Extended Web Services Framework to Meet Non-Functional Requirements

Zafar U. Singhera
Consultant, Cisco Systems
Application Oriented Networking Group
375 E. Tasman Dr.
San Jose,CA 95134, USA
zafar_singhera@hotmail.com

Abad Ali Shah
R & D Center of Computer Science
University of Engineering and Technology
Lahore, Pakistan
+92 42 682 9499
abad_shah@hotmail.com

ABSTRACT

Non-functional requirements/characteristics are important for providing effectively every kind of services including web services. A realistic web service must meet both functional and non-functional requirements of its consumers. Therefore, it is important that a web services framework is augmented so that non-functional characteristics of a web service can be determined at run-time and consumers are bound to a service that best meet their functional as well as non-functional requirements. In this paper, we propose an extension of the existing web services framework that enables a collection of functional and non-functional service characteristics at run-time, and usage of the collected data in discovery, binding, and execution of web services. Descriptions of new and enhanced components of the proposed framework are also presented. Typical publishing and usage scenarios in the proposed framework are also described.

Keywords
Web services, framework, non-functional requirements

1. INTRODUCTION

The idea of services business has been around from the very early days of mankind. As life became more complex and sophisticated, we learned to render services from others to fulfill our needs, and to serve others in their efforts and get rewarded. Those who provided services were rewarded in exchange of their services. Those who have provided larger number of services to larger consumer community got better rewards. We have learned to use services through providers who provide quality and reliable services at an affordable cost. Therefore, it became critical for consumers to find service providers who provided services that best meet their needs in the most economical, reliable, and timely fashion. To maximize their profits, it also became critical for service providers to publicize their services to attract more consumers, and improve their service quality and ensure efficient and reliable delivery to retain consumers and reduce costs.

Services paradigm went through a number of evolutions. Major evolutions were triggered by newer technologies. Focus of these evolutions has been to improve mechanisms to publicize, find, request, and deliver services. Web Services are the latest evolution in services paradigm [11]. Computers, the Internet, and web technologies are the catalyst technologies behind this evolution. Web Services offer an easier, effective, and efficient way to publish, find, request, and deliver services. It is based on the standard-based framework for publishing, discovery, binding and usage of independently developed distributed components, called *web services*.

During the past couple of years, there has been a lot of activity to develop infrastructure to support deployment, discovery, and use of web services. Major providers of computer and software infrastructure technologies, including BEA Systems Inc. [1], IBM [5], Microsoft [6], and Sun Microsystems [9], are aggressively working to extend their existing environments to support development, deployment, and maintenance of web services. Their major activities are underway to define and implement solutions to improve security [14], provide transaction support [15], and improve coordination among the web services [12].

So far, the web services research and development efforts have been primarily focused on definition and development of infrastructure to publish, discover, and deliver web services which meet specified functional requirements. The existing framework does not provide support for non-functional requirements, like performance, scalability, reliability, availability, flexibility, stability, cost, completeness, etc. For example, there is no mechanism available in the existing web services architecture to determine run-time performance characteristics of a web service, and select a service that best meets the performance requirements of the consumer.

In [10], Ran proposed an extension to the web services model to include Quality of Service (QoS) requirements in publishing and discovery of web services. This model extends Universal Description, Discovery, and Integration (UDDI) [7] schema to include the QoS information while publishing a service. A new component in Ran's model, called QoS Certifier, verifies QoS claims of a web service before it is published to the extended

UDDI registry. The publishing of a web service is split into two phases under the proposed model. The first phase involves verification of the QoS claims by the provider, and providing a certification to the provider if those claims are verified. During the second phase, service provider publishes the service along with its certification information in the UDDI registry.

The Ran model is a step in the right direction but it relies on static information and is incapable of providing run-time non-functional characteristics of a web service. It only works if variations in non-functional characteristics of a web service are minimal and are known at the time service is deployed. However, it is difficult to correctly predict majority of non-functional characteristics of a web service at time of its deployment because non-functional characteristics depend on many factors such as connectivity characteristics, usage profiles, work loads, computing infrastructure of service provider, wide variation in load on provider's web services infrastructure, etc.

A web services framework supports availability of multiple services that provide the exactly same functionality and interface. Multiple services can be available from the same or different service providers, and they provide the exact same functionality but only differ in non-functional characteristics. Ideally, a web services consumer should be using a web service that best serves his functional and non-functional requirements, and it is the most economical and reliable at the time of a request. The existing web services model does not support dynamic rebinding of a service that best meets a consumer's need at a particular time, frequent validation of the suitability of the existing consumer-service binding, and rebinding of a different service if that service meets consumer's needs better than the service consumer is currently bound to.

In this paper, we propose an extension to the existing web services framework. This proposed extension supports run-time collection of data/information related to non-functional characteristics of web services. It uses this data to rate web services on a variety of non-functional characteristics including performance, scalability, reliability, availability, stability, cost, completeness, etc. It further supports dynamic selection of the best service that meets functional and non-functional requirements of a consumer, and enables consumers to frequently evaluate their bindings to web services and update those bindings if a better service becomes available.

The remainder of this paper is organized as follows. Section 2 provides an overview of the evolution of the traditional services. It highlights the need for the proposed framework and describes why it is a natural evolution of the way traditional services and their supporting frameworks have evolved. In Section 3, we present the proposed extension and describe roles, and functionality of each component of the extended framework. Section 4 provides scenarios for publishing and usage of web services in the proposed framework. In Section 5, we give concluding remarks on the paper by highlighting the contributions made in the paper and future directions of this work.

RELATED WORK

2.1 Evolution of Services

The concept of services has been presented since we started depending on each other to leverage each other's skills to meet our goals. We render services from others and reward them for their services. Two fundamental roles of service provider and service consumer have always been presented since the very beginning. One of the challenges faced has been to find a service provider that meets the needs of a service consumer in the best manner. This resulted in evolution of a third role, called *broker*, which matches consumers with suitable providers and vice versa. A broker helps a service provider in advertising their services, and a service consumer in finding a service that best meets consumer's needs. Broker's reward depends on the number of consumers it introduces to a provider and vice versa, the number of services it introduces or sells to consumers, and the volume of those sales.
To increase short-term profits, the providers sometimes misrepresent their services and/or provide low quality services to their consumers. This results in emergence of another role, let us call it *Monitor*. A Monitor monitors the quality of services provided by providers, collects complements, feedback, and complaints from consumers, analyzes the collected data to rate services from various providers, and uses this rating during service discovery for consumers. Most of the times, brokers implicitly provide this service by monitoring quality of services provided by providers and collecting feedback from consumers. However, independent entities like Better Business Bureau [2], Department of Consumer Affairs [3], and independent rating agencies, have proven to be more effective. Figure 1 is a simple services model showing all the major players and their interactions in a traditional services paradigm.

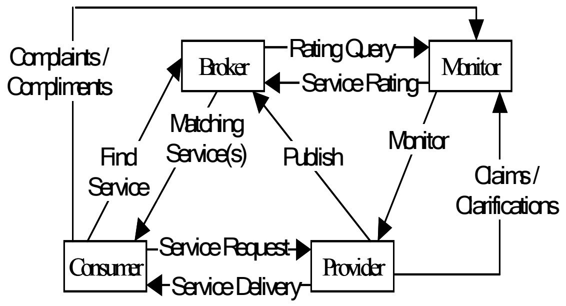

Figure 1: Traditional Services Paradigm

As the number and complexity of services, number of service providers, number of service consumers, and geographical areas and communities served by a service provider increased, it became increasingly challenging to find out a service provider that satisfactorily meets the needs of a service consumer at an affordable price. This objective increased the importance of a broker and monitor even further. These new challenges surfaced for consumers, providers, brokers, and monitors. These challenges and motivations for each of these roles are briefly described in the following sections.

2.2 Service Providers

The major challenges for service providers are to maximize their profits by taking care of the following factors:

Reaching the maximum number of consumers and convincing them that their service meets their needs at affordable price
Delivering good quality services to keep and maximize consumer retention
Reducing overheads and optimising order intake, service execution, and service delivery
Increasing the goodwill and projecting a better image

2.3 Service Consumers

The major challenges for service consumers are to get the best service at the most economical, efficient, and reliable terms. These challenges can be achieved by taking care of the following items.

Identifying the best service that meets their needs at an affordable price
Finding a service provider that offers most suitable service at the most attractive terms
Requesting a service in an easiest possible, most economical, timely, reliable, and secure fashion
Minimizing the cost to discover, evaluate, request, and render the needed service
Resolving issues and getting help if service is unsatisfactory
A reliable and fair mechanism to resolve disputes and complaint about poor quality of service

2.4 Brokers

The major challenges/motivations of brokers are to increase their profit, and they can achieve them by doing the following things:
i) Attracting more service providers to offer more services through them
ii) Attracting more consumers to discover and get more services through them
iii) Providing a friendly and easily accessible mechanism for publishing and discovery of services
iv) Building and maintaining the trust of both provider and consumer communities
v) Matching maximum number of consumers to providers that offer services of interest to the consumers
vi) Implementing mechanism to get feedback from both consumer and provider communities
vii) Quickly resolve dispute if it arises between these two communities

2.5 Monitors

The major challenges/motivations of monitors are to increase their image/effectiveness by taking care of the following things:
i) Providing an accessible and friendly environment to get compliments, suggestions, feedback, and complaints from both consumer and provider communities
ii) Establishing a reliable mechanism to collect and analyze data for a fair and realistic rating of service providers and their services
iii) Making the service provider ratings accessible to the consumer community in easily understandable means
iv) Continuously monitoring service providers to see any change in their service offerings and its quality
v) Winning trust of an increasing number of consumers
vi) Establishing an image and reputation to be the most attractive market place for both providers and consumers

Services and framework to discover and deliver services have been going through waves of evolution. New inventions, technologies, and resources have triggered each such wave. Some of the major phases in the evolution of services can be identified as Verbal, Paper, Mechanical, Electronic, Computer, the Web, and Web Services Phases [8]. Verbal communication, paper, machines, electronic devices, computers, The Web, and web services, respectively, were the catalysts for the emergence of each of these phases. The objective was always to improve the efficiency, reliability, and effectiveness of the discovery, request, and delivery of services. Newer technologies and inventions have been helpful in increasing the capabilities, effectiveness, efficiency, and reliability of the roles and mechanisms.

During the web services era, it increasingly used consumers, providers, and brokers. They have been playing the central role in automation of publishing, discovery, and delivery mechanisms. Computers interact with each other to publish, find, request, render, and provide services. Web services propose that UDDI registries take the role of a broker, web services the role of a provider, and web applications or other web services the role of a consumer. All the communication between these entities is done using a single communication protocol, i.e., HTTP [4], and all the documents and data exchanged among the involved entities is in the XML format [16] using SOAP [17]. Figure 2 shows the existing web services model.

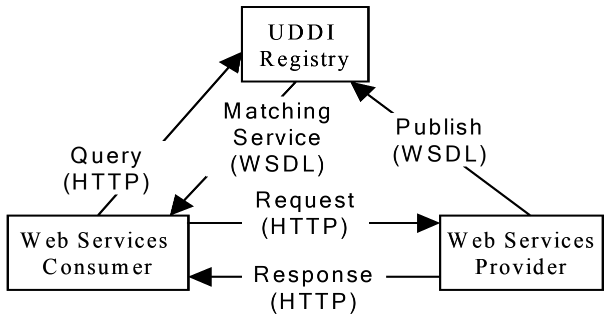

Figure 2: Current web services framework

As evident from Figure 2, one major drawback of the existing web service framework is the absence of a Monitor role. Traditionally, consumers, brokers, or monitor have continuously watched performance, reliability, availability and security characteristics of a service provider. The effort is always spent to determine these characteristics of a service or service provider either in real-time or in the nearest possible past. Consumers have been historically reluctant to get services from a service provider who did not have a good track record of non-functional characteristics. Therefore, it is natural to augment the current web services framework so that service providers can be continuously

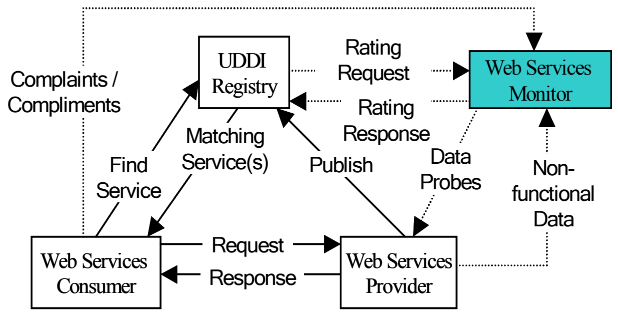

Figure 3: The Proposed Architectural Extension

monitored and rated for their functional and non-functional capabilities. Even more frequent, formal, and automated monitoring is needed because providers and consumers in web services framework are computers. With computers, speed, frequency, and volume of transactions are many times more than those in traditional services paradigm.
Now we list some of the limitations of the current web services framework:

i) The role of a monitor, that has been present in traditional services paradigm, is absent in web services framework.
ii) The information about a web service in UDDI registry is static and defined when the web service is registered or updated.
iii) Query posed by a client is entirely based on functional and statically defined characteristics of a service.
iv) Dynamic characteristics of a web service, like performance, availability, reliability, etc are not taken into consideration when the UDDI registry discovers a service for a consumer.
v) The UDDI registries are unaware of the run-time non-functional characteristics of a service or even its availability. Therefore, a query from a consumer always returns the same result irrespective of the real-time non-functional behavior or availability of the application.
vi) There is no mechanism so that consumers can influence the rating of a service by providing compliments or complaints.
vii) The existing web services framework does not impose a check on service providers that motivates them to improve and maintain their quality of service at all the time.

The above-listed limitations of the current web services framework motivates us to extend the existing framework to provide the customers run-time monitoring and rating of web services, more realistic discovery based on latest rating, and ability to frequently check and bind to a new service that better meets their functional and non-functional needs. In the following section, we present an extension to the existing web services framework to provide such capabilities.

3. THE PROPOSED FRAMEWORK

In this section we propose an extension of the existing web services framework, and it augments with additional components and protocols to support the above-mentioned capabilities (see Section 2). The proposed enhancements to the existing framework enable it to support run-time data collection, dynamic evaluation, and frequent query for suitable service identification at run-time, and binding and rebinding of the best available web service for a consumer. Figure 3 shows the proposed extensions. The grayed box in the figure represents the newly introduced component and broken lines represent new or extended information flows.

3.1 Web Services Monitor

Web services monitor is the only new component in the proposed extension to the web services architecture. It keeps information about a collection of web services and frequently probes those services to collect data about their non-functional characteristics. It stores the collected information and analyzes it to deduct additional non-functional characteristics of the service. It also responds to registration and query requests from registries. Such requests and queries include those to add/drop services or to get information about non-functional characteristics of the services. Some of the major modules of a Web Services Monitor and their functions are described below:

Registration Module: This module handles the requests from multiple UDDI registries to add or remove services that need to be monitored, along with the details of monitoring. As a single monitor can serve multiple UDDI registries, this module is also responsible to minimize duplication of information and store a super-set of information requested by all registries, and to maintain a minimal such superset. It is also responsible for maintaining the information about the deployment environments of listed services to determine if those environments support collection of non-functional information required for those services.

Service Monitoring Module: This module is responsible for poling services at pre-defined intervals for the required set of non-functional characteristics and storing that information into the repository. It is also responsible for collecting domain-specific data from a web service that implements a generic interface for this purpose.

Uplink Module: This module handles queries from UDDI registries and returns results of those queries.

Rules Engine: This Engine is responsible for the definition, maintenance, and execution of a set of global, registry-specific, characteristics-specific, analysis-specific, and service-specific rules. Such rules determine the amount of information to be collected for a service or category of services, frequency of data collection for a particular service or category of services, etc.

History Analyser: This module takes care of analysis of the historical data about the services, and it determines certain non-functional characteristics of the service. For example, availability data collected for a service over a long period of time is analyzed by this module to determine the reliability characteristics of the service, like the Mean Time to Failure (MTTF) reliability metric.

Synchronization Module: This module handles the interaction among Web-Services Monitors to replicate data, redirect requests, or consolidate data from multiple monitors.

3.2 Augmentation to UDDI Registry

The current architecture of UDDI registry stores only the static information provided by a web service provider at its registration time. This information is then used during selection of an appropriate service for a searching consumer. To support selection of services using latest run-time information about non-functional characteristics of a service, UDDI registry architecture must be enhanced so that it can request information about non-functional characteristics of a set of services from monitor(s) and then use this information to select the service that best meets consumer's requirements. Following are some of the additional functionality that registries have to implement to support the proposed framework:
(i) Mechanisms to register new web-services that monitors and the registry will rely on to determine non-functional characteristics of services. Additional intelligence should also be built into the registry so that it can support multiple monitors depending on the category, functionality, preferences, and claimed non-functional characteristics of a particular service.
(ii) Registration mechanism needs to be augmented to support the separation of functional and non-functional characteristics of new services, selection of monitor(s) for the new service, sending requests to the identified monitors to start monitoring the new services, and

informing a monitor when a service is de-registered or its run-time non-functional characteristics change.
(iii) Additional query processing capabilities so that UDDI registries can analyse the incoming service queries to separate functional and non-functional parameters. Non-functional parameters can be further decomposed into static non-functional parameters that are stored in the registry itself and run-time non-functional parameters for which registry has to rely on a monitor.

3.3 Augmentation to Web Service Provider

As the framework requires frequent collection of non-functional characteristics of a web service, therefore, a provider must provide a mechanism to collect such data/information frequently. Providers in the web services framework provide web services that are usually deployed on the provider's application server. Application servers currently provide a rich set of common services that can be leveraged by the all applications/services that are deployed on that server. Therefore, a support can be built into the application server or other deployment infrastructure of web services so that general non-functional characteristics, including the availability characteristics like uptime, performance characteristics like response time, throughput, etc, or the reliability characteristics like MTTF, can be collected without requiring individual web services to implement such support. The proposed framework suggests that the deployment infrastructures and application servers provide capabilities so that non-functional characteristics of deployed services can be monitored, collected, and reported to the requesting monitors. If domain-specific non-functional information about a web service is critical, then that service can implement special interfaces to provide that information. We are working on defining generic interfaces that web services can implement to provide domain-specific data.

3.4 Augmentation to Web Services Consumer

The proposed framework allows a consumer to frequently search for a service that best meets his needs. A mechanism is needed on the consumer side to define frequency and criteria for checking a better service, composing new search queries for UDDI registries that best suite the current needs of the consumer, analyzing results obtained from the registry, and rebinding new service if it is better than the one consumer is currently using.

Majority of web-service consumers are expected to be either serviced directly or through applications. Web services or applications these days do most of the processing on the server side, using a web server or an application server. Therefore, the needed additional functionality to support the proposed framework can be implemented in the infrastructure that deploys consumer services/applications, thus having minimal overhead on individual web services development. This will enable consumers to define the criteria for re-checking a better service in a generic and service-independent fashion. The criteria for the checking availability of better service and rebinding can be specified in the deployment descriptors of a web application or a web service. The deployment environment to implement the specified criteria can then interpret this descriptor.

4. SERVING WITH PROPOSED FRAMWORK

This section presents scenarios for publishing and usage of web services under the proposed framework. Following are the publishing and usage scenarios of web services that will consider both functional and non-functional characteristics of a web service.

4.1 Publishing

In this section, we list the steps that are involved in publishing a web service.

i) The provider creates, assembles, and deploys a web service on a deployment platform.
ii) The provider defines the web service in a deployment descriptor, using enhanced Web Services Description Language (WSDL) [13]. The enhanced description specifies whether and to what extent the service supports and is interested in being considered against queries involving non-functional characteristics also.
iii) He service provider registers the service in UDDI registries. The registration request also specifies if service provider prefers its service to be considered for queries that impose non-functional constraints.
iii) If the service provider requires that hiss service should be considered against the queries using non-functional parameters, then the UDDI registry informs appropriate monitor(s) to start monitoring non-functional parameters of the service.
iv) Monitor(s) start collecting non-functional data/information about the service using pre-defined and recommended policies for the services or categories to which the service belongs.
v) The collected data/information is stored in the repository of the monitor(s), and frequently analysed for more complex non-functional characteristics, like reliability, availability, robustness, etc.

4.2 Usage

This section presents the steps that are involved for using web services in the proposed framework.

i) An interested client sends a request for a service to a UDDI registry, specifying the functional and non-functional requirements.
ii) The registry separates functional and non-functional criteria in the query. Non-functional criteria are further split into static and dynamic non-functional criteria.
iii) The registry finds the candidate services that meet the functional and static non-functional criteria, by exploring through its own repository.
iv) The registry sends requests to the appropriate monitors to determine what services, among those selected in Step 3, meet the dynamic non-functional criteria and how well.
v) The monitors send non-functional characteristics of those services that meet the dynamic non-functional criteria to the registry.
vi) The registry collects data/information from all the monitors and analyses it to rate and sort the candidate services using non-functional criteria specified in the query and non-functional characteristics of the candidate services received from monitors.
vii) A reference to the service that best meets the non-functional requirements is returned back to the client
viii) The client uses the service until its non-functional requirements change, the non-functional characteristics of the selected service become unacceptable, or it decides to

check again if a better and more economical service is available.

ix) A new query with functional and non-functional requirements is sent to the UDDI registry again.

x) Steps i-ix are repeated, according to the policies defined by the consumer, until the consumer exits.

During the web services era, it increasingly used consumers, providers, and brokers. They have been playing the central role in automation of publishing, discovery, and delivery mechanisms. Computers interact with each other to publish, find, request, render, and provide services. Web services propose that UDDI registries take the role of a broker, web services the role of a provider, and web applications or other web services the role of a consumer. All the communication between these entities is done using a single communication protocol, i.e., HTTP [4], and all the documents and data exchanged among the involved entities is in the XML format [16] using SOAP [17]. Figure 2 shows the existing web services model.

5. CONCLUDING REMARKS AND FUTURE WORK

This paper presents web services just as another wave of evolution in centuries old services paradigm. We presented our perspective about web services as a vehicle to publish, discover, and deliver services. The shortcomings of the currently used web services framework and its inability to consider real-time non-functional characteristics during discovery of a web service have highlighted. An extension to the existing web services architecture is presented to support collection of real-time information about web services and using this information to discover services that not only meet functional requirements of a consumer but also meet consumer's non-functional requirements. The presented framework enables consumers to frequently check availability of more suitable services and renew their bindings as desired. Scenarios are presented that describe how web services are published and used in the proposed extended framework.

This paper serves only the purpose of introducing the idea of the proposed framework. Defining and implementing such a framework is a major task that will require further research and development efforts into a number of areas. We are currently working on addressing identified challenges. Some of those challenges include defining and comparing possible mechanisms to implement monitors, interaction between monitors and registries, enhancements to service deployment infrastructures for collection of run-time non-functional data/information, augmenting infrastructure to deploy consumer services and applications to support the frequent search for and rebinding to better services, handling and understanding of complex UDDI queries that include both functional and non-functional criteria, and prioritizing and sorting algorithms to prioritize services that qualify for particular consumer query.

REFERNCES

[1] BEA Systems Inc., http://www.bea.com/webservices

[2] Better Business Bureau, http://www.bbb.org

[3] Department of Consumer Affairs, http://www.dca.ca.gov/, http://www.ci.nyc.ny.us/html/dca/home.html

[4] HTTP – Hypertext Transfer Protocol, http://www.w3.org/Protocols/

[5] IBM Web Services, http://www-106.ibm.com/developerworks/webservices/

[6] Microsoft Web Services, http://msdn.microsoft.com/webservices/

[7] OASIS, 2002. Universal Description, Delivery, and Integration of Web Services (UDDI), Version 2.0, http://www.uddi.org

[8] Singhera, Z., Dynamic Monitoring and Binding of Web Services, The 2004 International Symposium on Collaborative Technologies and Systems (CTS'04), San Diego, CA, January 18-23, 2004.

[9] Sun Web Services, http://java.sun.com/webservices/

[10] Ran, Shupring, A Model for Web Services Discovery with QOS, ACM SIGecom Exchanges, Volume 4, Issue 1. March 2003.

[11] Web Services, http://www.w3c.org/2002/ws

[12] Web Services Coordination (WS-Coordination), http://www.ibm.com/developerworks/webservices/library/ws-coor

[13] Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl/

[14] Web Services Security (WS-Security) Version 1.0, http://www.ibm.com/developerworks/webservices/library/ws-secure

[15] Web Services Transaction (WS-Transaction), http://www.ibm.com/developerworks/webservices/library/ws-transpec/?dwzone=webservices

[16] World Wide Web Consortium. Extensible Markup Language (XML), http://www.w3.org/XML

[17] World Wide Web Consortium. Simple Object Access Protocol (SOAP), http://www.w3.org/2000/xp/Group/