platform [.NET framework 2006] was made in order to utilize extensive previous experience of the available developers. However, generality is not lost as other Web

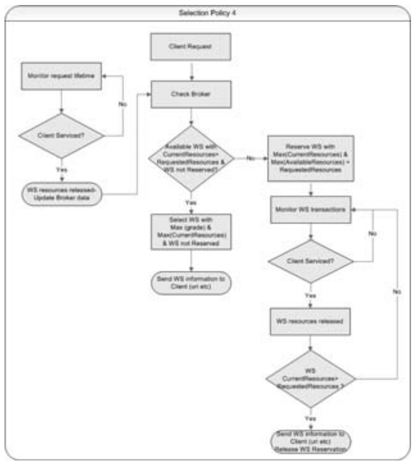
Selection Policy 4
Client Request
Check Broker
Monitor request lifetime
Client Serviced?
No
Yes
WS resources released-Update Broker data
Available WS with CurrentResources> RequestedResources & WS not Reserved?
No
Yes
Reserve WS with Max(CurrentResources) & Max(AvailableResources) + RequestedResources
Select WS with Max (grade) & Max(CurrentResources) & WS not Reserved
Send WS information to Client (url etc)
Monitor WS transactions
Client Serviced?
No
Yes
WS resources released
WS CurrentResources> RequestedResources ?
No
Yes
Send WS information to Client (url etc) Release WS Reservation

Figure 6: Selection Policy 4 Logical Diagram

Service enabling technologies may be used without any design changes on the proposed solution.

Authoring a service in WSRF.NET consists of three steps, 1) creating the service logic, 2) annotating the logic with meta-data through attributes, and 3) using the WSRF.NET tools to create and deploy the WSRF-compliant Web Service.

In order to find Web Services efficiently, we categorize the heterogeneous Web Services into different classes according to their functionality. Service availability is the probability that the service is available when a client invokes it. This only measures the server availability in terms of responding to a request, not the result quality.

The WSRF.NET stores the WS Resources in a database and the resources are automatically loaded from a WSRF.NET Web Service when a new request arrives. The WS-Resource Properties specification includes a set of port types that can be used by the clients in order to read and manage the Resource Properties of a service. In specific, these portTypes data can be retrieved, updated and aggregated.

The WS-Notification specification defines the port type Notification Producer to manage the subscription. The broker subscribes to the Notification Producer in order to be notified about the remaining Web Service resources when a new successful client request allocates server resources. The broker implementation includes a server thread that receives asynchronous notification messages using the WSRF.NET Asynchronous Notification Listener class.

The broker's WS-Resources represent the available Web Services. Each WS-

Resource has attributes such as the current availability and information on the last client and the amount of resources that was assigned to him. The destruction of a WS-Resource represents the removal of a Web Service from the service class.

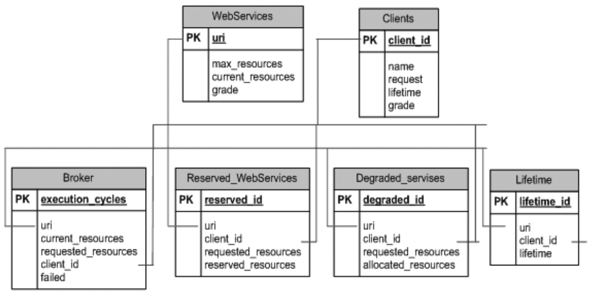

Figure 7: ER highlighting sketch of the Selection Module

5.1 Back-end Selection Module Entities

The proposed WSRF based selection algorithm needs only an limited database extensions to operate its basic functionality. In Figure 7 the entities participating in our solutions are depicted together with their interconnection relationships. In particular, *WebService* table stores information concerning the WS that the mediator has knowledge about at the moment. This table is updated with partial UDDI information and acts as a local replicating proxy. It includes:

- WSDL URI or Web Service key (The WS identifier that may be used by potential WS consumers), and
- Maximum resources available for consumers.

The two extra properties are utilized to store publisher's initial evaluation score and initial resource value.

Clients table records information concerning the incoming request to be handled by the broker:

- requested amount of resources,
- WS time duration while executing client request,
- feedback value about the Web Service upon job completion.

Broker table stores results of broker activity. For the incoming client requests for WS selection and consumption information is stored such as the number of successful and dropped requests as well as WS resource deallocation rates. It includes:

- the Web Service identifier selected for a specific consumption request,
- the client identifier, which made the request,
- number of resource allocation requested,
- current amount of resources for the WS selected and
- type of activity [successful, dropped, deallocation].

There are also three additional entities to facilitate the selection processes. The *Reserved_WebServices* table is utilized in the reservation based selection policy. It implements a buffer that records reservations for WS resource allocation. *Degraded_WebServices* table facilitates the selection policy where resource allocation is automatically degraded. It stores details for the client requests that received a WS selection with less resources than expected-requested. Finally, *Lifetime* table acts as a buffer for the WS execution duration before resource deallocation.

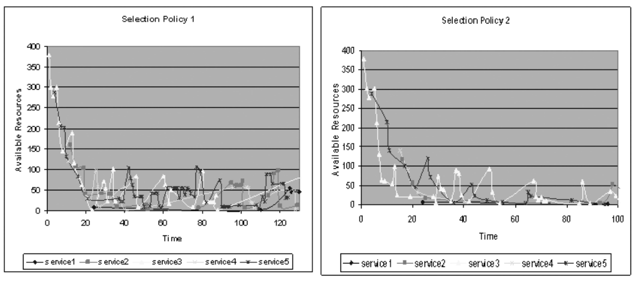

Figure 8: Selection Policy 1 and Policy 2

6 Experimental Results

We have implemented the broker framework to demonstrate its feasibility and efficiency. Development, implementation and experimental evaluation involved Pentium IV machines using 1GB RAM. The whole system is developed using MS .NET framework v.1.1 tools [.NET framework 2006] and C# language.

The experiments have been designed to satisfy cases and scenarios that have been derived while studying and working on web services for different enterprises

Table 1: Experimental Resources

WS Name	WS ax Resources
service1	61
service2	166
service3	423
service4	194
service5	289

and telecommunication carriers. In particular the mechanism has been evaluated for cases of processes which involve telephone service billing records and produce online bill reporting to "live" un-processed recorded transaction logs.

The testbed configurations and the experimental results are presented in the following. The experiments have been conducted to measure the performance of the broker in operational mode under the different selection policies followed. We have tested the broker using a single policy enabled at a time. The graphs below depict the experimental results:

1. Availability graphs (available system resources) of the Web Services managed by the broker versus time, for each one of the selection policies

2. Comparative graphs of the selection policies- number of serviced clients versus time. The initial server resources were:

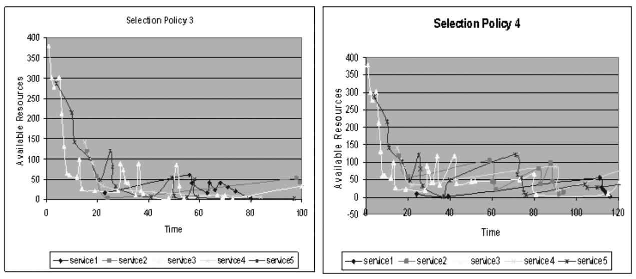

Figure 9: Selection Policy 3 and Policy 4

Figure 1 shows the availability of services versus time, while clients allocate and release resources based on selection policy 1. Web Service 1 which has the highest resource availability is used initially for all client requests until it is evened with the second highest. Next, according to the resources allocations and deallocations we select the Web Service with the highest availability.

Table 2: Experiments' Results

Selection	Unsuccessful	Successful	Time	Decreased Qual	Failures
Policy 1	33	67	136		33
Policy 2	47	53	114		47
Policy 3	47	53	115	5	47
Policy 4	0	100	149		48

Figure 2 shows Web Services' availability using selection policy 2. The graph in this case differs from the previous because during selection we take into account the grade. Figures 3 and 4 show Web Services' availability, using selection policy 3 and 4 respectively. In total, the experiments's results are shown in table 2.

The column "Decreased Quality" contains the number of customers that received less resources than requested (selection policy 3), column "Num. of Failures" is the number of failed tries and "Time" represents the clock cycles. Notice that when we take into account the client provided feedback concerning QoS, fewer clients are serviced in policy 2. We have to underline that the last policy is the only one that achieves to process all requests fulfilling all the QoS requirements by using the reservation features.

In figure 6 we see that initially the four selection policies have the same behavior up to the time moment "20" where they begin to be differentiated, (much faster than in the lower demand requirements case). This implies that the system is at an initial stage, where no request "'conflicts"' happen.

Overall, the experimental results confirmed the efficiency of the proposed architecture. We have also proposed an effective reservation QoS aware brokering policy. The functionality of the proposed WSRF.NET based WS broker was verified using different selection policies.

7 Conclusions and Future Work

The key aim of our solution is to drive knowledge sharing augmenting the existing web service technologies with ontology architectured qualitative characteristics that are possible to be dynamically updated. The target is to provide a competitive advantage to organizations and service consumers in general through quality evaluation. The proposed solution is a mediator that utilizes domain knowledge to provide better service consumption experiences and promote web services that implement good practices (that is marked with high reputation). In particular, we have presented a novel dynamic service knowledge management mediator based on the Web Services resource framework (WSRF). In this way, we enable for the first time *dynamic* knowledge management for Web Service non functional specifications (QoWS). In real world enterprise environments, quality of service requirements have to be met [Cepeda Carrion 2006]. As a result, the proposed mediator takes advantage of previous domain knowledge, that is quality information stored upon web service description combined also by user feedback. Our design supports different operational modes and it is fully scalable for different organizational configurations. Test-bed implementation includes four different selection policies, which have been presented and experimentally compared. The mechanism may be configured to work according to any of the selection policies as well as include more selection policies if necessary without architectural side-effects. Policies' performance in terms of requests serviced/time and requests dropped/time differs as expected. For the Quality of Service competitive world, we have proposed policies that takes advantage of QoS information (metadata and feedback) to best match among the available services. However, only when we enable the feature of reserving resources it is possible to fulfill all the incoming requests as a whole (introduced policy 4).

Future work includes the investigation of additional selection policies on the mediator mechanism in order to automate composition processes using reusable web service objects [Sicilia and Lytras 2005]. Moreover we look into the adjustment of the proposed strategy to support a peer to peer overlay service registry environment. Topic knowledge [Panagis et al 2006] can be also useful in order to further enhance web service matching using properties (tags) of the up-coming social-web Web 2.0. Further research will also focus on efficient adaptation of the mechanism to support recursive use and workflows of hypermedia services [Pandis et al 2005].

References

[Berners-Lee et al 2001] Berners-Lee, T., Hendler, J., Lassila., O.: "The Semantic Web"; Scientific American, 284, 5, (May 2001), 34-43.

[Cepeda Carrion 2006] Cepeda Carrion, G.:"Understanding the link between knowledge management and firm performance: articulating and codifying critical knowledge areas"; Int. J. of Knowledge and Learning 2,3/4 (2006), Inderscience Publishers, 238-262.

[Dobson et al 2005] Dobson, G., Lock, R., Sommerville, I.: "QoSOnt: a QoS Ontology for ServiceCentric Systems"; Proc. of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, 80-87.

[Garofalakis et al 2006] Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis A.: "Contemporary Web Service Discovery Mechanisms"; Journal of Web Engineering 5,3 (September 2006), Rinton Press, 265-290.

[Karousos et al 2003] Karousos, N., Pandis, I., Reich, S., Tzagarakis, M.: "Offering open hypermedia services to the WWW a step-by-step approach for developers"; Proc. of the WWW International Conference, 2003, 482-489.

[Liu et al 2004] Liu, Y., Ngu, A., Zeng, L.: "QoS computation and policing in dynamic web service selection";Proc. of the 13th International Conference on World Wide

Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, (May 2004), 66-73.

[Lytras and Sicilia 2005] Lytras, M.D., Sicilia, M.A.:"The Knowledge Society: a manifesto for knowledge and learning"; Int. J. of Knowledge and Learning 1,1/2 (2005), Inderscience Publishers, 1-11.

[Makris et al 2005a] Makris, Ch., Sakkopoulos, E., Sioutas, S., Triantafillou, P., Tsakalidis, A., Vassiliadis, B.: "NIPPERS: Network of InterPolated PeERS for Web Service Discovery"; Proc. IEEE International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, Las Vegas, Nevada, USA, 2005, 193-198.

[Makris et at 2005b] Makris, Ch., Panagis, Y., Sakkopoulos, E., Tsakalidis A.: "Efficient and adaptive discovery techniques of Web Services handling large data sets"; J. of Systems and Software 79,4 (April 2006), 480-495.

[Maximilien and Singh 2004] Maximilien, E.M., Singh, M.P.: "A Framework and Ontology for Dynamic Web Services Selection"; IEEE Internet Computing, 8, 5, (Sept 2004), 84-93.

[Ordonez De Pablos 2006] Ordonez De Pablos, P.: "Knowledge flows and learning at interorganisational level: implications for management in multinational corporations"; Int. J. of Knowledge and Learning 2,1/2 (2006), Inderscience Publishers, 58-72.

[Ouzzani and Bouguettaya 2004] Ouzzani, M., Bouguettaya, A.: "Efficient Access to Web Services"; IEEE Internet Computing 8,2 (2004), 34-44.

[Panagis et al 2006] Panagis, Y., Sakkopoulos, E., Garofalakis, J., Tsakalidis, A.: "Optimisation mechanism for web search results using topic knowledge"; Int. J. of Knowledge and Learning 2,1/2 (2006), Inderscience Publishers, 140-153.

[Pandis et al 2005] Pandis I, Karousos N, Tiropanis, T.: "Semantically annotated hypermedia services"; Proc. of the 16^{th} ACM Conf. Hypert., Salzburg, Austria, 2005, 245-247.

[Ran 2003] Ran, S.: "A Framework for Discovering Web Services with Desired Quality of Services Attributes"; Proc. of the International Conference on Web Services, ICWS '03 (June 2003), Las Vegas, Nevada, USA, 208-213.

[Russ and Jones 2006] Russ, M., Jones, J.K.:"Knowledge-based strategies and information system technologies: preliminary findings"; Int. J. of Knowledge and Learning 2,1/2 (2006), Inderscience Publishers, 154-179.

[Sakkopoulos et al 2006] Sakkopoulos, E., Kanellopoulos, D., Tsakalidis, A.: "Semantic mining and Web service discovery techniques for media resources management"; Int. J. Metadata, Semant. Ontol. 1,1 (2006), 66-75.

[Sicilia 2006] Sicilia, M.A.:"Metadata, semantics, and ontology: providing meaning to information resources"; Int. J. Metadata, Semant. Ontol. 1,1 (2006), 83-86.

[Sicilia and Lytras 2005] Sicilia, M.A., Lytras, M.D.:"Scenario-oriented reusable learning object characterisations"; Int. J. of Knowledge and Learning 1,4 (2005), Inderscience Publishers, 332-341.

[SOA 2004] "Web Services Architecture"; W3C, (2004) `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211`

[Tian et al 2004] Tian, M., Gramm, A., Ritter,H., Schiller,J.: "Efficient Selection and Monitoring of QoS aware Web services with the WS QoS Framework"; Proc. of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), Beijing, China, 2004, 152-158.

[Yu and Lin 2004] Yu, T., Lin, K.J.: "Service Selection Algorithms for Web Services with End-to-End QoS Constraints";Proc. of the IEEE International Conference on E-Commerce Technology (CEC 2004), (July 2004), San Diego, CA, USA, 129-136.

[Wiil and Leggett 1992] Wiil, U. Leggett, J.: "Hyperform: Using extensibility to develop dynamic, open and distributed hypertext systems"; Proc. 4^{th} ACM Conf. Hypert., ACM, New York (Nov 1992), 251-261.

[WSRF 2004] "The WS-Resource Framework"; IBM (2004) http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf

[WSRF.NET 2004] Wasson, G.: "WSRF.NET 2.0 Programmer's Reference"; (2004) http://www.cs.virginia.edu/~gsw2c/WSRFdotNet/WSRFdotNet.htm

[.NET framework 2006] ".NET framework"; MSDN (2006) http://www.microsoft.com/net/