I.J. Information Technology and Computer Science, **2014, 05, 18-27**
Published Online April 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.05.03

Evaluating Web Services Functionality and Performance

Tarek S. Sobh
Information Systems Department, Egyptian Armed Forces, Egypt
E-mail: tarekbox2000@yahoo.com

Medhat Fakhry
Chair of Management Information System Department, AAST, Egypt

Abstract **—**Traditional distributed database transaction applications within large organizations often involve a large number of resources. In this case, people and DDBMSs distributed over a wide geographic area, may introduce conflict between heterogeneous systems. Web services (WS) provide solution for this problem since WS have an independent platform, independent language, and independent object model. This work presents WS application to access heterogeneous and distributed database via horizontal data fragments that is designed to be reliable, flexible and scalable. It describes the setup of SOAP server and applications based on the SOAP for end user client. In addition, it allows the publishing of WS descriptions to submit user requests (goal) to retrieve the required information. Here we evaluate the functional, behavior and performance of WS among possible different alternatives with real-time and execution parameters. Implementation details and case study experiments are presented along with the corresponding results.

Index Terms **—** Web Services, Distributed Database, Horizontal Fragmentation, UDDI, WSDL, SOA, XML Relaxation

I. Introduction

The benefit of service-oriented architectures (SOA) is their support of loose coupling of software components, i.e. providing a high degree of interoperability and reuse [1]. WS is an example of a SOA that include three main entities: Consumers, Providers, and Registers of services. These entities work in concert to provide a loosely coupled computing paradigm. Web Services (WS) are self-contained, loosely coupled application modules with well-described functionality that can be published, located and invoked across the web [2].

The World Wide Web (WWW) is evolving into a medium for providing a wide array of e-commerce, business-to-business, business-to-consumer and other information based services [3]. WS is emerging as the enabling technology that bridges decoupled systems across various platforms, programming languages and applications. Interoperability among these applications is ensured using WS framework [4] [5].

WS provide a ubiquitously supported framework for application-to-application interaction, based on existing Web protocols and open XML standards. The WS framework is divided into three areas: communication protocol, service description, and service discovery. Several specifications have been developed like Simple Object Access Protocol (SOAP) [6] [7] [8], WS Description Language (WSDL) [9] [10] and Universal Description, Discovery and Integration (UDDI) [11], correspondingly.

The present work deals with problems of request-response behavior, where the response of a WS is time and cost sensitive to the end user. Intuitively, the most important factor for the end user has to do with the necessary time that a WS takes to produce and deliver its results, that is the WS execution period. There are situations where WS have to deal with very large data or may need ‘‘long’’ execution time period for other reasons (low infrastructure availability) before returning results back to the customer/user. The WS execution duration varies depending on the status of the infrastructure supporting it according to the available resources executing the WS. As a result the customer may asynchronously wait for the WS to respond so ‘‘long’’ as to even decide to request results from another available WS in a different location and with higher potential availability.

Such WS is particularly met in business environments where time and data intensive transactions are performed between customers and offered services/products. A typical example is found in the telecommunication carrier’s business network. A common telecommunication carrier’s business network WS paradigm is the presentation of online analytic details for telephone calls. Carriers exist with millions of customers and billions of daily telephone call transactions. In such case, the WS has to patiently search for all the unbilled calls of a whole day’s transactions to present analytically a single customer’s telephone calls. Imagine now that this long-taking-to-

respond WS may additionally be subject to extra delays. These delays depend on the availability and performance specifics of the carrier infrastructure resources.

The objective of this work is to develop an understanding of SQLXML WS application for accessing heterogeneous and distributed database of telecommunication carrier's network. It allows the publishing of WS descriptions to submit user requests (goal) to retrieve the required information. It measures the efficiency of WS among possible different alternatives with real-time and execution parameters.

The rest of the paper is organized as follows: Section 2 presents motivation of this work. In Section 3, we outline some details about WS and introduce a SQLXML WS usage. Section 4 presents the proposed system model. Section 5 presents our implementation details and results. Section 6 is a conclusion.

II. Motivation

Our WS application was developed as academic example while studying the nation-wide telecommunication carrier's network in Egypt. In general, telecommunication carriers utilize gateways to provide telephony services that have the capability to record call transactions on relational databases. In particular, existing web and distributed database (horizontal fragmentation) applications and services implementations have served in this work. Moreover, each area in telecommunication carrier's network may have certain database such as Oracle, SQL Server, My SQL...etc. In addition, the carrier utilizes the corporate network both for end-user internet service provision as well as for the needs of data interchange with its thousands of national business partners. As a result, business partners as well as other associates have had the need to consume several kinds of WS that deliver various billing reports and exchange data concerning the Customer/call Detailed Records (CDR).

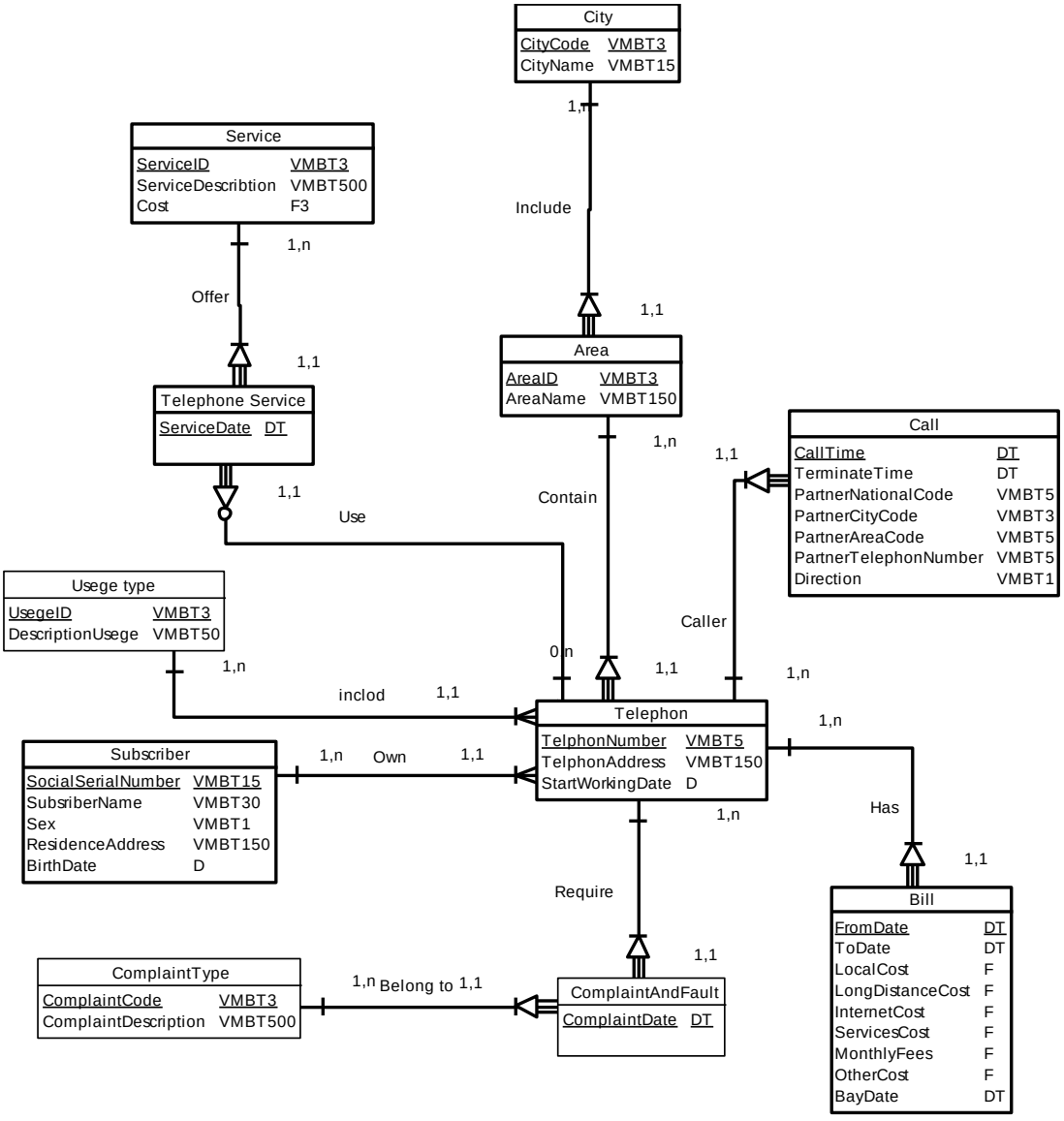

Fig. 1: Conceptual data model represents the needed relations to implement the proposed telecommunication carrier's network

In order to construct a relational data model for monitoring the telecommunication of central area, the design should consider the relevant information of the underlying area. The needed information is gathered from the customer, calls, complains,..etc. By analyzing the telecommunication data items and determining the corresponding function dependencies, several telecommunication carriers' entities can be obtained. These entities are submitted to an I-CASE tool to provide the required relational data model. Although other methods, instead of using I-CASE tool could be employed to accomplish the system relations (tables), the I-CASE approach is preferable because of its flexibility and its ability to generate easily and quickly modified versions of the data model [12][13]. These methods have enabled developers to deal with problems and their solutions at increasingly higher levels of abstraction in an increasingly higher-quality conceptual framework [12] [14] as shown in Fig. 1.

Moreover to make the database connectivity with WS independent of the database type, SQLXML WS is used for better connectivity over the network through the use of Active Server technologies [14][15][16]. The value of this model lies in its ability to meet the scalability requirements necessary to support several central area of telecommunication carrier's network.

Depending on the available database engine on the area server, the I-CASE tool determines the corresponding relational design. Once, the tables are created, they are filled with data of call gathering information system, customer, and customer complain. Actually, this procedure is not limited to any particular area server since it can be applied to any database engine as long as that server act as telecommunication area server.

III. Web Services

Web services are providing developers with suitable tools to provide an infrastructure for the development of scalable and interoperable systems.

Information systems are moving towards architectures that increase distribution, decoupling and collaboration with the aim to support the integration of autonomous and heterogeneous components. Autonomy and heterogeneity are based on so-called open environments.

In particular in the last years the attention has on Service-Oriented Architectures (SOA) that enable new kinds of flexible business applications of open environments in terms of their structure, productivity and administration, increasing the diffusion B2B (business two business) integration processes[1][8]. Fig. 2 shows service-oriented architecture. Service requester is the potential user of a service. Service provider is the entity that implements the service and offers to carry it out on behalf of the requester. Service registry is a place where all available services are listed. This Service registry allows providers to advertise their services and requesters to query for services [17] [18] [19].

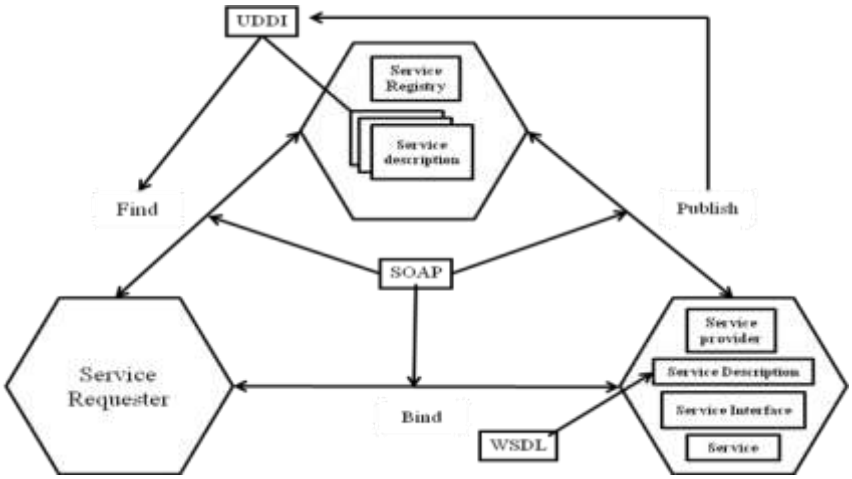

Fig. 2: Service-Oriented Architectures

3.1 Web Services Components

WS require several related XML-based technologies to transport and transform data into and out of programs and databases.

- XML (Extensible Markup Language), the foundation on which WS are built provides a base language for defining data, and processing it. XML represents a family of related specifications published and maintained by the World Wide Web Consortium (W3C).
- WSDL is used to provide an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. It is an XML-based technology, defines WS interfaces, data and message types, interaction patterns, and protocol mappings.

- SOAP, is a collection of XML-based technologies, defines an envelope for WS communication and provides a serialization format for transmitting XML documents over a network and a convention for representing RPC interactions [6][8][16]. A SOAP message represents the information needed to invoke a service or reflect the results of a service invocation, and contains the information specified in the service interface definition.
- UDDI, a WS registry and discovery mechanism, is used for storing and categorizing business information and for retrieving pointers to WS interfaces.

3.2 SQLXML Application

XML-based services provide interesting solutions to different business problems. It is no surprise that most modern DBMSs such as Oracle and SQL Server have extensive support for working with XML. DBMSs regularly need to work with and store data that may have originated in XML. Without this built-in support, getting XML to and from DBMS would require the application developer to translate XML data before sending it to DBMS and again after receiving it back [20][21][22]. It is clear this could quickly become very tedious given the pervasiveness of the language.

DBMS that support XML are called an XML-enabled DBMS. This means that it can read and write XML data. It can return data from databases in XML format, and it can read and update data stored in XML documents [20]. As Table (1) illustrates, out of the box, SQL Server's XML features can be broken down into eight general categories.

Table 1: SQL Server's XML Features [20]

Feature	Purpose
FOR XML	An extension to the SELECT command that allows result sets to be returned as XML
OPENXML	Allows reading and writing of data in XML documents
XPath queries	Allows SQL Server databases to be queried using XPath syntax
Schemas	Supports XSD and XDR mapping schemas and XPath queries against them
SOAP support	Allows clients to access DBMS functionality as a Web service
Updategrams	XML templates through which data modifications can be applied to a database
Managed classes	Classes that expose the functionality of SQLXML inside the .NET Framework
XML Bulk Load	A high-speed facility for loading XML data into a database server

WS technologies promise to deliver a loosely coupled, service-oriented architecture capable of integrating disparate software systems across proprietary platforms, networks and languages [23]. WS-based systems are not just technically interesting; they are the foundation for a new business paradigm. Businesses that evolve towards a loosely coupled nature can adapt to variant market conditions more quickly, allowing them to frequently add and change business partners with lower startup costs. Further, loosely coupled business interactions reduce dependence on proprietary protocols and methodologies, empowering businesses to reach across traditional client and partner boundaries into new markets.

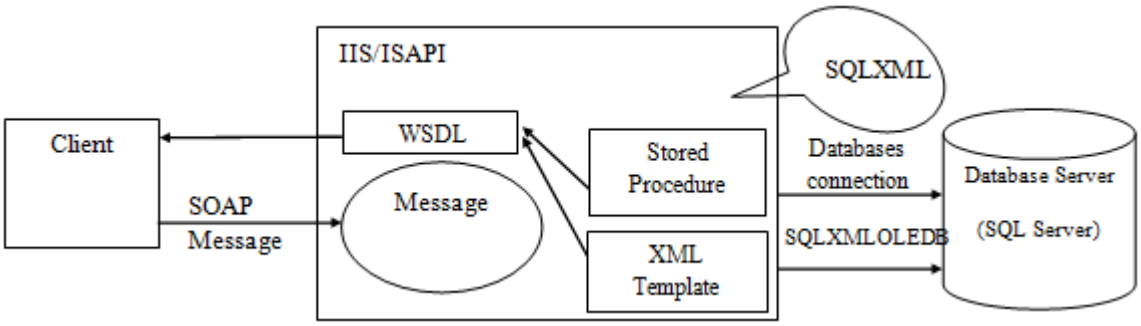

Fig. 3: Example of Microsoft SQLXML Web Services

One of the main loosely coupled business application is a SQLXML WS application using heterogeneous databases. SQLXML WS is used to simplify the interaction between SQL, multiple platforms, and various languages using different database connectivity such as Oledb, Active Data Objects (ADO), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), and so on [15]. SQLXML WS

help to better connect databases over an extranet, an intranet, or the Internet using Active Server technologies [15]. This enables multiple platforms to access the same services and attain the same results. This section contains a quick look at the functionality that SQL WS can bring to online database applications. One of the benefits of SQLXML WS is that they essentially cut out all use of ADO [15]. By using a WS to complete our data interaction we can remove a level of complexity and thus remove a layer of potential errors, for instance, ADO recordset object errors. Like all database applications, you have to provide the back-end before developing the front-end application and GUI. By using SQLXML WS, deployment time is reduced considerably and resources are increased [15]. SQLXML WS also opens the back-end to multiple front-ends that may be built by third parties, with no additional development work [15].

Most of organizations are now publishing a huge amount of data on the internet using XML for various purposes such as telecommunication carrier's business network for satisfying customer service needs. Due to users need, there is an increasing to query XML data. Relative to keyword search, a structured XML query allows a user to flexible way to present his/her queries needs. However, the structural XML query with large number of different XML data source makes it difficult to retrieve exact answer. Therefore, our future work will go to use relaxed query to the data sources.

IV. The System Model

As distributed software applications, WS let companies exchange data and share services with business units, customers, partners, and suppliers. WS facilitate machine-to-machine interactions for data collection, delivery, display, and processing. Here WS include an application that provides the telecommunication services that shares telephone information between customers and its billing. As a WS is something that has no user interface, we cannot interact with it without a special tool. That tool should allow composing XML requests via its own user interface. Commonly such user interfaces are text editors where we write our XML requests and controls for posting requests to the server. Therefore, this model has been designed and implemented as a relational database system as shown in Fig. 4, which is manipulated by a SQLXML WS application. The system interface is independent of the database type. One of the objectives of this work is to develop an understanding of SQLXML WS application for accessing heterogeneous and distributed database of telecommunication carrier's network. The proposed model allows the publishing of WS descriptions to submit user requests (goal) to retrieve the required information. It measures the efficiency of WS among possible different alternatives with real-time and execution parameters.

Here, telecommunication carriers utilize gateways to provide telephony services that have the capability to record call transactions on relational databases. In particular, existing web and distributed database (horizontal fragmentation) applications and services implementations have served in this work as shown in Fig. 4. Moreover, each area in telecommunication carrier's network may have certain database such as Oracle or SQL Server. In addition, the carrier utilizes the corporate network both for end-user as well as for the needs of data interchange with its thousands of national business partners. As a result, business partners as well as other associates have had the need to consume several kinds of WS that deliver various billing reports and exchange data concerning the Customer/call Detailed Records (CDR) as mentioned before.

Fig. 4 shows the four main players 1) WS provider, 2) WS requester, 3) Service registry, and 4) End user using thin client. They are working together according specific steps. Step 1 provider produces the WS and connects to distributed database. Step 2 provider deploys WS by its WSDL description, through WS registry (UDDI). Step 3 is called discovery process where the requester and provider, must become known to each other or at least one know the other. The discovery process somehow obtains both the WS description and an associated Functional Description (FD) of the service. FD is a machine-processable description of the functionality of the service that the provider is offering. Step 4 the service requester consumes selected services at his/her application. Requester adds appropriate WS to his/her application, (according to its contract with WS provider). Provider may change some parameters such connection string DB according to telephone location (end user of thin client) of telecommunication network and all changes are done in web.config file.

In the proposed model, requester can test WS to validate its functionality, behavior and performance using SoapUI testing tool. In addition, in this work the telecommunication carrier network consists of seven WS for customer service. These WS are: 1) Get subscriber telephones information, this WS like the guidebook telephone, the input of this service is the subscriber social serial number and the output is table contains all telephone in all centrals' telephone. This WS obtains the data from distributed and heterogeneous databases and retrieves it to appear to the customer transparently. 2) Telephone owner information, the input of this WS is city code and the telephone number and the output is the information about telephone owner. 3) New telephone service subscription, this WS allows customer to subscribe with available service. The inputs of this WS are city code, telephone number and service type while the output is just "OK" string if subscription done successfully or Null if subscription fail. 4) Customer complaint and telephone fault, this WS allows customer to take advice about specific fault or record complain about something. Input of this service is city code, telephone number and fault or complain type

while the output just “OK” string for successful submission of fault or complain. 5) Call detail record, in this WS the CDR record represents the footprint of a telecommunication carrier’s service. It includes customer call transaction, and communication information (caller, destination, duration, etc.). This WS allows customer to browse telephone traffic, at specific time interval period for both directions call and colleen. Inputs of this WS are city code, telephone number, time interval, and direction of call (i.e. customer calls another member or customer receives call from another member). Output of this WS is a table contains all telephone traffic at predefined time interval. 6) Telephone bill details, our objective from this WS is to allow associates and business partners to consume ‘‘online’’ updated information of a customers bill even for the unbilled CDR transactions not yet compiled. This WS allows service customer to browse telephone bills. Inputs of this WS are city code and telephone number while the output is a table contains all bill details. 7) Payment bills, WS payment bill provided to allow service customers to pay their telephone bills from any internet browser. In this WS, service consumer has been consuming WS to deal with banks. First, customer queries the telephone bills. Second, customer chooses which bill wants to bay, by its month. Third, customer chooses the bank would to pay through it (i.e. customers' bank) and enter his/her name and account number. All the above WS cooperate to observe telephone calls information such as billing and customers complains.

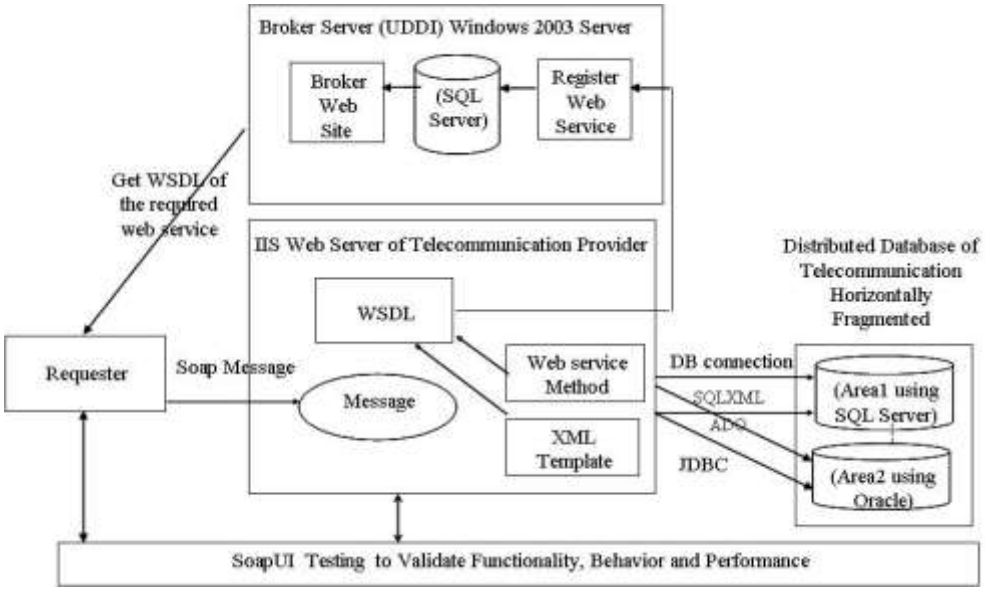

Fig. 4: The Proposed System Model

V. Implementation Details and Results

In this work, we will measure some significant parameter of WS. Also, we will validate its input and output, and assertion that it not broken under continually executing at specific period and implement several test strategy. It is very important to know WS behavior, before it has become available to end user (thin client).

In addition, we used SoapUI as a testing tool to validate functionality, analyze performance behavior under varying execute condition, find maximal performance available using thread strategies, define performance requirements and continuously validate using, maintain functional validations to see that they don't 'break'' under continually execution (repeat request). At first we construct project and then add the WSDL for WS we want to test it by its URL.

There are many advantages when we use test tools before publishing WS through web site: 1) If the WS contains too many methods with too many input and output parameters it may take several days to complete test of this WS, tools supports Groovy Script – a scripting language that is very similar to Java. 2) We can write scripts to access the database, which is used and / or influenced by the WS. 3) We can automate validation of the WS response against input. 4) We can automate the output of validation results into a file (which could be just a text file of HTML depending on our choice). 5) We can run an entire test suite.

5.1 SOA Testing

SOA tests are used for helping and realizing the full agility benefits of SOA with quality. Today's enterprises are leveraging SOA to provide greater flexibility and agility in meeting business needs [24]. This is enabled by exposing existing IT assets as reusable services and assembling them into composite solutions, with an appropriate governance infrastructure in place. However, SOA's distributed architecture

increases interdependencies and the underlying rate of change in the resulting application. This has exposed serious inadequacies with traditional software testing methods, requiring new approaches to maintain SOA application quality and agility in an environment of constant change.

Both web services (WS) and service-oriented architecture (SOA) presents a set of unique testing challenges. As services are distributed, it is necessary to test them using a distributed architecture [24]. Common SOA testing challenges as listed in www.itko.com/solutions/soatesting.jsp include:

- No visibility or traceability below the User Interface to isolate errors and problems
- Inability to test "headless" web services and components that do not have an available User Interface
- No ability to continuously validate application functionality, even as underlying components are being changed on their own lifecycles
- Inability to test composite solutions due to limited access or availability of dependent services and data needed for testing
- Inadequate or incomplete testing, resulting in costly problem identification and debugging once released into production
- Poor collaboration between development and quality assurance (QA), with minimal to no reuse of test assets between unit, functional, regression and performance testing

In this work, we use a testing framework called SoapUI, which allows different test cases to be selected based on built-in test strategies. The framework also has a windowing mechanism to evaluate and select test cases for both load testing and performance testing. Performance testing can help you identify bottlenecks in your application code and eliminate them proactively. After performance testing is completed, Load Testing can help identify inventory requirements and help you decide whether it is best for you to scale vertically (adding high-end servers) or horizontally (adding more web servers, databases etc).

5.2 Experimental Setup

The experimental setup and technological environment used for implementing and testing the proposed system model as the follows:

The Oracle Database 9i installed with operating system MS Windows XP, CPU 2 GHz and 2G RAM. The SQL Server 2005 installed with operating system MS Windows XP, CPU 2 GHz and 2G RAM. MS Visual Studio installed at Laptop with operating system MS Windows XP, CPU 2 GHz and 2 G RAM. The testing result has been obtained by standard testing software for validating WS performance, behavior and functionality called SoapUI version .2.0.2. We used IIS 5.1 as a web server. XML version 1.0, HTTP 1.1 and SOAP 1.2 are used as WS platform.

5.3 Measurement and Statistics

During execution, of WS the following measurement parameters in Table (2) are periodically collected and displayed in the statistical tables associated with web services. These parameters represent WS execution time, number of transferred bytes, Transaction per Second (TPS) ...etc as listed in Table (2) and rest of figures.

Table 2: Performance analysis of executing the proposed telecom WS

Execute WSs 1000 Count continuously					**Response (byte) or byte processed**	**Execution time(ms) of First Invocation**	**WS Provided by Egypt Telecom Provider**
K. byte per second	TPS	Avg. (ms)	Max (ms)	Min (ms)			
320~333.4	72~75	13	83	12	4524	15443	SubscriberTelephones -Information
206~211	93~104	9	68	9	2050	15141	TelephoneOwner
44~2169	212~103	12	35	8	440	15707	NewTelephoneService
43~215	108~120	8	77	8	402	10208	ComplaintTelephone
1293~197	607~92	9	31	9	2181	15174	CallDetailRecord
220~199	101~111	9	77	8	2015	15106	TelephoneBillDetails
0.889~0.922	2	388	704	365	350	15460	PaymentBills
47.5~48	120~123	8	40	7	404	15163	UpdatePaymentDate

In addition, minimum, maximum and average is execution time of web service depending. Each one depends on three main parameters: 1) complexity of web service internal functions, 2) web service response, and 3) available system resources

Fig. 5 shows WS performance (average execution time in ms) for different number of threads where each thread acts as virtual user with 500ms delay between each invocation.

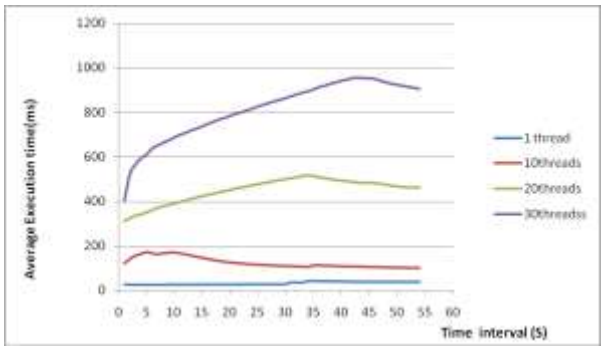

Fig. 5: Average execution time of WS with 500ms delay

Fig. 6 shows WS performance (average execution time in ms) for different number of threads immediately between each invocation.

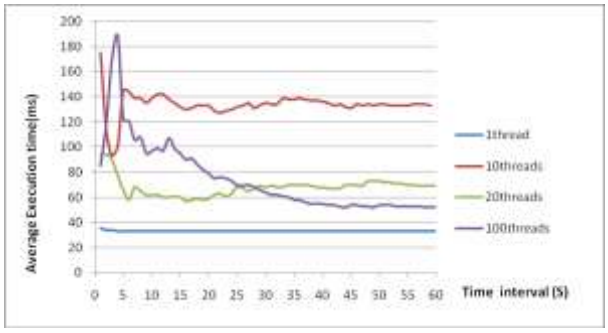

Fig. 6: Average execution time of WS immediately

Fig. 7 shows WS performance (count of executed WS) for different number of threads with 500ms delay between each invocation.

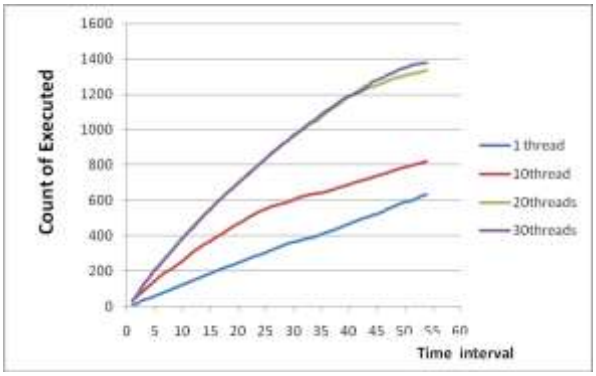

Fig. 7: Count of executed WS and number of threads

Fig. 8 shows WS performance (count of executed WS) for different number of threads immediately between each invocation.

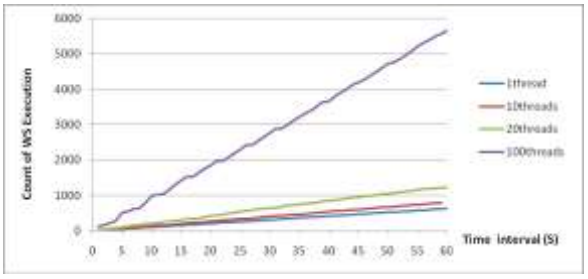

Fig. 8: Count of executed WS immediately

Fig. 9 shows WS performance (number of transactions per second) for different number of threads with 500ms delay between each invocation.

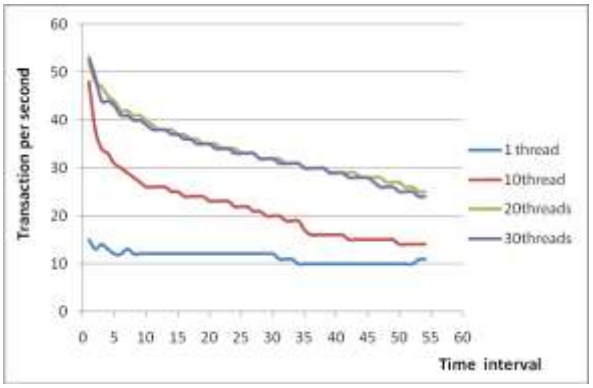

Fig. 9: Number of transactions per second of executed WS and number of threads

Fig. 10 shows WS performance (number of transactions per second) for different database operations such as insert, update and retrieve or select

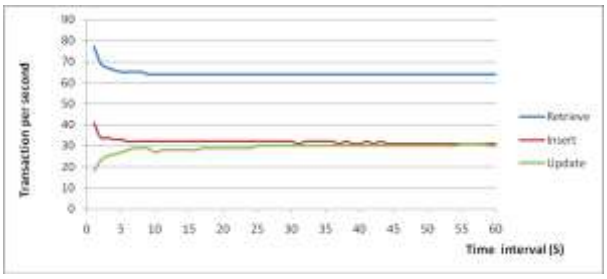

Fig. 10: Number of transactions per second of executed WS during database operations

Fig. 11 shows WS performance (average execution time in ms) for different database operations such as insert, update and retrieve

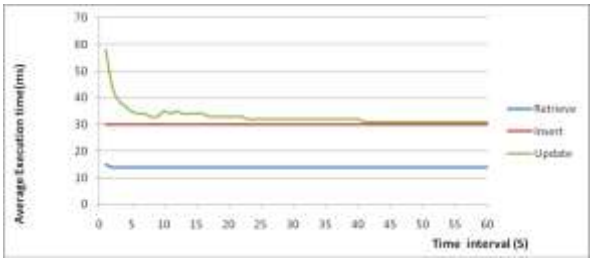

Fig. 11: Average execution time of WS during database operations

We analyzed the above results according to database type, database operation (transaction type), and number of threads at the same time where each thread acts as virtual user. In general execution time of web service is unpredictable (i.e. each invoke of web service mean different execution time), but averages execution time of web service can be used as a fingerprint of web service. At the beginning of invocating web service takes long time comparing with second or third web services invocation due to most of the time is consumed for network handshaking and host loading. Minimum, maximum and average execution time of web service is depending on three main parameters: 1) web service internal functions, 2) web service response time and 3) available resources.

In addition, type of database transaction (e.g. retrieve, insert, update...) affects the web service execution time and response time. The average execution time of the WS when it performs retrieve transaction is less than the average execution time of inserts and update transactions.

We find that, the WS average execution time depends on the number of threads and the average executing time of WS unpredictable when several threads immediately between these threads invoke WS. In addition, we notice the hardware limitations are common problem when we implement huge number of threads. Finally, recommending suitable hardware is an important task before publishing web services.

VI. Conclusion

This work shows the benefit of WS as an interface of the database and manipulating horizontal fragmentation transparently. Also, appear WS as a global middleware capable to deal with heterogeneous databases.

The architecture was tested in a multiplatform environment within a network and was proven to operate easily by employing SOAP and Web services. We have introduced WS application for retrieval information system over distributed and horizontal fragmentations. In addition to, the introduced application retrieved the telecommunication information from different heterogeneous databases (different databases platform such as Oracle, SQL Server, My SQL…etc). Also, we use testing tools for testing and validate a WS performance under different executing strategy, validations WS functionality to see that they do not 'break" under continuously executing, run several executing tests simultaneously to see how they affect each other. Using WS testing tools is very important to facility using such this WS when we purchase it from remotely vendor.

Testing is an essential part of the software development process, and an important area that is often misunderstood or overlooked is that of stress testing. Coverage of web service is benefit for analysis WSDL/interface functionality, particularly when web service contains several web methods (functions). By following the testing principles, we implement effective testing that aim to find some of the more devious problems associated with our WS development. Pre-written tool "SoapUI" is used, for testing. We used SoapUI for testing and validating a WS performance under different executing strategy and validating WS functionality to see that they do not 'break" under continuously executing.

References

[1] Laura Bocchi and Paolo Ciancarini, “On the Impact of Formal Methods in the SOA”, Electronic Notes in Theoretical Computer Science, 160 (2006): 113–126, 2006

[2] Christos Makris, Yannis Panagis, Evangelos Sakkopoulos and Athanasios Tsakalidis, “Efficient and adaptive discovery techniques of Web Services handling large data sets”, The Journal of Systems and Software, 79 (2006): 480–495, 2006

[3] Ouzzani, M., Bouguettaya, A., “Efficient access to Web Services”, IEEE Internet Computing. 8 (2): 34–44, 2004

[4] Makris, C., Sakkopoulos, E., Sioutas, S., Triantafillou, P., Tsakalidis, A., Vassiliadis, B., 2005. “Nippers: Network of interpolated peers for Web Service Discovery”. In: 2005 IEEE International Conference on Information Technology: Coding and Computing (ITCC_05), vol. II, Las Vegas, Nevada. pp. 193–198.

[5] Yu, T., and Lin, K.-J., “A broker-based framework for qos-aware Web Service composition”, In: 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services, 29 March–1 April 2005, Hong Kong, China. pp. 22–29.

[6] Gerhard Smiatek, “SOAP-based web services in GIS/RDBMS environment”, Environmental Modelling & Software, 20 (2005): 775-782, 2005

[7] W3C, 2003, “SOAP W3C Recommendation Documents”, http://www.w3.org/TR/SOAP

[8] XTRADYNE, “Protecting Web Services with the XML/SOAP Security Gateway”, XTRADYNE White Paper: 2004-2007 PrismTech, www.xtradyne.com

[9] W3C, 2004, “Web Service Description Language”, http://www.w3.org/TR/wsdl.

[10] W3C, 2004, “Web Services Architecture”. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211.

[11] UDDI, 2004, “UDDI Version 3.0.2”, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.

[12] M. Zaki and Tarek S. Sobh, "NCDS: Data Mining for Discovering Interesting Network Characteristics", Journal of Information and Software Technology (JIST), Volume 47, Issue 3, March 2005, PP. 189-198.

[13] Tarek S. Sobh, "Explanation-based Learning to Recognize Network Malfunctions", Information, Knowledge, System Management (IKSM), Volume 5, Issue 1, 2005/2006, PP. 1-21

[14] Boris Motik, Ian Horrocks and Ulrike Sattler, "Bridging the gap between OWL and relational databases", Web Semantics: Science, Services and Agents on the World Wide Web, 7 (2009): 74–89, 2009

[15] D. Jorgensen, "Building a SQLXML WS Application", Developing .Net WS with XML, 2002, pp 299-336, Elsevier Inc.

[16] S. Sioutas, E. Sakkopoulos, Ch. Makris, B. Vassiliadis, A. Tsakalidis and P. Triantafillou, "Dynamic Web Service discovery architecture based on a novel peer based overlay network", The Journal of Systems and Software, 82 (2009): 809–824, 2009

[17] Marco Crasso, Alejandro Zunino, and Marcelo Campo, "Easy web service discovery: A query-by-example approach", Science of Computer Programming, 71 (2008): 144–164, 2008

[18] Maria Cavalcanti, Rafael Targino, Fernanda Baia˜o, Shaila Rossle, Paulo Bisch, Paulo Pires, Maria Campos and Marta Mattoso, "Managing structural genomic workflows using Web services", Data & Knowledge Engineering, 53 (2005): 45–74, 2005

[19] Yih-Ling Hedley, Muhammad Younas, Anne James and Mark Sanderson, "Sampling, information extraction and summarization of Hidden Web databases", Data & Knowledge Engineering, 59 (2006): 213–230, 2006

[20] H. W. Kenton SQLXML, Chapter 18, Henderson_book.fm, pp 675-790 Thursday, September 25, 2003 5:23 AM

[21] Microsoft, 2004, "Performance monitoring, browsing counters". http://msdn.microsoft.com/ library/en-us/perfmon/base/getting_counter_ information.asp

[22] Microsoft, 2005, "C Sharp Programming Language Specification". http://msdn.microsoft.com/library/ en-us/csspec/html/CSharpSpecStart.asp.

[23] Vikas Agarwal, Girish Chafle, Koustuv Dasgupta, Neeran Karnik, Arun Kumar, Sumit Mittal and Biplav Srivastava, "Synthy: A system for end to end composition of web services", Web Semantics: Science, Services and Agents on the World Wide Web, 3 (2005): 311–339, 2005

[24] Xiaoying Bai; Yinong Chen; Zhongkui Shao, "Adaptive Web Services Testing", Proceeding of Computer Software and Applications Adaptive (COMPSAC), 24-27 July 2007 Page(s):233 – 236, 2007

Authors' Profiles

Tarek Salah Sobh received his B.Sc. degree in computer engineering from Military Technical College, Cairo, Egypt in 1987. Both M.Sc. and Ph.D. degrees from Computer and System Engineering Department, Faculty of Engineering, Al-Azhar University, Cairo, Egypt. He has managed, designed and developed several package for business applications and security systems. He has authored/co-authored of many refereed journal/conference papers and booklet. Some of the articles are available in the ScienceDirect Top 25 hottest articles. His research of interest includes computer networks, security systems, distributed systems, knowledge discovery, data mining, and software engineering**.**

Late professor **Medhat Fakhry** received his B.Sc. degree in computer engineering from Military Technical College, Cairo, Egypt. He worked as Deputy Director of Information Systems Department, Egyptian Armed Forces. In addition, he worked as a teaching staff in Military Technical College. He finished his career as Chairman of Management Information System Department AAST, Egypt. My professor **Medhat Fakhry** died to the mercy of God in beginning of 2012.

How to cite this paper: Tarek S. Sobh, Medhat Fakhry,"Evaluating Web Services Functionality and Performance", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.5, pp.18-27, 2014. DOI: 10.5815/ijitcs.2014.05.03