Web Services Procurement Based on WSLAs

Giner Alor-Hernandez [1], Juan Miguel Gomez [2]

[1]Division of Research and Postgraduate Studies
Instituto Tecnologico de Orizaba.
Av. Instituto Tecnologico 852, Col Emiliano Zapata. 09340 Orizaba, Veracruz, México.
e-mail: galor@itorizaba.edu.mx

[2]Departamento de Informática
Escuela Politécnica Superior, Universidad Calos III de Madrid.
e-mail: juanmiguel.gomez@uc3m.es

Abstract. This paper describes a framework for providing differentiated levels of Web services to different customers on the basis of service level agreements (SLAs). Under the framework described in this paper, service providers can offer Web services at different service levels. In general, the service levels are differentiated based on many variables such as responsiveness, availability, and performance. The framework comprises the Web Service Level Agreement (WSLA) language to specify SLAs in a flexible and individualized way, a system to monitor the compliance of a provided service with a service level agreement, and a workload management system that prioritizes requests according to the associated SLAs.

1 Introduction

A Web service is a software component that is accessible by means of messages sent using standard web protocols, notations and naming conventions, including the XML protocol [1]. The notorious success that the application of the Web service technology has achieved in B2B e-Commerce has also lead to consider it as a promising technology for designing and building effective business collaboration in supply chains. Deploying Web services reduces the integration costs and brings in the required infrastructure for business automation, obtaining a quality of service that could not be achieved otherwise [2], [3]. Therefore, Web services offer a new way for the development of distributed applications which can integrate any group of services on the Internet into a single solution. It may involve, possibly, the use of web services provided by different organizations, cooperating in complex collaborations. Thus, there is a need of agreements in order to establish the obligations to both sides, i.e. customers which use Web services and providers which supply them. Commonly, these agreements are defined by using Web Service Level Agreement (WSLA) Language. A WSLA document defines assertions of a service provider to perform a service

© *A. Gelbukh, S. Torres, I. López (Eds.)*
Advances in Computer Science and Engineering
Research in Computing Science 19, 2006, pp. 15-28

according to agreed guarantees for IT-level and business process-level service parameters such as response time and throughput, and measures to be taken in case of deviation and failure to meet the asserted service guarantees [4]. The assertions of the service provider are based on a detailed definition of the service parameters including how basic metrics are to be measured in systems and how they are aggregated into composite metrics [5]. In addition, a WSLA expresses which party monitors the service, third parties that contribute to the measurement of metrics, supervision of guarantees or even the management of deviations of service guarantees [6]. Interactions among the parties supervising the WSLA are also defined. Having this into account, we have developed a framework for Web Services procurement which provides needed functionalities to business services and allows developers of business services to focus on their business domains rather than on support issues. Our framework features provisioning services, such as contracting, metering, accounting, notification and SLA based management of web services.

The rest of this paper is structured as follows. In the next section we present the general architecture and its main components of the framework for Web services procurement. In the following sections, we discuss the functionality of each component of the service Hub which is the main component in our architecture and discuss their relationships among them. Next, we present a scenario for Web services procurement among services requestors and providers. Then we describe the future directions and review the related work. Finally, we emphasize the contributions of our work.

2 Architecture

Our architecture was designed to show how to extend a Web Service by adding common provisioning services that most business infrastructure will need. Under our architecture, there are four steps that occur during the processing to carry out the procurement of Web services:

1. A Business Service (Web Service) is hosted on a machine. This service is registered with a Service Hub.
2. A Hub administrator will create an Offering package. An Offering is a contract that specifies what service is being made available and at what performance levels.
3. A Service Requestor will then choose an Offering package, thus creating a Usage Contract - agreeing to the terms of the contract - including the billing terms.
4. Once the Usage Contract is activated the requestor is now free to use the Service.

Fig. 1 shows a multi tier configuration of our framework for procurement of Web services. Below is a high-level view of the configuration. In Fig.1, the first tier is the Service Requestor (Client). In this tier, an initial SOAP request is generated which is sent to the second tier - a Service Provider. This message passes through an Axis handler which places the identity of the Requestor into the SOAP message so that the

Service Provider can properly identify who is trying to use the service. In the second tier, the service Hub, acts as a gateway to the services being offered. A hub administrator is responsible for registering and managing the services that are available from this service hub. The service hub has the responsibility of invoking all of the provisioning services and then ultimately routes the request to the desired Supplier to actually do the business logic of the Web Service. The internals of the service hub are shown in Fig. 2.

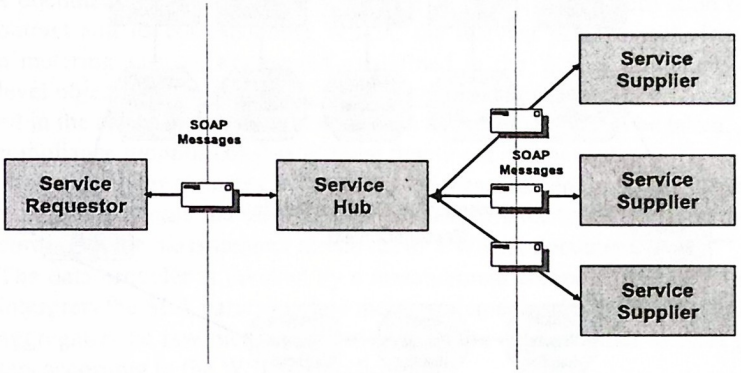

Fig. 1 A multi tier configuration of the framework for Web services procurement

When a SOAP message enters the service hub (as shown in Fig. 2), the Axis servlet passes it through a series of handlers responsible for interacting with each of the provisioning services. First a profile handler uses the Profile Service to validate the Service Requestor identity and gets its unique profile key. This key is stored in the SOAP message context so that it can be accessed by other handlers. Then, the contract handler invokes the Contract Service to verify that the service requestor has a valid Usage Contract and places the contract ID into the message context. Next, the metering request handler generates a start metering event, which is used for accounting purposes. The management request handler loges the request for statistical purposes. Next, the request is processed by the Web Services Management Middleware (WSMM) handler. This handler takes the contract ID from the message context and retrieves the WSLA performance expectation data from the contract. Based on this data, plus the load on the machines hosting the business services, the WSMM handler determines when to allow the request to continue down the chain of handlers. Finally, the message is passed to the Service Desk handler which routes the request to the proper machine hosting the Business Service. The WSMM handler acting in conjunction with the Service Desk acts as a load balancing mechanism - to help ensure all of the performance criteria expected by the contracts are met.

On the output side, the response from the Business Service is processed by the Metering, Management and WSMM Response handlers. Each one uses its specific service to make a note of the completion of the Business Service's processing. The response message is returned to the Service Requestor.

The third tier, the Service Supplier, is a machine hosting the actual Web Service. This machine does not need any special set-up beyond the normal Web Services configuration (SOAP server and the Business Service itself). The Hub machine must be aware of its existence in order for this supplier can participate. A hub administrator is responsible for handling this as part of the process of managing the services available from the service hub.

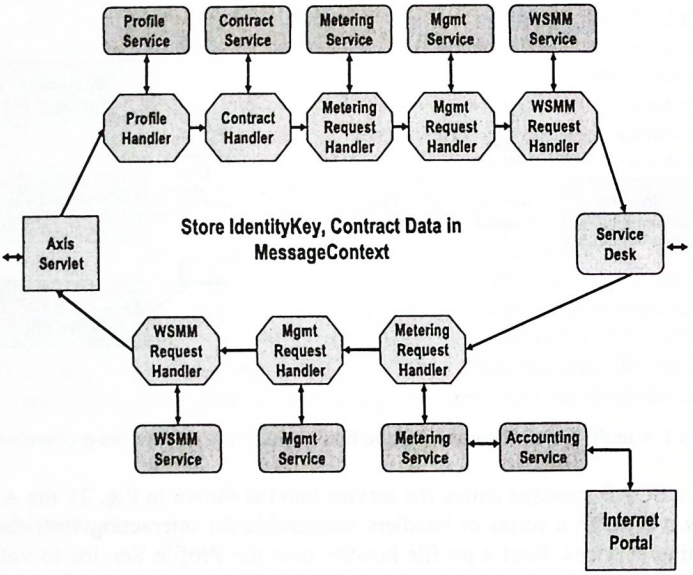

Fig. 2 Internals of the service hub

Managing the services available from the service hub involves registering the interface for a service that is available and then registering any Suppliers that actually provide the implementation for this service interface. Once a service is registered, it can be made available to requesters through either fixed of flexible offering packages. A service can be used as long as there are any Service Suppliers registered that provide the service. The service hub manages which Supplier handles requests for a service.

Based on this understanding, our approach is based on offerings and usage contracts. An offering is created by the Hub administrator and indicates which business services are available to a service requestor and what performance guarantees it is offering for that service. The service requestor establishes a usage contract before using a business service. Furthermore, the user profiles for the service requestor and hub administrator must be registered.

In the next section, we describe with more detail the functionality of each internal from the service Hub and their relationships among them.

3 Internals of the Service Hub

3.1 Compliance Monitor

The Compliance Monitor supervises whether the service level objectives specified in a WSLA document are met and raises an alarm otherwise. Upon activation of a new usage contract and its corresponding WSLA, the relevant performance data is read from the metering service, aggregated as defined in the WSLA document and the service level objectives are evaluated. If violations or other relevant conditions occur, as defined in the action guarantees, actions such as notifications can be taken.

The compliance monitor consists of three functional components:

1. A data provider connects to the source of measurement, e.g., the instrumentation or in our case the measurement service, and reads raw measurements according to the measurement directives of a WSLA document.
2. The data provider is invoked by a measurement component. This component interprets the SLA parameter and metric statements of a WSLA document and aggregates the raw metrics as retrieved by the data provider to SLA parameters according to the WSLA specification.
3. New SLA parameter values are forwarded to the compliance monitor component. This component evaluates the conditions of the service level objectives and executes the activities defined in the action guarantees of the WSLA document, typically sending notifications to the notification service.

The compliance monitor provides the following operations:

- **add** - add a WSLA to be monitored
- **remove** - stop monitoring a WSLA and remove it
- **getActive** - get the list of currently monitored WSLAs

3.2 Web Services Management Middleware (WSMM)

WSMM is a feedback control mechanism for transparent performance management of web services, in a way that maximizes expected business value in the face of service level agreements (SLAs) and fluctuating offered load. The main objective of WSMM is to provide support for differentiated services based on Service Level Agreements (SLAs). The introduced mechanisms enable service providers to offer the same web service at different performance levels (e.g., different average response time thresholds), in a way that is transparent to the service and client developers.

WSMM performs resource allocation, scheduling, and overload protection by means of a collection of real-time mechanisms, which are applied to individual requests, as well as slower time scale optimization and coordination mechanisms.

3.3 Service Desk

Web Services binding, interoperability, sharing, and aggregation of multiple heterogeneous Web Services are key integration problems. We have used Service Desk technology [7] that provides intelligent Web Services clustering. Service desk is a specific service domain object described in WSDL documents that represents the basic processing unit of a collection of services. Service Desk technology allows create, cluster, organize, route, recover, and switch Web Services in an autonomous way [7]. The cluster can represent a group of comparable or related services through a common services entry point, in fact a service grid. Subsequently, responsive to the receipt of service requests, the grid can select suitable ones of the computing services instances to process the received service requests, monitor the performance of the selected instances, and perform fail-over processing if required. The selection is according not only to availability, but also according to QoS characteristics, as specified via WSLAs, and business arrangements. The operation is automatically performed based on a service policy. The set up process follows the eUtility model [8] of creation of service offering, and customer subscription, for both the service desk clients and service desk suppliers. This technology demonstrates an integration benefit of Web Services, Autonomic computing and Grid computing utilized as a whole.

3.4 Metering Service

The Metering Service receives meter events from clients and provides meter events upon request. The Accounting Service gets metering information from the Metering Service and contract information from the Contract Service and uses this to produce a usage report for a particular client using a particular service. The Metering Service supports three types of WSDL-defined operations from a client: (1) recordMeterEvent, (2) recordMeterEvents, and (3) getMeterEvents.

Metering is possible on an operation level. Meter events contain the service name and the operation name of the service that was called, timestamps, as well as the id of the contract used to handle the request. Meter events vary by type so various ways of charging a service call are possible (specified in the service contract set up by the Contract Service):

- Start/end events are used when access to a service is charged by the amount of time used to perform the service;
- Ad-hoc events are used when access is charged for by the number of times that the service is accessed, or on some other basis besides time.

In addition to the above, two more types of events are available: cancelled, which is used to cancel an event which has already been sent to the metering service, and unknown, which is used when the type of event was not supplied by the service requestor.

3.5. Contract Service

The Contract Service handles the relationship between service providers and service requestors. It provides information about the type of contract between a service provider and the service hub (deployment contracts) and between a service requestor and the service hub (usage contracts). Usage contracts can be used to subscribe to any combination of operations of any service provided through the service hub. They can include fixed services, or allow for services to be added or removed over the life of the contract. A usage contract contains information such as how calls to service operations are to be charged for (by time, by number of uses, among others) and how much the subscribed service operations should cost for that client. For each usage contract the Contract Service defines the payment model and rating model to be used, the effective dates for that contract. Contracts may optionally store the digital signatures of both parties (service hub and service provider/requestor) to the contract. Under our approach, contracts are added to the Contract Service via our framework and a valid contract must be in place between a service hub and a service requestor before the requestor can use the service. The Contract Service supports WSDL-defined operations such as the following: (1) createContract, (2) getContractModel, (3) getContractState, (4) updateConstractState, (5) getContractType, (6) setContract-Property, (7) getContractProperty, and (8) getUsgaeContractsValidForIdentity.

3.6 Accounting Service

The Accounting Service is used to calculate billing data according to rating models, using provider contracts and corresponding meter events as input. A rating model describes the pricing scheme for the service, and can be implemented according to a service provider's specific requirements and plugged into the accounting service.

3.7 Profile Service

The Profile Service provides access to user profile information for a user. Basic profile information is collected and supplied by this service, including name, address, user id, to mention a few. In time, this may expand to include more information. The service requestor saves and gets profile information using the Profile Service. The profile key provided by the Profile Service is used by the Contract Service to determine what users have valid contracts with a service provider. In this context, all users of business services must have a profile assigned by the Profile Service. Profiles may be created in advance by using the framework for Web Services Procurement.

To illustrate the functionality of our implementation, we describe next a scenario for Web services procurement that integrates services requestors and providers that has already been implemented.

4 Case of Study

The case of study describes a basic scenario which involves simple accounting information associated with a single web service.
Suppose the following scenario:

1. A services provider, who is a book seller (like Amazon) brings access to its database by means Web services interfaces. The functionalities provided by these interfaces are to get the stock quote, price, availability and other technical features of its products.
2. A client wants to create a virtual enterprise through a services provider. The service provider should offer their product catalogs to the client for the development of the virtual enterprise.

In this scenario, how can client find to the service provider and establish a usage contract with him to carry out the development of the virtual enterprise?

To solve this issue is necessary to use our framework. Firstly, is necessary register the Web services interfaces provided by the services provider within Service Hub. For doing this, our framework present a set of graphic interfaces where new services can be added to view information about the types of services that have already been defined. Under our framework, if a client wants to add her own application there are just a few simple steps to follow:

1. Deploy the service along with an interface WSDL document that defines the service interface and an implementation WSDL for the new service. The WSDL documents must be accessable through a URL.
2. Create an HTML page (e.g. JSP or servlet) that can be used to access and invoke the service. This HTML page must accept the following parameters:
   a. **wsdl** - a URL to the WSDL document describing the service
   b. **namesapce** - the targetNamespace to use in the WSDL document
   c. **servicename** - the service name to use in the WSDL document
   d. **portname** - the port name to use in the WSDL document

In a similar way, we need to register at least one service provider which provides an implementation of this service. In Fig. 3, a screenshot to add new services in the Service Hub is shown. Once the service and service provider have been registered, is necessary to establish an offering for this service. In this sense, the service provider must create a service offering -- fixed packages including service operations, associated service levels, penalty upon violation, as well as price for using this service -- expressed as a SLA template. A screenshot of our framework where offerings are created is shown in Fig. 4. Next, the client (a service requestor) in order to use this service she must first create a Usage Contract agreeing to the terms and conditions specified in it. Then, she must subscribe to a selected service offering creating a new SLA. The service provider may provide some customization flexibility in its offerings. The customization capability may range from mere selection of a few SLA parameter values (e.g., from a set of fixed throughput levels) as expressed in an offer, to some negotiation of parameters (e.g., negotiation of price for a customer-specified throughput level) to composing new service level objectives. To provide this flexibil-

ity, a provider should not only have the required capability of cnline negotiation, but also its business ability to support any new customer-specified service level objectives (SLOs), i.e., runtime infrastructure for supporting this service level as well as its ability to price this new service level. Therefore, before accepting a new SLA, the provider must ensure its ability to support this new SLA. In Fig. 5, a screenshot to create usage contract is shown. Once the usage contract is established, the client can invoke the service by using our framework. The framework provides Web services dynamic invocation by creating GUIs for consuming the service. The process of Web services invocation is carries out by analyzing WSDL documents. WSDL documents employ XML Schema for the specification of information items either product technical information or business processes operations. Our framework reports the business processes operations, input and output parameters, and their data types in a XML DOM tree which is a XML document. This XML document is presented in HTML format using the Extensible Style-sheet Language (XSL). In Fig. 6, a screenshot for Web services invocation is shown. In this figure, the stock price is displayed given a product code. Each time this Web service is invoked by a service requestor, our framework gathers data which contain accounting information for this usage contract. Among the gathered data are: (1) invocation date, (2) basic price, (3) unit price normal and (4) unit price reduced of usage. With these data, our framework can generate a table where the accounting and billing details for the selected user contract are listed. A screenshot for accounting and billing information is shown in Fig. 7.

Through our framework, the client could find a service provider who offers their product catalogs and provides operations to get the stock quote, price, availability and other technical features of its products. By means of our framework, the client established a usage contract with the service provider and obtained the accounting/bill generation for this service.

As could be observed, our framework demonstrates the various roles and steps that make up the lifecycle of managing and hosting a Web Service, starting with offering up the service to potential requestors, using the service and finally ending with the accounting/bill generation for specific services.

5 Future Directions

As future work, we are considering include management for composite Web services. A composite Web service is one that uses other web services in addition to its own business logic, to fulfill its clients' requests. The underlying Web services may also be composite, thus leading to complex chains of service-to-service interactions. Each service in the composition hierarchy may be independently owned and operated, so that each request must be metered and charged to its immediate client. In addition, it may contribute towards the accounting for a higher-level request in the hierarchy. Thus, the accounting process must include correlation and aggregation of metering data. In a composite service scenario, each web service may potentially be hosted on a separate Service Hub, resulting in a distributed deployment. A request to one service may result in multiple requests to other services on different Service Hubs. The

service usage reported by each such *child* request needs to be correlated with the *parent* request that caused it, and aggregated together to compute the composite usage of the *parent* request. Since each service is potentially a composite service, each *child* request may itself act as the parent of other *child* requests. Thus, the correlation and aggregation must be performed recursively, to cover the entire web service call graph generated by an end-user request.

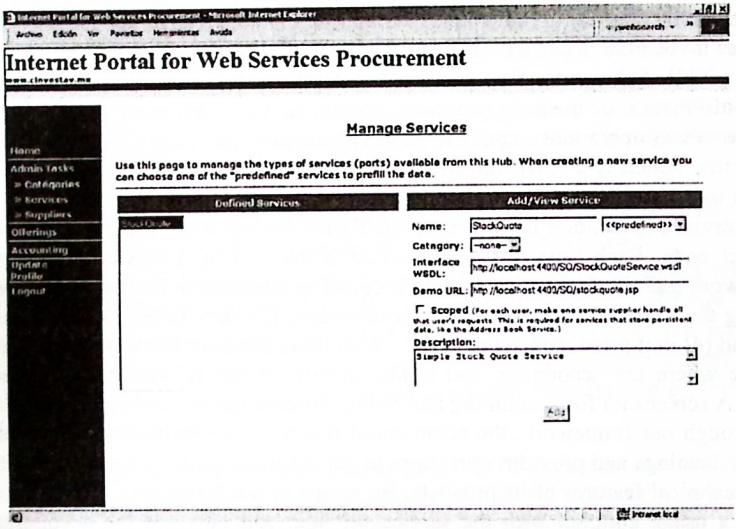

Fig. 3 Graphic interface to add new Web services interfaces within Service Hub

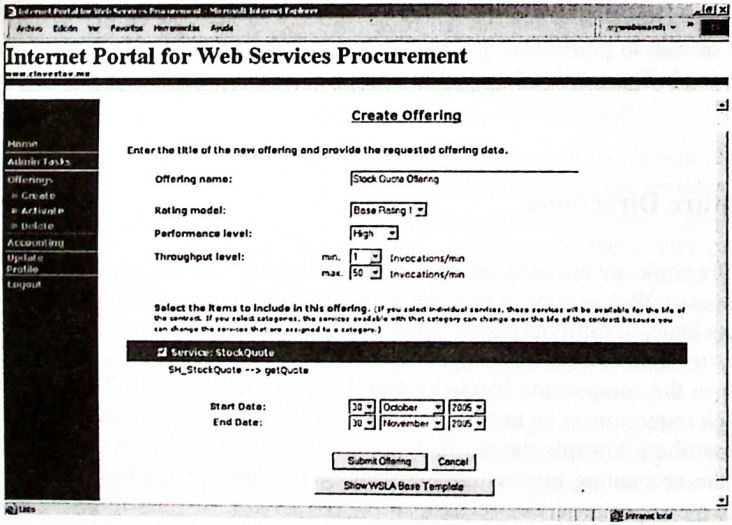

Fig. 4 Graphic Interface where offerings can be created by services providers

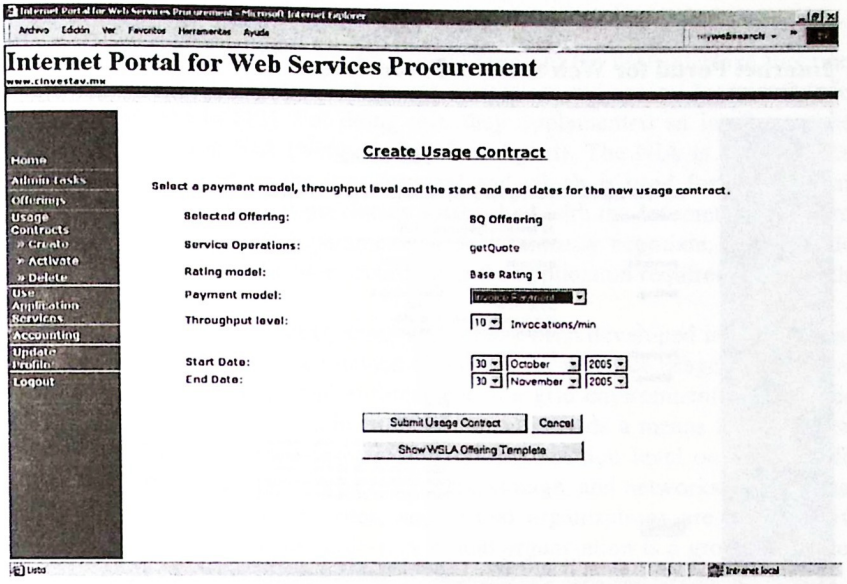

Fig. 5 Graphic Interface where usage contract can be created by services requestors

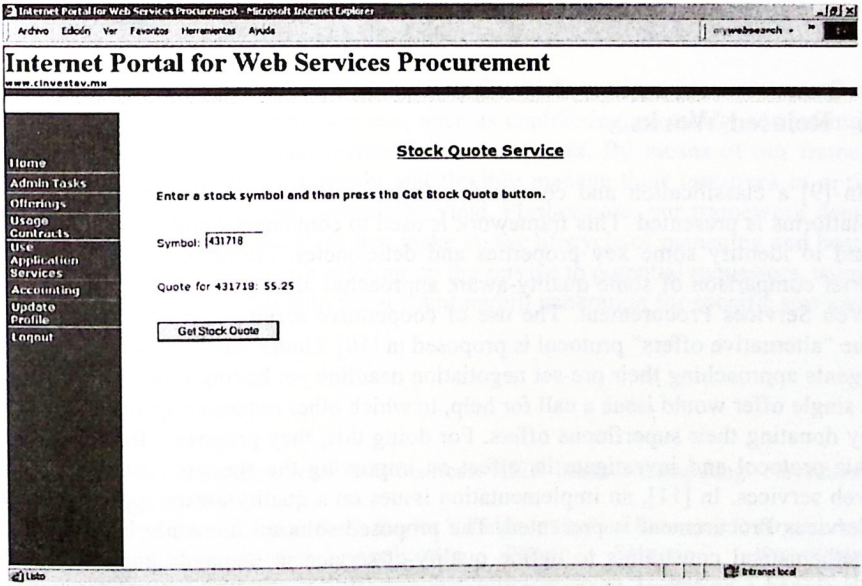

Fig. 6 Graphic Interface for invoking the Web service that get the stock quote given a product code

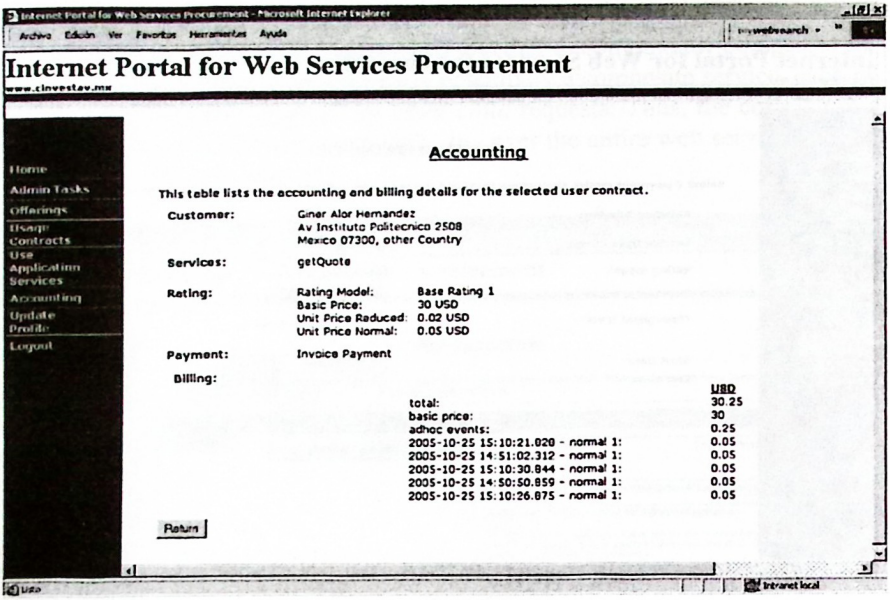

Fig. 7 Graphic Interface for accounting the Web service that get the stock price

6 Related Works

In [9] a classification and comparison framework for Web Services Procurement platforms is presented. This framework is used to compare several existing platforms and to identify some key properties and deficiencies. Furthermore, they present a brief comparison of some quality-aware approaches among different frameworks for Web Services Procurement. The use of cooperative service requester agents within the "alternative offers" protocol is proposed in [10]. Under this idea, service requester agents approaching their pre-set negotiation deadline yet having failed to secure even a single offer would issue a call for help, to which other requester agents will respond by donating their superfluous offers. For doing this, they propose a formal model of this protocol and investigate its effect on improving the success rates in procuring web services. In [11], an implementation issues on a quality-aware approach to Web Services Procurement is presented. The proposed solution is mainly based on using mathematical constraints to define quality-of-service in demands and offers. They developed a prototype of the run-time framework for management and execution of multi-organizational web-based systems. This prototype includes a quality trader web service as the main component, which offers services such as checking for consistency and conformance, and searching for the best choice. Among the main characteristics of implementation, they used the QRL language to specify quality-of-service, and XML to specify QRL-based documents, the definition of XSLT transformations

to get the appropriate CSP for carrying out the WSP-related tasks, and the use of a constraint solver as ILOG's OPL Studio. An extension to the user interface functionality to the dynamic SLA negotiation between a customer and several Internet Service Providers is proposed in [12]. For doing this, they implemented an intelligent user interface which is called NIA (Network Interface Agent). The NIA is a multi-agent system which is installed on the user terminal and which is used for the dynamic negotiation of SLS. The SLA is previously established with the Internet Service Provider and specifies that the SLS parameters are dynamically negotiate. The NIA determines the SLS on behalf the user according to the application requirements and the user's needs.

Under the Computing Grid context, some works have been developed in this address. In [13], an architecture and toolkit named GRUBER for resource usage service level agreement (SLA) specification and enforcement in a grid environment is presented. The novelty of GRUBER consists in its capability to provide a means for automated agents to select available resources from virtual organization level on down. It focuses on computing resources such as computers, storage, and networks; owners may be either individual scientists or sites; and virtual organizations are collaborative groups, such as scientific collaborations. A virtual organization is a group of participants who seek to share resources for some common purpose.

7 Conclusions

In this work we have presented a framework for Web services procurement. Our framework features provisioning services, such as contracting, metering, accounting, notification and SLA based management of web services. By means of our framework, service providers can efficiently and flexibly manage their resources to optimize customer satisfaction and, potentially, yield. Furthermore, our framework demonstrates the various roles and steps that make up the lifecycle of managing and hosting a Web Service - starting with offering up the service to potential requestors, using the service and finally ending with the accounting/bill generation for specific services

References

1. Steve Vinoski. Integration with Web Services. IEEE Internet Computing. November-December 2003 pp 75-77.
2. Adams, H., Dan Gisolfi, James Snell, Raghu Varadan. "Custom Extended Enterprise Exposed Business Services Application Pattern Scenario," http://www-106.ibm.com/developerworks /webservices/library/ws-best5/, Jan. 1, 2003
3. Samtani, G. and D. Sadhwani, "Enterprise Application Integration and Web Services," in Web Services Business Strategies and Architectures, P. Fletcher and M. Waterhouse, Eds. Birmingham, UK: Expert Press, LTD, pp. 39-54, 2002a.
4. Alexander Keller, Heiko Ludwig: Defining and Monitoring Service Level Agreements for dynamic e-Business. In Proceedings of the 16th USENIX System Administration Conference (LISA'02), November, 2002.

5. Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King. A Service Level Agreement Language for Dynamic Electronic Services. In Proceedings of WECWIS 2002, Newport Beach, CA, pp. 25 - 32, IEEE Computer Society, Los Alamitos, 2002.
6. Alexander Keller, Gautam Kar, Heiko Ludwig, Asit Dan, Joseph L. Hellerstein. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. IBM Research Technical Report RC22162, 2002.
7. HP OpenView Service Desk 4.5. Release Notes. First Edition. July 2002. Hewlett-Packard Company. 3000 Hanover Street. Palo Alto, CA 94304 U.S.A.
8. Frank Leymann. Web Services: Distributed Applications without Limits - An Outline. IBM Software Group. June 25, 2003.
9. Octavio Martín-Díaz, Antonio Ruiz-Cortés, Rafael Corchuelo, Miguel Toro. A Framework for Classifying and Comparing Web Services Procurement Platforms. Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03).
10. A.M.Abdoessalam and N.Mehandjiev. Collaborative Negotiation in Web Service Procurement. Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04).
11. Octavio Martín-Díaz, Antonio Ruiz-Cortés, David Benavides, Amador Duran, and Miguel Toro. A Quality-Aware Approach to Web Services Procurement. In B. Benatallah and M.-C. Shan (Eds.): TES 2003, Lecture Notes on Computer Science 2819, pp. 42–53, 2003.
12. Gilles Klein and Francine Krief. Mobile Agents for Dynamic SLA Negotiation. In E. Horlait (Ed.): MATA 2003, Lecture Notes on Computer Science 2881, pp.23 –31, 2003.
13. Catalin L. Dumitrescu and Ian Foster. GRUBER: A Grid Resource Usage SLA Broker. In J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, Lecture Notes on Computer Science 3648, pp. 465–474, 2005.