Towards a Methodology for Performance Measurement of Service-Based Systems

Marcelo Silva, Fernando Lins, Erica Sousa

Abstract— Currently, the use of web services in systems engineering and execution is increasing considerably. In addition, service oriented computing (SOC) is being viewed as an interesting approach for building more complex systems based on primitive services. Based on the widely adoption and use of this approach, web services performance evaluation acquires a considerable relevance, because this performance directly impact the performance of the entire system. This work proposes a methodology for measuring the performance of Web services, in which a set of steps can be used to accomplish this task. In special, considering that two relevant communication protocols, SOAP and REST, were proposed in the last years to support web services project and execution, a case study focuses on a comparative study of these protocols in order to evaluate performance aspects in real world scenarios.

Index Terms— Computer network, Distributed Systems, Internet, Methodology for Performance evaluation, REST, SOAP, web services.

1 Introduction

In recent years, one interesting topic that has gained interest is Service Oriented Computing (SOC) [8]. This paradigm involves, in general, the use of Web services for the implementation of business capabilities. Service-oriented systems, in theory, promote reuse and interoperability, and they are developed using widely disseminated and accepted standards (mostly based on the Internet HTTP protocol), thus facilitating its adoption by the scientific community. In this context, two communication protocols have been used to invoke Web Services: SOAP (Simple Object Access Protocol) [9] and REST (Representational State Transfer) [5]. The SOAP protocol is considered more established and is supported by relevant companies (e.g., IBM, Microsoft and others), while the REST protocol experienced a considered growth in recent years.

A crucial issue in the development of services-based applications is the choice of the protocol to be used. Both the SOAP and REST can (and has been used) for this purpose. However, they differ considerably with respect to I) necessary tooling/mechanisms for their execution II) execution time and III) support to non-functional requirements (such as security, performance and cost). More specifically, some current related work emphasize the performance impact that can appear depending of the adopted communication protocol [2], [5], [6], [7]. In the current scenario, depending of the system type, there is always a question regarding which web service configuration should be used.

Currently there is a lack of an approach that describes how to conduct a web service performance measurement study, focusing on each step necessary to plan, execute and analyze this study. Considering this context, this article proposes a methodology for measurement of Web services, and this methodology adresses from the objectives of the measuremente study from its execution and evaluation. One interesting application of this methodology is the perforamnce evaluation of Web services communication protocols (more specifically, REST and SOAP). This evaluation is important because it present relevant results that helps to understand which scenarios are more favorable to use the SOAP protocol and what are other scenarios in which the use of the REST protocol is most appropriate.

The structure of this paper is presented as follows. Section II describes theoretical aspects of the SOAP and REST communication protocols. A methodology for performance measurement of Web services is carried out in Section III. In Section IV, a case study is used to showcase the proposed methodology and to perform a comparative performance evaluation of the communication protocols SOAP and REST. Section V describes related works. Finally, Section VI presents the conclusions and some future work.

2 Basic Concepts

This section introduces basic concepts that are relevant for the understanding of this work.

2.1 Web Services

According to W3C [9], Web services provide a standard mean of interoperability between different software applications, running on a variety of platforms. Web services are characterized by their considerable level of interoperability and extensibility, as well as its processing performed between machines (applications) due to descriptions using representation standards, such as XML. Web services can be combined flexibly in order to build more complex applications, and allows systems that provide simple services can interact with each other in order to provide sophisticated services. Another interesting

- *Marcelo José Santos da Silva is currently student of masters degree program in Applied Informatics in DEINFO - Department of Statistics and Informatics of University Federal Rural de Pernambuco, Pernambuco. E-mail: celosop@gmail.com.*
- *Fernando A. Aires Lins and Erica T. G. Sousa are professors in the Department of Statistics and Informatics at Federal Rural University of Pernambuco, Brazil, PH. E-mail: fernando.aires@deinfo.ufrpe.br and erica.sousa@deinfo.ufrpe.br.*

definition is presented in Benharref [3], which defines Web services as an application that exposes its functionality through a description of a standardized interface that makes it available for use by other programs (clients).

2.2 Web Services Comunication Protocols

A considerable number of protocols have been proposed in last years to support functional and non-functional aspects of Web services. The first generation of these protocols [4] are composed by SOAP, WSDL and UDDI. In short, SOAP was used as the communication protocol (i.e., provides the messages format for the communication between client and service), WSDL was used to describe the services interface (i.e., what the service can actually do and which are the necessary parameters to invoke it) and UDDI was intended to be a standard format/facility for service registry.

The first generation protocols were largely used in the last decade. However, they were viewed as insufficient for more "complex" applications (i.e., enterprise systems which several functional and non-functional requirements). Based on that, a set of protocols were proposed and were referenced as second generation web service protocols [4] (WS-Policy, WS-Security, WS-Transactions and so on). In addition, during this period, other communication protocol has gained interest by the industry: REST [5]. Competing with SOAP, REST is viewed as a lightweight protocol in which services can be invoked requiring a lighter infrastructure. The REST functions are available for the four most common HTTP commands: GET, POST, PUT and DELETE. Based on the fact that all second generation protocols (WS-*) are based on SOAP, this new approach imposed several changes in the service oriented computing scenario, because most of the developed protocols and applications were not initially designed to be executed with REST.

One central point in the development on the development of service oriented systems is the choosing of the communication protocol. This choice impacts not only in the performance of the service invocation, but also in the system development and necessary support tools. Based on that, web service communications protocols (SOAP and REST) plays a very important role in service systems. Based on this fact, they are detailed on the next subsections.

2.2.1 SOAP

SOAP [9] is a web service protocol that focus on information exchange in a decentralized, distributed environment. This protocol is an XML based standard that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols [9].
Figure 1 illustrates how a request message is composed by the HTTP headers and the SOAP Envelope. The name of the service being created is identified by HTTP header, whereas a whole SOAP Envelope of 490 bytes is used to identify the type of the service.

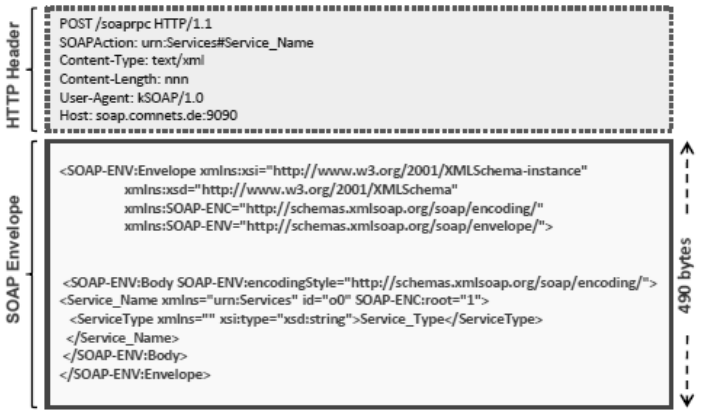

Figure 1. SOAP Service Creation Request.

2.2.2 REST

The Representational State Transfer (REST) is considered a simple approach for building Web services fast, using a set of well-defined operations that are commonly used to access information resources. The HTTP itself defines a small set of operations, the most important are POST, GET, PUT and DELETE. Currently, the main REST reference is [5], which has been used for diverse others authors and developers to reason and build REST-based systems. The REST architectural style is an abstraction of the architectural elements within a distributed hypermedia system. According to Fielding [5], the architectural elements encompasses the fundamental constraints upon components, connectors, and data that define the basis of the Web architecture, and thus the essence of its behavior as a network-based application. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components and their interpretation of significant data elements. It encompasses the fundamental constraints upon components, connectors, and data that define the basis of the Web architecture, and thus the essence of its behavior as a network-based application.

A REST messaging example in presented on the Figure 2, in which the request message is specified in the HTTP header, and both the name and type of service are identified by the URLstring.

Figure 2. REST Service Creation Request.

3 Web Service Performance Measurement Metodology

This section describes a methodology for performance measurement in web service-based applications. The need for a

methodology for this purpose is based on the fact that a considerable number of applications are using Web services currently, and to reason about performance can help the business analysts to choose with protocol and hardware configurations should be used considering a specific workload. The methodology is composed by activities, and these activities are presented on Figure 3, which aims to support the Web services performance evaluation though measurement. This methodology was inspired in a seminal work of this area, [10]. Which was adapted to the Web services context. This methodology is illustrated on Figure 3.

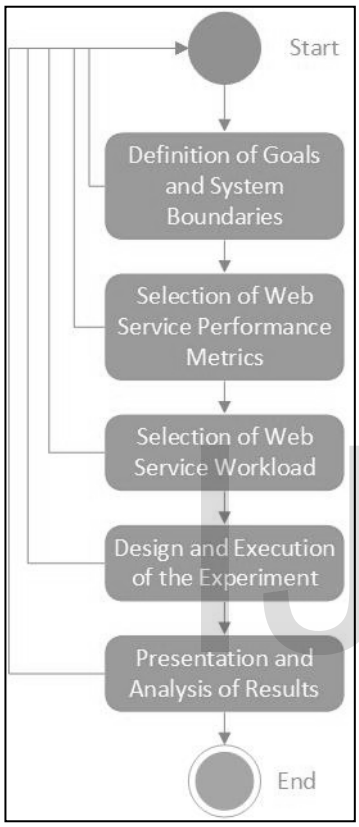

Figure 3. Activities for Web Services Performance Evaluation.

3.1 Definition of Goals and System Boundaries

This activity is responsible for stating the goals of the study and to define what constitutes the system by delineating system boundaries. All this has to be clearly defined at the beginning of the experiment. Given the same set of hardware and software, the definition of the system may vary depending upon the goals of the study. The choice of system boundaries affects the performance metrics as well as workloads used to compare the systems.

For example, one has to be set if the Web service operation system server will be Windows or Linux. It is also important to know if there will be some kind of cluster, or if you have some kind of data mirroring with the raid for example.

3.2 Selection of Web Service Performance Metrics

The next step is to select criteria to compare the services performance. These criteria are named metrics. In general, metrics are related to the speed, accuracy and availability of services. The goal of this services performance measurement study is either to compare different alternatives or to find the optimal parameter value.

Three metrics are generally adopted in service-based systens performance evaluation, and are detailed as follows.

- **Response Time -** This metric is defined as the interval between an user's request and the system response [12]. To illustrate this metric, Figure 4 is presented. In step 1, the user performs the initial request. After that, in the step 2, the user's application sends the request to the specified web service. Then, in the step 3, the Web Server receives and processes the user request and sends the response to the user's application (step 4). Finally, in the step 5, the user receives the required result.

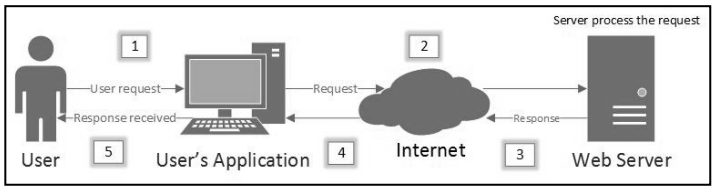

Figure 4. Step by step of user request of web server.

- **Processor Utilization -** Processor Utilization is the percentage of time during which the server is busy processing jobs. To measure this fraction of time that the resource is busy, it is possible to use a Windows tool called Performance Monitor [11]. This tool helps to compute the ratio of the request arrival rate and the service rate. For a stable system, the value of this ratio must be less than 1 (100 %), because the request arrival rate should not exceed the service rate.
- **Throughput –** Is defined as the rate (requests per unit of time) at which the requests can be serviced by the system. The following formula was used for calculating throughput expressed as clients per second:

$$\textbf{Throughput} = \frac{\textbf{Number of Clients (Requests)}}{\textbf{Response Time (ms)}}$$

For networks, the throughtput is meansured in packets per second or bits per second. For computer networks, the nominal capacity is called the bandwidth and is usually expressed in bits per second. In our case (Web Service), it was-used Number of Requests divided by the time in milliseconds.

3.3 Selection of the Web Service Workload

The workload consists of a list of service requests to the system. For example, the Web services comparison may consist of a set of requests. Depending upon the evaluation technique chosen, the workload may be expressed in different forms. For

analytical modeling, the workload is usually expressed as a probability of various requests. For simulation, one could use a trace of requests measured on a real system. For measurement, the workload generally consists of user scripts to be executed on the systems that represents a specific amount of request per second. In all cases, it is essential that the workload be representative of the system usage in real world scenarios.

3.4 Design and Execution of the Experiment

An important factor to be focused is how the experiment should be performed, considering both the client side that sent the request and the server that processes it. On the server side it is necessary to make measurements on its internal processes in order to obtain the needed results. In the case of metric response time and processor utilization, we have to isolate the process and leave you with the real-time type of priority so that the result really is as integrity as possible.Another relevant point in this activity is the design of the experiment itself. Before executing the required mensurements, it is important to define, for example, how many iterations should be executed for obtaining a valid statistical result and to define the size of the payload that should be sent in the network.

This is undoubtedly the main phase of the work. One should choose the right tools for each type of task that will be developed. First, in the case of choice of programming language to be used in the development of Web service. Each application can have its own "language", which is translated into a universal language, an intermediate format such as XML, JSON, CSV, etc. Soon after the choice of language, you must decide which Web Server, can be Apache, Glassfish [17], Tomcat, etc.

After the installed environment, it can be to chosen a framework for REST and SOAP. It has several options frameworks, including the axis, JAX-RS [15], JAX-WS [14].

As for taking a measurement of HTTP requests, one can use some form of software that servers as a stress test JMeter [12], SoapUi [16], etc.

Some tools can be used for the measurement of requests from potential customers for a particular web service. All these tools can be used to capture the time it requests lead. Besides being a well used and referenced by the academic community software, because it is an Open Source software.

3.5 Presentation and Analysis of Results

The final step of a Web service performance study is collect, analyze and communicate the results to the interested users. Graphic charts such as line charts, bar charts, pie charts, and histograms are commonly used in performance results. There are a number of reasons why a graphic chart may be used for data presentation in place of a textual explanation. A graphic chart is also a good way to emphasize or clarify a point, to reinforce a conclusion, and to summarize the results of a study.

Considering experiments focused on web services, it is important to remember the tree relevant metrics that were already cited in this work: Response Time, Processor Utilization and Throughput. Most statisticas analysis and evaluation can be based on both metrics, and support the business user in the decision making process.

For the analysis of the results has been a lot of possibilities to be highlighted. Depending on the line of analysis that is chosen for the project and the data were collected. Regarding the response time metric can be used an average of time each request made going back to the server and the user.

When several requests and different amounts of load has, can be some outliers, which is nothing more than an aberrant or outlier value, is an observation that has a substantial departure from the others in the series, or is inconsistent. In this case it is interesting to take them out of the "normal" samples to the result to be as close to a real case of stress on the server.

Such data analysis can also be used in simulation. After the created simulation model, you can enter the time for each stage of the model and test.

4 Case Study and Results

Based on the methodology defined in the Section 3, a Web services performance study is carried out in this section.

To illustrate in practice what was proposed in this paper, a case study was developed exactly following the acitivities of the methodology. In this case the server side was set up on a desktop machine a Web server containing 2 Web Services, SOAP and REST. In the client used a notebook to the load test.

4.1 Definition of Goals and System Boundaries

Firstly, we define the goals and system boundaries of the study. The goal of this case study is a performance comparative study of the Web services communication protocols SOAP and REST in order to compare their results. The System Boundaries consists of the Figure 4, when the requests are sent via the channel from the client computer to the Web Server.

4.2 Selection of Web Service Performance Metrics

The study was limited to operations that actually occurred and were confirmed, requests that have errors were not considered. For each service, the rate at which the service can be performed, the time taken for the service, and the resources consumed was be compared. The resources are the local computer (client), the remote computer (Web Server), and the time of the process.

In order to measure web services response time, there are several tools to perform this operation, such a JMeter, SOAPUI.

Considering the general objectives of this work, JMeter was adopted because it supports both REST as SOAP requests. JMeter measures the Response Time from just before sending the request to just after the first response has been received. Thus the time includes all the processing needed to assemble the request as well as assembling the first part of the response, which in general will be longer than one byte. Protocol analysers (such as Wireshark [13]) measure the time when bytes are

being sent/received over the interface. Jmeter was also used to calculate the throughput.

To capture the data related to CPU utilization at the server side, we used Windows Performance Monitor.

4.3 Selection of the Workload

The study of this work was in the form of measurement. Several tests were massively developed with specific load testing tools. In these tests requests were made with 490 bytes each. In this case, each request, it would be a different user, making a request in the web service.

After the definition of metrics, the workload parameters were chosen:

a) Type of Protocols - In this case the work was to use the two most used Web Services, REST and SOAP;

b) Time between successive calls - The time of this case study were all requests sent to the server at the same time;

c) Number and sizes of the call parameters - The interval between our minimum and maximum requests were respectively 100 and 600 requirements. The size of the samples was each 490 bytes.

The key factor chosen for this study was Number n of consecutive calls and the comparison of the Web service communication protocols SOAP and REST.

4.4 Design and Execution of the Experiment

Firstly, it is important to register the conditions in which this study was executed. On the server side, the configuration is as follows: Intel® Core ™ 2 Duo CPU @ 2.33GHz 2.33GHz, Memory (RAM) 5GB, Windows Operating System 7 Ultimate 64bit, Hard Drive 128GB, Internet 10MBps. The two Web Services (REST and SOAP) were installed in the same server mentioned above. The tests were performed by isolating each of the services in order to not disturb the experiment results.

The programming language used was the Java programming language. To execute the experiment, the application server Open-source Glassfish was installed. In the study, we used the version of Netbeans IDE 8.0.2 without Glassfish. After the environment instalation, the Java API for XML Web Services (JAX-WS) for SOAP and Java API for RESTful Web Services (JAX-RS) for REST were installed.

The POST verb is most-often utilized for creation of new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI). POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests. The HTTP GET method is used to retrieve (or read) a representation of a resource.

In JAX-WS, a web service operation invocation is represented by an XML-based protocol, such as SOAP. The SOAP specification defines the envelope structure, encoding rules, and conventions for representing web service invocations and responses. These calls and responses are transmitted as SOAP messages (XML files) over HTTP. Although SOAP messages are complex, the JAX-WS API hides this complexity from the application developer. On the server side, the developer specifies the web service operations by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Soon after the creation it was made the configuration of Web Service with Glassfish. After that the server was already online and available to any user / application. With REST was also made a creation of a Java project, but we used the JAX-RS, which is specific to REST. Have the user side, to test whether the Web Services were working properly, a request directly by the web browser with the ip of the server with the correct address of each web service was made.

The case study was executed and some relevant results were obtained. In Figure 5, the experiment was executed by observing the % CPU usage according to the number of requests that were arriving. A comparison was made between the Web Services SOAP and REST, with the number of requests 100, 200, 300, 400, 500 and 600 respectively.

4.5 Presentation and Analysis of Results

In the first experiment, the Figure 5 presents the average of CPU usage considering the time that the service requests were executed. To measure CPU usage, it was used to the Windows Performance Monitor which gives complete information of all processes, memory ram usage, disk usage etc. While the requests were sent, the process was analyzed and verified in accordance with Figure 5. It is important to note in the Figure 5 there is always a difference between the CPU usages. The SOAP header is higher compared to the REST consequently it sends more data to the server process. Based on this fact, SOAP takes longer to process on the server side.

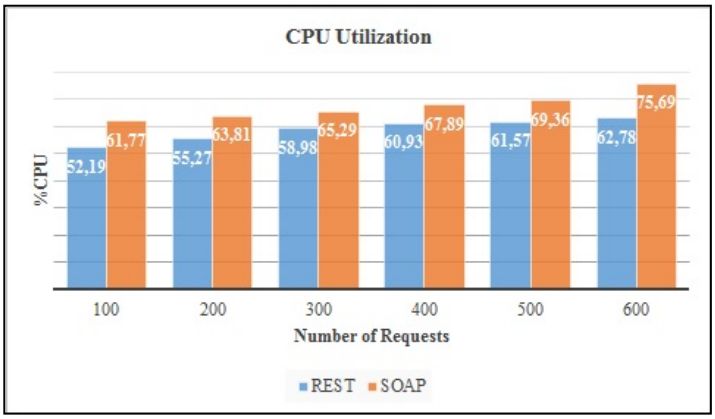

Figure 5. CPU vs Number of Requests.

Figure 6 shows a comparison of the number of requests compared to the response time. It is important to note that once again SOAP shows a visible performance disadvantage than REST. It is possible to choose any of the HTTP shipping methods, POST or GET. In the case of Figure 6, it was used the

same script that returned the same value in both REST and in SOAP. These results show that there is a difference between the two Web Services, and it apparently increases at the same rate in this experiment. In this case, the Web server to send the response of the same script, returned to the requesting a message of 314 and 279 in SOAP to REST respectively.

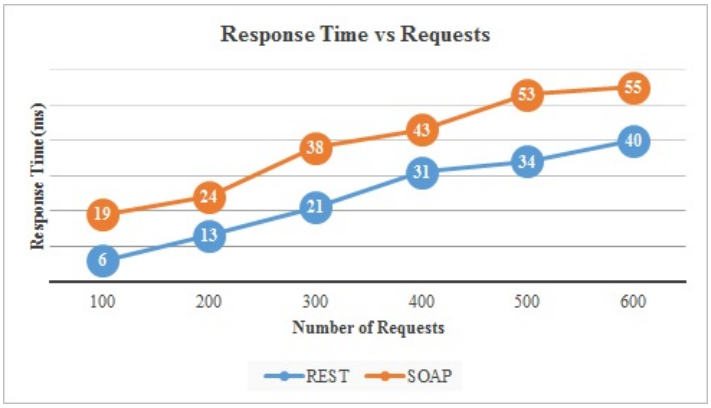

Figure 6. Time Response vs Number of Requests.

The throughput was measured according to the formula mentioned in section 3.2. It can be observed in the Figure 7 that as the number of simultaneous service requests increased, REST has a higher throughput than SOAP.

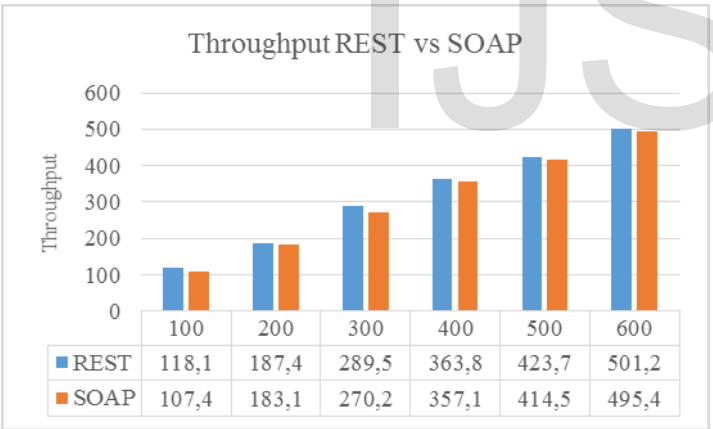

Figure 7. Throughput vs Number of Requests.

When looking at the metric of Processor Utilization, CPU usage in all tests was higher in SOAP. In the case of requests regarding the Response Time, SOAP was also slightly higher, and this is considered a disadvantage in the performance evaluation.

5 Related Work

The main focus of this work was the proposal of a methodology for Web Services performance evaluation. This methodology details all the steps to conduct a measurement evaluation in this context. In addition, we compared the performance of the two most used communication protocols, REST and SOAP, in order to showcase the methodology. It is important to highlight that there are other existing approaches that focuses of Web services performance. Gary Clinks and Philip Malley [18] evaluated the performance of SOAP and RESTful Web Services for implementing Service and Resource Oriented Architectures. The main contribution of Gary Clinks and Philip Malley is the overview of both of the Web Service styles and their associated architectures, as well as presenting experimental results in which to demonstrate the performance of each style, over a number of test scenarios. It can be stated that this work and Gary's work have a common goal, but use different points of view to produce conclusions; they are complementary, and can be used together to provide a more complete evaluation of existing Web Services.

Another study that also focused on this subject is Pavan Kumar Potti [19]. In this initiative, the author compares the performance between two Web Services SOAP and REST. At work of Pavan, they make changes to client requests wired and wireless, also do some tests to upload files and compares all results both REST and SOAP. But the author does not explain in details characteristics about the adopted environment or how the experiments were carried out. In other words, this initiative lacks a general approach to better present to the reader how to replicate the experiments in other contexts and environments.

6 Conclusions and Future Work

The main focus of this work was to create a methodology focused on Web Services performance through measurement. This work also made a comparative analysis of two communication protocols most commonly used in Web Services. An extensive comparison of the process of creating services in both mechanisms have been studied. Web services performance is generally evaluated based on parameters such as server response time and throughput. The study shows that the performace of REST is better than SOAP calculated for each metric, both in response time and in throughput. As future work, we intend to develop a mathematical representation to in order to also use simulation to evaluate service-based systems. In this case, formalisms such as Stochastic Petri Net [21] or Markov chain model [20] appears with considerable relevance. Using these mathematical representations and simulation, it is possible to increase the number of requests, since there is only the need to calculate the simulation result, and it is not required to execute all of these requests.

References

[1] AlShahwan, F. Moessner, K. and Carrez, F. "Distributing resource intensive mobile web services," International Conference on Innovations in Information Technology, 2011 ,pp. 41-46

[2] Aijaz, Fahad ; Shaikh, Shuja Jamil ; Walke, Bernhard, "Performance Analysis of a Framework for Multi-Interfaced Service Level Agreements on Mobile Devices," 11th European Wireless Conference 2011 - Sustainable Wireless Technologies, pp. 1-8. Intalio. Intalio

BPMS Designer. Available at http://www.intalio.com/bpms (last visit: 30rd December 2014)

[3] BENHARREF, A.; SERHANI, M. A.; BOUKTIF, "S. Online Monitoring for Sustainable Communities of Web Services," Poster Session. 12th IFIP/IEEE, 2011

[4] Erl, T. "Service-oriented Architecture: Concepts, Technologies and Design," Prentice Hall, 2005

[5] Fielding, R.T. "Architectural styles and the design of networked-based software architectures. PhD thesis. Department of Information and Computer Science, University of California, Irvine.Cardoso, J. Business process control-flow complexity: metric, evaluation and validation," 2000, International Journal of Web Services Research, v. 5, pp. 49-76. 2008

[6] Li, G and Sun, H. "RESTful Dynamic Framework for Services in Mobile Wireless Networks. International Conference on E-Business and Information System Security (EBISS'09),", pp. 1-5.Liu, H. *et al*. Comparison between Collaborative Business Process tools. In 2011 Fifth International Conference on Research Challenges in Information Science (RCIS), pp. 1-6. 2011

[7] Mohamed, K.E. George Mason Univ., Fairfax, VA, USA ; Wijesekera, D. "A Lightweight Framework for Web Services Implementations on Mobile Devices. IEEE International Conference on Mobile Services (MS),"2012, pp. 64-71

[8] Papazoglou, M. and Heuvel, W. "Service oriented architectures: approaches, technologies and research issues," In VLDB Journal, v. 16, 389- 415. 2007

[9] W3C. "SOAP v. 1.2," 2007, Available at http://www.w3.org/TR/2007/REC-soap12-part000704 27/ (last visit: 28th February 2015).

[10] Raj Jain, "The Art of Computer Systems Performance Analysis: techniques forexperimental design, measurement, simulation and modeling," John Wiley,1991, ISBN: 0-471-50336-3

[11] Enbody, R, "Perfmon: Performance Monitoring Tool," Michigan State University, 1999, http://www. cps. msu. edu/enbody/perfmon. html (last visit: 28th February 2015)

[12] Halili, Emily H, "Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites,", 2008

[13] Gerald Combs, "Wireshark," Available at https://www.wireshark.org/ (last visit: 22rd March 2015).

[14] Chinnici , Sun Microsystems. "The java api for xml-based web services (JAX-WS) 2.0," 2006

[15] Burke, Bill " O'Reilly Media, Inc. RESTful Java with JaX-RS," 2009

[16] Ole Lensmar, Smartbear, "SoapUi," 2005, Available at http://www.soapui.org/about-soapui/release-history.html (last visit 31rd March 2015).

[17] Heffelfinger, David R, "Java EE 6 with GlassFish 3 application server," 2010

[18] P Markey, G Clynch, "A performance analysis of WS-*(SOAP) and RESTful Web Services for Implementing Service and Resource Orientated Architectures," 2013, Available at http://arrow.dit.ie/ittscicon/1/ (last visit 31rd March 2015).

[19] PK Potti, S Ahuja, K Umapathy, "Comparing Performance of Web Service Interaction Styles: SOAP vs. REST," 2012, Available at http://scholar.google.com/scholar?cites=14866246770289479857&as_sdt=2005&sciodt=0,5&hl=en&num=20 (last visit 31rd March 2015).

[20] WR Gilks, "Markov chain monte carlo," 2005, Book

[21] M Ajmone Marsan, G Conte, G Balbo,"A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," 1984, Available at http://dl.acm.org/citation.cfm?id=191 (last visit 31rd March 2015).