Evaluating the second generation Web services specifications for satisfying non-functional requirements

Manoj Lall
School of Computing
PO Box 392, UNISA, 0003

Lucas M. Venter
North West University

John A. van der Poll
School of Computing
PO Box 392, UNISA, 0003

ABSTRACT

Web services have enjoyed rapid acceptance in recent years. One of the motivating factors for this is the reliance on open standards for loose coupling and platform independent interface definition. Besides satisfying the functional requirements of an application, a web service also has to cater for the equally important non-functional requirements (NFRs) such as availability and reliability. Though Web services specifications such as WSDL, SOAP and UDDI have been around for quite some time, they however, lack the ability to adequately provide for the NFRs associated with a specification. Second-generation Web services specifications have emerged in an attempt to alleviate these and other short-comings. This paper evaluates these second generation Web services specifications to determine its ability to cater for the NFRs which can then be used to identify the areas that need further attention with respect to creating specifications, stimulating discussion around non-functional properties of Web services, and encouraging formal treatments to those specifications.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Online Information Systems - *Online Information Storage Retrieval, Information Services, Web-Based Services*

General Terms

Theory

Key Words

Web services, WSDL, SOAP, UDDI, Non-functional requirements.

1. INTRODUCTION

An overwhelming number of organizations have and continue to reap the benefits of the web by making their applications available to their customers and partners through a combination of interactive interfaces and dynamically generated web pages. The next step in the evolution of Web technologies is the emergence of Web services [1]. There are several definitions of Web services, but most agree on stating that Web services are software entities available on the Web (through a URI), communicating through XML messages over a Internet transport protocol and whose capabilities and modus operandi are described in XML [5].

In comparison to its predecessor technologies i.e. CORBA, DCOM or RMI, Web services are seen as a new model for distributed computing. Web services offer a programming model for building distributed applications using open Internet standards. Basing it on open Internet standards, Web services address many of the interoperability issues associated with CORBA and DCOM. Unlike previous concepts in distributed computing - objects or components, Web services are more modular and self-contained. It offers reusable functionality that is contractually defined in a service description. Each service provides access to a well-defined collection of functionality and the system as a whole is designed and implemented as a set of interactions amongst these services [6].

Despite much progress in creating component based and service oriented solutions, the problem of addressing the non-functional properties of distributed applications is far from being solved [19], [26]. Functional requirements define what software is expected to do whereas NFRs specify global constraints that must be satisfied by the software [10], [23]. In the context of Web services, considerations of non-functional attributes are critical for a number of reasons: firstly, autonomous services depend on one another for their functioning and they need to be aware of the Quality of service (QoS) of the services involved; secondly, a service requestor may decide on the use of a particular service depending on its non-functional properties. Similarly, a service provider may offer the same function with different QoS's, for example, different prices [1], [14]. Although it is clear that the NFRs are as important, if not more, they are often ignored until the end of the development cycle or neglected altogether [26] - reasons often cited for this is that they are difficult to define and represent precisely [20], [18]. For a pragmatic web

service, it is important that it meets **both** the functional and NFRs for its customers.

This paper analyzes the non-functional characteristics of Web service applications at three distinct levels. It identifies the key NFRs of the messaging framework, the NFRs of the Web services interface, and the NFRs of the services. The major second generation Web services specifications are scrutinized against the identified NFRs. This can then be used to identify the areas that need further attention with respect to creating specifications, stimulating discussion around non-functional properties of Web services and encouraging formal treatments to those specifications.

The core Web services standards such as SOAP (Simple Object Access Protocol), WSDL(Web Services Description Language), and UDDI(Universal Description, Discovery, and Integration) provide a solid foundation for integrating loosely coupled web-based components that expose well-defined interfaces, while abstracting the implementation and platform specific details. These specifications are adequate for development of simple Web services applications that can conduct simple interactions but fall short in its ability to compose and integrate complex business-to-business applications. No matter how simple or complex the web service application may be, its basic ingredients are the services themselves, a means for describing these services, and a messaging framework to facilitate communications between the services. We use these three components to identify the major NFRs.

2. CATEGORIZATION OF NFRs:

The NFRs can be categorized in several ways. One such categorization is obtained by analyzing the domain in which the Web services operate. In this categorization, the non-functional attributes may either be grouped as domain specific or generic. The generic attributes reflect the characteristics that are common to all services such as response time, availability, and reliability whereas, domain specific non-functional attributes are business context oriented, e.g. the customer satisfaction index could be one such NFR for a hotel reservation service [21].

Another approach to categorizing NFRs is by analyzing Web services from the perspectives of the service provider and the service user. From the providers point of view characteristics such as security, accessibility, and performance may be important NFRs whereas the users may see cost, usability, and reputation as important NFRs. Similarly, different categories of NFRs can be obtained if runtime characteristics are considered or configuration management characteristics are analyzed. For the purpose of this research, we have adopted the categorization as presented in [12] and [15]. In this categorization, the NFRs are classified either as deterministic or as non-deterministic. The former indicates that the NFRs are known before a service is evoked, such as cost of invocation while the latter indicates the absence of any such certainty, for example service reliability. Using this categorization mechanism, we present in section 3, the NFRs of the services, followed by NFRs of the service interface in section 4, and in section 5, we present the NFRs as applicable to the messaging framework.

3. NFRs OF SERVICES

As mentioned earlier, each service encapsulates a small, distinct unit of well defined business logic capable of existing autonomously. Though, these units of business logic have the ability to exist separately, they are required to conform to a set of design principles to ensure their use and enhance their reusability. For example, a design principle to enhance the reuse of a service would be that it be as loosely coupled as possible with other components of the application for which it was designed. This can be achieved by requiring the services to minimize retaining information specific to an activity or of each other thereby reducing dependencies. Reuse can also be improved by requiring the services to be designed such that the logic governed by a service reside within an explicit boundary and that the services have complete control over the logic they encapsulates [9].

Related to the principle of reusability of Web services is the principle of extensibility. Services should be designed in a manner such that it is easy to extend its functionality. Extensibility of the functionality is in-turn supported by the encapsulation of the underlying logic. Another important characteristics of applications based on Web services is that the services be designed such that they can be discovered be other entities involved in that application. In other words, the services should be designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanisms.

The design principles mentioned above have a great influence on satisfying the NFRs of the applications. As it is not possible to consider all possible NFRs, our efforts are focused on the non deterministic NFRs that could be applied to all Web services applications. The paragraphs below list and define these NFRs.

- **Composibility:** Typically, a Web service application makes use of several services each encapsulating a small, distinct and well defined unit of business logic. Hence, it is expected of services to be capable of participating as effective composition members, regardless of whether they need to be immediately enlisted in a composition. Besides the influence exerted by the design principles mentioned above, composibility is improved by complying with WSDL specifications.

- **Availability**: Availability is the absence of service downtime. It signifies the probability of a service being present and ready for immediate use [17]. Following the URL from the WSDL does not necessarily lead to a Web service available to satisfy the request.

- **Accessibility**: For a service to be accessible, it has to be available. Just because a service is available does not imply that it is accessible too. Accessibility refers to the quality aspect of a service that represents the degree and ease with which it is capable of serving a Web service request. Accessibility of Web services can be improved by building more scalable systems.

- **Scalability**: Scalability refers to the ability of Web services to handle growth in request volumes efficiently. This could be achieved by having more than one instance of the same service or having more than one server to cater for the growth in demand.

- **Integrity**: Integrity is the quality aspect of how the Web service maintains the correctness of an interaction with respect to its description in the WSDL and conformance with its Service-Level Agreement. A higher measure of integrity means that the functionality of a service is in line with its intended use.

- **Reliability**: Reliability refers to the assured and ordered delivery for messages being exchanged between the service requestors and service providers. It represents the degree to which Web services are capable of maintaining the service and service quality in spite of system or network failure.

- **Interoperability**: The goals of Web service interoperability are to provide seamless and automatic connections between software applications written in different language or deployed on different platforms. In addition, interoperability may be improved by providing additional information (metadata) that supplement the information provided by the WSDL.

- **Robustness**: Characterizes the behaviour of Web services under erroneous input conditions. Having a robust service will be helpful to users in picking up the web service that best meets their requirements and help providers in selecting the best services for a given composition [23]. Robustness is supported by the application server infrastructure and features such as exception handling mechanisms of the application software.

- **Performance**: Performance is the quality aspect of Web service, which is measured in terms of throughput and latency. Higher throughput and lower latency values represent good performance of a Web service. *Throughput* represents the number of Web service requests served at a given time period. *Latency* is the round-trip time between sending a request and receiving the response. Performance is influenced by factors such as the design choices of the services, the infrastructure on which the Web services are operating, and the scalability of the services.

- **Security and trust**: As the invocation of Web service invocation occurs over the public Internet, security is of a paramount importance. Providing security for Web services involves, amongst other things, making provision for confidentiality and non-repudiation by authenticating the parties involved, establishing trust, and providing access control.

Although we acknowledge that issues such as availability, accessibility, integrity, reliability, and performance of Web services can be influenced by security threats, this article is focused at determining whether there is a second generation Web services specification that addresses these important NFRs as per the definition/ description provided above.

4. NFRs OF SERVICE INTERFACE

Services express their purpose and capabilities via a service descriptor. The design of descriptors for Web services involves a lot more issues than in conventional middleware which basically involves interfaces and interface definition languages (IDL). In this context, IDL specifications are needed to automatically generate the stubs and to assist in dynamic binding. Issues such as semantics of different operations, the order in which they are to be invoked and other properties of services are assumed to be available to the programmer of the clients. Furthermore, the middleware platform defines and constrains many aspects of service description and binding process that are implicit, and do not need to be specified as part of the service description. In Web services such implicit contexts are missing and therefore need to be covered by a richer service descriptor [3]. For example, it is necessary to specify the address (URI) of the service and the transport protocol (e.g. HTTP) in order to invoke the service.

The Web services interfaces gets exposed as service descriptions using a service description languages such as WSDL. The design of the service descriptor should abstract all information (functional and non-functional) other than those described by the descriptor. The service interface being the public “component” of a web service application has to measure-up to several of these NFRs. The NFRs identified for the service interface are as follows:

- **Robustness**: Agents and other services make use of service interfaces to access the services. The interface should be designed keeping robustness as one of its main requirements. For example, trying to invoke service method via an interface operation with wrong parameter types should not lead to the entire application coming to an abrupt halt [12]. Hence it is important for the interface to be in a position to handle such conditions. Robustness is improved by complying to the WSDL design specification.

- **Consistency**: To develop service-based applications and business processes, which comprise service assemblies, Web services need to be described in a consistent manner. This is essential in enabling services to be published by service providers, discovered by service clients and developers, and assembled in a manageable hierarchy of composite application assemblies. However, to accomplish this, the interface must be consistent in its design.

- **Configurability / Customizability**: As a service may be accessed by several users with slightly varying needs, it is important to present to the users different interfaces to meet those needs. For instance, in a commercial environment, a web service could be designed to offer a product at a discounted rate to only its preferred customers. This fact should also be represented by the web service's interface. In other words, the service interface should be customizable to meet different needs. Furthermore, it should be also be possible for a web service interface to be configurable to support multiple bindings to different network protocols. Configurability and customizability is influenced by WSDL specification.

- **Extensibility**: It should also be possible for the interface to be augmented with additional information such as QoS or management information. For example, performance related information or the pricing model for the use of the Web services.

- **Integrity with the service**: Since the interfaces reflect the operations that the Web services provide. It is important that a high level of integrity be maintained between the two entities. Supported by the tools used in generating the Web services from the WSDL or vice-a- versa.

- **Publishable**: For services to be useful, they need to be found by as many users as possible. This can be facilitated by having the interfaces to those services to be published with ease. One of the ways in which this can be achieved is by keeping the coupling between the services and its interface as loose as possible. This is supported by the design principles mentioned earlier.

- **Trustworthy**: Web services are represented by their interfaces. It is often the only thing seen by the service requester. Hence it should maintain a high level of trustworthiness.

- **Availability**: Web services are accessed via its interface. If the interface is unavailable or its availability is limited, it is obvious that the availability of the Web services will also be affected and hence its usefulness.

- **Security**: Agents and other services make use of Service interfaces to access the services. This exposes the service interface to various forms of exploits. Hence service interface descriptions should be as secure as possible.

- **Composibility**: As the services can be composed together to provide larger functionality, their interface should also be composable to reflect this added functionality.

- **Reusability**: Since the services are reusable entities, they should be described in such a way that their descriptions too are reusable.

5. NFRs OF MESSAGING FRAMEWORK

For services to interact and accomplish something meaningful, they must exchange information. Even before any interaction can take place, services need to be aware of each other. This need to know each other's whereabouts should be designed in such a manner that further interactions are possible by maintaining a relationship with minimum dependencies. In other words, the locations of the services are not hard coded into the services. The design of the communication mechanism should be such that it promotes loose coupling between the communication parties. The term coupling, as used here, indicates the degree of dependencies parties have on each other. In line with this design principle, the Web services communicate primarily by exchanging messages between each other. This exchange of information follows a strict pattern and promotes autonomy by being sufficiently intelligent thereby eliminating help in processing by the recipient.

To facilitate exchange of messages regardless of heterogeneous environment in which they function, the Web services rely on SOAP. SOAP provides a wire protocol specifying how service-related messages are structured and exchanged across disparate systems in a distributed environment. To this end, SOAP makes use of XML as an encoding mechanism for its request and response parameters and uses HTTP as a means of transport. The combination of XML encoding style and the pervasive HTTP makes SOAP highly suitable for use in web service environment [11]. Analysis of the messaging framework to derive the NFRs is presented below.

- **Reliability**: In this context, reliability is defined as the ability to ensure the delivery of messages to the correct location, in the presence of software, system or network failure. This should form a critical requirement of such a framework else their use in mission critical complex business-to-business application is not possible. (supported by WS-ReliableMessaging, WS-Security, WS-Addressing)

- **Efficiency**: Efficiency of a messaging infrastructure relates to the overhead in relation to its payload in exchanging messages. Obviously, the more overhead required the less the efficiency of the framework.

- **Security**: As the messages, often have to travel through public domain to reach the intended recipient, security becomes an important NFR. This can be achieved by using techniques such as message encryption, and digital signatures.

- **Interoperability**: Refers to the ability of the protocol to work with a variety of computing platforms. In addition, the difference in XML schemas used in various requests/response, and data type representations of the information also contribute to the problems of interoperability. Thus it is necessary to ensure the interoperability at higher level (Application levels)

- **Support for multiple messaging styles**: The messaging framework should be able to support both synchronous and asynchronous messaging styles. In addition, there should be a provision for dynamic routing.

- **Scalability**: refers to the ability of the protocol to work with an increasing number of potential recipients. As the numbers of services coordinating their efforts grow, the performance of the messaging infrastructure should not suffer.

- **Fault handling**: The messaging infrastructure should have the capabilities to transfer errors / exceptions messages to the users, no matter in which of the three components it happens.

- **Context and transaction Management information**: For Web services to be able to part-take in any serious business-to-business applications, it must be able to maintain context and transactional information. As Web services are inherently stateless, this information has to be supported by means of a transport infrastructure.

- **Coordination of multiple messages**: Web services achieve their objectives by sending messages between each other. The messaging infrastructure must be able to coordinate these messages for effective functioning.

- **Extensibility**: As the messaging infrastructure is responsible for more than just assisting in sending messages, it must have the potential to be extended to cater for additional application specific information and should be extensible to transport data that is not encoded according to the standard encoding used in the SOAP body.

Having identified the NFRs of the three major components of a typical Web service application, the following section analyses the second generation Web services specifications in an attempt to map them to the identified NFRs.

6. MAPPING OF THE NFRs TO THE SECOND GENERATION WEB SERVICES SPECIFICATIONS

To cater for the short-comings of the core specifications (WSDL, SOAP, and UDDI) in addressing lack of QoS features, the second generation Web services specifications have been proposed. The introduction of these second generation web service specifications, bring enough advanced business functions to Web service platform to enable it to rival traditional distributed platforms [8]. A high-level description of the major second generation Web services specifications are presented below.

6.1 WS-Transaction

Decentralization in distributed systems poses reliability problems that are not frequently encountered in centralized systems. Decentralization allows parts of the systems to fail while other parts remain functioning, which leads to possibility of abnormal behavior for executing applications. The WS-Transaction specification describes coordination types that are used with the extensible coordination framework. It defines two coordination types: Atomic Transaction for individual operations, and Business Activity for long running transactions. The Atomic Transaction is similar to traditional ACID transactions intended to support short-duration interactions whereas Business Activity support business logic to handle exceptions that occur during the execution of activities of a business process.

6.2 WS-Coordination

When executing a unit of programming logic that is shared between groups of Web services, it is often required of the services to share a common context. This provides the task with an identity that is propagated to each participating service. The WS-Coordination specification defined an extensible framework for description of protocols that coordinate distributed activities [7]. The framework defined by this specification enables an application service to create a context needed to propagate an activity to other services. In addition, it describes the structure of the context and the requirements for propagation of the context between cooperating services.

6.3 Business Process Execution Language (BPEL)

BPEL is a language for describing the behavior of business process based on Web services [16]. It specifies a comprehensive syntax for describing business workflow logic in composite Web services applications. It allows for

the creation of both abstract processes that describe business protocols and executable processes that can be compiled into runtime scripts [13].

6.4 WS-Security Framework

As the first generation Web services specification did not provide for any real security, the acceptability of Web services in conducting serious business over the internet was threatened. WS-Security framework institutes a fairly complete security model to a lay this fear. This framework consists of fundamental and conceptual security standards as well as a set of supplementary specifications with the aim of providing end-to-end security. The WS-Security framework consists of the following main specifications:

6.4.1 WS-Trust

This specification establishes a standard trust model to unite existing trust models, so that the validity of the exchanged security tokens can be verified. It provides a communications process for requesting the involvement of third-party trust authorities to assist with this verification.

6.4.2 WS-Privacy

This specification can be used to communicate their privacy policies and check to see whether requestors intend to comply to these policies. WS-Privacy works in conjunction with WS-Policy and WS-Trust.

6.4.3 WS-SecureConversation

Various security models can be supplemented with WS-SecureConversation, which establishes a standard mechanism for exchanging security information between Web services. It provides formal definitions for the creation and exchange of security context and associated session keys.

6.4.4 WS-Federation

This specification defines mechanisms that are used to enable identity, attribute, authentication, and authorization federation across different trust realms.

6.4.5 WS-Authorization

This specification describes how access policies for a Web service are specified and managed. The manner in which claims are represented within security tokens is part of this specification. This specification has a number of commonalities with XACML specification.

6.5 WS-ReliableMessaging

It is often a requirement for Web services to communicate with each other over the network. The Web services should be able to achieve this in spite of software, system or network failure. This specification aims at provides for a truly robust communication framework. This is ensured by establishing standards for the acknowledgement of successful message delivery and the notification of transmission failure.

6.6 WS-Policy Framework

The WS-Policy framework establishes a means of extending service metadata beyond the WSDL definition. It allows the Web services to annotate their interface definitions to include service assurance qualities such as security and reliable messaging [4]. WS-Policy framework consisting of three interrelated specifications - WS-Policy, WS-PolicyAssertions, and WS-PolicyAttachments. WS-Policy specification provides a general purpose model and corresponding syntax to describe and communicate the policies to a Web services. WS-PolicyAssertions indicates a set of common message policy assertions that can be specified within a policy. Whereas, the WS-PolicyAttachement specification specify attachment mechanisms for the use of policy expressions with existing Web services technologies [17].

6.7 WS-Attachments

To enable payload data outside of the SOAP body, a standard for encoding this data as an attachment to the message is required. The WS-Attachments specification introduces a compound SOAP structure and specifies DIME as the attachment description format. DIME provides a lightweight encapsulation format that was considered more suitable for SOAP message documents [8].

6.8 WS-Addressing

WS-Addressing provides transport-neutral mechanism to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages [24]. Web services endpoints are referenceable entity, processor, or resource where Web service messages can be targeted. This specification establishes a standardized addressing syntax that is independent of transport protocols.

6.9 WS-MetadataExchange

Web services use metadata to describe what other endpoints need to know to interact with them. WS-

MetadataExchange specification enables Web services to be self-descriptive in a standardized manner [24].

6.10 WS-Notification Framework

Like WS-Policy Framework, WS-Notification Framework too consists of three interrelated family of specifications that define a standard Web services approach to notification via a topic-based publish/subscribe pattern. The WS-BaseNotification provides for a basic point-to-point communication mechanism. It defines the standardized interface used by services invoked on either end of notification exchange (producer/consumer). Message structure and operational requirements for producers and consumers are fully defined by this specification. The second specification in this framework, WS-BrokeredNotification, defines the web-service interface for the notification broker—an intermediary that allows publication of messages from service providers and other entities. The third specification in this family - WS-Topics defines a mechanism to organize and categorize items of interest for subscription known as "topics" into a tree structure. The specification determines the form of the subscription expression as well as the form of the metadata to be added to topics [2].

Table 1 evaluates the Web services specifications against the selected NFRs.

Table 1: Evaluation of the Specifications

	WSDL	SOAP	UDDI	WS-I Basic Profile	BPEL4WS	WS-Coordination	WS-Transaction	WS-Security Framework	WS-Policy Framework	WS-Attachments	WS-ReliableMessaging	WS-Addressing	WS-MetadataExchange	WS-Notification Framework
NFRs of services														
Availability												√		√
Accessibility									√			√		
Composibility	√				√									
Scalability														
Integrity									√					
Reliability														
Interoperability	√			√									√	
Robustness														
Performance														
Security								√						
NFRs of service interface														
Robustness	√													
Consistency	√								√					
Configurability/ Customizability	√													
Extensibility	√								√					
Integrity with the service														
Publishable	√													
Trustworthy								√						
Availability			√											
Security								√						
Composability	√				√									

Reusability	√													
NFRs of messaging framework														
Reliability								√			√	√		
Efficiency														
Security								√	√					
Interoperability		√		√										
Support for multiple messaging style		√												
Scalability		√												
Robustness		√			√						√			
Context and transaction Management information						√	√							
Coordination of multiple messages		√				√								
Extensibility		√								√				
Performance														

7. CONCLUSIONS

The evaluation shows that though the core specifications seem to be adequate for supporting several of NFRs, it is mostly due to the design principles applied in those components and not the specifications themselves. For example, composability or reusability of service interface is supported by the design principle of loose coupling that is embedded in a typical Web service application. The same goes for the messaging framework and the services themselves. The evaluation further demonstrates that though the second generation Web service specification such as WS-ReliableMessaging and WS-Security have provides some measure of support for the NFRs there are several areas such scalability, Reliability, and robustness that need attention.

The contribution of this paper could be extended in several directions. For example, encourage research towards creation of specifications for NFRs that are seen to be lacking. These specifications could be specified using formal methods. Organizations or individuals involved in creating specifications could benefit by eliminate duplications of their efforts.

8. REFERENCES

[1] Aiello M. and Giorgini P. 2004. Applying the TROPOS Methodology for Analysing Web Services Requirements and reasoning about Quality of Service. Upgrade, 5(4), 20-26.

[2] Aiello, M., Marchese, M., Busetta, P. and Calabrese, G. Opening the Home: A Web service approach to domotics. In: Applied Computing 2005, vol. 1. pp. 271-279.

[3] Alonso, G., Casati, F., Kuno, H., and Machiraju, V. 2003. Web Services – Concepts, Architectures and Applications, Springer Verlag, Berlin.

[4] Baresi, L., Guinea, S., Plebani, P. 2006. WS-Policy for service monitoring. 6th VLDB Intl. Workshop on Technologies for E-Services, LNCS vol. 3811, pp 72-83, Springer-Verlag, Berlin.

[5] Benatallah, B., Dijkman, R., Dumas, M., and Maamar, Z. 2005. *Service Composition: Concepts, Techniques, Tools and Trends.* In Stojanovic, Zoran and Dahanayake, Ajantha, (eds.) Service-Oriented Software System Engineering: Challenges and Practices. IDEA Group, Hershey, PA, 48-66.

[6] Brown, A., Johnston, S., and Kelly, K. 2002. Using Service-Oriented Architecture and Component- Based Development to Build Web Service Applications. Rational Software Corporation.

[7] Curbera, F., Khalaf, R., Mukhi, N., Tai, S., and Weerawarana, S. 2003. The next step in Web services. *Communication of the ACM* 46(10), 29-34.

[8] Erl T. 2004. Service – Oriented Architecture : A field guide to integrating XML and Web Services, Prentice Hall PTR, Upper Saddle River, NJ, USA.

[9] Erl, T. 2005. Service-oriented architecture: concepts, technology, and design, Prentice Hall PTR, Upper Saddle River, NJ, USA.

[10] Guan, Y., Ghose, A. K., and Lu, Z. 2006. HCLP Based Service Composition. In Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence and intelligent Agent Technology (December 18 - 22, 2006). WI-IATW. IEEE Computer Society, Washington, DC, 138-141. DOI= http://dx.doi.org/10.1109/WI-IATW.2006.77

[11] Kangasharju, J., Tarkoma, S., and Raatikainen, K. 2003. Comparing SOAP performance for various encodings, protocols, and connections. LNCS vol 2775/2003, 397–406. Springer-Verlag.

[12] Laranjeiro,N., Vieira, M., Madeira, H. 2009. Improving Web Services Robustness, In: IEEE International Conference on Web Services, 397-404.

[13] Liu, Y., Ngu, A. H., and L. Zeng, L. 2004. QoS Computation and Policing in DynamicWeb Service Selection. In Proceedings of the 13th International Conference on World Wide Web (WWW'04), ACM Press, New York, NY, 66–73.

[14] Lohmann N., Massuthe, P., Stahl, C., and Weinberg, D. 2008. Analyzing interacting WS-BPEL processes using exible model generation. Data Knowl. Eng., 64(1),38-54.

[15] Mani A. and Nagarajan A. 2002. Understanding quality of services for Web services, IBM developerWorks, available at: http://www.ibm.com/developerworks/java/library/ws-quality.html

[16] Michlmayr, A., Rosenberg, F., Leitner, P., and Dustdar, S. 2009. Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In Proceedings of the 4th international Workshop on Middleware For Service Oriented Computing (Urbana Champaign, Illinois, November 30 - 30, 2009). MWSOC '09. ACM, New York, NY, 1-6. DOI= http://doi.acm.org/10.1145/1657755.1657756

[17] OASIA. 2007. Web Services Business Process Execution Language Version 2.0. Available at: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf Accessed on 13th April 2010.

[18] Papazoglou, M.P. 2008. Web services: Principles and Technology, Pearson - Prentice Hall.

[19] Rosa, N. S., Justo, G. R., and Cunha, P. R. 2000. Incorporating Non-functional Requirements into Software Architectures. In Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, LNCS, vol. 1800, 1009-1018. Springer-Verlag, London.

[20] Saleh, A., Justo, G.R.R., and Winter S. C. 2003. Non-functional Oriented Dynamic Integration of Distributed Components. Electronic Notes in Theoretical Computer Science, Vol 68(3), 405-418.

[21] Saleh, K and Al-Zarouni, A. 2004. "Capturing Non-Functional Software Requirements Using the User Requirements Notation", The 2004 International Research Conference on Innovations in Information Technology (IIT2004), Dubai.

[22] Sha, L., Shaozhong, G., Xin, C., and Mingjing, L. 2009. A QoS based Web Service Selection Model. In Proceeding of IEEE International forum on Information Technology and Applications (IFITA). vol. 3, 353-356.

[23] Singhera, Z. U. and Shah, A. A. 2006. Extended web services framework to meet non-functional requirements. In *Workshop Proceedings of the Sixth international Conference on Web Engineering* (Palo Alto, California, July 10 - 14, 2006). ICWE '06, vol. 155. ACM, New York, NY, 21. DOI= http://doi.acm.org/10.1145/1149993.1150017

[24] Vieira, M., Laranjeiro, N., Madeira, H. 2007. Assessing Robustness of Web-Services Infrastructures. In 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). 131-136.

[25] W3C. 2006. Web Service Addressing 1.0–Core. Available at: http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/ accessed on 26th April 2010.

[26] W3C. 2008. Web Services Metadata Exchange 1.1 (WS-MetadataExchange). Available at: http://www.w3.org/Submission/WS-MetadataExchange/. Accessed on 28th April 2010.

[27] Zou, J. and Pavlovski, C. J. 2006. Modeling Architectural Non-functional Requirements: From Use Case to Control Case. In Proceedings of the IEEE international Conference on E-Business Engineering (October 24 - 26, 2006). ICEBE. IEEE Computer Society, Washington, DC, 315-322. DOI= http://dx.doi.org/10.1109/ICEBE.2006.71