ISSN 0973-9777

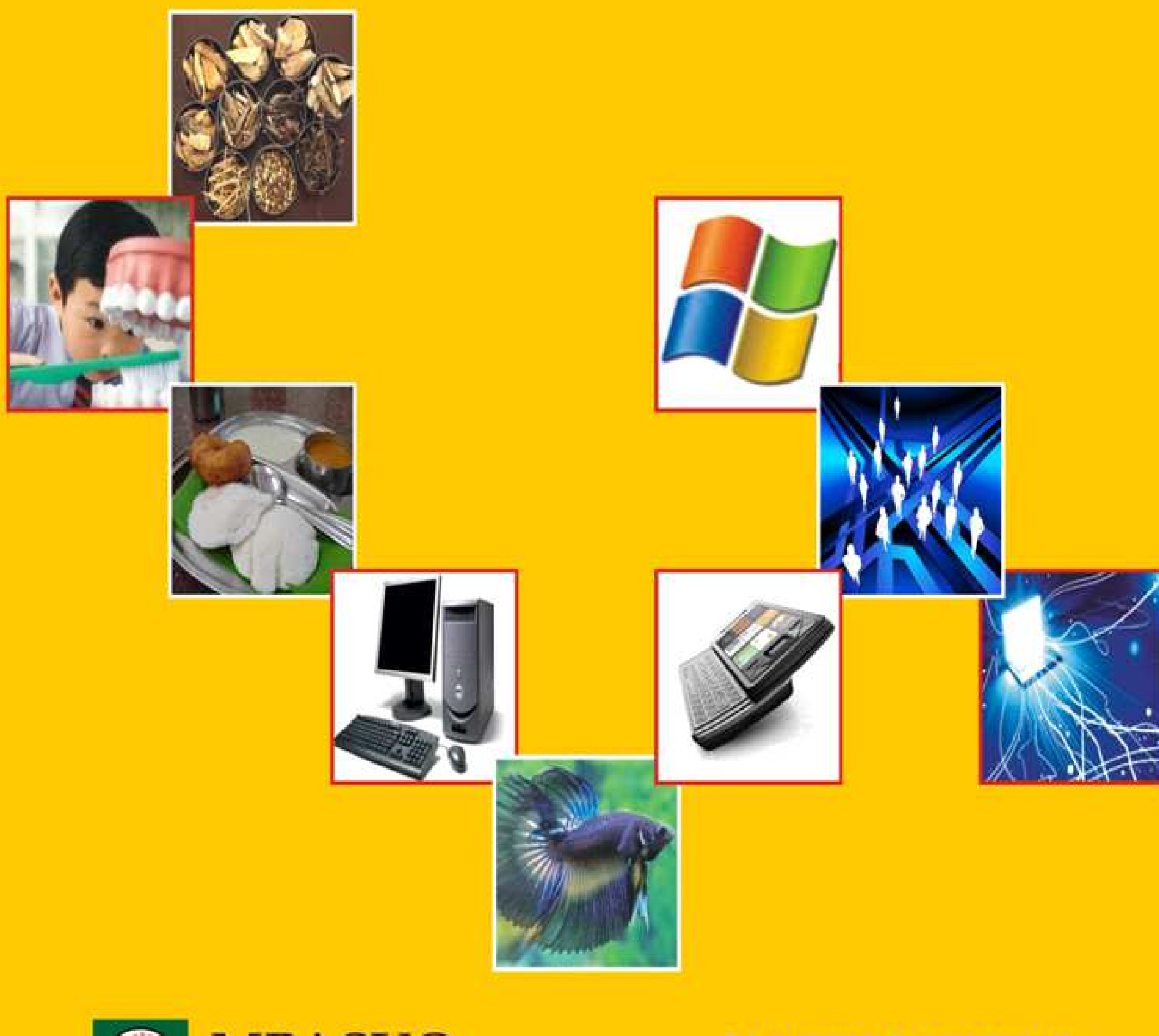
Volume-6 Number-4 July-August 2012

The Indian Journal of Research

ANVIKSHIKI

Bi-monthly International Journal of all Research

Science



MPASVO

Published on behalf of the MPASVO in association with the Member's of Anvikshiki

MANEESHA PUBLICATIONS

www.anvikshikijournal.com

Anvikshiki
The Indian Journal of Research

Bi-Monthly International Journal of All Research

Abstracts and Indexing

http//nkrc.niscair.res.in/browseByTitle.php?Keword=A, ICMJE ICMJE ,www.icmje.org, Academia.edu, banaras.academia.edu, ebookbrowse.com, BitLibrary! http:/ www.bitlib.net/, Tech eBooks ,freetechebooks.com, ARTAPP.NET ,artapp.net,Catechu PDF / printfu.org, File Away,www.fileaway.info, KMLE 의학 검색 엔진 ,docslibrary,http://www.docslibrary.com, MyCelular.ORG ,Android Tips, Apps, Theme and Phone Reviews http://dandroidtips.com, Edu-Doc, http://www.edu-doc.com, www.themarketingcorp.com,Dunia Ebook Gratis duniaebook.net, DOC CAFES,www.cn.doc-cafes.com., Google scholar ,http://scholar.google.co.in, Website : www.onlineijra.com.Motilal Banarasi Das Index,Varanasi, Motilal Banarasi Das Index,Delhi. Banaras Hindu University Journal Index,Varanasi. www.bhu.ac.in, D.K.Publication Index, Delhi. National Institute of Science Communication and Information Resources Index, New Delhi.

Subscriptions

Anvikshiki,The Indian Journal of Research is Published every two months (January,March,May,July,September and November) by mpasvo Press,Varanasi.u.p.India. A Subscription to The Indian Journal of Research : Anvikshiki Comprises 6 Issues in Hindi and 6 in English and 3 Extra Issues. Prices include Postage by Surface mail,or For Subscription in the India by Speed Post. Airmail rates are also available on request. Annual Subscriptions Rates (Volume 3,6 Issues in Hindi,6 Issues in English and 6 Issues of science 2012):

Subscribers

Institutional : Inland 4,000 +500 Rs. P.C., Single 500+51 Rs.P.C.,Overseas 6000+2000Rs. P.C., Single 1000+500 Rs.P. C.

Personal : 2,500+500 Rs. P.C., Single 500+51 Rs. P.C., Overseas 5000+2000Rs.P.C., Single 1000+500Rs. P.C.

Advertising & Appeal

Inquiries about advertising should be sent to editor's address. Anvikshiki is a self financed Journal and support through any kind or cash shall be highly appreciated. Membership or subscription fees may be submitted via demand draft in faver of Dr. Maneesha Shukla and should be sent at the address given below. Sbi core banking cheques will also be accepted.

All correspondence related to the Journal should be addressed to

B.32/16 A., Flat No.2/1,Gopalkunj,Nariya,Lanka, Varanasi, U.P.,India
Mobile : 09935784387,Tel.0542-2310539.,e-mail : maneeshashukla76@rediffmail.com,www.anvikshikijournal.com
Office Time : 3-5 P.M.(Sunday off)

Journal set by

Maheshwar Shukla,maheshwar.shukla@rediffmail.com
9415614090

Printed by

mpasvo Press

Maneesha Publication
(Letter No.V-34564,Reg.533/2007-2008)
B-32/16-A-2/1,Gopalkunj,Nariya,Lanka
Varanasi,U.P.,India

Anvikshiki
The Indian Journal of Research
Volume 6 Number 4 July 2012

Science
Papers

PRINT ISSN 0973-9777,WEBSITE ISSN 0973-9777

Letter No.V-34564,Reg.533/2007-2008 INDIAN JOURNAL OF RESEARCH(2012)**6,77-84**
ANVIKSHIKI ISSN 0973-9777 Advance Access publication 26 Apr. 2012

WEB SERVICE DISCOVERY SYSTEMS FOCUSING ON SYSTEMS THAT SUPPORT QOS

VIKAS PORWAL*

Declaration

The Declaration of the author for publication of Research Paper in The Indian Journal of Research Anvikshiki ISSN 0973-9777 Bi-monthly International Journal of all Research: I, *Hemant Kumar Chaudhary* the author of the research paper entitled A NOVEL METHODOLOGY FOR SOFTWARE TEAMBUILDING USING ANNS declare that , I take the responsibility of the content and material of my paper as I myself have written it and also have read the manuscript of my paper carefully. Also, I hereby give my consent to publish my paper in Anvikshiki journal , This research paper is my original work and no part of it or it's similar version is published or has been sent for publication anywhere else. I authorise the Editorial Board of the Journal to modify and edit the manuscript. I also give my consent to the Editor of Anvikshiki Journal to own the copyright of my research paper.

Abstract

Service-oriented Architecture supports software to be composed from services dynamically. With the growing popularity of web services, web services standards enable the development of large-scale applications in open environments. To deploy web services, quality of service (QoS) becomes a significant concern for service consumers and providers for service discovery. QoS-based web service discovery mechanisms will play an essential role in service-oriented architectures, as e-Business applications want to use services that most accurately meet their requirements. However, representing and storing the values of QoS attributes are problematic, as the current Universal Description, Discovery and Integration (UDDI) was not designed to accommodate these emerging requirements. To solve the problems of storing QoS in UDDI and aggregating QoS values through the tModel approach. The aim is to study these approaches and other existing QoS tModel representation for their efficiency and consistency in service discovery. In this paper, we present an overview of web service discovery systems, focusing on systems that support QoS. The article also elaborates on open issues relating to such discovery systems.

Keywords: UDDI, Web Services Discovery, QoS, Web Service Broker, tModel

1. Introduction

Web service technology enables e-business and e-commerce to become a reality. It has become a competitive tool of companies by reducing cost through fast, effective, and reliable services to customers, suppliers, and partners over the Internet. It enables more efficient business operations via the Web and enhances business opportunities to companies. The Service Oriented Architecture (SOA) is "an

*M.Tech in C.S., Singhania University (Rajasthan) India.

architecture that represents software functionality as discoverable services on the network"[4]. As an implementation of SOA, web services are defined as a set of standards [3], SOAP (Simple Object Access Protocol), UDDI, WSDL (Web Services Description Language), which enable a flexible way for applications to interact with each other over networks. SOAP is the protocol for applications to communicate with each other. UDDI defines a registry for service providers to publish their services. WSDL is used to describe a web service's capabilities and the interface to invoke it.

A WSDL document is self-describing so that a service consumer can examine the functionality of the web service at runtime and generate corresponding code to automatically invoke the service. All these standards are XML-based, which allows applications to interact with each other over networks, no matter what languages and platforms they are using. The two features, self-description and language-independence or platform-independence, distinguish web services from other distributed computing technologies, like CORBA and DCOM. Web services have won an increasing popularity.

Fundamentally, standards like WSDL can support service providers describe their services' functionality through standard interfaces and advertise them in some UDDI registries. When a service request is issued, available Web service descriptions and the service request description is matched together in order to find the services that can provide expected functionality (the matching step). However, Web service functional descriptions are not sufficient for service discovery process. There are several reasons for this fact. *Firstly*, a key advantage of Web service technology is to enable Web services to be dynamically and automatically discovered and selected at runtime. In this case, an automatic mechanism is needed to support a system determine the best services to be chosen. *Secondly*, as more and more Web services are created by many providers and vendors, there is often the case where a number of Web services can satisfy functional requirements of a service request. Those reasons lead to the issue of ranking and selecting of the best Web services for a request among a list of candidate Web services which can provide similar functionality for the request. Primarily, QoS information is used for computing the quality degree of candidate Web services. Such QoS information can be performance (in terms of response time, latency…), availability, accessibility, security, etc. [14]. These QoS information have substantial impacts on user's expectation and experience of using a Web service. Therefore they can be used as a main factor to distinguish quality of Web services. A Web service with highest ranking value will be selected. This ranking step is often performed after the above matching step. Because different service providers and requesters may use different languages [5,6,7,8] and models [9,10,11,12,13] for QoS advertisements and requirements, it is necessary to find a way for a system understanding different QoS concepts in QoS descriptions. Besides that, different domains and applications may require different QoS properties; therefore we need a more efficient and flexible method to express QoS information. A survey of Web service discovery system is needed to explore existing techniques and to highlight the advantages and disadvantages of each system.

The rest of the article is as follows. The section 2 introduces the background of Web service model. The section 3 introduces UDDI registry with QoS. The section 4 presents the issues related to Web services discovery, followed by the conclusion in the section.5.

2. Web Service Model

In a Web service model, a *service provider* offers Web services which provide functions or business operations which can be deployed over the Internet, in the hope that they will be invoked by partners or customers; a Web *service requester* describes requirements in order to locate *service providers*. Publishing, binding, and discovering Web services are three major 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.

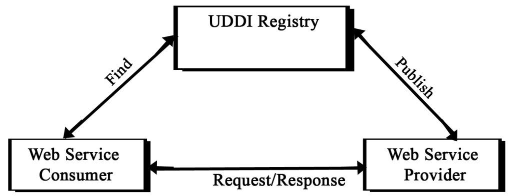


Figure 1. Web Service Model

The Web service model in figure 1 shows the interaction between a service requester, service providers, and a service discovery system.

⇒ The service providers offer Web services which provide functions or business operations. They are created by companies or organizations. In order to be invoked, the Web services must be described. This will facilitate discovery and composition. WSDL or service profile of semantic Web service is used to carry out this function.

⇒ The Web service requester describes 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 usually described by WSDL, service template or service profile.

⇒ The Web service discovery or service registry is a broker that provides registry and search functions. The service providers advertise their service information in the discovery system. This information will be stored in the registry and will be searched when there is a request from service requester. UDDI is used as a registry standard for Web service.

3. The Uddi Registry

A UDDI registry is a directory for storing information about Web services. A service provider makes its services available to public users by publishing information about the service in a UDDI registry. Individuals and businesses can then locate the services by searching public and private registries. For example, airlines can publish their fare services to a UDDI registry. Travel agencies then use the UDDI registry to locate Web services provided by different airlines, and to communicate with the service that best meets their requirements. The information about Web services in a UDDI registry includes a description of the business and organizations that provide the services, a description of a service's business function, and a description of the technical interfaces to access and manage those services. A UDDI registry consists of instances of four core data structures including the *businessEntity*, the *businessService*, the *bindingTemplate* and the *tModel*. This information comprises everything a user needs to know to use a particular Web service. The *businessService* is a description of a service's business function, *businessEntity* describes the information about the organization that published the service, *bindingTemplate* describes the service's technical details, including a reference to the service's programmatic interface or API, and *tModel* defines various other attributes or metadata such as taxonomy and digital signatures.

3.1 Storage of QoS Information in the UDDI Registry

As a current feature in the UDDI registry, the *tModel* is used to describe the technical information for services. A *tModel* consists of a key, a name, an optional description and a Uniform Resource Locator (URL) which points to a place where details about the actual concept represented by the *tModel* can be found. *tModel*s play two roles in the current UDDI registries. The primary role of a *tModel* is to represent a technical specification that is used to describe the Web services. The other role of a *tModel* is to register categorizations, which provides an extensible mechanism for adding property information to a UDDI registry. Blum [2,15] proposes that the categorization *tModels* in UDDI registries can be used to provide QoS information on *bindingTemplates*. In the proposal, a *tModel* for quality of service information for the binding template that represents a Web service deployment is generated to represent quality of service information. Each QoS metric, such as average response time or average throughput, is represented by a *keyedReference*, that is a general- purpose structure for a namevalue pair, on the generated *tModel*. Blum gives an example of the *bindingTemplate* reference to the *tModel* with the QoS attribute categories, and an example of the QoS Information *tModel*, which contains a *categoryBag* , which is a list of name-value pairs specifying QoS metrics. Figure 2 shows the *tModel* with the QoS Information. This *tModel* contains a *categoryBag* that specifies three QoS metrics of Average ResponseTime, Average Throughput and Average Reliability. The *tModelKey* in each *keyedReference* is used as a namespace which provides a uniform naming scheme.

```
<tModel tModelKey=”mycompany.com:StockQuoteService:
PrimaryBinding:QoSInformation”” >
<name>QoS Information for Stock Quote Service</name>
<overviewDoc>
<overviewURL>
http://<URL describing schema of QoS attributes>
<overviewURL>
<overviewDoc>
<categoryBag>
<keyedReference
tModelKey=”uddi:uddi.org:QoS:ResponseTime”
keyName=”Average ResponseTime”
keyValue=”fast” />
<keyedReference
tModelKey=”uddi:uddi.org:QoS:Throughput”
keyName=”Average Throughput”
keyValue=”>10Mbps” />
<keyedReference
tModelKey=”uddi:uddi.org:QoS:Reliability”
keyName=”Average Reliability”
keyValue=”99.9%” />
</categoryBag>
</tModel>
```

Figure 2. The *tModel* with the QoS Information [2]

4. Web Services Discovery

Web services discovery is “the act of locating a machine-processable description of a Web service that may have been previously unknown and that meets certain functional criteria”. The goal is to find appropriate Web services that match a set of user requirements. A discovery service, which could be performed by either a consumer agent or a provider agent, is needed to facilitate the discovery process. There are three leading approaches on how a discovery service should be designed: a registry, an index, or a peer-to-peer (P2P) system. Their differences are discussed in the following section.

4.1 Discovery: Registry, Index and P2P Approaches

A registry is an authoritative, centrally controlled repository of services information. Service providers must publish the information of their services before they are available to consumers. The registry owner decides who has the authority to publish and update services information. A company is not able to publish or update the information of services provided by another company. The registry owner decides what information can be published in the registry. UDDI is an example of this approach. Centralized registries are appropriate in static or a controlled environment where information does not change frequently. An index is a collection of published information by the service providers. It is not authoritative and the information is not centrally controlled. Anyone or any company can create their own index, which collects information of services exposed on the web usually using web spiders. The information in an index could be out of date but can be verified before use. Google is an example of the index approach. P2P computing provides a de-centralized alternative that allows Web services to discover each other dynamically. Each Web service is a node in a network of peers. At discovery time, a Web service queries its neighbors in search of a suitable Web service. If any one of its neighboring peers matches its requirements, it replies and the query is ended. Otherwise, the query is propagated through the network until a suitable Web service is found or certain termination criteria are reached. P2P architecture is more reliable than registry approach since it does not need a centralized registry, but introduces more performance costs since most of time a node acts as a re-layer of information.

4.2 Manual versus Autonomous Discovery

Depending on who is actually performing the discovery, service discovery could be manual or autonomous. Manual discovery is typically done at design time and involves a human service consumer that uses a discovery service to find services that match its requirements. Autonomous discovery involves a discovery agent to perform this task at design time or run time. One situation in which autonomous discovery is needed is when a service consumer needs to switch to another service because the current service is either no longer available or cannot satisfy its requirements anymore.

4.3 Research on Web Services Discovery with QoS

QoS is “a combination of several qualities or properties of a service”. It is a set of non-functional attributes that may influence the quality of the service provided by a Web service. Some examples of the QoS attributes are given below:

⇒ *Availability* is the probability that system is up and can respond to consumer requests. Generally it is slightly parallel to reliability and slightly opposite to capability.

⇒ *Capacity* is the limit of concurrent requests a service can handle. When the number of concurrent requests exceeds the capacity of a service, its availability and reliability decrease.

⇒ *Reliability* is the ability of a service to perform its required functions under stated conditions for a specific period of time.

⇒ *Performance* is the measure of the speed to complete a service request. It is measured by latency (the delay between the arrival and completion of a service request), throughput (the number of requests completed over a period of time) and response time (the delay from the request to getting a response from the service).

⇒ *Cost* is the measure of the cost of requesting a service. It may be charged per the number of service requests, or could be a flat rate charged for a period of time.

The QoS requirements for Web services are more important for both service providers and consumers since the number of Web services providing similar functionalities is increasing. Current Web service technologies such as WSDL and UDDI, which are for publishing and discovering Web services, consider only customer functionality requirements and support design time, or static service discovery. Nonfunctional requirements, such as QoS, are not supported by current UDDI registries. The web service selection and ranking mechanism uses the QoS broker based architecture [17]. The QoS broker is responsible for selection and ranking of functionally similar web services. The web service selection mechanism [16] ranks the web services based on prospective levels of satisfaction of requester's QoS constraints and preferences. QoS can be used to select and rank the Web services by extending standard service oriented architecture (SOA). In this architecture, the Web service is selected by matching requested QoS property values against the potential Web service QoS property values. In literature, the Web service is selected by taking the requester's average preference for QoS properties. In Maximilien propose an agent ontology supported framework for dynamic Semantic Web services selection. Service consumers and providers are represented as agents, QoS data about different services are collected from agents, and then shared by other agents (knowledge sharing). This agent-based framework is implemented in the Web Services Agent Framework (WSAF). The WS-QoSMan applies an external resource approach in which it uses a tModel called QoSMetrics that contains information to an external reference . This is very similar to the tModel used for pointing to WSDL files. QoSMetrics uses overview URL to point to an XML-based file generated by WS-QoSMan and that contains QoSMetrics for a specific web service.

In , authors present a description and an implementation of broker-based architecture for controlling QoS of web services. The broker acts as an intermediary third party to make web services selection and QoS negotiation on behalf of the client. Delegation of selection and negotiation raises trustworthiness issues mainly for clients. Performance of the broker is not considered in this approach. Moreover, performance of the broker can be a key to the success of any proposed architecture; if the user does not get a response to his/her request with an acceptable response time, he/she will switch to another provider. Some similar broker based architectures were presented in and that focus more on the QoS specification using XML schema, and dynamic QoS mapping between server and network performance. Maximilien and Singh propose an agent framework and ontology for dynamic Web services selection.

Service quality can be determined collaboratively by participating service consumers and agents via the agent framework. Service consumers and providers are represented and service-based software applications are dynamically configured by agents. QoS data about different services are collected from agents, aggregated, and then shared by agents. This agent-based framework is implemented in the Web Services Agent Framework (WSAF). A QoS ontology, which captures and defines the most generic quality concepts, is proposed in their paper.

Adam Blum and Fred Carter [1,2] present four different QoS storing methods in UDDI by utilizing tModels. The first method [1] employs a QoSInformation tModel referring to a QoS file which is not a part of UDDI. The location of the QoS file is stored in the overviewURL of the QoSInformation tModel which is represented in XML, so the detailed QoS information, including QoS values and QoS description, is stored in an XML file. Each bindingTemplate contains one QoSInformation tModel and adds the QoSInformation tModel to the tModelInstanceDetail. The method provides a set of APIs, such as save_business, save_service and save_tModel, to store QoS values. To obtain the desired QoS values requires the built-in APIs (i.e. find_tModel, find_service) to acquire the QoS values in the XML file. The second method [1] creates many different QoS tModels for various QoS information. These categories are added to the bindingTemplates. Each categoryBag has multiple keyedReferences, which represent different types of QoS. Each QoS value is stored in the keyValue of the keyedReference. Apart from the aforementioned API supported by this approach to store the QoS values, two functions are provided (i.e. save_binding and save_tModel) to increase the system flexibility. The third method [1] is similar to the first one by utilizing the QoSInformation tModel, but it contains a bindingTemplate. The categoryBag of QoSInformation tModel has many keyReferences, which represent various types of QoS. The APIs supported by this method is similar to the second approach. The fourth method [2] stores the QoS values in the categoryBag of businessService in UDDI. The method needs save_business, save_tModel and save_service to store QoS values. The method requires find_service to search the Web service according the QoS value. These studies have provided the researchers with better understanding and the usage of UDDI to manage Web Service QoS.

Many researchers work on how to take QoS information for Web services into account in the service discovery process to find services that best meet a customer's requirements. Ran [9] proposes a model in which the traditional service discovery model is extended with a new role called a Certifier, in addition to the existing three roles of Service Provider, Service Consumer and UDDI Registry. The Certifier verifies the advertised QoS of a Web service before its registration. The consumer can also verify the advertised QoS with the Certifier before binding to a Web service. This system can prevent service providers from publishing invalid QoS claims during the registration phase, and help consumers to verify the QoS claims to assure satisfactory transactions with the service providers. Although this model incorporates QoS into the UDDI, it does not provide a matching and ranking algorithm, nor does it integrate consumer feedback into service discovery process. Gouscos et al.[17] propose a simple approach to dynamic Web services discovery that models Web service management attributes such as QoS and price. They discuss how this simple model can be accommodated and exploited within basic specification standards such as WSDL. The key Web service quality and price attributes are identified and categorized into two groups, static and dynamic.

The Price, Promised Service Response Time (SRT) and Promised Probability of Failure (PoF) are considered as static in nature and could be accommodated in the UDDI registry. The actual QoS values that are the actual SRT and PoF are subject to dynamic updates and could be stored either in the UDDI registry or in the WSDL document, or could be inferred at run time through a proposed information broker. The advantage of this model is its low complexity and potential for straightforward implementation over existing standards such as WSLA and WS-Policy specifications.

In, DAML-QoS ontology is proposed as a complement for the DAMLS ontology to provide a better QoS metrics model. QoS requirements and various constraints can be specified explicitly and precisely using this novel ontology. Although these works address some form of service discovery with QoS, none considers feedback from consumers. The result of service discovery and selection is based solely on advertised QoS, which may be invalid (in the one case though, advertised QoS is verified by the Certifier in the model proposed by Ran[9]). In , web service architecture employs an extended UDDI registry to support service selection based on QoS, but only the certification approach is used to verify QoS and no information is provided about the QoS specification.

5. Conclusions

QoS plays an important role in Web service selection in order to evaluate and rank candidate Web services that are able to provide expected functionality. Similarly to the problem of Web service description; there may be various QoS models which are adopted by different service providers and service requestors for describing QoS information. It is necessary to match different QoS concepts such as QoS properties, metrics, and units which are specified in a QoS advertisement and a QoS requirement. A number of models for web service discovery with QoS have been developed recently. This paper reviews those approaches, analyze their advantages and shortcomings as well as indicate open issues that need further investigation in ongoing works aiming at developing models and related algorithms for matching and ranking Web services based on QoS.

References

[1] Adam Blum & Fred Carter, *"Representing Web Services Management Information"*, Available: http://www.oasis-open.org/committees/ download.php/5144/

[2] Adam Blum, *"UDDI as an Extended Web Services Registry "*, SOA WORLD MAGAZINE, Available: http://webservices.sys-con.com/ read/ 45102.htm

[3] Walsh, A. E., *"UDDI, SOAP, and WSDL:The Web Services Specification"*, Reference Book (1st edition ed.) 0130857262: Pearson Education, 2002.

[4] W Han, *"Integrating Peer-to-Peer into Web Services"*, Master thesis, University of Saskatchewan, 2006

[5] S. Frolund & J. Koisten, *"QML: A Language for Quality of Service Specification"*, Hewlett-Packard, http://www.hpl.hp.com/techreports/98/ HPL-98-10.html, 1998.

[6] A. Dan, *"Web Service Level Agreement (WSLA) Specification"*, http://www.research.ibm.com/ wsla/, 2002.

[7] A. Sahai, A. Durante & V. Machiraju, *"Towards Automated SLA Management for Web Services"*, HP, www.hpl.hp.com/techreports/ 2001/ HPL-2001-310R1.pdf, 2002.

[8] V. Tosic, B. Pagurek & K. Patel, *"WSOL: Web Service Offerings Language"*, In the book of Web Services, E-Business, and the Semantic Web, LNCS, Springer- Verlag, pp. 57-67, 2003.

[9] S. P. Ran, *"A Model for Web Service Discovery with QoS"*, ACM SIGecom Exchanges, Vol. 4, No. 1, ACM Press, pp. 1–10, 2003.

[10] M. Matinlassi & E. Niemela, *"The Impact of Maintainability on Component-based Software Systems"*, In Proc. of the 29th Euromicro Conference (EUROMICRO.03), Turkey, 2003.

[11] B. Sabata, S. Chatterjee, M. Davis, J. J. Sydir & T. F. Lawrence, *"Taxonomy for QoS Specifications"*, In Proc. of the 3rd International Workshop on Object-Oriented Real-Time Dependable Systems, USA, 1997.

[12] L. Chung, B. A. Nixon, E. Yu & J. Mylopoulos, *"Non-Functional Requirements in Software Engineering"*, In Kluwer Academic Publishing, 2000.

[13] S. W. Choi, J. S. Her & S. D. Kim, *"QoS Metrics for Evaluating Services from the Perspective of Service Providers"*, In Proc. of the IEEE International Conference on e-Business Engineering, ACM, pp. 622–625, 2007.

[14] J. C. Laprie, B. Randell & C. Landwehr, *"Basic Concepts and Taxonomy of Dependable and Secure Computing"*, In IEEE Transactions on Dependable & Secure Computing. Vol. 1, No. 1, pp. 11–33, 2004.

[15] A.Blum,*"Extending UDDI with Robust Web Services Information"*. Retrieved from http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci952129,00.html, 2002.

[16] A. Demian, D'mello & V. S. Ananthanarayana, *"A QoS Model and Selection Mechanism for QoS-Aware Web Services"*, Proceedings of the International Conference on Data Management (ICDM2008), 2008.

[17] D.Gouscos, M. Kalikakis & P. Georgiadis, *"An Approach to Modeling Web Service QoS and Provision Price"*. Proceedings. of the 1st International Web Services Quality Workshop - WQW 2003 at WISE 2003, Rome, Italy, pp.1-10, 2003

Note for Contributors

SUBMISSION OF PAPERS

Contributions should be sent by email to Dr. Maneesha Shukla Editor-in-Chief, Anvikshiki, The Indian Journal of Research (maneeshashukla76@rediffmail.com). www.onlineijra.com

Papers are reviewed on the understanding that they are submitted solely to this Journal. If accepted, they may not be published elsewhere in full or in part without the Editor-in-Chief's permission. Please save your manuscript into the following separate files-***Title; Abstract; Manuscript; Appendix.*** To ensure anonymity in the review process, do not include the names of authors or institution in the abstract or body of the manuscript.

Title**:** This title should include the manuscript, full names of the authors, the name and address of the institution from which the work originates the telephone number, fax number and e-mail address of the corresponding author. It must also include an exact word count of the paper.

Abstract**:** This file should contain a short abstract of no more than 120 words.

MANUSCRIPT: This file should contain the main body of the manuscript. Paper should be between 5 to 10 pages in lenth,and should include only such reviews of the literature as are relevant to the argument. An exact word count must be given on the title page. Papers longer than 10 pages (including *abstracts*, *appendices and references*) will not be considered for publication. Undue length will lead to delay in publication. Authors are reminded that Journal readership is abroad and international and papers should be drafted with this in mind.

References should be listed alphabetically at the end of the paper, giving the name of journals in full. Authors must check that references that appear in the text also appear in the References and *vice versa*. Title of book and journals should be italicised.

Examples**:**

BLUMSTEIN,A.and COHEN,J.(1973),'A Theory of Punishment' *Journal of Criminal Law and Criminology*,64:198-207

GUPTA,RAJKUMAR(2009),*A Study of The Ethnic Minority in Trinidad in The Perspective of Trinidad Indian's Attempt to Preserve Indian Culture*, India: Maneesha Publication,

RICHARDSON,G(1985),Judicial Intervention in Prison Life', in M. Maguire ,J. Vagg and R. Morgan, eds., *Accountability and Prisons*,113-54.London:Tavistocs.

SINGH,ANITA.(2007),*My Ten Short Stories*,113-154.India:Maneesha Publication.

In the text,the name of the author and date of publication should be cited as in the Harvard system(e.g.Garland 1981: 41-2;Robertson and Taylor 1973;ii.357-9)If there are more than two authors, the first name followed by *et al.* is manadatory in the text,but the name should be spelt out in full in the References. Where authors cite them as XXXX+date of publication.

Diagrams and tables are expensive of space and should be used sparingly. All diagrams, figures and tables should be in black and white, numbered and should be referred to in the text.They should be placed at the end of the manuscript with there preferred location indication in the manuscript(e.g.Figure 1 here).

Appendix: Authors that employ mathematical modelling or complex statistics should place the mathematics in a technical appendix.

NOTE : Please submit your paper either by post or e-mail along with your photo, bio-data, e-mail Id and a self-addressed envelop with a revenue stamp worth Rs.51 affixed on it. One hard copy along with the CD should also be sent.A self-addressed envelop with revenue stamp affixed on it should also be sent for getting the acceptance letter. Contributors submitting their papers through e-mail, will be sent the acceptance letter through the same. Editorial Board's decision will be communicated within a week of the receipt of the paper. For more information, please contact on my mobile before submitting the paper. All decisions regarding members on Editorial board or Advisory board Membership will rest with the Editor. Every member must make 20 members for Anvikshiki in one year. For getting the copies of 'Reprints', kindly inform before the publication of the Journal. In this regard, the fees will be charged from the author.

Search Research papers of The Indian Journal of Research Anvikshiki-ISSN 0973-9777 in the Websites given below

http://nkrc.niscair.res.in/BrowseByTitle.php?keyword=A

National Knowledge Resource Consortium

www.icmje.org

www.scholar.google.co.in

www.kmle.co.kr

www.fileaway.info

www.banaras.academia.edu

www.www.edu-doc.com

www.docslibrary.com

www.dandroidtips.com

www.printfu.org

www.cn.doc-cafes.com

www.freetechebooks.com

www.google.com

IJRA

www.onlineijra.com

ISSN 0973-9777

09739777

₹ 1500/-