Using Semantic Web for Selection of Web Services with QoS

Luis H. V. Nakamura
Institute of Mathematics and Computer Science
University of São Paulo (USP)
São Carlos, Brazil
nakamura@icmc.usp.br

Julio C. Estrella
Institute of Mathematics and Computer Science
University of São Paulo (USP)
São Carlos, Brazil
jcezar@icmc.usp.br

Marcos J. Santana
Institute of Mathematics and Computer Science
University of São Paulo (USP)
São Carlos, Brazil
mjs@icmc.usp.br

Regina H. C. Santana
Institute of Mathematics and Computer Science
University of São Paulo (USP)
São Carlos, Brazil
rcs@icmc.usp.br

ABSTRACT

This paper introduces a module named UDOnt-Q (Universal Discovery with Ontology and QoS) that uses the Semantic Web and an Ontology based on Service Level Agreements (SLA) and attributes of Quality of Service (QoS). In order to classify and select Web Services without drastic changes in the standard Web Service architecture, two semantic algorithms were developed and included in the module. At the end, a performance evaluation was executed to compare their performance in different settings, trying to identify an efficient way to promote the selection of Web Services.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Online Information Services – Web based service

General Terms

Algorithms, Languages, Measurement, Performance

Keywords

Semantic Web, Web Service, QoS, UDDI

1. INTRODUCTION

In the current context of the Web, more companies do business through this virtual environment. Thus, a large number of Web applications are becoming available and demanding greater attention on two main points: (1) Interoperability and (2) Quality.

Interoperability is a key issue when there are several applications developed in various programming languages and running on different architectures. For example, it is not acceptable the fact that an online trading company cannot communicate with the operator credit card of its client because their applications were not developed with the same programming language. In this case, the client is not worried if the implementation of e-commerce companies do not understand the language or communication protocol used by the operator, for him the purchase must be conducted as transparently as possible. The lack of interoperability between different systems is solved with the use of Web Services, which adopt a standard communication protocols based on Web.

The Quality of Services (QoS) provided on the Web should also be observed by its developers and providers. Ensuring quality of service in Web Services is not a trivial task [8], as well as in any distributed system. Web Services are submitted to various failures that may occur during the process of the service and transport of the information over the network. Therefore, for a service to obtain quality, it must have the assurance that other important factors, such as availability, latency, security, etc., are provided. These factors are the attributes of QoS, that despite they are not functional, i.e. not having an effect on operation of the service, they become essentials when the customer begins to demand a quality of service.

This paper presents a proposal of a Semantic Web application that use an ontology based on Service Level Agreements (SLA) and Quality of Services (QoS) attributes to classify and select Web Services that have the quality required by the clients. The ontology created is used as a knowledge base and it can be accessed by a module called UDOnt-Q (Universal Discovery with Ontology and QoS) that was developed as a platform for algorithms that use semantics become able to perform the search for quality services.

This article is composed of five sections. In section 2 the related work are mentioned. In section 3, the UDOnt-Q and the ontology are briefly explained. In section 4 is carried out the performance evaluation and analysis of the results, and in the section 5 the conclusion is commented.

WebMedia'11: Proceedings of the 17th Brazilian Symposium on Multimedia and the Web. Short Papers.
October 3 -6, 2011, Florianópolis, SC, Brazil.
ISSN 2175-9650.
SBC - Brazilian Computer Society

2. RELATED WORK

A SOA architecture for Web Service [1], in its original specification, does not present preoccupation with the quality of service. The UDDI (Universal Description, Discovery and Integration) is a register used in the architecture for Web Service to store functional information about the services, such as the interface description (WSDL - Web Service Definition Language) [4]. However, it lacks the ability to record information that are not functional, so it has no information about the QoS services [11].

Furthermore, the services can have different levels of QoS and can become a challenge to find out which ones have the appropriate QoS for a particular activity. To deal with this limitation, several architectural models are discussed in the literature [13], [2]. Some of these works propose the use of mechanisms for managing and monitoring service providers.

Another approach discussed for the implementation of QoS is through the use of Semantic Web. The semantics provides not only the search for explicit information, but also those that can be deduced by an inference engine. Ontologies have been used for descriptions of knowledge domains that can be shared and help the semantic interoperability in different QoS models and languages. Several works suggest the use of ontologies to represent QoS in Web Services [10]. But, some of them are not implemented in a real environment, and when they are, they do not usually present the results of a performance evaluation.

Targeting a more specific point (the quality of services in Web Services), there are studies [11] [12] which developed ontologies, but do not provide information about service agreements. The use of agreements is one of the most reliable ways to achieve a good level of QoS [6]. On this account, both the client and the service provider should monitor themselves and conduct periodic evaluations to determine whether the goals stated in the contract are being achieved, nevertheless the service provider is subject to fines established in the pre-contract if the QoS level required by the customer is not satisfactory. A proposed ontology based in modeling contract SLA (Service Level Agreement) is the *SLAOnt* [3], which works with the parties involved (clients and providers), obligations and services. SLAOnt is the closest ontology with the ones that is presented in this article, but in UDOnt-Q, there is the definition of clients and services classes that are inferred according to the values assigned of QoS required in the agreements.

3. UDONT-Q COMPONENTS

3.1 UDont-Q Module

The UDOnt-Q module was created to serve as a platform for the search algorithms and semantic selection. It is designed to be reusable and configurable to accept new algorithms (not only semantics), requiring minimal changes to source code. It was developed with the Java programming language to be portable to different platforms. The use of this language is also justified by the fact that several semantic tools make available APIs (Application Programming Interfaces).

During the module construction, the inference engine (*reasoner*) called Pellet was used to perform the consistency check and execute inference on ontologies, including those created in the OWL language. Pellet is based on algorithms developed for expressive logics description and includes the characteristics defined in OWL [7]. The UDOnt-Q is divided into components (java packages), each of them performs a specific function, the main ones are:

- **Command Components (CC):** Responsible for leading, directing and analyzing the requests of the clients.
- **UDDI Components (UC):** responsible for the access to the UDDI registry.
- **Common Information Shared (CIS):** responsible for maintaining the information in the ontology and the UDDI registry.
- **Ontology Components (OC):** responsible for searching for QoS service in the ontology, using the following semantic algorithms:
  - **OntAlgorithmObject:** this algorithm uses the API provided by Jena to manipulate the information in the ontology programmatically.
  - **OntAlgoritmSPARQL:** it makes use of packages that allow the use of query language SPARQL-DL a variant of SPARQL (Simple Protocol and RDF Query Language) [9].

3.2 UDOnt-Q Ontology

The ontology for the module UDOnt-Q was created in order to represent the key elements involved in the field of Web Services with QoS. Some of these elements deserve attention:

- **Clients:** Elements that order quality services. For this, they sign agreements with service providers.
- **Providers:** They must provide the Web Services with the quality agreed.
- **Services:** Web Services provided by Providers and consumed by Clients. Web Services should be examined for their functional characteristics and particularly for the non-functional (QoS attributes) be specified.
- **Agreements:** Agreements between clients and providers. The client agreement indicates which is the QoS desired to him.
- **QoS:** Quality of service belonging to a certain service or that contracted in a particular agreement. Additionally, it contains levels and quality attributes.

Each element is represented with a class in the ontology of UDOnt-Q that may has subclasses forming a hierarchy class. Furthermore, they can be linked through properties that are known as "*object properties*". There is also the "*data properties*" which relate a class or instance of a class to a data type (e.g. an integer). These properties can have attributes or characteristics that express information about the relationship allowing the machine to be able to "*understand*" the semantic meaning of the relationship.

The Client, Service and QoS classes have three subclasses that correspond to the clients, services and QoS (Gold, Silver and Bronze). What determines whether a service belongs to a particular subclass is its QoS (related by the property *hasServiceQoS*). Moreover, the client is related to a type of agreement which may belong to three subclasses (*HeavyAgreement*, *MediumAgreement* and *LightAgreement*). Each agreement is related to a QoS that determines its subclass. To determine whether QoS is Gold, Silver or Bronze, the inference engine checks the values (or levels) of the attributes in each QoS and by comparisons with the restrictions set forth in the "Equivalent Classes" it determines what is its subclass. For example, some constraints (or conditions) for any other element of the ontology be an equivalent element to a QoSGold class can be:

- Have the QoS class as "superclass", or be a subclass of QoS.
- Have the value of property *hasResponseTimeContentValue* less than or equal to 500.00 (meaning that the response time should be less than or equal to 500 milliseconds).
- Have the value of property *hasAvailabilityContentValue* greater than or equal to 98.00 (meaning having an availability of service greater than or equal to 98% of the time)

Likewise, there are also restrictions on equivalent classes to subclasses *QoSSilver* and *QoSBronze*, but their values of the ranges are different in each property. Once consistent, the ontology can be used as a knowledge base for semantic search and the classification through inference that allows new individuals (instances of a class) be created to represent the real world without the need to be previously labeled with respect its quality. To find the correct service, the algorithm must seek individuals who represent clients in the ontology and find the services that have a QoS in the same subclass of the client agreement's QoS.

4. PERFORMANCE EVALUATION

4.1 Environment Configuration

The Environment Configuration details the elements of hardware and software used in the experiments. Such information are interesting because it facilitates the reproduction of the environment used during the experiments.

The Table 1 shows the computing infrastructure and Table 2 lists the main software elements used in the performance evaluation.

Table 1: Hardware Elements

Component	Quantity	Configuration
Providers	5	Intel QuadCore Q6000 (2.4GHz) 2GB RAM, HD 120GB, 7200RPM
Clients	3	Intel QuadCore Q9400(2.66GHz) 4GB RAM, HD 500GB, 7200RPM
UDOnt-Q and UDDI	1	Intel QuadCore Q9400(2.66GHz) 8GB RAM, HD 320GB, 7200RPM
Switches	2	Gigabit 3Com Baseline: 2916 and 2913

Table 2: Software Elements

Element	Version
Linux Ubuntu Server	10.04 kernel 2.6.32-26
Apache Web Server	2.2.14
Apache Tomcat	6.0.26
Apache Axis2	1.4.1
jUDDI	0.9rc4
MySQL Server	5.1.41-3ubuntu12.8
Pellet	2.2.2
Jena	2.6.3
Log4J	1.2.16

4.2 Experiment Design

The design of experiments seeks to obtain maximum information with a minimum number of experiments [5]. This experimental planning can facilitate the understanding of the behavior and performance of the modules in certain situations. It also seeks to identify what are the possible responses of the system to be analyzed. In the experiments were considered as response variables the load of CPU utilization and the time spent by the module to find the right service, this time is referenced in this article as the response time of the module.

Another point to be identified is in which situations the module should be evaluated, i.e., to determine which factors are influencing the system performance and which levels of each factor may be of interest.The factors and levels chosen in this planning of experiments are listed in Table 3.

Table 3: Factors and Levels of the Experiments

Factors	Levels
Number of Services (Factor A)	300 and 600
Number of Clients (Factor B)	15 and 30
Algorithm (Factor C)	*OntAlgorithmObject* and *OntAlgorithmSPARQL*

The technique of complete 2^k factorial design was used and determines that each factor (k) has at most two levels. Each variation is a new experiment that should be analyzed. Thus, the complete factorial design to evaluate the module is of 2^3 possible combinations shown in Table 4. In the experiments performed were considered two constraints of equivalent classes (properties): response time and availability.

The hardware elements available and listed in Table 1 were not sufficient to represent 15 and 30 clients, so it was necessary to create threads on the client machines to represent the number of concurrent requests in the proposed experiments (5 or 10 threads per client) and each experiment was replicated 30 (thirty) times.

Table 4: Experiments, Factors and Levels

Exp	A (N. Services)	B (N. Clients)	C (Algorithm)
1	300	15	*OntAlgorithmObject*
2	300	15	*OntAlgorithmSPARQL*
3	300	30	*OntAlgorithmObject*
4	300	30	*OntAlgorithmSPARQL*
5	600	15	*OntAlgorithmObject*
6	600	15	*OntAlgorithmSPARQL*
7	600	30	*OntAlgorithmObject*
8	600	30	*OntAlgorithmSPARQL*

4.3 Result Analysis

The results obtained in the experiments were analyzed statistically, being possible to find the mean, standard deviation, upper and lower limits and the confidence interval at 95% confidence for each response variable chosen in the previous section (CPU load and response time).

The experiments results can be seen in the graph of Figure 1. It shows that doubling the number of concurrent clients, there is on average, an increase in the response time of the module. The same occurs when the number of services from 300 to 600 doubles. The behavior of the algorithms was similar, but the response time were higher when the *OntAlgorithmSPARQL* was used. This shows that the *OntAlgorithmObject* has better performance. The reason for this result is that its development was done specifically for the ontology of UDOnt-Q. While the *OntAlgorithmSPARQL* requires an additional java package for its operation and its characteristics allow it to be flexible with the changes in the structure of the ontology (an amendment only requires the reformulation of the query).

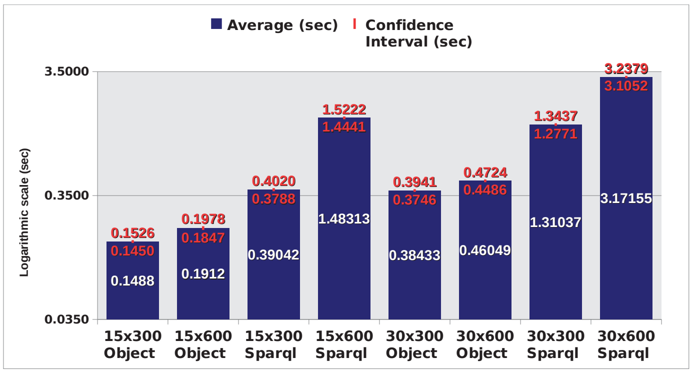

Figure 1: Comparison among concurrent clients, services and algorithms.

Another variable response observed during the experiments was the average of the load of CPU utilization. This load was measured using a Perl script and the results are illustrated in Figure 2. It is noteworthy that in these results can be included the processing fees by the operating system, so to minimize this influence were considered only the rates that exceeded 1% of CPU utilization. Furthermore, it is possible to observe that the CPU usage is higher when the algorithm *OntAlgorithmSPARQL* is used. The largest number of clients also increases the CPU utilization.

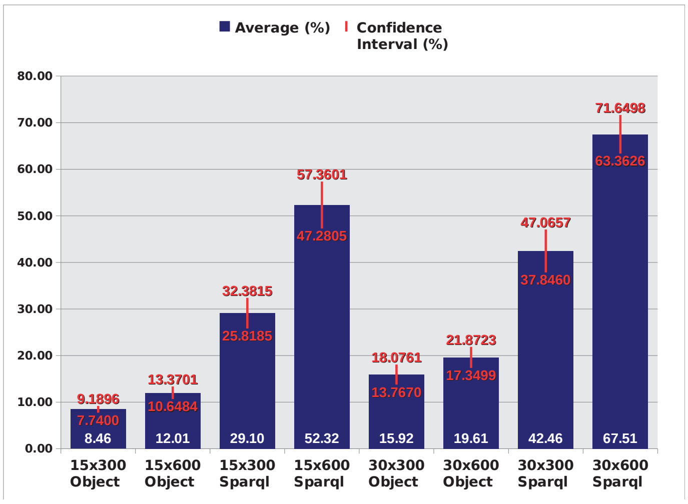

Figure 2: CPU Load during the experiments.

5. CONCLUSION AND FUTURE WORK

This work presented the creation of an ontology specifically designed to serve as a knowledge base for research and selection of Web Services with quality of service. This ontology is accessed by the module UDOnt-Q that actually performs the search and selection of Web Services, so it uses algorithms that make use of Semantic Web resources. The results obtained with the algorithm *OntAlgorithmObject* encourage their use. Regardless the algorithm used, the inference process allows that new clients and services be created in the ontology to represent the real world without requiring changes to the source code.

6. ACKNOWLEDGMENTS

The authors thank the Brazilian Foundation FAPESP for the financial support given to this work.

7. REFERENCES

[1] T. Erl. *SOA Princípios de design de serviços*. Pearson Prentice Hall PTR, 2009.

[2] J. C. Estrella. *WSARCH: Uma arquitetura para a Provisão de web services com Qualidade de Serviço*. Phd thesis, ICMC-USP, São Carlos, SP, 2010.

[3] K. Fakhfakh, T. Chaari, S. Tazi, K. Drira, and M. Jmaiel. A comprehensive ontology-based approach for sla obligations monitoring. *The Second International Conference on Advanced Engineering Computing and Applications in Sciences*, 2008.

[4] P. Farkas and H. Charaf. Web services planning concepts. *Journal Of .net Technologies*, pages 9–12, 2003.

[5] R. Jain. *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1991.

[6] B. Kandukuri, V. Paturi, and A. Rakshit. Services computing. *SCC '09. IEEE International Conference*, pages 517 – 520, 2009.

[7] H.-U. Krieger, B. Kiefer, and T. Declerck. A hybrid reasoning architecture for business intelligence applications. In *Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on*, pages 843 –848, 2008.

[8] S. Lee and D. Shin. Web service qos in multi-domain. In *Proceedings of Advanced Communication Technology, - ICACT*, 2008.

[9] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, 2008. Available on: http://www.w3.org/TR/rdf-sparql-query/. Accessed: 05/03/2011.

[10] V. X. Tran and H. Tsuji. A survey and analysis on semantic in qos for web services. *In.Proceedings of the international Conference on Advanced Information Networking and Applications*, 2009.

[11] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-enhanced qos-based web services discovery. *Proceedings of the 2007 IEEE International Conference on Web Services (ICWS)*, 2007.

[12] G. Ye, C. Wu, J. Yue, and S. Cheng. A qos-aware model for web services discovery. *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, 2009.

[13] S. Zilora and S. Ketha. Think inside the box! optimizing web services performance today [web services in telecommunications, part ii]. *IEEE Communications Magazine*, 46(3):112 –117, march 2008.