International Journal of Computer Applications (0975 – 8887)
Volume 180 – No.18, February 2018

To Study the Functionality and Performance of Web Services

Dharmendra Choukse
Institute of Engg. & Science
IPS Academy
Indore, India

Umesh Kumar Singh
Institute of Comp. Science
Vikram University
Ujjain, India

ABSTRACT

Nowadays, the ability to smoothly exchange information between internal business units, customers, and partners is essential for success. On the other hand, most administrations service a variety of disparate applications that store and exchange data in different ways and therefore cannot "talk" to one another effectively. Net Application and Access models have evolved as a practical, cost-effective solution for consolidate information distributed between critical applications over the operating system, platform, and language barriers that were previously blocked. This study presents the analytical review of Web services functionality and their performance through latency and throughput.

Keywords

Webservices,UDDI,SMTP,XML

1. INTRODUCTION

Web services are software mechanisms that communicate using pervasive, standards-based Web technologies including HTTP and XML-based messaging and this structure are based on a collection of standards and protocols that allow us to make handling requests to remote systems by delivering a standard, nonproprietary language and using conventional transport protocols such as HTTP and SMTP. The efficient e-business perception calls for a smooth integration of business processes, applications, and Web services over the Internet. Web service technology enables e-business and e-commerce to become a reality. It has become a competitive tool for companies by reducing cost through fast, efficient, and reliable services to clients, dealers, and partners over the Internet. It permits more efficient business processes via the Web and improves business chances for companies, Web services are planned to be accessed by other applications and differ in complication from primary activities, such as examine a banking account balance online, to complicated processes running CRM (customer relationship management) or enterprise resource planning (ERP) systems because these are based on open standards such as HTTP and XML-based protocols including SOAP and WSDL. Web services are powered by XML and three other core technologies: WSDL, SOAP, and UDDI. In a Web service model, a service supplier proposals Web services which deliver tasks or business operations which can be arranged over the Internet, in the hope that they will be invoked by partners or customers; a Web service requester defines requirements to trace service provider. Publishing, binding, and discovering Web services are three key tasks in the model. Discovery is the process of finding Web services provider locations which satisfy specific requirements. Web services are useless if they cannot be discovered. So, discovery is the most important task in the Web service model. The Web service model in Fig. 3.2 shows the interaction between a service requester, service providers, and a service discovery system.

1. The service providers proposal Web services which deliver functions or business operations. They are formed by companies or societies. In order to be invoked, the Web services must be defined. This will facilitate discovery and arrangement. WSDL or service profile of semantic Web service is used to carry out this task.

2. The Web service requester defines requirements in order to locate service providers. Service requesters usually contain a description of the Web service, though it is not a Web service which can run on the Internet. The requirements are typically defined by WSDL, service template or service profile.

3. The Web service discovery or service registry is a broker that provides registry and examine tasks. The service providers advertise their service info in the discovery system. This info will be kept in the registry and will be searched once there is a demand from service requester. UDDI is used as a registry typical for Web service.

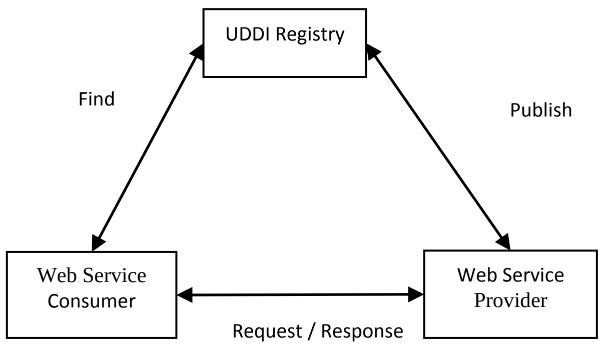

Figure 3.2: Web Services Model

The above three mechanisms interact with each other via publishing, discovery, and binding operations. These operations are elaborated upon as follows:

1. Publish: the Web service providers publish their service information through the discovery system for requesters to discover. Through the publishing operation, the Web service provider stores the service description in the discovery system.

2. Discovery: the Web service requesters repossess service providers from the service archive. Based on service explanations, which describes the requests of the Web service clients, the discovery system will output a list of Web service suppliers which satisfy the requirements.

3. Bind: After discovering, the discovery system provides some Web service providers. The Web service requester invokes

these Web service providers. The binding occurs at runtime. The Web service requesters and Web service providers will communicate via SOAP protocol which is an XML based protocol for Web service exchange information.

2. LITERATURE SURVEY

While an effective feedback mechanism has been identified as a key feature in building a successful electronic marketplace [7], the lack of effective mechanisms to provide feedback, performance measures and reputation systems have been identified by practitioners as significant issues in the adoption of Web service electronic marketplaces [8]. In an information service based situation, an effective device that reports on the quality and performance of the information service is an important feature. Since Web services are provided in information-based service environments, it is important to evaluate their quality & performance and to develop a feedback mechanism to provide adequate information for the client's decision-making activities.

Maximilien and Singh (2002)[10] proposed a conceptual model for Web service standing, using which repute information can be prearranged and shared and service selection can be facilitated and automated. [10] Discuss the significance of considering key attributes of the Web services' presentation and assigning them weights based on their relative influence on their overall reputation in reputation systems in the business domain. [10] Present a theoretical model of presentation driven Web service collection and highlight the importance of defining the dimensions to measure the quality and performance of Web services. While[10] address the technical condition of such Web service selection mechanism, they do not specify the presentation and quality dimensions of Web services that are essential to inform this quality-driven Web service selection process. Our literature review reveals that while extant literature is familiar with the significance of measuring the quality and presentation of Web services, it does not provide metrics that take into account the technical and business dimensions of Web services in an electronic marketplace.
IBM defines Web services as a technology that allows applications to communicate with each other in a platform- and programming language-independent manner [11]. In added words, a Web service is a software interface that describes a gathering of processes that can be accessed over the network using identical messaging protocols. A Web service is basically a special type of electronic service. The literature stream in the quality and performance of electronic services is a applicable and useful literature stream to help describe the quality and performance measures of a Web service [12], considered one of the early assistances to the field e-services quality, have proposed measures for the quality of e-services as a multidimensional construct. However, [12] assert that the various potential dimensions have to be examined more methodically since no consensus on the related dimensions of this multidimensional construct has been reached.

3. REQUIREMENT FOR WEB SERVICES MODEL

The service providers of Web services extend their service descriptions with logical statements about quality oriented services (QoS) associated with the entire interfaces or individual components. Fig. 3.3 gives the sequence of events and communication between the entities involved in quality oriented Web services.

The service requestor requests the binding information with the QoS. Depending on the QoS requirements, the broker searches the UDDI for the listed services available, and while performing the QoS negotiation by comparing the required and offered QoS, the broker finally determines a QoS that is acceptable to both parties. In this method, the binding is built, and the communication between the service provider and service requestor eventually starts. To support QoS, the developers should be willing to incorporate significant design changes to the system, because individual QoS attributes cannot be utilized independently of the existing components.

The other requirements of the web services model are Reliability, Scalability, Integrity, Availability, Accessibility, and interoperability is well defined in.

4. EXEPERIMENTAL SETUP

The ASP Dot net Mobile Store Sample Application [MSSA] was developed within the ASP BluePrints Program [MSSA] at Visual Studio 2016, Inc. It demonstrates how to use the capabilities of the Dot net platform to create robust, scalable and portable e-business applications. It comes with full source code and documentation, allowing application developers to experiment with dot net technologies and learn how to use them effectively to build their own enterprise solutions. Mobile Store also includes Web Service interfaces to some of its services. The goal is to showcase the use of Web Services within the Dot Net platform. Following is a brief description of Mobile Store based.

The Web site presents an online interface to the store, through which customers can shop and place orders. When a customer finishes a request, the latter is sent to the order fulfillment center for processing. Hence the Website functional unit can be thought of as the front end of the enterprise. The fulfillment center, on the other hand, has an order fulfillment component and a supplier component. The fulfillment center processes orders based on the enterprise's business rules manage financial transactions and arranges for products to ship to customers. Because not all products are in stock at any given moment, order processing may occur over a period. Managers and other suppliers may relate through the fulfillment center. This portion of the business is discussed to as the back end. Although the supplier component is part of the sample application, it could just as easily be a service external to the application.

The new sample application comprises four separate sub-applications, each of which is an ASP Dot Net application:

1. Mobile Store E-Commerce Web Site: A Web application that shoppers use to purchase merchandise through a Web browser.

2. Mobile Store Administration Application: A Web application that enterprise administrators use to view sales statistics and manually accept or reject orders. While being a Web application, it features a rich client that uses XML messaging by adding to a plain HTML interface.

3. Order Processing Center (shortened "OPC"): A process-oriented application that manages order fulfillment by providing the following services to other enterprise participants:

• Receives and processes orders placed through the Mobile Store Web Site. Orders are received as XML documents.

• Provides the MobilestoreAdmin application with order data using XML messages over HTTP.

• Sends an email to customers acknowledging orders using WebLogic JMS Mail.

• Sends purchase orders (described as XML documents) to suppliers via JMS.

• Maintains purchase order database.

4. Supplier Application (shortened "Supplier"): A process-oriented application that manages shipping products to customers by providing the following services:

• Receives purchase orders (in the form of XML documents) from the OPC via JMS.

• Sends products to clients.

• Provides handbook inventory management through a Web-based interface.

• Maintains inventory database.

Figure 3.4 depicts the most important Mobile Store sub-applications and shows the mechanisms and protocols used for communication between them.

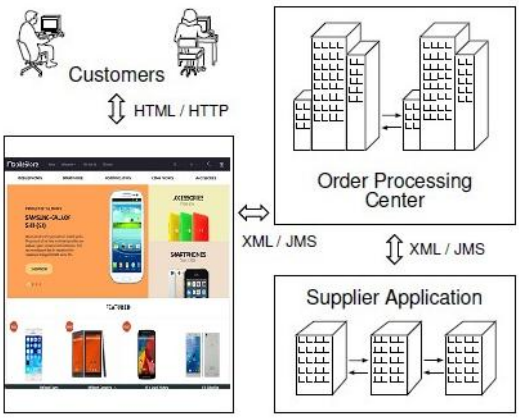

Figure 3.4: Main Mobile Store Sub-Applications

Mobile Store's Asynchronous, Document-Oriented Architecture: The Order Processing Center defines the business process for handling purchase orders placed at the Web site. The business method consists of a workflow, which is a sequence of steps with transitions between them. Transitions between phases are handled by individual classes called transition delegates, which allows for flexibility if the type of communication between levels is to change. Mobile Store uses a document-oriented business process to coordinate its internal workflow and to communicate with its supplier application. Transition delegates pass XML documents between workflow steps by placing JMS messages in message queues (or topics). The JMS queues are the transition points between steps. Messages arriving in the queues are processed asynchronously by Message-Driven Beans (MDBs). The asynchronous architecture allows components to call each other without having to block and wait for a response.

Figure 3.5 depicts the flow of messages upon reception of a purchase order at the OPC. The OPC receives the order through its purchase order queue, validates it and passes it to the order approval queue. The order approval MDB confirms if funds are available for the order. Once the order is accepted, it is sent to the supplier. The latter receives the order through its supplier order queue, validates it and ships products to the customer. It also creates an invoice and returns it to the OPC by sending a JMS message to the invoice topic. Two MDBs, invoice and mail invoice, both listen to the invoice topic and process they arrived invoice. The invoice MDB triggers the OPC's invoice processing workflow. The mail invoice MDB sends a message to the mailer queue to notify the customer by email that his order is completed.

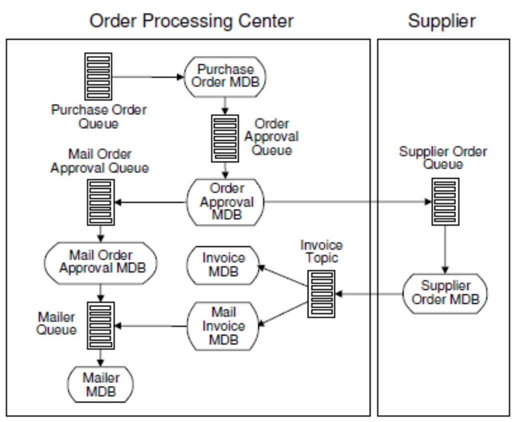

Figure 3.5: Message Flow in the OPC and Supplier

In particular, Web Service interfaces to the supplier application are provided, allowing for the communication with it to take place through Web Services.

Figure 3.6 shows the new message flow leveraging Web Services for the communication with the supplier. Instead of sending orders openly to the supplier through JMS, the OPC now sends them to a Web Service endpoint in the supplier using a modified transition delegate. The supplier, Web Service endpoint, receives orders as XML forms and authorizes them against a public XML schema. It then places them on the supplier order line, which triggers the usual order processing workflow. Likewise, instead of sending invoices directly to the OPC through JMS, the supplier now sends them to a Web Service endpoint in the OPC. The latter receives incoming invoices, validates them against a public XML schema, and places them in the invoice topic, triggering the usual invoice processing workflow.

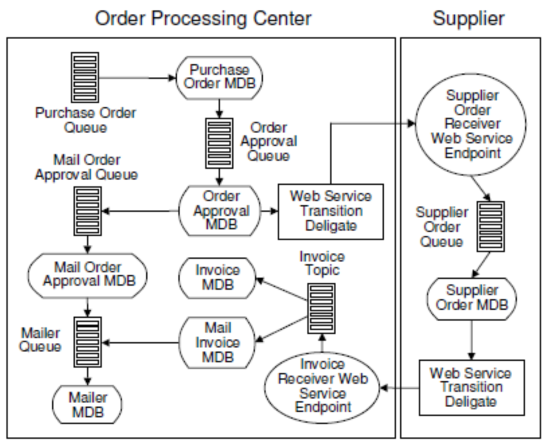

Figure 3.6: Message Flow when using Web Services

The new Mobile Store code can be compiled in two variants. In the first one (JMS variant), no Web Services are used, and communication with the supplier is done by directly sending messages to JMS targets. In the second one (Web Service variant), Web Services are used as explained above.

Deployment Environment

We deployed Mobile Store in the deployment environment using WebLogic JMS .NET as a Dot Net container and Oracle 11g as a database server. The database server is used for persisting application data and JMS messages. The client machine is used to create the load (using the Gatling tool) and control the progression of the experiments. In addition, the client machine is running a mail server, used by the application server for sending Mobile Store's email notifications to customers about the progress of their orders.

5. RESULT ANALYSIS

In order to measure system performance under the considered workload, dissimilar measurements were taken during the experimentations. For this purpose, Mobile Store was instrumented to log timestamps at various points during the execution (measuring points), and performance metrics were calculated based on differences between timestamps. The first measurement determines the start of an order's processing and is done on Mobile Store's front end right after the reception of a request before it is forwarded to the OPC. The second measurement determines the end of an order's dealing out and is done in the Mailer MDB at the point an email is sent to the client to confirm order completion. The difference between timestamps taken at these points gives us the total order processing time, also referred to as the response time. Dividing the number of orders processed during an experiment by the time elapsed between its start and end, we obtain the throughput. To further monitor the performance of the back end, three other dimensions were done. The first one determines the time at which the OPC calls the supplier to submit a new order, i.e., the beginning of the calling operation. When running in JMS mode, this is the time of sending a JMS message to the supplier order queue. When running in Web Service mode, it is the time of calling the supplier Web Service endpoint. The second dimension determines the time at which the Web Service call (or the sending of the new order, respectively) completes. The difference between timestamps taken at these points gives us the time to which the OPC has blocked when calling the provider. We mention to this time as the call time. The last measurement is done in the supplier order MDB and determines the time of arrival of order as an XML document and the start of its processing. The time elapsed between the beginnings of a call to submit an order to the provider and the start of its processing will be mentioned too as the latency.

A. Performance Analysis

To gather the measurements for our analysis, we performed three series of experiments. Each series consists of six experiments characterized by the number of client threads used for simulating concurrent clients (15, 25, 40, 60, and 100 client threads, respectively). The first series signifies the base case and uses the modified JMS mode (referred to as "JMS" in the following figures). Send processes at the OPC/supplier interface in this mode are not part of the workflow transactions. The second series covers the Web Service variant of Mobile Store ("Web Service"). Finally, the third series uses the original JMS variant of Mobile Store as published by the Blueprints program, i.e., the JMS send operations at the OPC/supplier interface are part of their respective workflow transactions. These measurements represent the typical operational behavior of the Mobile Store, but cannot be used for comparison with the Web Service variant. They are accessible to highlight the performance differences of our base case system (JMS) with respect to the fully transactional, original JMS mode system. The CPU utilization on the client machine and database server machine were monitored during all the experiment runs and stayed below 10 and 35 percent, respectively, over the range of all simulated load levels.

B. Throughput

Figure 3.8 demonstrations the throughput effects (as the number of completed orders per minute) for each experimentation in JMS and Web Service mode. The unique JMS mode is not represented as it exhibitions equal throughput characteristics as the improved JMS mode (less than 1 percent eccentricity) over the full load range. The JMS mode reaches the highest throughput of 163 orders per minute with 80 concurrent client threads. Under the same load, the Web Service variant reaches an about 5 percent lower value of 155 orders per minute. Under light to a moderate load of up to 40 customer threads, both variations scale linearly and exhibit almost identical performance. The modifications remain below 1 percent in this load range; then we observe about 10 percent higher CPU time demand for the Web Service variant. This difference in resource consumption has no impact on throughput, as enough CPU time is readily available at that point. In the load range beyond 40 client threads towards system saturation, the higher CPU consumption begins to have an impact on throughput. The point of system saturation where no additional increase in throughput can be accomplished is reached at a load of 80 client threads in the JMS case. Due to the higher CPU time claim of the Web Service variant, it reaches that point already at a load close to 70 threads and thus lags behind in maximum throughput.

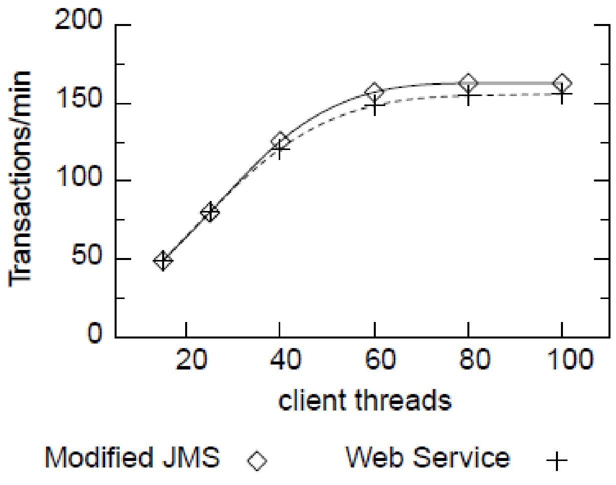

Figure 3.8: Throughput Measurements

C. Latency

While throughput characterizes the speed of order processing as a whole, the latency is directly related to the OPC/supplier communication, and it is significantly affected by the introduction of Web Service interfaces. Figures 3.9 shows the latency measurement results of the modified and original JMS variants. Considering the latency of the modified JMS variant, it strikes that messages sent, arrive at their destinations (latency — Figure 3.9) prior to the sending entities having completed.

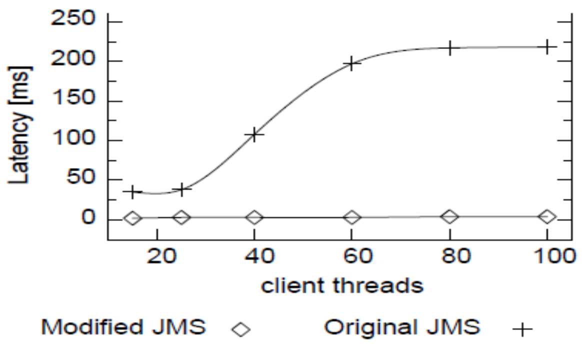

Figure 3.9: Latency Measurements

The reasons for this become clear from looking at Figures 3.10, which shows the sequence of actions performed when sending JMS messages to the supplier.

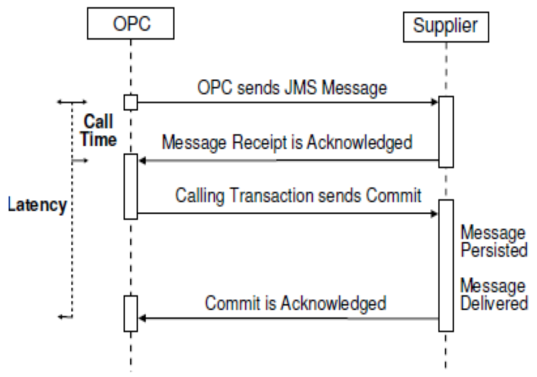

Figure 3.10: JMS Call in the Original JMS Variant

The non-transacted send procedure can only return to the caller after the messaging system can guarantee that the message will be sent at some point. The configuration at hand persists the message in a database to provide this guarantee without waiting for an actual effective reception to complete.

Should a fault occur through the activities of the receiving dealer order MDB causing the operation to roll back, the messaging system can always redeliver the message since it can be re-claimed from persistent storage. The call time for the improved JMS variant starts at 11 ms under light load and increases slowly until it stabilizes at values around 100 ms under high load. These appearances are related to the additional database activity for persisting messages due to increasing throughput as well as the increased time wanted to in serialize the message for persistence due to growing CPU operation. The first sending of the message to the supplier order MDB does not, however, have to wait for the achievement of the operations to persist the message but can start directly. This allows the improved JMS variant to bear a very low latency of 2 to 4 ms over the whole load spectrum. This low latency of the OPC/supplier interface has a substantial impact on the overall response time of the modified JMS variant, as it regulates the time the next treating step can begin and as such is part of the time-critical path. Paralleling these results to the original JMS variant reveals very different latency behavior for the latter. While the sent message has to be persisted before the send method call can return in the modified JMS variant, the send process being part of a transaction in the original JMS variant can defer persistence to the later commit task. This reasons the call time to halt very low at 2-3 ms throughout the full load range. On the other hand, the ease of use of the sent message to the receiving end of the supplier application is delayed until after the completion of the transaction in the commit operation. This prolongs the latency not only by the time required to persist the message (as measured by the call time readings for the non-transacted send in the modified JMS variant) but also by the interval needed to commit the work of all other actions taken in the business logic of this processing step. This prolonging effect of measured call time in the modified JMS variant on latency in a transacted option should be kept in mind when considering the measurements of the following comparisons to the Web Service variant.

6. CONCLUSION

We calculated the effects on performance when part of the ASP Dot Net Mobile Store application is implemented using Web Service interfaces. The reference implementations were the original Mobile Store application with transactional messaging and a modified version of Mobile Store with non-transactional, but still reliable messaging. The latter modification was needed for a fair comparison since today's Web Services Platforms do not provide support for transactional calls. The measurement results of our experiments show that the system throughput penalty from using Web Services in the ASP Dot Net Mobile Store is only marginal. Under light to moderate load, the throughput decreases by less than 1% and falls behind the modified JMS variant by at most 5% under high and extreme pressure. The throughput difference stems from the Web Service variant reaching the threshold of CPU capacity under slightly lighter load due to this variant's higher CPU consumption. In adding to the already mentioned higher CPU consumption, the typical 20% increase results from the more senior latency of the transition between the sub-applications across the Web Service interface.

7. REFERENCES

[1] David Booth, Michael Champion, Chris Ferris, Francis McCabe, Eric Newcomer, and David Orchard and. "Web services architecture", 2003. W3C Working Draft 14 May 2003

[2] Patrick Cauldwell, Rajesh Chawla, Vivek Chopra, Gary Damschen, Chris Dix, Tony Hong and Francis Norton, Uche Ogbuji, Glenn Olander, Mark A Richman, Kristy Saunders, and Zoran Zaev. Professional XML Web Services. Wrox Press, 2001

[3] Kai S. Juse. "Performance of J2EE-based Web Services". Master thesis, Darmstadt University of Technology, June 2003. In German

[4] Georgios Meditskos and Nick Bassiliades, "Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S", Member, IEEE Transactions On Knowledge And Data Engineering, Vol. 22, NO. 2, February 2010

[5] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng, "Semantic Web Services", Stanford University, IEEE Intelligent Systems 2001

[6] Guo Wen-Yue, Qu Hai-cheng, Chen Hong, "Semantic web service discovery algorithm and its application on the intelligent automotive manufacturing system", International Conference on Information Management and Engineering, IEEE Xplore, 2010.

[7] Bachlechner, D., Siorpaes, K., Fensel, D. and Toma, I. " Discovery of Web Services- A Reality Check", 2006, DERI – Digital Enterprise Research Institute

[8] Bo Zhou, Tinglei Huan, Jie Liu, Meizhou Shen, "Using Inverted Indexing to Semantic Web Service Discovery Search Model", 5th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE Xplore, 2009

[9] Pavlou, P. and Gefen D. "Building Effective Online Marketplaces with Institution-Based Trust," 2004, Information Systems Research, Vol. 15, No. 1, March, pp. 37–59.

[10] Maximilien M.E and Munindar P. Singh, "Conceptual model of Web service reputation", 2002 ACM SIGMOD Record, v.31 n.4 "New to SOA and Web services" Developer Works Retrieved August 8, 2009,

[11] "Web services Glossary" W3C Working Group Note 11 February 2004, Retrieved January 7, 2009, from W3C Official Site < http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/ >.

[12] Zeithaml, A., Parasuraman, A. and Malhotra, A. "Service Quality: Definition, Dimensions, and Conceptual Model," 2000, working paper, Marketing Science Institute, Cambridge, MA.