IBIMA Publishing
Communications of the IBIMA
http://www.ibimapublishing.com/journals/CIBIMA/cibima.html
Vol. 2010 (2010), Article ID 552843, 13 pages
DOI: 10.5171/2010.552843

An Approach of Web Service Quality Attributes Specification

Samer Hanna and Ali Alawneh
IT Faculty – Philadelphia University, Jerash, Jordan

Abstract

Web Services are considered a new way of building software applications based on Services that are available through the Internet. However, Web Services still face many problems that are limiting their adoption. One of the causes of this problem is the lack of metadata about the quality attributes of Web Services, which make Service Requesters reluctant to integrate Web Service with their applications. This paper proposes a novel ontology that describes a model of the requester-oriented Web Services' quality attributes. The ontology is based on previous quality models which have been refined and modified specifically to address the quality issues as they relate to the requester of Web Services. Also an analysis will describe how some of the quality attributes in the previous model can be evaluated using different types of test cases.

Keywords: Service Oriented Architecture, Web Services, Quality Attributes, Testing

1. Introduction

Service Oriented Architecture (SOA) is not a new concept – although it has been around for over a decade now; SOA has gained extreme popularity lately among researchers and practitioners due to Web Services. While many believe that Web Services are SOA, they are in fact an implementation (or realization) of SOA based on a set of open XML-based technologies or standards.

Web Services are considered a new paradigm to build software application. In the Web Service paradigm, the requester of a Web Service can only see an interface which contains information about the operations provided by the Web Service and how to bind to this Service, the requester need not worry about how this Web Service was implemented or where it is located. Although similar to previous paradigms especially to component based development, one of the main differences is that Web Services rely on open standards or technologies for communication and integration of heterogeneous applications.

However, Web Services still face many challenges. One of those is that the current Web Services standards, such as WSDL, only describe the functional aspects of Web Services and not the non-functional aspects related to the Quality of Service (QoS)0. Another challenge is that there is no shared understanding of quality attributes of Web Services among providers and requesters – in other words, terms for quality attributes are used without following a standard or clear definition of their meaning.

This work aims at producing an ontology that will help in providing the requesters of Web Services with more semantics or metadata about the quality attributes, and also providing a shared understanding for those. OWL (Ontology Web Language) 0 is used to describe the quality attributes of Web Services.

This paper is organized as follows: Section 2 presents a background on relevant concepts and terms used in the rest of the paper. Section 3 discusses the analysis of different quality models in order to identify quality attributes relevant to requesters of Web Services. Section 4 shows what quality attributes can and cannot be assessed by applying current and adapted testing techniques. In Section 5, we discuss the proposed ontology for the quality attributes of Web Services. An example application of the ontology is presented in section 6, and section 7 concludes this work and discusses future work

2. Background

2.1. Service-Oriented Architecture (SOA)

SOA is an architectural style for building distributed application that depends on loosely coupled services that are available on the Internet. SOA consists of four main entities: service requester, service provider, service registry, and contract:

- Service Requester: The Service requester could be any type of software that needs a specific Service; the requester can be a human, an application, or even another Service.
- Service Provider: The Service provider implements a Service and publishes his Service's contract or description in a registry.
- Service Registry: The Service registry stores contracts from Service providers.
- Service Contract (Description): The contract specifies what tasks or methods a certain Service provides and also how the requester of a service will bind to the provider. The contract may also specify Quality of Service (QoS) levels.

The Service provider publishes a description of his Service in the registry. The Service requester asks the registry about Services that accomplish a certain task, and once the registry founds the right Service, it returns the Service information (such as the contract location) to the Service requester, which in turn uses the information in the contract to bind to the Service.

2.2. Web Services

Web Services implement SOA using open standards such as XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), and UDDI (Universal Description, Discovery, and Integration). Web Services are used mainly for the interoperability between heterogeneous applications over the Internet.

There is no universally accepted definition of Web Services, as it has been under debate for quite some time. An extensive literature survey on Web Services showed us that none of current definitions (given by different people and organizations) contained all the relevant characteristics of Web Services. In the context of our work, our (proposed and adopted) definition for Web Service includes those relevant characteristics for our work, as it is defined as follows:

Web Services are network (Internet) based modular applications designed to implement SOA, and support interoperable, loosely coupled, integration of heterogeneous application. Web Services are discovered using UDDI and It has an interface that is describe in WSDL, Other systems interact with the Web Services in a manner prescribed by its description using SOAP, these SOAP messages (as well as all other technologies of Web Services) are based on XML and typically conveyed using HTTP.

Web Services implement most of the SOA characteristics using the previous technologies. However, there are other characteristics of SOA that are still not implemented by Web Services. For example, few specifications provide QoS levels for a Service and they do not cover all the quality attributes of Web Services. In addition, the requesters can get the WSDL document of a Web Service without using UDDI (registry) and this is violation to the SOA.

2.3. Testing

Testing is a quality assurance Software Engineering technique that is part of almost any software development project. Testing is mainly used to assess the quality attributes and detect faults in a software system and demonstrate that the actual program behaviour will conform to the expected behaviour. Many studies show that testing may involve 50% to 60% of the effort involved in building software applications and this percentage may be significantly higher for critical software systems 0. Testing includes designing test cases, exercising software with these test cases, and then examining the results with the objective of evaluating the quality attributes such as correctness, robustness, and reliability.

2.4. The Semantic Web

The Semantic Web is an extension to the Web that aims to give meaning to the data and information on the Web to make them machine-understandable and automatically process-able 0.

XML is considered as a first step towards the Semantic Web vision; however XML provides no semantic (meaning) to the data. Another important technology for developing the Semantic Web is the Resource Description Framework (RDF), a XML-based data-model that allows the description of meanings for concepts and resources 0. A third technology was introduces after XML and RDF which is a collection of information called an ontology. An ontology is an explicit and formal specification of a conceptualization, describing formally a domain of discourse, and typically consists of a finite list of terms and the relationship between these terms 0. One ontology language is OWL which is a rich vocabulary description language for describing properties and classes 0. Logic is used as a formal language for expressing knowledge in the ontologies and to uncover ontological knowledge that is implicitly given 0

Table 1. Quality model (attributes and sub-attributes that concern the requesters of Web Services are in bold).

Attribute	Sub-attributes
Functionality	Suitability, **Accuracy** (or **Correctness)**, **Security** , Interoperability, Compliance
Reliability	Maturity, Fault Tolerance, Recoverability, Compliance, **Robustness**, **Availability**, **Integrity**.
Efficiency	**Time Behavior** (or **Performance**) (**Latency** and **Throughput**), Resource Behavior, Compliance, **Scalability**. **Accessibility**
Maintainability	Analyzability, Changeability or Modifiability, Stability, Testability, Compliance
Portability	Adaptability, Install-ability, Co-existence, **Replace-ability**, **Compliance** (or **Regulatory**)
Usability	**Understandability**, **Learn-ability**, Operability, **Attractiveness**, Compliance, **Documentation**.

3. Web Services Quality Attributes

According to Garvin 0 quality can be described from five different perspectives. One of these is the user view. A user sees quality as "fitness of purpose", i.e., quality is defined as the product characteristics that meet the user needs or expectations whether explicit or not. Although quality attributes may vary between Web Services applications according to the domain where the Web Services are used, we analyze and focus our work on the general abstract quality attributes that affect most of the requesters of Web Services, which are, therefore, the quality attributes that concern the user. A general quality model was developed based mainly on ISO 9126 0, and other relevant quality attributes from McCall 0 and Boehm 0 were also added to the model when related to Web Services (see Table 1).

When building the quality model, we noticed that there is no agreement between researchers about a fixed general quality attributes because there is no shared understanding about the quality attributes (or characteristics). For example, the terms accuracy and correctness are used by different researchers to mean the same quality attribute. This issue happens with other terms as well, such as compliance and regulatory. We also noticed that some sub attributes are related to different attribute. For example: accuracy is related to functionality attribute in ISO 9126, while it is related to reliability attribute in Boehm's model; and although being mainly related to security, access control is related to integrity in McCall's model, and so on.

There are only a very few research publications discussing about QoS for Web Services. Among these, Looker et. al. 0 mention that the non-functional quality attributes for Web Services include: availability, accessibility, integrity, security, performance (latency and response time), reliability, and regulatory. They also provide the definition of those quality attributes. Other important quality attributes for Web Services include accuracy, robustness, and scalability. We define below the quality attributes for Web Services that are important for us and that were not previously defined by Looker et al at 0:

- Accuracy: the quality aspect of whether a Web Service returns the right (correct or intended) response to its requester. Accuracy is also important for reliability.
- Robustness: the quality aspect of whether a Web Service continues to perform despite some violations of the constraints in its specification.
- Scalability: the quality aspect of whether a Web Service can handle increasing the number of requesters.
- Replace-ability: the quality aspect of a Web Service that relates to the difficulty of using it on the place of other software.
- Understandability: the quality aspect of Web Services that relates the requesters' effort to understand what a Web Service can do or what is its purpose.
- Learn-ability: the quality aspect that relates to the requesters' effort for learning Web Services application.
- Attractiveness: the quality aspect of Web Services that relates to its capability to be attractive to the requester.
- Documentation: the quality attributes of Web Services that relates to how much information and description is available with a Web Service.

The next section shows what quality attributes can be assessed by applying traditional and adapted testing techniques, and also propose methods to assess those quality attributes using testing. And also next section will show what quality attributes need other techniques to be assessed or evaluated.

4. Assessing the Quality of Web Services

QoS for Web Services is very important since the QoS is considered one of the main issues in Web Services applications as many requesters now are reluctant to use Web Services because of the trustworthiness issues 0. Very little work has been done to assess Web Services quality through testing. Traditional testing methods and tools are not adequate because they do not address the characteristics of Web Services and its applications 0.

Testing Web Services is more difficult than testing previous paradigms for software application development because Web Services' applications may be composed dynamically from different available Services that may be located in different places and have different quality attributes. Not only is the source code of the Service unavailable, but the Service might be hosted on servers at remote, even competing organizations 0. In addition, a Web Service may contain unknown faults and, since any requester can bind to a Web Service after being deployed, it may experience intruding attempts.

Testing Web Services can be viewed from two perspectives: the Service provider and the Service requester. One big difference between the two perspectives is the availability of the Service's source code. The Service provider has access to the source code, whereas the requester typically does not. The lack of source code for the requester of the Service limits the testing techniques that can be performed.

According to Bloomberg 0, Web Services testing tools employ the following range of traditional software testing techniques: black box or functional testing, white box or structural testing, regression testing, load testing, unit testing, and system testing. Black box testing includes boundary value testing, robustness testing, special value testing, worst case testing, equivalence class testing, and random testing 0.

Bloomberg 0 lists a variety of desirable capabilities for testing Web Services, such as:

- Testing WSDL files and using them for test plan generation: using the information in WSDL files to generate black box test plans.
- Web Service requester emulation: emulating the requester of a Web Service by sending test messages to another Web Service and analyzing the results.

Since we are mainly concerned with quality as perceived by the requesters of Web Services, this work focus on black box testing, using WSDL to generate test cases, and requester emulation.

Offutt et. al. 0 used data perturbation which is considered a black box testing technique to generate test cases for Web Service. They stated that most of the current testing tools for Web Services focus on the testing SOAP messages, testing WSDL files, and requester provider emulation. Looker et. al. 0 used fault injection to assess the dependability of Web Services.

After analyzing how researchers tackled Web Services testing, we noticed that they focused mainly on testing the composition of Web Services using integration testing and in most cases their work did not explicitly specify what quality attributes are being assessed. Examples include 000. Besides, some quality attributes such as robustness and accuracy have not been addressed by researchers in this field.

Many quality attributes of Web Services can be assessed by using requester emulation and using WSDL to generate test case together with adapted traditional testing techniques (assuming that the WSDL file for the Web Service is available to the user).

Due to space limitation, this work focuses on the following quality attributes:

- Accuracy or Correctness

Correctness of Web Services can be assessed by using the information in WSDL file to apply the black box boundary value testing and random testing and then requester emulation.

In more details:

- For each operation in WSDL build the boundary value and random test cases according to the data types of the input parameters which also can be obtained from the WSDL.

- For the Boundary value testing:

If the input parameter data type is numeric (like int or float) then the test cases for this parameter will be: minimum value allowed for this parameter (which can be obtained from WSDL file by the minInclusive tag); minimum value + 1, nominal value, maximum value allowed for this parameter (which can be obtained from WSDL file by the maxInclusive tag); and maximum value – 1. If the input parameter data type is String then the test cases for this parameter will be: Minimum length of this string, only one character, nominal string, maximum length allowed for the string, maximum length – one character.

- For the Random testing:

Choose a statistical distribution like the normal or uniform distribution to generate the test cases depending on the data type of each parameter.

- For each test case from the previous steps requester emulation is applied by sending a SOAP message to the Web Service under test and then analyzing the response to compare it with the expected return value of the specific test case.

- Repeat the previous step until an estimation of the accuracy of each operation in WSDL has been reached.

- Robustness

Robustness of a Web Service can be assessed by the same steps followed with Accuracy, but changing the testing techniques from boundary value testing and random testing to robustness testing (also called negative testing) where maximum_value+1 and minimum_value-1 are also added to the test cases of boundary value testing.

- Scalability

Scalability of a Web Service can be assessed by requester emulation and load testing, where a testing tool should simulate many requesters trying to bind to a Web Service under test at the same time and then check if the Web Service still performs as it is supposed to do.

- Availability

Availability can be assessed by constant requester emulation and using WSDL, where SOAP messages are regularly sent to the Web Service under test to check if this Web Service will send back a response.

- Security

Verifying that a Web Service gives the correct responses and that it is robust, scalable, and available is important; however we must also verify that this Web Service is not vulnerable to attacks by intruders. Security of Web Services has many factors, one of them is access control and can be assessed by applying Penetration testing 0 (also called Bypass testing) and then requester emulation. For example, we can change the data type of the input parameter (that obtained from WSDL) or use a combination of letters and numbers and any unexpected inputs that are often used by intruders, and then send (valid and invalid) SOAP messages with those inputs to the Web Service under test to check if the Web Service is vulnerable to such attacks.

From the above analysis, we noticed that many Web Services quality attribute can be assessed by adaptation of traditional testing techniques (to make them suitable for the new characteristics of Web Services). In addition, those traditional testing techniques should be merged with requester emulation and using WSDL to generate test cases in order to assess Web Services quality. We also notice that there is no shared understanding about some of the testing terms used among researchers, as Bypass testing and Penetration testing are used to mean the same testing technique. Another example is Robustness testing and Negative testing. This is one motivating factor for the ontology proposed in next section.

The quality attributes of Web Services that cannot be assessed using testing and need other techniques include: adaptability, replace-ability, attractiveness, and regulatory.

5. The Proposed Ontology

The lack of semantics about the quality of Web Services makes it difficult for a Service requester to find the Web Service that will accomplish the task he wants with the expected quality. To solve this problem, the operations of a Web Service should be described in a way that lets the requester understands the tasks this Service can do, how to use them, and what type of testing have been applied. Quality attributes should also be described and published with the Service interface. One way to write this description is by using an ontology language.

This section combines the quality model from section 3 and the set of testing techniques that can be applied to Web Services, as discussed in section 4, to devise an ontology about the quality attributes and the test cases for each of the available Web Services (we are assuming that we have a Web Service pool of different domains). In other words, our goal is to have an ontology that represents a registry of a quality enabled Web Services. This ontology is not a replacement for the WSDL or UDDI, but it provides the requesters of Web Services with more semantics about the quality attributes. The URL for this ontology can be published inside the WSDL file.

The ontology was built using the following steps:

1. Create a class in the ontology that represents the general description for a Web Service called *Web Services Description* (see step 1 in Figure 1.)

2. Add the following properties to the class created in step 1: *Web Service Location*, *Web Service Owner*, *Web Service Name*, *Web Service Domain* (e.g. financial, medical, etc.), and *Documentation Location* which represents a URL for a document containing more details in a human readable format about the specific Web Service (see Figure 2.). Then populate the individuals of this class by the properties of the available Web Services in the pool.

3. Map the taxonomy from section 3 to the ontology, by associating each quality attribute that can be evaluated using testing to a class of this ontology. All the sub attributes are then mapped to a sub class of the corresponding attribute class and the whole taxonomy will be in a class called *Quality Attribute Evaluate by Testing* (see step 3 in Figure 1.).

4. Add a property to the general class about the quality attributes from step 1 that represents the weight, or estimate, obtained for each quality attribute by running and evaluating the test cases.

5. Build another class for the test cases that contain a sub class for the relevant test cases to each of the sub quality attributes in step 3 (see step 5 in Figure 1.)

6. Add the following properties to the test cases class in step 5: *operation name*, a property for each of the input parameters, and the *expected output*.

7. Populate the individuals of the class in step 5 using the testing techniques described in section 4 to each of the quality attributes in this class for each of the available Web Services under test. Then, use the results of the testing techniques applied to obtain a weight or estimate to the corresponding quality attributes in step 3, according to the number of test cases passed and failed.

8. Add a class to the ontology that represent a hierarchy of the quality attributes that cannot be evaluated by testing.

6. A Quality Enabled Web Service

The proposed ontology in section 5 can describe the quality attributes and test cases for a Web Service, and the Triangle Web Service example will be used to illustrate this ontology. The Triangle Web Service consists of one operation that accepts three integers as input and the output is the type of the triangle according to the three sides which represents the 3 input values.

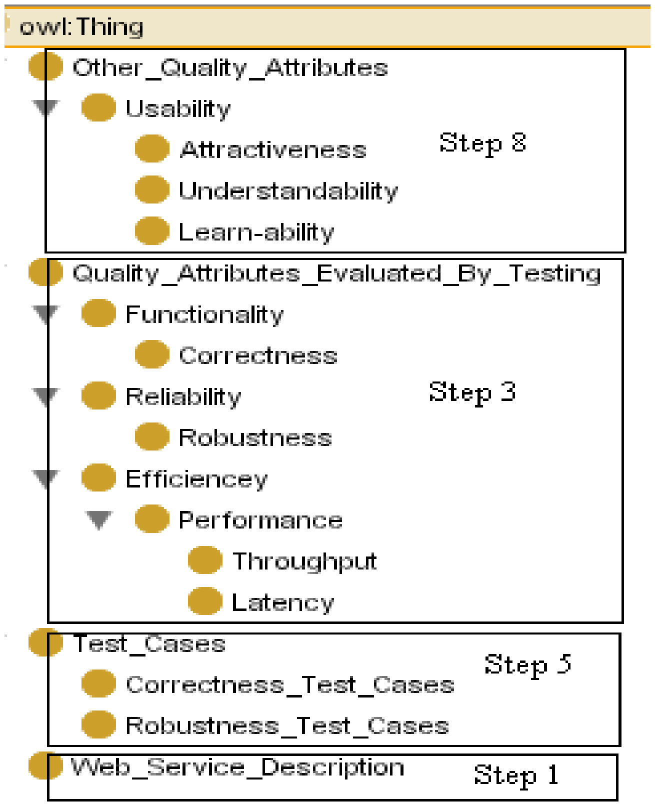
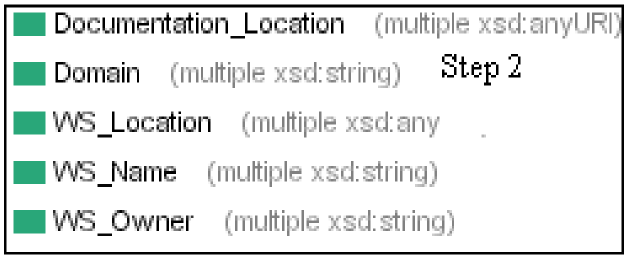


Figure 1. A fragment of the taxonomy of the classes of the proposed ontology

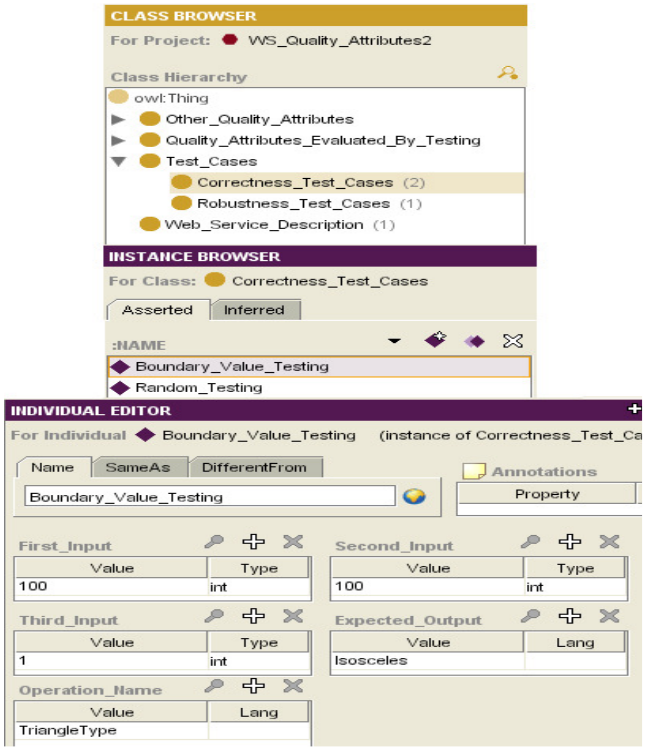


🟩 Documentation_Location	(multiple xsd:anyURI)
🟩 Domain	(multiple xsd:string)
🟩 WS_Location	(multiple xsd:any
🟩 WS_Name	(multiple xsd:string)
🟩 WS_Owner	(multiple xsd:string)

Figure 2. Properties of the Web Services Description Class

The output is either: scalene, equilateral, isosceles, or not a triangle 0. The Triangle Web Service was implemented using Axis 1.2, which also enables obtaining WSDL file automatically. To add a description of this Web Service to the ontology, the following steps were taken (See Figure 3.):

- *An instance of the class Web Service Description was created that is called Triangle Type (see Step 1 in Figure 1.)*
- *The individuals for the instance in step A were added where the web Service Name was obtained from WSDL (see Step 2 in Figure 2.)*
- *Two instances were added to the Correctness Test Cases class which is a subclass of Test Cases* Class



CLASS BROWSER
For Project: WS_Quality_Attributes2
Class Hierarchy
owl:Thing
Other_Quality_Attributes
Quality_Attributes_Evaluated_By_Testing
Test_Cases
Correctness_Test_Cases (2)
Robustness_Test_Cases (1)
Web_Service_Description (1)
INSTANCE BROWSER
For Class: Correctness_Test_Cases
Asserted
Inferred
:NAME
Boundary_Value_Testing
Random_Testing
INDIVIDUAL EDITOR
For Individual Boundary_Value_Testing (instance of Correctness_Test_Ca
Name
SameAs
DifferentFrom
Annotations
Property

Boundary_Value_Testing

First_Input

Value	Type
100	int

Second_Input

Value	Type
100	int

Third_Input

Value	Type
1	int

Expected_Output

Value	Lang
Isosceles	

Operation_Name

Value	Lang
TriangleType	

Figure 3. Two instances for the Correctness_Test_Cases class and part of the individuals for the Boundary_Value_Testing instance for the Triangle Web Service.

(see Step 5 in Figure 1), the first instance called Boundary_Value_Testing and the second instance called Random_Testing (see Instance Browser in Figure 3) the purpose of these instances is to add the test cases that will check the correctness of the operation in the Triangle web Service.

- The individuals of the instances in step C (Boundary_Value_Testing and Random_Testing) were populated using the techniques for testing the Correctness quality attributes described in Section 4. Applying the testing techniques for the Boundary Value Testing on the Triangle Web Service, we obtained a set of test cases that contained, among others, the following test cases:

Input: (0,0,0) Exp Output: "Not a Triangle"
Input: (100,100,1) Exp Output: "Isosceles" (see Individual Editor in Figure 2)
Input: (100,1,100) Exp Output: "Isosceles"
Input: (100,99,2) Exp Output: "Scalene"
Input: (99,100,101) Exp Output: "Scalene"
Input: (99,99,99) Exp Output: "Equilateral"
For the Random Testing technique, we obtained a much larger set of test cases containing, for example, test cases such as:
Input: (70,42,70) Exp Output: "Isosceles"
Input: (64,2,21) Exp Output: "Scalene"
Input: (20,1,2) Exp Output: "Not a Triangle"
Input: (36,42,53) Exp Output: "Scalene"

- An instance of the Correctness class which is a sub class of Functionality was then created for the Triangle Web Service (see Step 3 in Figure 1.)

- The Requester emulation testing was then used repeatedly to test the Triangle Web Service by sending SOAP messages to this Web Service using the test cases that were obtained in step D reporting the success/failure rate (weight) of the Correctness quality attribute.

- The weight that was obtained in step F was populated to the weight property of the instance of Correctness created in Step E.

- The same steps from C to G can be done to other quality attributes in Quality Attributes Evaluated by Testing class such as robustness.

Building the ontology instance for the Triangle Web Service, it is possible to associate how the results from executing the test cases can measure and be used to assess the quality attributes of the Web Service.

7. Conclusions and Future Work

Web Services are considered an application integration technology on the Internet that will shift the way that distributed systems are built. Being an implementation of Service Oriented Architectures, Web Services have many characteristics different from the previous technologies for building applications. Some of the most important characteristics are loose coupling and dynamic discovery and invocation of heterogeneous services.

One problem that Web Service applications still face is that current standards do not support the description of the quality attributes of Web Services. This paper addressed this issue by proposing an ontology for describing quality attributes that concerns the requesters of Web Services.

The benefits of this ontology include:

- A solution toward reaching the required quality in Web Services to achieve the desired level of trustworthiness by requesters.
- Both providers and requesters can add test cases to the proposed ontology and the increase in test cases means obtaining a better evaluation or weight for the specific quality attribute.
- Taxonomy of the quality attributes for Web Services and how they are related to each other and to the traditional testing techniques, and also to new Web Services testing techniques.
- The provision of a shared understanding for quality attributes of Web Services among providers and requesters, which will reduce the semantic gap and verify the conformance of Web Services with requester requirements.
- The provision of a shared understanding about the testing techniques of Web Services.
- An ontology that describes both the quality attributes that can be evaluated using testing and also other attributes that cannot, such as understandability and attractiveness.
- Help for the requester to find the Service that best fits his or her requirements among many competing Services.
- Help for the requester to search for Web Services in some domain only or that satisfies certain criteria of the quality attributes or functions.
- The ontology will help in automating the process of testing Web Services based on its description.

The proposed ontology describes the quality attributes that can be evaluated by testing techniques. We described how to evaluate some of the quality attributes in this paper, while others will be discussed in future work. Also future work will solve the problem of automatically testing and describing Web Services quality attributes.

We are currently investigating ways to evaluate the other quality attributes that can not be evaluated with testing. We also plan to use reasoning to infer about the quality attributes of Web Services using the test results. Also we plan to extend this work by describing also the composition of Web Service in an ontology and how the quality attributes of one Web Service will affect the composition overall quality attributes.

References

Antoniou, G. and van Harmelen, F. (2004), "A Semantic Web Primer," The MIT Press, Massachusetts Institute of Technology, USA.

Arkin , B., Stender, S., Cigital, G. (2005), “Software Penetration Testing,” Published by the IEEE Computer Society, IEEE Security and Privacy, 532-535.

Berners-Lee, T., Handler, J., and Lassila, O. (2001), "The Semantic Web". Scientific American 284, 34-43.

Bloomberg, J. (2002) “Testing Web Services Today and Tomorrow”, Rational Edge E-zine for the Rational Community [Online] [Retrieved January 29th, 2007], http://www.ibm.com/developerworks/rational/library/content/RationalEdge/oct02/WebTesting_TheRationalEdge_Oct02.pdf.

Boehm, B. W., Brown, J. R., and Lipow, M. (1976), "Quantative Evaluation of Software Quality," Proceedings of the 2nd International Conference on Software Engineering (ICSE), California, USA, 592-605.

Canfora, G. (2005), "User-side Testing of Web Services," Proceedings of the IEEE Ninth European Conference on Software Maintenance and Reengineering (CSMR'05), Manchester, UK, 301-301.

Garvin, D. (1984), "What does 'Product Quality' Really Mean?" Sloan Management Review, USA, fall. 25-45.

ISO 9126-1: 2001 Software Engineering – Product quality – Part 1: Quality Model, International Organization of Standardization, Geneva, Switzerland.

Jargensen, P. (2002), "Software Testing, A Craftsman's Approach," Second Edition, CRC Press, USA, 2002.

Looker, N., and Xu, J. (2003), "Assessing the Dependability of SOAP-RPC-Based Web Services by Fault Injection", 9th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS'03), Mexico, 163-170.

Looker, N., Munro, M., and Xu, J. (2004), "Assessing Web Services Quality of Service with Fault Injection," Presented at Workshop on Quality of Service for Application Servers, Symposium on Reliable Distributed Systems, SRDS, 2004, Brazil.

McCall, J. A., Richards, P. K., and Walters, G. F. (1977), "Factors in Software Quality," vol. 1, 2, and 3, AD/A-049-014/015/055, Nat'l Tech. Information Service, Springfield, Va.

Milanovic, N. (2005), "Contract-based Web Services Composition Framework with Correctness Guarantees", In proceedings of the 2nd International Service Availability Symposium (ISAS 2005), Berlin. 52-67.

Offutt, J., Xu, W. (2004) "Generating Test Cases for Web Services Using Data perturbation", ACM SIGSOFT, Software Eng. Notes, 29(5), 1-10.

Osterweil, L. (1996), "Strategic Directions in Software Quality," ACM Computing Surveys, 4(4), 738-750.

Singh, M. P. And Huhns M. N. (2005), "Service-Oriented Computing," John Wiley & Sons Ltd, England.

Tsai, W., Chen, Y., Paul, R. (2005), "Specification-Based Verification and Validation of Web Services and Service-Oriented Operating Systems," 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS05), Sedona, 139-147.