

MEASURING OF QoS PARAMETERS AND DEVELOPMENT OF QoS-AWARE WEB SERVICE

Muhammad Abul Kalam Azad

Student ID: 012171026

A Project
in
The Department
of
Computer Science and Engineering



Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

June 2021

© Md. Abul Kalam Azad, 2021

Approval Certificate

This project titled "**Measuring of QoS parameters and development of QoS-Aware web service**" submitted by Muhammad Abul Kalam Azad, Student ID: 012171026, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on June, 2021.

Board of Examiners

- | | | |
|----|---|------------|
| 1. | Dr. Salekul Islam
Professor & Head of Dept.
Department of Computer Science & Engineering | Supervisor |
| 2. | Khondaker Abdullah Al Mamun
Professor and Director
Department of Computer Science & Engineering | Member |
| 3. | Prof. Dr. Mohammad Nurul Huda
Director, MSCSE Program
Dept. of Computer Science and Engineering | Ex-Officio |

Declaration

This is to certify that the work entitled “**Measuring of QoS parameters and development of QoS-Aware web service**” is the outcome of the research carried out by me under the supervision of Dr. Salekul Islam, Professor and Head of Dept.

Muhammad Abul Kalam Azad, ID: 012171026

Department of Computer Science and Engineering

In my capacity as supervisor of the candidate’s project, I certify that the above statements are true to the best of my knowledge.

Dr. Salekul Islam

Professor & Head of Dept.

Abstract

Web service plays an important role in the new paradigm of distributed computing over the Internet. Web service allows to combine different publicly available services together to create new functionality. Many public web services can be used at free of cost. Two important elements of web services are Web Services Description Language (WSDL) used for describing and locating web services and Simple Object Access Protocol (SOAP) used for communicating between applications via the Internet. The consumers of web services are always keen to differentiate between QoS-aware and QoS-unaware web services. Measuring QoS is the prerequisite for designing QoS-aware web services. Therefore, a number of metrics need to be identified. Because of the dynamic and unpredictable nature of the web services infrastructure, providing the acceptable QoS is really a challenging task. In this paper we have identified three dominating features—response time, reliability and security—that governs the QoS of a web service. Using open-source tools we propose a simplified approach to measure those QoS metrics for publicly available web services. Based on our approach, QoS aware web services can easily be built from publicly available services.

This project provides a modeling process using some practical and operable measurement techniques for the QoS requirements of Web services and develops a QoS-aware web service using the modeling result.

Acknowledgement

I would like to begin by expressing my heartfelt gratitude to Almighty Allah for providing me with the capacity and strength to complete the assignment successfully and on time.

This project, named "Measuring QoS parameters and developing a QoS-Aware web service," was created to satisfy the MSCSE degree requirement. I consider myself quite fortunate to have received honest direction, supervision, and cooperation from a variety of people.

I'd like to thank my supervisor, Dr. Salekul Islam, Professor and Head of Department, United International University's Department of Computer Science and Engineering, for his constant advice, support, and patience, as well as for allowing me to undertake this job. His insightful advice and firm direction enabled us to put together a well-organized presentation.

Finally, I want to express my heartfelt thanks and affection to my parents for their unwavering support, encouragement, and unconditional love.

Table of Contents

LIST OF TABLES.....	vii
LIST OF FIGURES	viii
1. Introduction.....	1
1.1 Overview.....	1
1.2 Components of web service.....	2
1.3 Web service Architecture	5
1.4 Characteristics of Web Services	7
1.5 Objective.....	7
1.6 Organization of the project book	8
2. Background and Literature Review	9
2.1 Introduction.....	9
2.2 Related work	9
2.3 QoS Parameters of web service	10
2.4 Performance:	11
2.5 Reliability	12
2.6 Robustness	13
2.7. Security	14
2.8 Scalability	18
3. Methodology.....	20
3.1 Introduction.....	20
3.2 Experimental design	24
3.3 Public web service provider.....	33
3.4 Web services used.....	33

4. Results	34
4.1 Introduction.....	34
4.2 Experiment Results.....	35
A. Response Time:.....	35
B. Reliability.....	36
C. Security	37
4.3 Result Discussed.....	38
5. Conclusion	40
6. References.....	41

LIST OF TABLES

Table 3. 1: SOAP UI Features	21
Table 3. 2: List of web service provider	21
Table 3. 3: www.programmableweb. Com web service features.....	22
Table 3. 4: http://free-web-services.com web service features	23
Table 3. 5: http://community.workday.com web service features.....	23
Table 4. 1: Reliability value of weather & Currency web service	36
Table 4. 2: Security measure of weather and currency converter web services	38

LIST OF FIGURES

Figure 1. 1: SOAP Architecture.....	2
Figure 1. 2: WSDL Architecture	3
Figure 1. 3: Web service communication steps.....	5
Figure 1. 4: UDDI Model.....	6
Figure 2. 1: Response time calculation.....	11
Figure 2. 2: Robustness measure	13
Figure 4. 1: An example screenshot of load test executed by Visual Studio.net.	34
Figure 4. 2: Response time graph of weather web service	35
Figure 4. 3: Response time graph of currency web service.....	36

Chapter 1

Introduction

1.1 Overview

Web services are platform-independent, Internet-enabled services that are developed for some specific business needs. Reusability and interoperability are two important features of web services, and thus web services are built by mashing up a set of existing services. While building new web services we need to maximize the Quality of Services (QoS) and meet the user's requirements. This is an open-ended problem and in the literature a number of researches are found to address this issue.

For constructing distributed applications, web services can be published and accessible over the Internet and corporate intranets. It can be developed as a software system model [1] that is designed to ensure interoperability, and it may be then published and invoked through the Internet from anywhere if the connectivity is available [2]. In case of building mash-up services, selecting and composing different inter organizational and mixed services at run-time are challenging, and thus, are considered as important issues in web service applications. We are not able satisfy both the user's functional and non-functional requirements with a single web service, therefore we need to blend a collection of composite currently existing web services to meet the user needs. Web service composition of related web services based on functional and non-functional behavior is the current research problem. To provide languages and platforms for allocating web services and composing them among heterogeneous mixed systems. Web services enable applications from many sources to communicate with one another without the need for time consuming customized coding. Web services are independent of any operating system or programming language because all communications are based on XML. With the increased adoption of web services, QoS has become a critical criterion for determining a service provider's success. QoS impacts the service's usability and utility, both of which have an impact on its popularity.

Consumers of web services will frequently desire a way to distinguish QoS-aware web services from QoS-unaware web services as web services become more popular and widely deployed. According to the World Wide Web Consortium (W3C) definition, QoS criteria for web services encompass 10 primary parameters: performance, reliability, scalability, robustness, accuracy, integrity, accessibility, availability, interoperability and security. However, determining the QoS requirements has proven to be a nontrivial task. Providing the appropriate QoS is a difficult undertaking due to the dynamic and unpredictable nature of the web services infrastructure. Furthermore, the various applications that collaborate for web services interface with varying requirements will contend for network resources. Ladan proposed methodology for measuring web service major QoS metrics [3].

1.2 Components of web service

SOAP (Simple Object Access Protocol): SOAP (Simple Object Access Protocol) is a transport-independent messaging protocol. SOAP uses SOAP Messages to send XML data. A document known as an XML document is included with each message. The XML document's structure, but not its content, follows a pattern. The nicest thing about Web services and SOAP is that everything is communicated through HTTP, the most used web protocol.

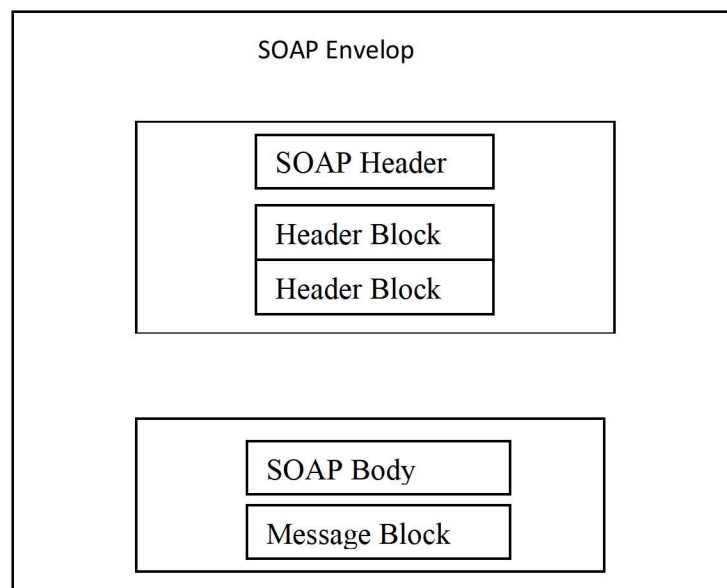


Figure 1. 1: SOAP Architecture

SOAP message consists of

- There must be a root element known as the `envelope` in every SOAP document. In an XML document, the root element is the first element.
- The "envelope" is separated into two halves. The header comes first, followed by the body.
- The header provides routing data, which is information that tells the XML document to which client it should be sent to.
- The body will include original message.

WSDL (Web services description language): If a web service can't be found, it can't be used. The client invoking the web service should be aware of the location of the web service. Second, the client application must understand what the web service does in order to invoke the correct web service. The WSDL file is another XML-based file that explains what the web service does to the client application. The client application will be able to understand where the web service is located and how to use it by using the WSDL document.

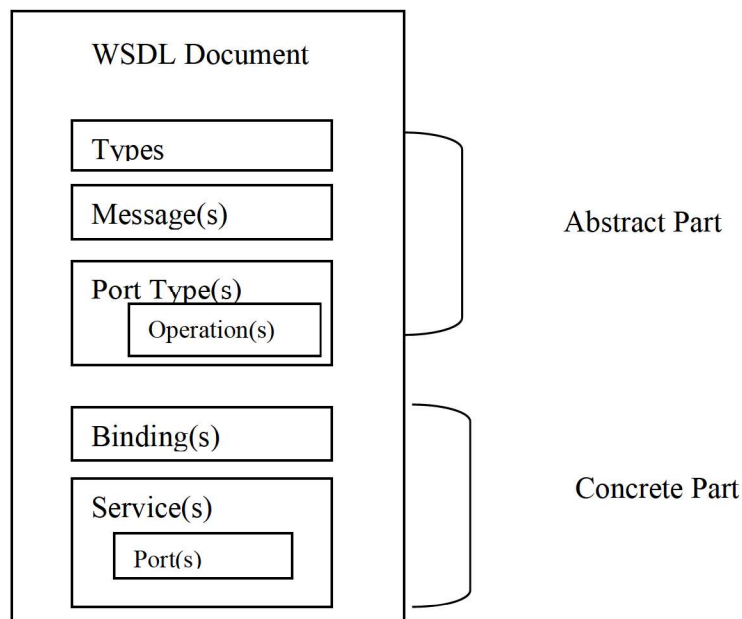


Figure 1. 2: WSDL Architecture

The following are the key points to remember about the aforementioned WSDL declaration:

1. <message> - The message parameter in the WSDL description is used to define the various data pieces for each web service operation. So, in the preceding example, we have two messages that can be exchanged between the web service and the client application: the "TutorialRequest" operation and the "TutorialResponse" action. The element "TutorialID" of the TutorialRequest is of the type string. In the same way, the TutorialResponse action has a type string element called "TutorialName."

2.<portType> - This actually indicates the action which may be performed by the web service, which in our instance is named Tutorial. This operation can accept two messages: one as an input and the other as an output.

3. binding> - This element specifies the protocol that will be used. In our example, we're specifying http (<http://schemas.xmlsoap.org/soap/http>) as the protocol. Other details for the operation's body include the namespace and whether or not the message should be encoded.

UDDI stands for Universal Description, Discovery, and Integration, and it is a standard for describing, publishing, and discovering online services supplied by a specific service provider. It provides a specification that aids in the hosting of data via web services.

Now, in the previous article, we talked about WSDL and how it provides information on what a Web service does. However, how can a client program find a WSDL file in order to comprehend the many activities provided by a web service? UDDI is the solution to this problem, as it provides a repository where WSDL files can be stored.

As a result, the client application will have full access to the UDDI, which serves as a database for all WSDL files. The UDDI registry will hold the required information for the

online service, just like a telephone directory has the name, address, and phone number of a certain individual. So that a client application may figure out where it is.

1.3 Web service Architecture

The high-level architecture of a typical web service is shown in Fig. 1.3. There are three main roles or entities involved in a web service: provider, requester and broker. They have distinct responsibilities and they depend on each other too. Web Services Description Language (WSDL) is commonly used to describe the data types and the structure of a web service. Another important component is Universal Description, Discovery and Integration (UDDI), which is a platform-independent XML protocol that contains a XML based registry service. UDDI allows businesses around the world to advertise themselves on the Internet, as well as a system for registering and finding web service applications.

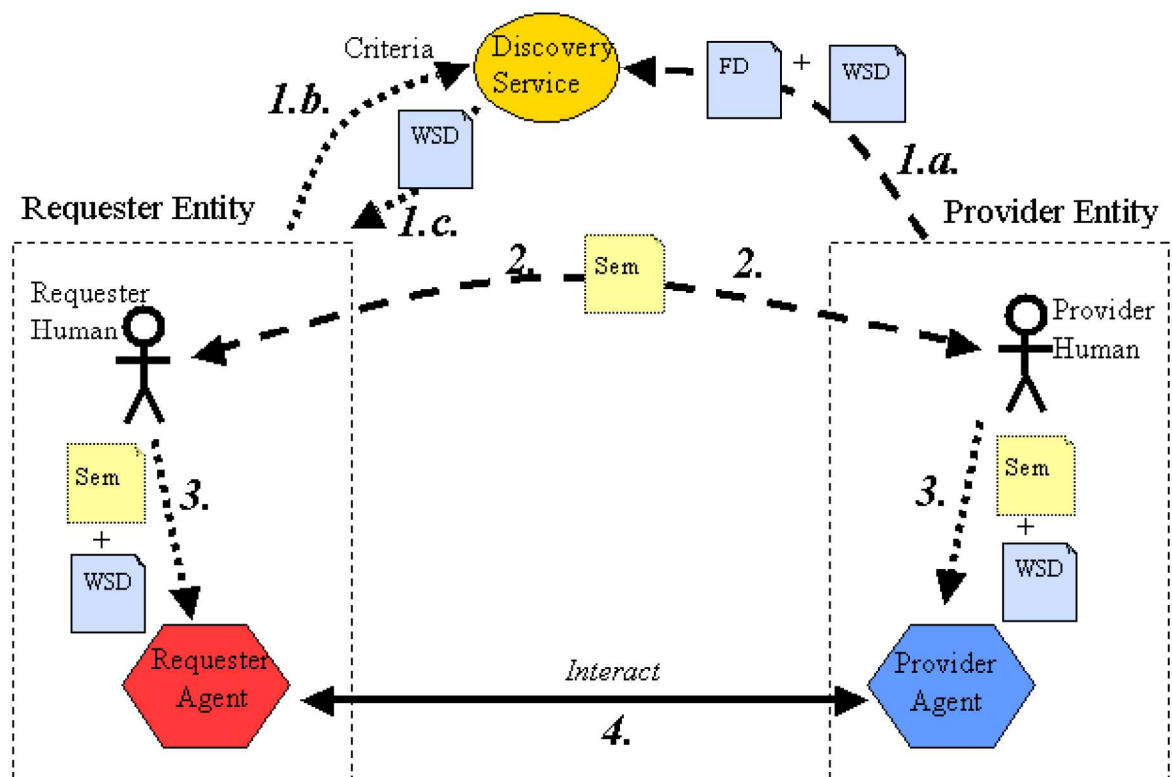


Figure 1. 3: Web service communication steps

In the following we explain the basic functionalities of the provider, the requester and the broker:

Provider - The provider is responsible for providing a collection of software (i.e., the application program, middleware and platform) required to create a web service. The provider also makes the web service available when a client application requests for it. If a service registry is available, a provider publishes a web service and its description in a registry service. Note that WSDL is used for describing a web service and UDDI is used for the publishing step.

Requester - A requester is a client application that needs to communicate with a web service. It is possible for the client application to be an application written in .Net, Java, or other languages that requires functionality from a web service. The service requester binds with the service provider and interacts with the web service implementation using the service description.

Registry - The registry is the application which provides access to the description of a web service. As we have mentioned earlier, the provider publishes service description to the registry and a client application (requester) later can find it. Note that UDDI is being used to find the appropriate web services. The diagram below showcases how the Service provider, the Service requestor and Service registry interact with each other.

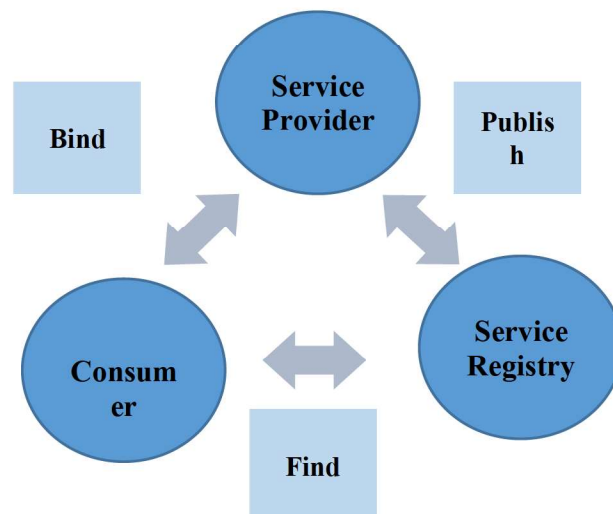


Figure 1. 4: UDDI Model

1.4 Characteristics of Web Services

The following are some of the unique behavioral traits of web services:

- They are XML-based - At the representation and data transit layers, Web Services employ XML to represent data. Because XML is a universally known language, it eliminates any networking, operating system, or platform dependencies.
- Loosely Coupled - Loosely coupled means that the client and the web service are not inextricably linked, which means that even if the web service changes over time, the way the client calls the web service should not change. Adopting a loosely linked architecture makes software systems easier to maintain and enables for easier integration between them.

Synchronous or Asynchronous functionality: - Synchronicity refers to the client's connection to the service's execution. The client will wait for the web service to complete an operation in synchronous operations. A scenario where a database read and write operation is being done is possibly an illustration of this. If data is read from one database and then written to another, the actions must be performed in a specific order.

1.5 Objective

In this project, we have identified various parameters those are related to QoS of web service composition. We have illustrated the related technologies by analyzing QoS parameters. Our proposed methods are based on available network analysis tools. The main goal of this work is to measure QoS parameters quantitatively using practical and operable measurement techniques while demonstrating a way for developing a QoS aware web service. The contributions of this project can be summarized as follows:

1. Identification of practical QoS measurement techniques.
2. Measure QoS parameters quantitatively for publicly available web service.
3. Identify techniques to improve QoS parameters.
4. Developing QoS-aware web service.

Outcome: Metrics for measuring the QoS of web service will be established through this work.

1.6 Organization of the project book

The project is organized as follows:

Chapter 2 presents QoS parameters of web service like performance, throughput, response time, security etc.

Chapter 3 presents experimental setup & design, public web service features and test case development in SOAP UI.

Chapter 4 presents result of project in tabular and graph format. Here future work and conclusion also presented.

Chapter 2

Background and Literature Review

2.1 Introduction

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, web services are not tied to any one operating system or programming language. With the growing popularity and an extensive deployment of web services, the consumers of web services will rather want a means to distinguish QoS-aware web services from QoS-unaware web services. According to the specification designed by the world wide consortium (W3C), QoS requirements for web services include ten major parameters: performance, reliability, scalability, robustness, accuracy, integrity, accessibility, availability, interoperability and security. However, the measurement of the QoS requirements has been a difficult area. Ladan L.M. proposed methodology for measuring web service major QoS metrics.

2.2 Related work

A web service selection algorithm is proposed by Zhang [4] that is based on dynamic replanting and targets to optimize global QoS. The proposed algorithm meets the requester's needs and can be effectively integrated into the development trend of QoS. Rodr'iguez et al. [5] carried out a linear regression-based statistical investigation to find out the relationship between the quality attributes provided by Web services and the metrics related to their interfaces defined in WSDL. They also executed experiments on public QoS web service datasets and service interface metrics. Many web service providers provide similar web services. Only checking service registration QoS of a web service cannot be determined. A novel method has been proposed by Rangarajan et al. [6] that predicts the web service QoS by inspecting source code metrics. Silva et al. [7] propose a genetic algorithm based technique to make a composition of web services and

optimize the solution to enhance the QoS. Zheng et al. [8] show an integrated resource service selection method by which requester can find the optimal manufacturing services. The authors use a fuzzy theory based computational model and later they adopt a particle swarm optimization (PSO) method to find the optimal service composition.

2.3 QoS Parameters of web service

QoS is not a functional requirement rather it is a nonfunctional requirement. It represents numerous non-functional aspects such as total service execution time, cost, availability, and security characteristics, among others. Web-based service composition is a collection of services that are combined to run in a sequential order. When there is parallel implementation, branching, and loops in a complex composite problem, it is considered as an NP-hard problem. Throughput is chosen based on the user's requirements for performance, availability, security, and response time. The validity of published QoS information and reputation for web service discovery for basic or complicated tasks has been a hard issue for practitioners as the number of functionally identical web services available on the web has grown. To obtain the QoS of composite service we need to calculate the QoS of each component service that is invoked fairly. Invoking a low-quality component service can degrade the composition's overall performance. During web service monitoring, the majority of existing approaches ignore the registry's rating. A QoS evaluation and monitoring methodology with several novel QoS parameters is presented in Access Rate and Overall Aggregated Effective Quality of Service (OAEQoS) [9]. To ensure the quality of service, the proposed parameters are evaluated in the working environment of the trusted third party registry upon new service registration. The value of the OAEQoS option can be used by the registry to assign a rank. The register publishes services with their overall quality score using established parameters, which might assist service consumers in finding the best service. The evaluation of proposed parameters is performed at the trusted third party registry's operating environment during new service registration to assure the quality of service. The registry can assign rank from the value of OAEQoS parameter. With the help of proposed parameters, the registry publishes the services with their overall quality score that can help service consumer to retrieve best services. Although we have mentioned ten parameters identified by the W3C, we are proposing here to reduce it to three: response

time (performance), reliability and security. In the following we have explained briefly how these three parameters can be measured.

Cost: The cost quality is the amount that a service requester needs to pay to execute a service using task. There are many free web service which user can use free of cost and many vendor is offer web service at certain cost. The cost factor C_{pq} can be calculated by the amount required to pay for a service p using task q which is: $C_{pq}, p \in [1 \dots n], q \in [1 \dots m]$

The value of C_{pq} is not determined when the service p cannot execute the task.

2.4 Performance:

Performance of a web service concerns how fast a web service request can be processed and serviced. This requirement can be determined by measuring response time and throughput.

Response Time: It indicates the time elapsed between a client sends a request to a web server and the time when the client receives the entire response. Considering the following definitions:

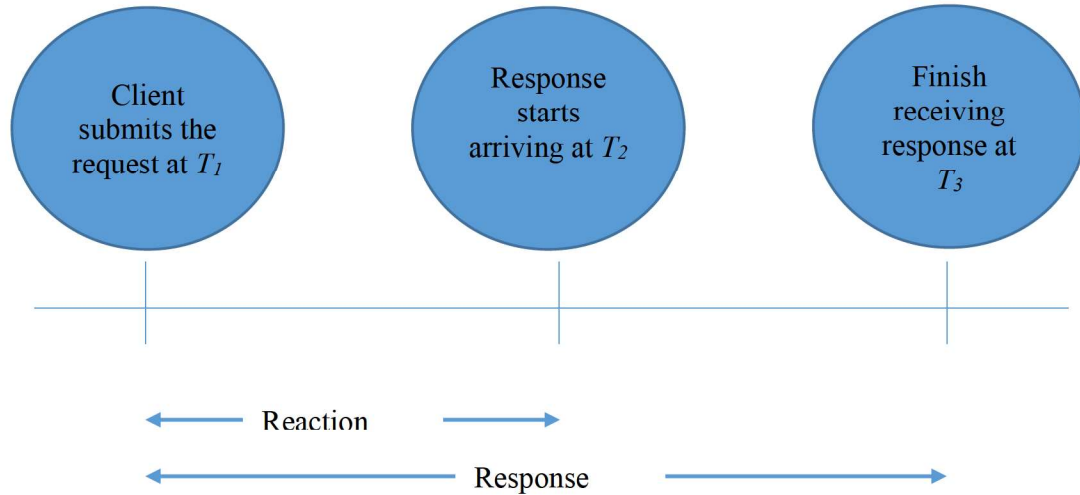


Figure 2. 1: Response time calculation

where,

$T1$: when a request is sent by the client to the web server.

$T2$: when the server's response first arrived at the client, and

$T3$: when the entire response from the web server is received by the client.

We can define the Response time as follows:

Response time = Reaction time + Receiving time ($T3-T1$), where

Reaction time = $T2 - T1$, and

Receiving time = $T3 - T2$

Throughput: Throughput is the measure of the number of requests serviced in specific amount of time.

Throughput = number of requests processed / unit time

2.5 Reliability

It is calculated by finding out the percentage of times an event is successfully completed. This number will allow users to evaluate the probability of a transaction's success or failure. The number of dropped, duplicate, incorrect and out-of-order messages are used to calculate the number of failures. There is no ambiguity in this matter; an event can either succeed or fail. As a result, the total number of events will be equal to the sum of the number of failures and the number of successful events. Let us consider that

t : the number of days that a web service is observed for the purpose of collecting the number of failures,

n : the total number of failures that occurred during that time, and

N : the number of events recorded during that period.

Then we can calculate the reliability as follows:

Ratio of failure in t days = n/N

Daily average failure rate = $n/(N * t)$, and

Success rate or Reliability = $1 - (n/(N * t))$

2.6 Robustness

The robustness criterion dictates how quickly a service recovers from failure and how well it performs with high network traffic and other worst-case circumstances, such as when a client intentionally or unintentionally sends erroneous data to a web service. For delivering and receiving messages, most web service systems employ HTTP as the application layer protocol. HTTP, on the other hand, is a best-effort service, which means it may discard messages without warning. When a client attempts to connect to another system but does not receive a positive acknowledgement, it may incorrectly think that the server has not received its request and submit another connection request. A reliable system always maintains control over such critical situations and is capable of recovering from or detecting failure. Let's pretend that our system is up and running. It takes some time for it to quit working once more. The average time it takes for a system to fail again is known as the Mean Time to Failure (MTTF).

And remember that the mean time to recover (MTTR) is the average time it takes for a system to recover after a breakdown. The shorter the MTTR, the more robust the system is, because less clients will be affected by system failure.

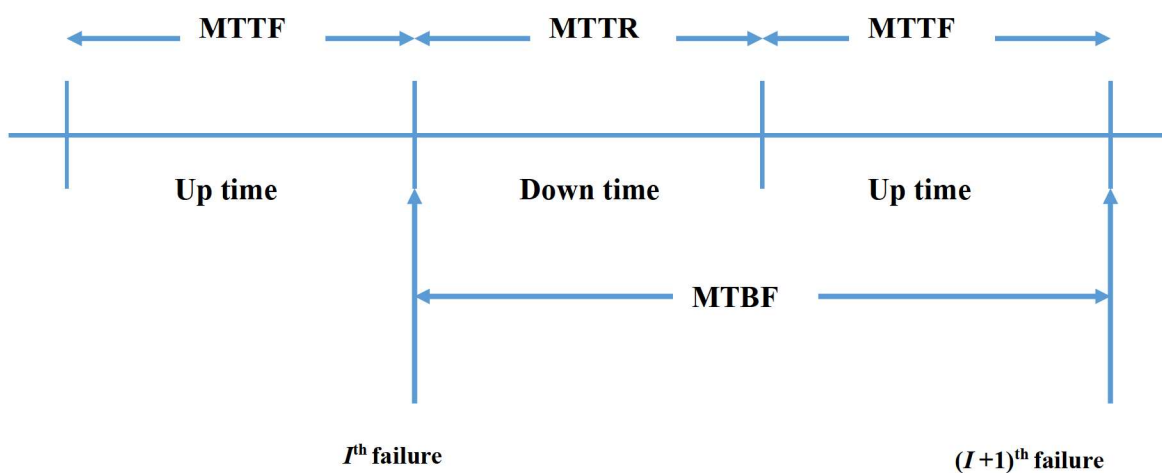


Figure 2. 2: Robustness measure

2.7. Security

The weakest link of a web service is the desired level of security. Security greatly influences the client's level of trust on a specific web service. For web services, there are few well known security vulnerabilities including SQL injections, buffer overflow attack, denial of service (DoS) attack, XML bombs, etc. A web service is scanned through well prepared security tests through which different well known attacks are simulated to find out the possible vulnerabilities of a web service against those attacks. Security scans are used to find potential security flaws in the services you're trying to protect. Each scan makes a series of fraudulent queries to your service in an attempt to elicit and uncover behavior that could signal a security flaw that needs to be addressed. Finally, security is measured through a fractional number using the following equation:

$$\text{Security} = (\text{The number of detected attacks}) / (\text{The number of simulated attacks}).$$

Security scans are used to discover potential security flaws in your target services. Each scan makes a series of fraudulent queries to your service in an attempt to elicit and uncover behavior that could signal a security flaw that needs to be addressed. For online services, the following security scans were evaluated.

SQL injection : tries to use incorrect coding to integrate databases. It was used for the validation of the input. It corrupts a database or recovers sensitive data from the database by giving invalid input. Data from other users or any data the application itself can access may be included. An attacker can modify or delete this data in many cases, causing persistent modifications in the content or behavior of the application.

In some situations an assailant may intensify an Injection SQL attack to jeopardize the underlying server or other back-end infrastructure or attack it. The unlicensed access to sensible data, such as passwords, credit card details, or personal user information, can result from a successful SQL injection attack. In recent years SQL attacks have resulted in many high profile data violations, resulting in reputational damage and regulatory penalties. In some cases, an attacker may have a persistent backdoor in the systems of an

organization, which leads to a long-term compromise that can be unaware of for a long time.

SQL injection examples:

In different situations, there are a wide array of vulnerabilities to SQL injection, attacks and techniques.

Some typical SQL injection examples include:

- Hidden data retrieval that allows you to modify a SQL query to return extra results.
- Subvert application logic, where the application logic can be interfered with, by changing a query.
- Attacks from UNION, where you can get data from various tables of databases.
- Database screening, where the version and database structure information can be extracted.
- SQL injection blind, if in response to the application the results of a query you control are not returned.

Detect vulnerabilities in SQL injection

Use the Burp Suite web vulnerability scanner to identify the majority of SQL injection vulnerabilities quickly and confidently.

By a systematic test set against each entry point of the application, SQL injection can be detected manually. This usually involves:

- Submit the single quote character, and search for anomalies or errors.
- Send a specific syntax to the basis (original) value of the input and to a different value and systematically identify differences in the application responses resulting in the evaluation. •

- Enter Boolean conditions like OR 1=1 and OR 1=2 and seek differences in the responses to the application.
- Send payloads to trigger time delays when performed within a SQL question and search the time needed to respond to differences.
- Submitting OAST payloads to trigger an out-of-band network interaction and monitoring of any resulting interactions when performed within a SQL query.

‘OR 3=3 --

```
public String login(String user, String password)
throw SQLException {
String sql = "SELECT * FROM username WHERE "+
"username='" + login + "' AND "+
"password='" + pass + "'";
ResultSet ds = statement.executeQuery(sql);
(...)
}
```

XPath injection: It is trying to take advantage of poor XML processing within your target service. XPath Injection attacks occur when a website uses information provided by the user in order to build an XPath query for XML data. When an attacker deliberately sends malformed information to the website, it can see how the XML data is structured or access data, to which it normally cannot access. If the XML data are used for authentication, they could even increase their privileges on the website (such as an XML based user file).

XML-Query is done by XPath, a simple descriptive statement that can find a piece of information from the XML query. Just like SQL, certain attributes and patterns you can find can be specified. When XML is used for a website, some form of input on the query string is commonly accepted to identify the content to be found and displayed on the page. This input needs to be sanitized to ensure that the XPath query is not messed up and the incorrect data are returned. Path is a standard language, always independent of the implementation and therefore the attack may be automated. Its notation/syntax. In requests for the SQL databases, there are no different dialects.

Due to the absence of a level access control, the whole document can be obtained. As we may know from SQL injection attacks, we will not encounter any limitations.

Boundary scan: It tries to exploit the misuse of values beyond defined ranges. The border scan checks whether your service correctly processes unexpected input values.

Typically, attackers try to send values that fall outside of the expected range, causing the service to reveal the system data through error messages or stack traces. The Boundary scan checks how your service acts in such cases by sending various unexpected inputs. For example, for an input field that accepts any integer value between 1 and 10, the Boundary scan checks its behavior if a user enters 20 or -5. Being the data-driven test executed in a loop, it replaces existing values with new ones. If the scan does not reveal any information about possible vulnerabilities, it passes successfully. If the scan has *Failed*, that may mean your service is sensitive to values that fall outside of the normal range, and its vulnerabilities are exposed.

Invalid types: tries to use invalid data handling. It is easy to forget about managing the values you don't expect while building web services, particularly if the input is already restricted on the client side. The Invalid Type Scan helps you to make sure that this sort of issue is handled gracefully by your server. By throwing values that lead to unexpected behavior in web service activities, attackers could gain useful system information from the target server (typically via error messages or stack traces).

Mismanaged XML: attempts to exploit misuse in your server and service of the invalid XML. If an assailant sends specially crafted malformed XML, it can crash a vulnerable server or even perform arbitrary server commands. This is done by inserting malformed XML snips, allowing items or attributes to be opened, adding undefined attributes etc.

XML bomb: tries to exploit malicious XML application handling. The XML Bomb scan checks if your service has efficient protection against an XML bomb. Typically, attackers try to use an XML bomb to cause service response delays and general instability leading to the denial of the service. The XML bomb scan creates such a bomb for you to ensure attacks of that type would not cause significant harm. For example, the following XML snippet will deny the service of a vulnerable server and possibly cause a crash.

Malicious Attachment: attempts to exploit poor file handling. It put malware by inserting attachment to software. It malware potentially create backdoor and this backdoor used in future for hacking. The Malicious Attachment scan generates corrupted files to check how your service behaves when receiving malicious files.

Typically, attackers will attach malicious files to a request for one of these purposes:

- Trying to cause the server to crash with corrupted or very large files.
- Damaging servers with viruses or files containing harmful code.
- Passing viruses or files containing harmful code through the server, when the attack target is a client.

The Malicious Attachment scan provides a convenient way to ensure that your server handles malicious files safely. If the scan does not reveal any information about possible vulnerabilities, it passes successfully. If the scan has *Failed*, that may indicate your service cannot handle malicious files efficiently, which may cause security problems.

Cross Site Scripting: tries to find vulnerabilities in cross-site scripting. Download the malicious script to user computer when the user visits the valid website. This malicious script creates the system's potential harm vulnerability. Cross-Site Scripting (XSS) attacks are a type of injection that injects malicious scripts into user-friendly and trusted websites. XSS attacks occur when a web-based attacker uses a malicious code to send a different end-user, usually in the form of a browser side script. Mistakes that allow such attacks to succeed are very common, and occur wherever a web application uses the input of a user without valve.

Customized Script: allows you to create customized fuzzing values by using a script. Custom script fools the system for unauthorized input and malicious events.

2.8 Scalability

The requirement for scalability defines how extensible a Web service could be. Every day, web services are introduced to new interfaces and techniques and this makes it extremely important to keep a Web service up to date. If more computer capacity and more requests are required, the system must be able to support additional systems and

new technologies. The Web service should be able to handle high loads while ensuring that its customers' performance with regard to response time is not inappropriate.

PNL performance is a relatively new technique to predict whether the system can withstand higher traffic loads without affecting the level of performance. This metric is used to calculate the load intensity at which the system can operate unless response time and throughput have been decreased. PNL calculation involves producing potential workloads and investigating the system's behavior which is similar to the reaction of the system in the light of this varying workload. By calculating the expected performance loss for a certain workload, the PNL method helps prevent performance disasters. In terms of their behavior, two states can exist. The state of behavior may be acceptable (0) or unacceptable (1). Instead of the constant values, we can indicate the status with relative values between 0 and 1 (in the interval $[0, 1]$). This indicates the degree to which the system is not compatible.

For the program P in relation to the operational distribution Q according to the formal definition of PNL:

$$\text{PNL}(P, Q) = \sum \text{Pr}(s)C(s)$$

Where,

$\text{Pr}(s)$ indicates the stable-state probability of a State,

$C(s)$ indicates the degree of performance degradation under operational distribution Q,

$C(s) - 1$ indicates an unacceptable performance, and

$C(s) = 0$ indicates a fully acceptable performance.

The interval $[0, 1]$ is $C(s)$.

Chapter 3

Methodology

3.1 Introduction

In this section we explain how we have designed our experiments to measure the QoS parameters we have mentioned earlier for different well known web services at different server loads. In this experiment we have used different open-source tools including simple object access protocol user interface (SoapUI) [10], [11], WebInject [12], etc.

SoapUI is being used for web service testing. This is an open-source tools being used for SOAP and representational state transfers (REST) applications. It is being used to test a wide range of functionalities of web services including web service invoking, development, functional testing, inspection, simulation and mocking, load and compliance testing, etc. While using SoapUI in our experiments, we have sent different requests from a client at the same time to a specific web server and measures the QoS metrics. SoapUI has been configured to call different publicly available web services including weather, currency exchange and email address validation services.

Initially, in September 2005 SoapUI was released to SourceForge. It is free software, licensed under the public license of the European Union. More than 2,000,000 downloads have taken place since the original release. It is built completely on the Java platform and uses Swing for the UI. SoapUI is cross-platform therefore. SoapUI supports IDEA, Eclipse and NetBeans today as well. SOAP, REST, JMS, AMF and any HTTP(S) and JDBC calls can be tested by SoapUI. Include web services, key features:

- Inspection
- Simulation and mocking
- Functional checkup,
- Compliance and safety testing

Functions of SOAPUI:

Feature	Benefit
Covey for WSDL	To see how well a WSDL contract is being tested/mockd/utilized, analyze request and answer messages.
Rebuilding WSDL	When the underlying WSDL changes, automatically update requests/response/assertion.
Projects composite	Work with a team on the testing projects of your web services.
Management of requirements	Project requirements manage, import and export and link them to test cases.
Editor of the application form	For technical and non-technical users to easily edit and understand request messages.
Viewer of the overview message	To get an overview of the answer, request and monitored message.
Request/answer editor based on tree.	To make navigation easier and to understand larger messages.

Table 3. 1: SOAP UI Features

Selection of Web service providers

We have considered five public web services repository for our experiment. These web services are available at free of cost for public use. Therefore, we carry out our experiments by invoking the public web services. Table 3.2 shows the list of web services that we are using in this experiment. Note that <https://stackoverflow.com> is not a web service provider, but it provides the link to use few web services.

SL	Web Service provider
1	http://www.programmableweb.com
2	http://free-web-services.com
3	http://www.webservicex.com
4	http://community.workday.com/sites/default/files/file-hosting/productionapi/index.html
5	https://any-api.com
6	https://public-apis.xyz

Table 3. 2: List of web service provider

Salient Service of www.programmableweb.com provider

API Name	Description	Category
YouTube API	The Data API allows users to integrate their program with YouTube and to carry out many of the website's operations. It offers the ability to look for videos.	Video
Flickr API	The Flickr API is suitable for collecting photos via various feeds – public photos, videos, favorites, friends, group bags, discussions, and more from the Flickr Photo-Sharing service.	Photos
Amazon Product Advertising API	What used to be the ECS - eCommerce Service - was converted into the Product Advertising API. This API enables product information to be obtained from developers.	eCommerce
Microsoft Bing Maps API	AJAX Map Control includes Bing Maps API and Interactive SDK. To create maps with routes and traffic information, use BM. Enables developers to encode the controls, forms and layers.	Mapping
Google Gadgets API	From your location: The Google Gadgets API provides a means of placing content from third parties on Google's homepage. The Google Gadgets API mainly uses gadgets to generate gadgets, which users use.	Widgets

Table 3. 3: www.programmableweb.com web service features**Salient Service of <http://free-web-services.com>**

API Name	Description	Category
Conference Calling API	To your service, site, or app, you can add conference calling skills. Register for the permissions to use our API and seamlessly incorporate your current project with our international conference service. No software chaotic.	Communication
Free SMS API	With attached advertisements you can use all 160 characters. Free SMS API can be integrated with your website in various programming languages such as PHP, Java, Perl, Python, Ruby, C, VB.NET and C#.	Communication
Online Messenger Service	Use the Online Messenger Service to add a Messenger to your OWN applications (such as ICQ, MSN Messenger and America Online Instant Messenger).	Communication

	It interacts completely with any other available OMS client.	
Twitter API	Twitter exposes its data through a programming interface for applications (API). These documents constitute the official reference to this feature. There are three parts to the Twitter API: two REST APIs and one Streaming API.	Communication

Table 3. 4: <http://free-web-services.com> web service features

Salient service of <http://community.workday.com>

Service	Description
Absence_Management	Including employee time off and absence inputs for time off and accumulated adjustments and overrides and leave requests in Workday Human Capital Management business services, the Absence Management Web Service contains transactions that expose the absence related data.
Admissions	Web services such as applicants for the creation, edition, and retrieval of admission-related objects.
Compensation	The Web Service for Compensations includes operations which expose data relating to compensation related to workday human capital management services.
Human_Resources	The human resources web service includes business services, including data about employees, contingency and organizational information, that expose Workday Human Capital Management Business Services. This Web Service can be used for integrated enterprise systems such as business directories, data management tools, email or any other system requiring worker or organizational data. The system may be integrated with enterprise systems.
Identity_Management	The Web Service Identity Management includes work-day identity and access management operations.
Recruiting	The Web Service includes operations that disclose business data for integration with talent management and application tracking systems for Workday Human Capital Management Business.
Resource_Management	The web service for resource management includes activities that disclose data on resource management in the workday financial sector. Data on providers, supplier accounts, expenses, business assets and projects are included here.
Staffing	Web service staffing operations disclose business services and data for the day-to-day human capital management. These services relate to employees' and contingent workers' staffing transactions, including involving employees and contingent workers.

Table 3. 5: <http://community.workday.com> web service features

We have invoke the mentioned web service in our project to identify different QoS parameters. From available public web services, we have used the following two web services for this experiment:

- Weather web service
- Currency conversion web service

3.2 Experimental design

We have already defined the mathematical representation of different QoS parameters. We have designed the test cases targeting those QoS parameters so that we can test those parameters.

The following stages will be performed in the project:

i. Represent all QoS parameters measurement mathematically: Quality-of-Service (QoS) is usually used as an important differentiating point for describing the non-functional characteristics of web-services. A thorough knowledge of the web service QoS is becoming increasingly important as web services are prevalent on the Internet. Here we represent a mathematical overall description of the QoS needs for Web Services. The following are the web service QoS measures:

- Time of response = Reaction time + Receiving time ($T_3 - T_1$)
- Throughput = Number of requests processed / time of the unit
- Reliability or success rate = $1 - (n / (N * t))$
- Robustness
- Security

ii. Design test cases using SoapUI, Webinject: A collection of test cases can be combined into a logical unit to function as a single group. These types of combinations are known as TestSuites. Note that many test scenarios can be supported inside SoapUI by creating any number of TestSuites. After a test case has been added, a cookie has been added to the cookie jar. Note that this has been added before sending the test case request to the web server. This is particularly useful when the external cookie is set in the response header outside of a Set-Cookie directive. An example test case is shown below.

```

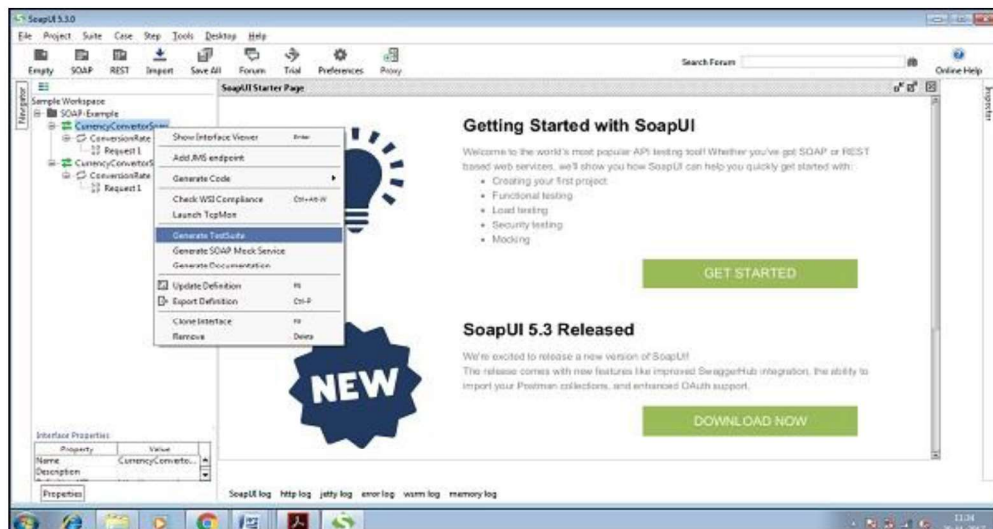
<SOAPtestcases>
  <testcase
    id          = "1"
    description1 = "Sample SOAP Test Case"
    method      = "get"
    url         = "{BASEURL}/test.jsp"
    verifypositive = "All tests succedfull"
    warning     = "5"
    critical    = "15"
    label       = "SOAP testpage"
    errormessage = "got error: {PARSERRESPONSE TIME}"
  />
</SOAPtestcases>

```

Creation of TestSuite

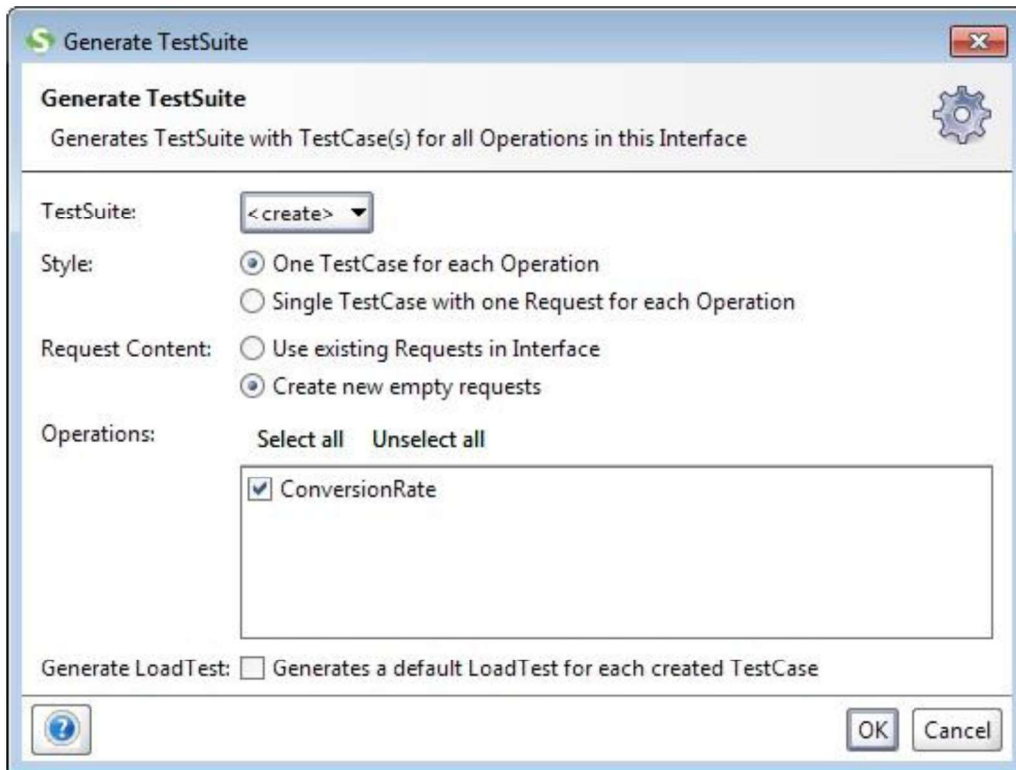
Step 1 – Within a project, right-click on the interface (next to the project name) and then click ‘Generate TestSuite’.

Here, SOAP – Example is a Project name while CurrencyConvertorSoap and CurrencyConvertorSoap12 are interfaces.



Step 2 – A new wizard opens. Select the choices based on the requirement.

Step 3 – Once the selection is made, click OK.



Step 4 – Check the checkbox of Generate LoadTest. This will generate a LoadTest for each TestCase created in this TestSuite.

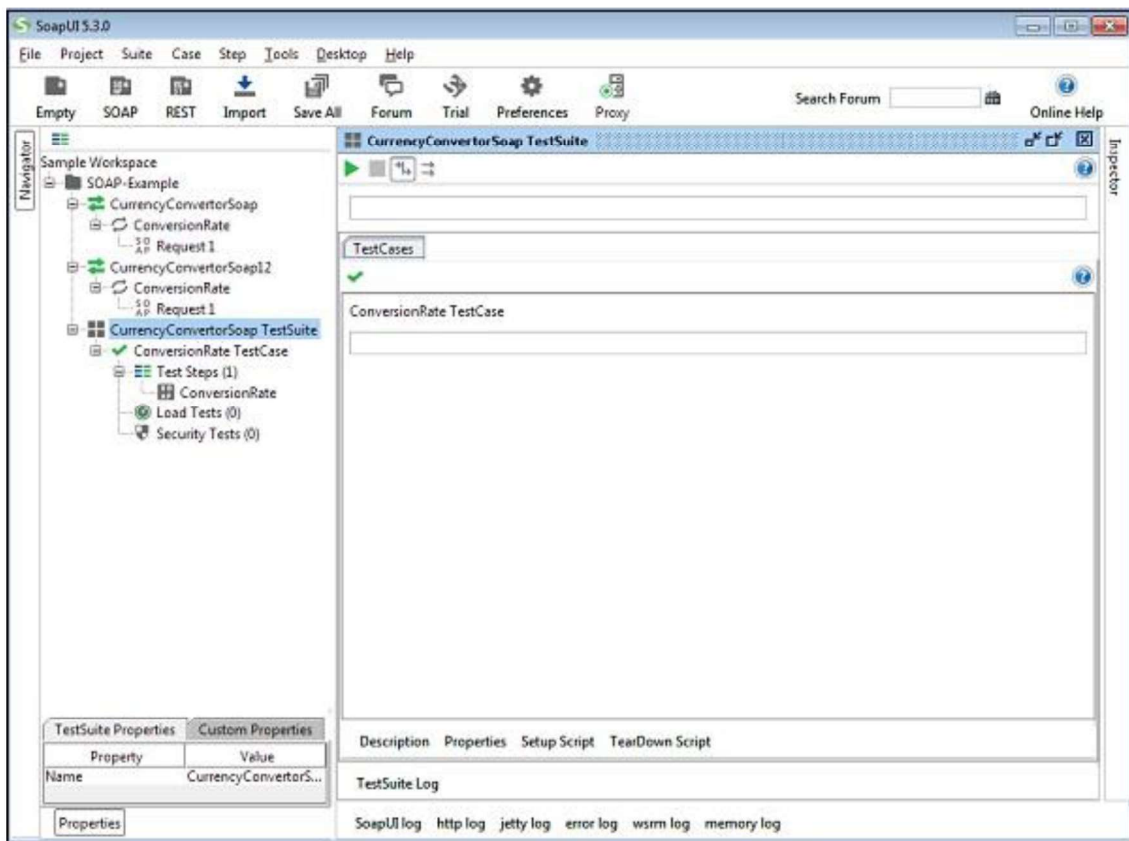
Step 5 – Enter the TestSuite name in the new wizard and then click OK.



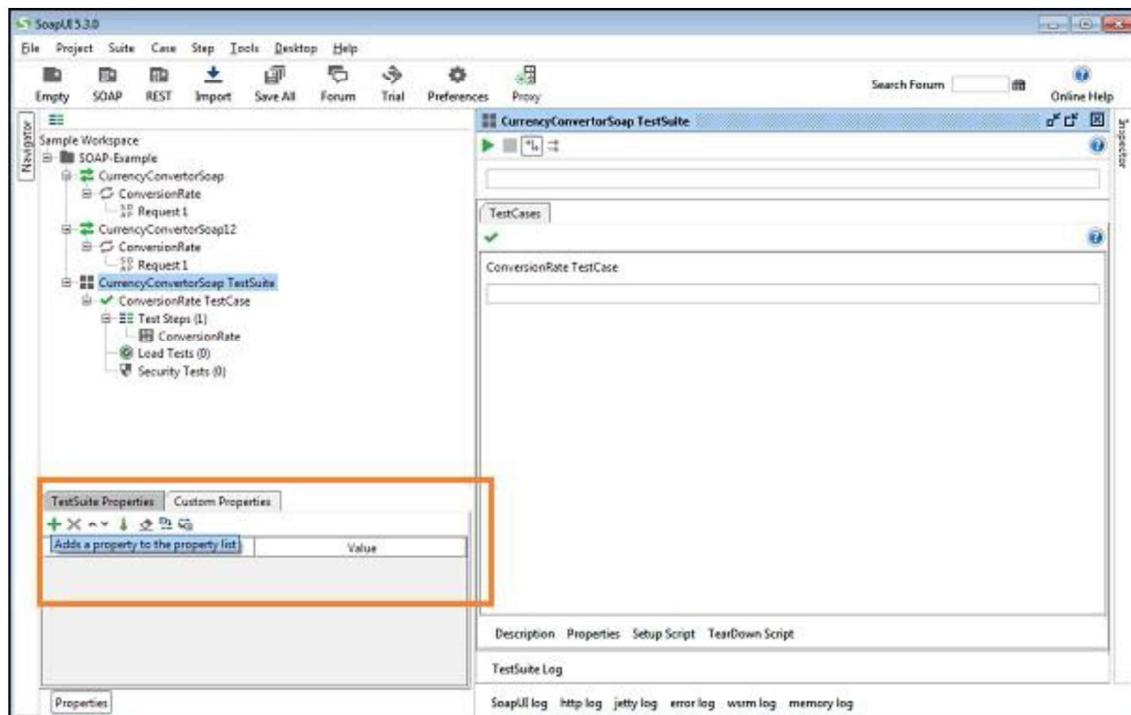
The created TestSuite displays in the navigation panel as shown in the following screenshot.



Step 6 – Double-click the TestSuite Name, and the TestSuite window opens at the right panel. Since there no TestCases are added, it is blank.



TestSuite properties can be seen at the bottom of the Navigation Panel. New Custom Properties can be added at the TestSuite level.



iii. Simulate the experiment in Visual Studio.net environment for measuring QoS parameters: We used the component of Visual Studio's web performance and charging tools. First we must inside Visual Studio install the component Web performance and load test tools. Next we created a load test project for C#, which allows us to add load test scenarios. The load tests can now be performed. A summary is displayed when the test is completed. Also available are charts, tables or details of various information on load test results. We use the SQL database server in this experiment to store different QoS parameter results from the load test results. According to details steps:

1. Open Windows Start Menu Visual Studio Installer.

The new project dialog box in Visual Studio or Tools > Get Tools and Features from the menu bar allows you to access it too.

2. Select the Individual Components tab in the Visual Studio Installer and scroll down to debugging and testing. Choose Web performance and load tools for testing.

3. Select the button Change.

The following steps must be taken once the Web performance and load testing tool component has been installed:

Create a project for a load test

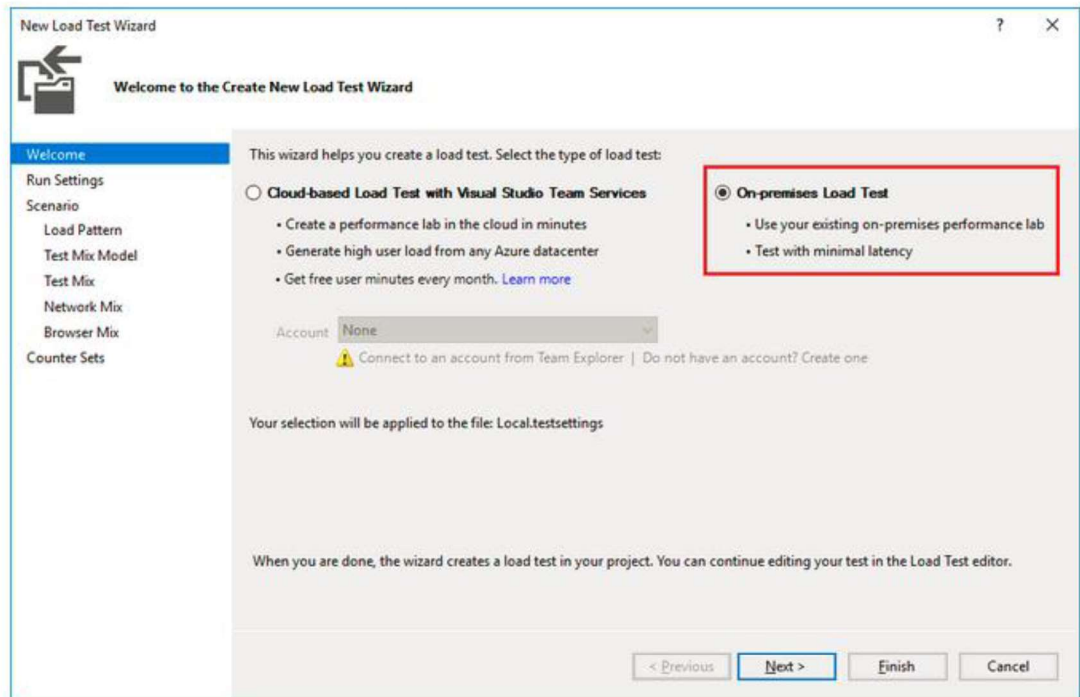
We will create a load test project C# in this section. You can also create a load test project Visual Basic, if you like.

1. Open the Studio Visual.
2. Select create a new project on the start window.
3. When creating a new project page, type the test web into the search box and select the [Deprecated] template for C# web performance and load. Select next. Next.
4. Enter a project name if the default name is not used and then select Creating.

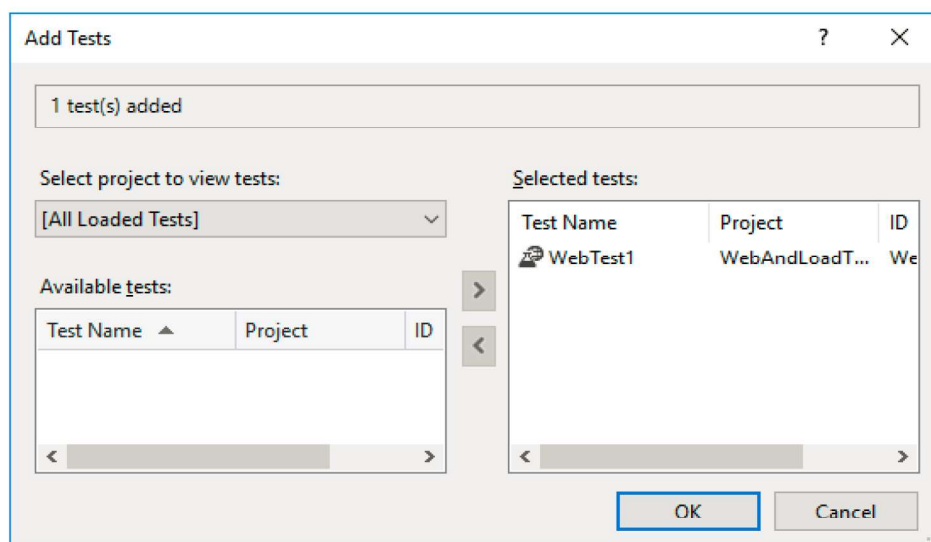
Visual Studio creates the Solution Explorer project and displays the files. Initially, the project contains a WebTest1.webtest file.

Fill the project with a load test

1. Select Add > Load Test from the Solution Explorer right-click menu or context menu of the project node. Open the New Charging Test Wizard.
2. Select the Load Test option on-site, and then select Next.



- Choose Next then you step through the wizard and edit the test mix page. Tests are added to the load test scenario. Select the Add key.
- The dialog Add Tests opens. Under Available tests, select WebTest1, and then choose the right arrow to move it over to the selected tests box. Choose the OK button.

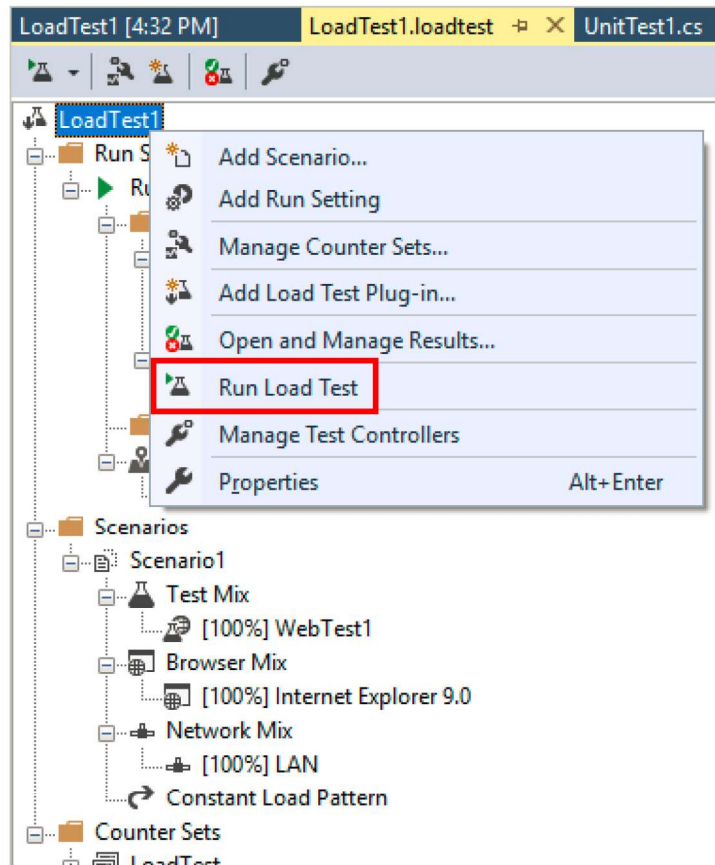


5. Choose Finish button, return to the New Load Test Wizard.

The load test is added to the project, and in the editor window the load test file opens.

Run the load test

A load test has been created that doesn't do much, but let's run it anyway. Choose Run Load Test from the load test opened in the editor from a right-click menu, or from the pop-up menu.



The load test begins. The test results window demonstrates that the test is being carried out and that the load test analyzer is shown in the editor window. After the test has been done, a summary of five minutes should be given in the editor if you accept the defaults.

iv. Store simulation result for QoS parameters: We store the QoS result into SQL Server database for further analysis.

v. Analyses simulation result to identify improvement techniques: We analyses the result to identify the problem, whether it can be performance or response time. We also review the security of each hosted service.

vi. XML web service will be developed by ASP.net using improvement techniques. Based on problem identification, we resolve the issue using improvement technique using ASP.net. Here we address the response time or throughput or security issue.

vii. Final QoS-Aware Web service will be hosted in Internet Information Server (IIS) environment.

We have designed different test cases for testing the QoS metrics of web services. WebInject [12], an open-source automated testing tool, has been used to design these test cases. If a web service has HTTP interfaces, WebInject can be used to test the individual system components of that web service. Different test cases targeting acceptance, regression and functional tests can be designed. Through a test harness, many test cases can be executed and results of these tests are then displayed real-time. WebInject also allows to monitor the system response time.

When added to a test case, a cookie is added to the cookie jar in advance of the test case request (i.e. the test case this is attached to will include any cookies specified in this parameter). The fields for this parameter are a one-to-one direct correlation to the HTTP::Cookies::set cookie method parameters and are a direct link to each parameter. Multiple cookies can be defined, as with the add header parameter, by separating with a "|" character.

For this experiment we used below configured system:

- Operating system: Windows 10 Pro
- Hard Disk : 500 GB
- RAM: 8 GB
- Internet Connection

Software used:

- Microsoft Visual studio Team System 2019
- Soap UI
- WebInject

3.3 Public web service provider

There are many public web service available. Again, there are many proprietary web service. For this project I used public web service as it is freely available. Below are the public web service provider from where we can invoke web service.

<http://www.service-repository.com/>

<http://www.programmableweb.com>

<http://free-web-services.com>

<http://www.websvcex.com>

<http://stackoverflow.com>

<http://www.xmethods.com/ve2/index.po>

<http://www.gcomputer.net/webservices/dilbert.asmx>

<http://www.ripdev.com/webservices/localtime.asmx>

http://www.xml-ebsservices.net/services/conversions/euro_convert/euro_conver.asmx

<http://www.perfectxml.net/webservices/salesranknprice/bookservice.asmx>

3.4 Web services used

From available public web services, we used below listed web service for experiment

Weather

Currency conversion

Unit conversion

We invoke the mentioned web service in our project to identify different QoS parameters.

Chapter 4

Results

4.1 Introduction

In this project we have analyzed different QoS parameters mathematically. For most of the cases we have analyzed historical data. We have identified response time, reliability and security of listed web service as these are the key parameters of QoS. By analyzing these parameters we can identify which provider is performing best as per their results. As we have mentioned earlier we have measured the weather and currency conversion web services for our experiments.

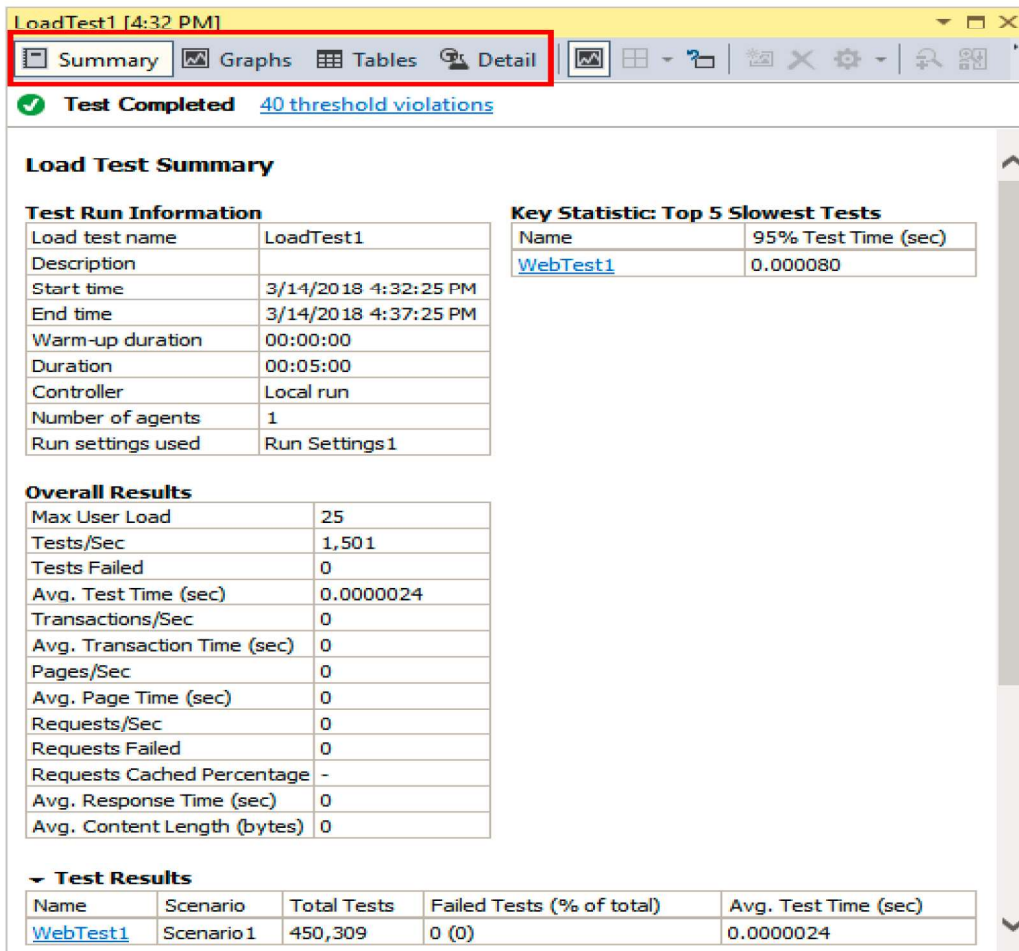


Figure 4. 1: An example screenshot of load test executed by Visual Studio.net.

4.2 Experiment Results

A. Response Time:

We have invoked the web services from the provider sites and then logged the results we have found in the database. The response time has been calculated by the mathematical equations we have already mentioned, i.e., subtracting the first request time from the time when receiving has been completed. In addition to recording these times, we have also recorded additional information such as the web service and method names, request date and time, response date and time, etc. Note that, one-time request and response do not provide reliable information. Therefore, we have gathered the historical data, from where we can find the minimum, maximum and average response times for a time period. For the web services we have studied we have collected historical data from different service provider for the previous 10 days. Fig. 4.2 and 4.3 show the data collected from different service providers for weather and currency conversion web services, respectively.

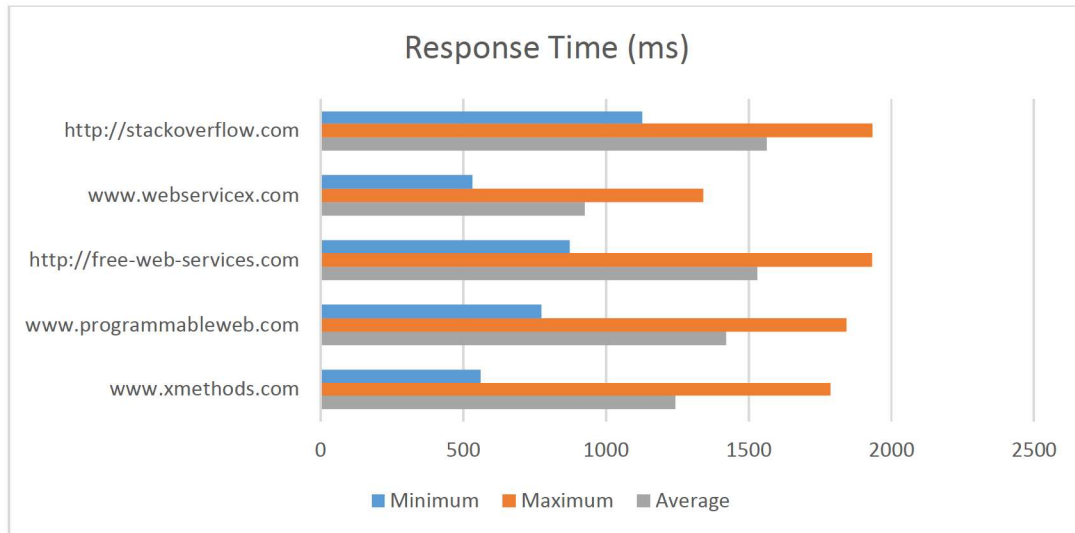


Figure 4. 2: Response time graph of weather web service

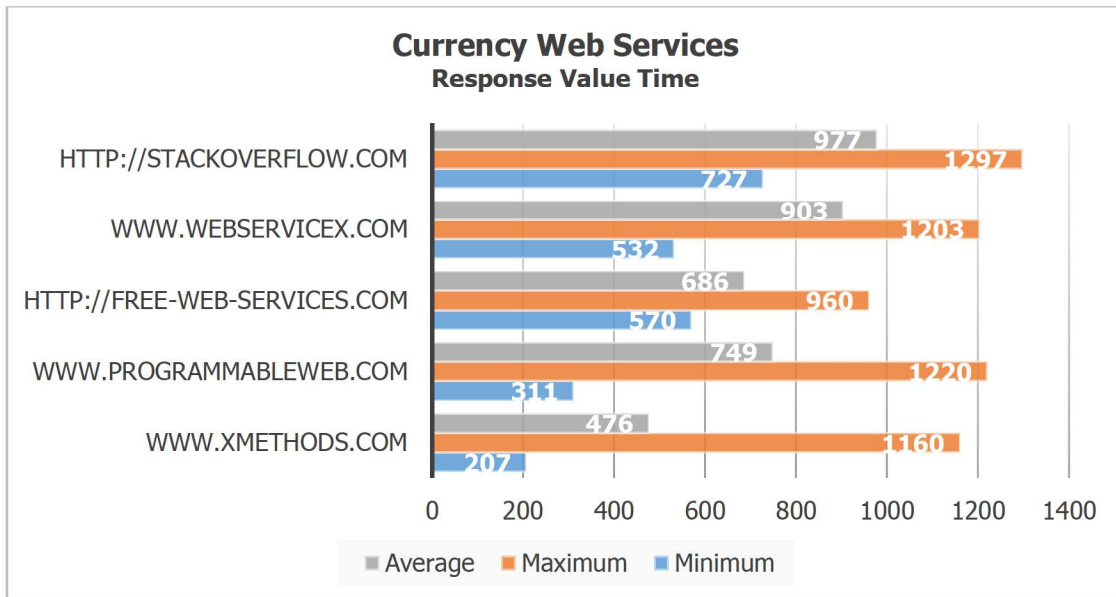


Figure 4. 3: Response time graph of currency web service

B. Reliability

Reliability confirms that something is dependable and thus the outcome will not vary over time. Reliability is usually measured for a specific period of time. A web service is verified to function without any failure for that period of time in a particular environment. In this testing, a web service confirms that it is free from bugs and also reliable to generate its expected outputs. Web service reliability is measured between 0 to 1. For the same web service reliability can vary significantly over different service providers. The reliability factor of various service providers that we studied for weather and currency conversion web services are shown in Table 4.2.

Source	Weather Web service	Currency Web service
www.xmethods.com	0.9673	0.9890
www.programmableweb.com	0.9453	0.9304
http://free-web-services.com	0.9475	0.9654
www.webservicex.com	0.9743	0.9801
http://stackoverflow.com	0.9541	0.9645

Table 4. 1: Reliability value of weather & Currency web service

C. Security

The security test of a web service is performed by authorized hacking attempt which is carried out to identify vulnerabilities in the architecture and configuration of a web service. Through these authorized hacking attempts the attacks generated by the intruders or the attackers are simulated. Thus unauthorized access to organization's virtual assets are attempted. In the following we mention the security breaches that we have looked for by executing several tests.

SQL Injections:

Dynamic SQL declarations are inserted into an execution input field through SQL injections and therefore creates a security breach. This testing is used to break the perimeter security by an approved hacker and to transfer fixed entrances to the SQL statement. This identifies the vulnerability of a web service.

Buffer Overflows:

Buffer overflow occurs when the input size is not inspected before storing. If inputs are not validated, a buffer overflow attack is possible even remotely via a SOAP request or submission of web form. An attacker targets is to feed more data than the size of the allocated memory to store that data. In testing this attack, test cases send over-sized data to check if the required verification has been implemented or not to prevent this attack.

Denial of Service Attacks:

DoS is a well-known attack for any kind of publicly accessible servers, and therefore, in web server as well. During DoS attacks, the attacker attempts to keep the server too busy to reply his numerous bogus queries so that a legitimate user fails to receive services from that server. In this test case, similar attack attempts are carried out.

Improper Error Handling:

When an error occurred inside an application server, often the server return back the details of that error, which may include the stack trace. During development and debugging phases, these stack traces are very useful. However, when a service is being deployed, these traces should not be leaked to the user ends. Otherwise, the attackers may exploit those traces to find out the possible vulnerabilities.

Table 4.2 shows the security measurement of the weather and currency conversion web services. This security index values indicate the probability of successfully defending against the attacks. The value closer to 1.0 would imply a more secure service.

	Weather Web Service			Currency Web Service		
Source	Attack Simulated	Attack Detected	Security	Attack Simulated	Attack Detected	Security
www.xmethods.com	9	6	0.6666	9	7	0.7777
www.programmableweb.com	9	5	0.5555	9	4	0.4444
http://free-web-services.com	9	5	0.5555	9	5	0.5555
www.webservicex.com	9	8	0.8888	9	6	0.6666
http://stackoverflow.com	9	7	0.7777	9	7	0.7777

Table 4. 2: Security measure of weather and currency converter web services

4.3 Result Discussed

Response time:

From Fig. 4.2 and 4.3 it is clear that for the same service different service providers' response times are different. It is shown that www.webservicex.com and www.xmethods.com have the least minimum, maximum and average response times for weather and currency conversion web services. Note that the response time depends on many factors including the distance, network traffic, number of hop etc. This unpredictability manifests itself in particular through highly variable online service response times and unpredictable Internet data transport delays. In the specification of web service composition, response time is quite important. Load balancing is a technique used to optimize response time proactively by routing user requests to the systems that

are currently least busy and to servers that are reasonably close in vicinity from where a request arrives.

Reliability:

According to Table 4.1 among the service providers we have studied, www.webservicex.com and www.xmethods.com appear to be the most reliable service providers for weather and currency conversion web services, respectively. The use of SOAP by Java Messaging Service (JMS) rather than HTTP can help improve the reliability of critical web services. SOAP significantly increases the reliability of SOAP messages in web services to be transported in a stronger and more reliable way than HTTP.

Security:

The security measure of the studied service providers are shown in Table 4.2. It shows that www.webservicex.com in case of weather, and both www.xmethods.com and <http://stackoverflow.com> in case of currency conversion have the highest security indices. Therefore, these service providers have defended the maximum number of simulated attacks. Security has always been a prominent engineering trend in the early stages of online service development. However, in order to ensure the security of web services, a thorough and tangible security review is required. If completed early in the development phase, the results of such an evaluation can be used to improve the quality of the target online service. On the other hand, during the security examination of web services, it is impossible to eliminate all security flaws. As a result, total security will never be attainable, and a security failure during the execution of a web service is possible. A measured level of fault tolerance must be established by partial achievement of security goals to avoid security failures.

4.4 Future Works

Designing web service in terms of Restful API will help in terms of elasticity. Again hosting of web service play vital role in terms of response time. In future we can design web site by introducing local web service broker.

Chapter 5

Conclusion

In this project we have defined three main QoS parameters namely response time, reliability and security mathematically and measure the QoS value for publicly available web service from various service providers. We have found that the response time varies a lot from different service providers for the same service. Again, the response time depends on the network traffic, server load, hop count, etc. To select a web service the historical data will be very helpful. From the historical data we can get better idea about the response time of the service. The percentage of times an event is executed successfully is determined by web service reliability. This number will allow customers to estimate the likelihood of a transaction failure. Web service security determines the trust the clients have to the service.

Although web service providers strive to meet higher-level QoS standards for their web services, the many levels of QoS needs that can be met should be assessed against one another using a variety of tests, measurements, technical analysis, and logic. Most of the time, addressing one QoS criterion has a positive or negative impact on other QoS criteria. Before creating the web service, the application developer might use the values. Because online services are hosted over the Internet, predicting their behavior is extremely challenging. Statistical data may not always provide an accurate estimate of QoS parameters. Historical data will be very helpful for more accurate QoS measurement.

References

- [1] M. Rajeswari, G. Sambasivam, N. Balaji, M. Saleem Basha, T. Vengattaraman, and P. Dhavachelvan, "Appraisal and analysis on various web service composition approaches based on QoS factors," *Journal of King Saud University - Computer and Information Sciences*, vol. 26, no. 1, pp. 143–152, 2014.
- [2] M. Papazoglou, "Web services and SOA: principles and technology 2nd," Harlow, Essex: Pearson Education Limited, 2012. [3] M. I. Ladan, "Web services metrics: A survey and a classification," *Journal of Communication and Computer*, vol. 9, no. 7, pp. 824–829, 2012.
- [4] N. Zhang, "Service discovery and selection based on dynamic QoS in the internet of things," *Complexity*, vol. 2021, 2021.
- [5] G. Rodríguez, C. Mateos, and S. Misra, "Exploring web service qos estimation for web service composition," in *Information and Software Technologies*, A. Lopata, R. Butkienė, D. Gudonienė, and V. Sukackė, Eds. Cham: Springer International Publishing, 2020, pp. 171–184.
- [6] S. Rangarajan, H. Liu, and H. Wang, "Web service QoS prediction using improved software source code metrics," *PLOS ONE*, vol. 15, no. 1, pp. 1–25, 01 2020. [Online]. Available: <https://doi.org/10.1371/journal.pone.0226867>
- [7] A. S. da Silva, H. Ma, and M. Zhang, "Genetic programming for QoS aware web service composition and selection," *Soft Computing*, vol. 20, no. 10, pp. 3851–3867, 2016.
- [8] H. Zheng, Y. Feng, and J. Tan, "A fuzzy QoS-aware resource service selection considering design preference in cloud manufacturing system," *The International Journal of Advanced Manufacturing Technology*, vol. 84, no. 1-4, pp. 371–379, 2016.

- [9] M. Rathore and U. Suman, "Evaluating QoS parameters for ranking web service," in 2013 3rd IEEE International Advance Computing Conference (IACC), 2013, pp. 1437–1442.
- [10] C. Kankanamge, Web services testing with soapUI. Packt Publishing Ltd, 2012.
- [11] "SoapUI: the web services testing tool," <http://www.eviware.com/soapUI/features.html>, accessed: 2021-04-19.
- [12] "Free open-source test automation," <http://www.webinject.org/webservices.html>, accessed: 2021-04-19.
- [13] X. Wang et al, "QoS Prediction of Web Services Based on Reputation-Aware Network Embedding" in proceeding of IEEE Access Volume 8, September 2020.
- [14] Y.Fanghua, L.Zhiwei, C. Chuan ,Z. Zibin, and Hong H., "Outlier-ResilientWeb Service QoS Prediction" in proceeding of The Web Conference, April 19–23, 2021 on pages 12.
- [15] S. Aruna and P. Abinaya in proceeding of Webology "A Web service recommendation system based on QoS attribute based collaborative filtering", Volume 17, Number 1, June, 2020.
- [16] S. Maheswari and S. Justus , "A Review on the Quality of Service of Web Service", in proceeding of International Journal of Mechanical Engineering and Technology Volume 9, Issue 12, December 2018, pp. 414–424,
- [17] Web Service Architecture Working Group, <http://www.w3.org/2002/ws/arch/>; last accessed on 19.04.2021.
-
- [18] G.Song , "A New Approach for Web Service Composition Based on Semantic" in proceedings of International conference on Computational intelligence and software engineering (CiSE),2010 on page(s) 1-4.

[19] D. Weider, R.B. Radhakrishna, S.Pingali, and V. Kolluri, “Modeling the Measurements of QoS Requirements in Web Service Systems” in proceeding of *SIMULATION*, Vol. 83, Issue 1, January 2007 75–91.

[20] K. Mahbub and S. George, “Run Time Monitoring of Requirements for System Composed of Web-Services: Initial Implementation and Evaluation Experience” in proceedings of IEEE International Conference on Web Service (ICWS), 2005 page(s) 257-265.

[21] S. Youcef, M.U. Bhatti, L. Mokdad and V. Monfort, “Simulation-based Response-time Analysis of Composite Web Services” in proceeding of IEEE multitopic conference, 2006, page(s) 349-354.

