Semantic Web and Ontology applied to Web Services Discovery with QoS

Luis H. V. Nakamura, Julio C. Estrella, Marcos J. Santana, Regina H. C. Santana
Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
São Carlos - SP, Brasil
{nakamura, jcezar, mjs, rcs}@icmc.usp.br

Abstract

This paper presents a study on the use of Semantic Web and Ontology created with the OWL (Web Ontology Language) to sort and select Web Services according to its Quality of Service (QoS) attributes. The focus of this paper is to evaluate the performance of this approach in accordance with some adopted parameters. To this end, two algorithms that use Semantic Web resources have been developed in order to make the discovery of appropriate Web Services in the ontology. At the end of this article, the results of a performance evaluation are presented and analyzed.

1. Introdução

A Qualidade de Serviço (QoS – *Quality of Service*) em *Web Services* é uma preocupação atual da indústria de TI. Porém, garantir a qualidade de serviço em *Web Services* não é uma tarefa trivial, uma vez que é um desafio crítico e necessário devido à Internet ser uma rede dinâmica e de natureza imprevisível [10]. Esta garantia é alcançada com o estabelecimento de fatores importantes, como por exemplo, identificar o nível (valor) de cada atributo de QoS e a combinação entre eles para se obter uma eficiência adequada. Contudo, estas medidas tornam-se ineficientes se o provedor não garantir serviços com os níveis adequados de QoS. Uma possível solução para este problema é a utilização de acordos de níveis de serviços (SLA – *Service Level Agreement*) que definem as obrigações das partes envolvidas (provedores de serviços e clientes).

Neste contexto, ainda surge a questão de como determinar quais os serviços que possuem um nível de QoS aceitável do ponto de vista do cliente. O registro UDDI (*Universal Description, Discovery, and Integration*) armazena a lista de *Web Services* de um ou mais provedores e o cliente consulta este registro para encontrar os *Web Services* que deseja. Porém, uma limitação do UDDI é não armazenar informações não-funcionais (QoS) dos serviços. Assim, o usuário não consegue identificar a priori quais serviços apresentam melhor qualidade em relação a outro. Várias propostas na literatura ([21], [22], [23]) objetivam solucionar esta questão. Porém, outra abordagem busca resolver este problema por meio da criação de ontologias e do uso da Web Semântica para a representação e a classificação dos serviços, tendo como base propriedades de QoS ([15], [19] e [2]). Apesar do bom embasamento teórico destes trabalhos, são raros aqueles que apresentam um desenvolvimento prático e informações sobre desempenho.

Diante disso, neste artigo é apresentada uma proposta de ontologia criada com a linguagem OWL (*Web Ontology Language*) para atender as necessidades do módulo UDOnt-Q (*Universal Discovery with Ontology and QoS*). O módulo tem a função de descobrir os melhores *Web Services* para os clientes, com base em atributos de qualidade de serviço. Além disso, o módulo foi concebido com o intuito de ser reutilizável, flexível, extensível e também foi instrumentado para que informações de desempenho possam ser coletadas.

Este artigo está estruturado em sete seções. Na seção 2 é feita uma breve introdução sobre Ontologia e Web Semântica. Alguns trabalhos relacionados são discutidos na seção 3. Na seção 4 são apresentados o módulo UDOnt-Q e sua ontologia. A seção 5 descreve informações a respeito da configuração do ambiente para a realização dos experimentos. Na seção 6 é apresentado o planejamento dos experimentos. Na seção 7 é descrita a avaliação de desempenho do módulo e os resultados são discutidos e analisados. Por fim, a seção 8 apresenta as considerações finais e sugestões para trabalhos futuros.

2. Ontologia e Web Semântica

O termo ontologia que ficou restrito à Filosofia durante anos [4], começou a ser empregado na ciência da computação por volta do final dos anos 70 [11]. Com base no conceito do estudo do ser, as ontologias facilitam a descrição dos seres, estudando suas características, capacidades, relacionamentos, etc. Nesse contexto pode-se notar

o quão importante elas são para criação de semânticas.

As ontologias podem então ser utilizadas para a criação de um vocabulário baseado em um determinado domínio de conhecimento, o qual pode ser compartilhado e reutilizado. Este vocabulário pode ser fornecido de forma explícita e com um grau de formalidade suficiente o qual permite o seu processamento por agentes computacionais. O fato de os agentes computacionais serem capazes de "entender" o conteúdo de ontologias fez com que elas fossem utilizadas na representação de informações da Web e isto permitiu a criação de mecanismos de buscas semânticas de uma forma mais efetiva [8].

A informação contida na Web é geralmente expressa em linguagem natural para que, dessa forma, seja interpretada por pessoas. Portanto, torna-se necessária uma formalização sistemática da informação e dos recursos contidos nela para que os agentes computacionais sejam capazes de processar ("entender") os seus significados [8]. Para representar tais informações na Web, a W3C desenvolveu um *framework* denominado RDF (*Resource Description Framework*) que permite adicionar informações sobre determinado recurso na Web. Dessa forma, é possível adquirir não apenas o conteúdo, mas também informações capazes de atribuir uma semântica àquele recurso [20]. A Figura 1 exibe os três componentes básicos de um arquivo RDF, no qual a Página 1 é o **Sujeito**, a Página 2 é o **Objeto** e há uma propriedade em comum entre elas (*creator* – criador) denominada **Predicado**.

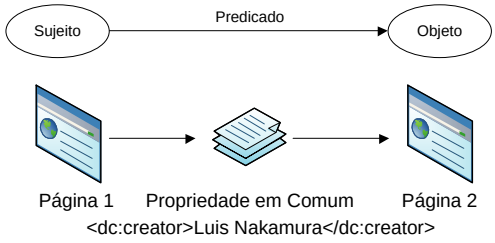

Figura 1. Exemplo de um grafo RDF.

A evolução da Web Semântica estendeu o RDF para que ele ficasse mais expressivo de modo que ontologias mais representativas e formais fossem utilizadas. Dessa forma, foi criada a linguagem OWL (*Web Ontology Language*) que estende o conceito do RDF para a criação de ontologias. O sucesso da linguagem OWL fez com que ela se tornasse o padrão utilizado pela comunidade de Web Semântica [8].

3 Trabalhos Relacionados

A ampla proliferação de *Web Services* despertou a preocupação do meio acadêmico e da indústria de TI (Tecnologia da Informação) em relação à qualidade de serviço desses componentes. O número de empresas utilizando a Web como plataforma para negócios é crescente e aplicações do tipo B2B (*Business to Business*) e SaaS (*Software as a Service*) precisam garantir um mínimo de QoS para os clientes mais exigentes.

Algumas propostas nos últimos anos ([10], [22], [23], [6]) sugerem a criação de novos modelos, ontologias, componentes e arquiteturas que visam um melhor suporte à QoS em *Web Services*. Algumas ontologias criadas para *Web Services* têm o objetivo da descoberta e composição de serviços, como é o caso da GSO (*Goal-Based Service Ontology*) [3] e da OWL-S [12]. Todavia, a OWL-S não menciona os possíveis contratos e obrigações entre os clientes e os provedores. Por outro lado, a ontologia SLAOnt, que é baseada no modelo SLA (*Service Level Agreement*), tem a preocupação com essas características.

4 UDOnt-Q - Universal Discovery with Ontology and QoS

Esta seção explica resumidamente os principais componentes e funcionamento do módulo UDOnt-Q, bem como a ontologia desenvolvida com a linguagem OWL e destinada ao módulo.

4.1 Componentes do UDOnt-Q

O módulo UDOnt-Q foi desenvolvido com o objetivo de realizar uma busca semântica na base de conhecimento representada pela ontologia discutida na próxima subseção. Ele foi projetado desde o início para ser modular e configurável, permitindo a fácil adição de novas classes, pacotes, componentes e algoritmos de busca [14]. O UDOnt-Q faz uso do *framework* Jena que possui uma API a qual permite a manipulação programaticamente de ontologias criadas em RDF e OWL [5].

Além disso, a máquina de inferência Pellet (um *reasoner* de ontologias) também foi utilizada. Pellet é baseado em algoritmos criados para a descrição de lógicas expressivas e inclui as características definidas na linguagem OWL [9]. Ele possui seu código aberto sob uma licença de utilização livre e permite que ontologias desenvolvidas sejam analisadas, classificadas e inferidas de acordo com lógicas de descrição (DL). O módulo foi desenvolvido em componentes (Figura 2)[14]:

- **Command Components (CC)**: comanda, analisa e direciona as requisições;
- **UDDI Components (UC)**: acessa informações no registro UDDI;
- **Common Information Shared (CIS)**: responsável pela manutenção de informações na ontologia e no registro UDDI;

- **Ontology Components (OC)**: busca serviços com QoS na ontologia, por meio de dois algoritmos: (1) ***OntAlgorithmObject*** utiliza a API fornecida pelo Jena para manipular as informações da ontologia instanciando objetos e realizando iterações (*loops*) em busca do cliente e dos serviços adequados, e (2) ***OntAlgorithmSPARQL*** faz uso de uma variação da linguagem de consulta SPARQL (*Simple Protocol and RDF Query Language*) [16], conhecida como SPARQL-DL [18] para buscar informações na ontologia.

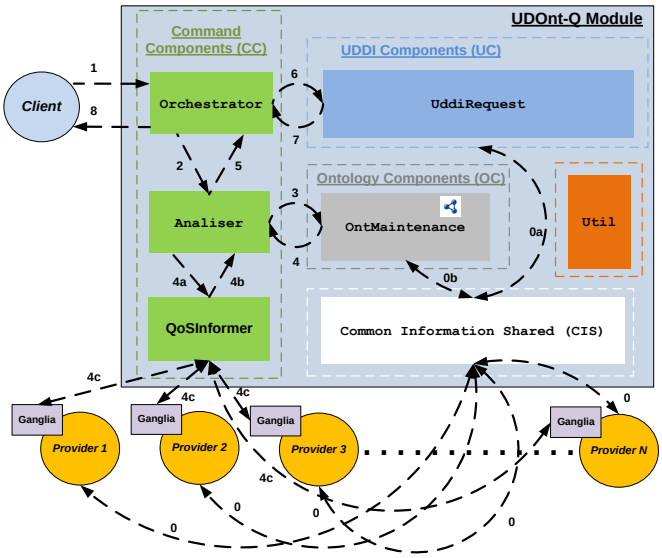

Figura 2. Componentes e funcionamento do módulo UDOnt-Q.

O funcionamento do UDOnt-Q foi apresentado resumidamente em [14]. Esse funcionamento corresponde aos passos exibidos na Figura 2 e descritos a seguir:

0. Inicialmente os provedores de serviço implementam as classes do componente **CIS** para registrar seus serviços no registro UDDI (**0a.**) e na ontologia (**0b.**). Os provedores e clientes devem criar acordos nos quais os clientes especificam os possíveis atributos e os valores (níveis) da QoS desejada.

1. O Cliente informa o nome do serviço desejado e seu endereço IP.

2. O componente **CC** encaminha a requisição para a análise que por sua vez transfere os dados ao componente de busca semântica.

3. O componente **OC** recebe a requisição, instancia o algoritmo e procede com a busca. Esta busca considera o IP do cliente para encontrar o cliente e seu acordo na ontologia. Assim, descobre-se qual a QoS que ele contratou e quais serviços atendem aos requisitos de QoS estipulados. O resultado (nome do serviço, nome do provedor e endereço do provedor) é retornado ao **CC**.

4. O **CC** recebe o resultado da busca semântica e analisa o resultado. Se houve apenas um resultado ele retorna este resultado para a classe *Orchestrator*. Caso haja mais de um registro o sistema pode seguir dois fluxos: (1) se a configuração de monitoramento estiver desativada, então será retornado o primeiro resultado da lista de resultados. Caso a opção esteja ativa (2), (**4a.**) a lista de resultados será encaminhada para a classe *QoSInformer* que analisará o atual estado dos provedores que constam no resultado (IPs dos provedores). (**4b. e 4c.**) Baseando-se nas informações fornecidas pelo monitor (neste caso o Ganglia [13]) é determinado qual é o melhor provedor naquele momento (menos sobrecarregado em CPU e Memória) para atender a requisição do Cliente.

5. A classe *Orchestrator* recebe o resultado e repassa os nomes do serviço e do provedor para o componente **UC**.

6. O componente **UC** realiza a pesquisa no registro UDDI e retorna a WSDL (*Web Service Definition Language*).

7. O componente **CC** recebe a WSDL que será encaminhada ao Cliente.

8. O Cliente recebe a WSDL e, com as informações contidas nela, ele poderá então consumir o serviço. Caso nenhum serviço for encontrado com a qualidade desejada, a resposta *Null* será enviada ao cliente.

4.2 Ontologia do UDOnt-Q

O domínio da ontologia do UDOnt-Q é baseado nos elementos participantes do contexto de *Web Services* com qualidade de serviço. Alguns destes elementos são destacados [14]:

- **Clients:** Os clientes que requisitam os serviços e possuem um acordo com os provedores.
- **Providers:** Os provedores que provêem os serviços.
- **Services:** *Web Services* providos pelos provedores e consumidos pelos clientes.
- **Agreements:** São os acordos estabelecidos entre os clientes e os provedores.
- **QoS:** É a qualidade de serviço de um determinado serviço ou àquela contratada em um determinado acordo (possui os atributos e níveis da qualidade).

Na ontologia, estes elementos tornam-se classes que possuem subclasses, estabelecendo uma hierarquia de classes. Além disso, elas podem se relacionar por meio de propriedades que por sua vez podem possuir características. A hierarquia de classes, propriedades e suas características estão presentes na linguagem OWL, sendo utilizadas para expressar a semântica necessária na ontologia.

Na ontologia proposta há cinco classes, quatro delas com três subclasses, conforme exibe a Figura 3. A primeira classe representa os Acordos (*Agreement*) que possuem as subclasses (*HeavyAgreement*, *MediumAgreement* e *LightAgreement*). As subclasses do Acordo são classificadas considerando as três subclasses de QoS (*QoSGold*, *QoSSilver* e *QoSBronze*). Por exemplo, o acordo relacionado com a subclasse *QoSGold* estará relacionado à subclasse *HeavyAgreement*, pois um cliente *Gold* apresenta um maior nível de exigência. A classe de Serviço (*Service*) também está relacionada com a classe QoS, e o que determinará a subclasse de um serviço será a sua subclasse de QoS. A classe Cliente (*Client*) é dividida nas três subclasses (*ClientGold*, *ClientSilver* e *ClientBronze*) conforme o tipo de Acordo estipulado. Por exemplo, na inserção de informações na ontologia, os Clientes não precisam escolher um Acordo em específico, basta apenas possuir um "Acordo Genérico"(SuperClasse ou Classe Pai) que está relacionado a uma "QoS Genérica". Porém, o cliente deve informar os valores dos atributos de QoS que deseja e conforme estes valores, a QoS será classificada em uma de suas subclasses. Dependendo da subclasse da QoS, o Acordo relacionado também é inferido e classificado em uma subclasse. Consequentemente o tipo de cliente será determinado por meio do seu acordo. Esta classificação é possível somente após a ontologia passar pelo processo de inferência (feita pelo *reasoner* [1]) sem apresentar inconsistências.

A inferência proporciona a classificação em razão das restrições de *Equivalências de Classes* impostas na ontologia. Estas restrições impõem que uma determinada classe ou indivíduos dessa classe (instâncias) sejam equivalentes a outra classe. A Figura 3 exibe a hierarquia das classes da ontologia proposta. As restrições (ou condições) para que qualquer outro elemento da ontologia seja um elemento equivalente da classe *QoSGold* são as seguintes:

- Apresentam a classe QoS como Classe Pai (*Superclass*) ou ser subclasse de QoS;
- Apresentam o valor da propriedade *hasResponse_TimeContentValue* menor ou igual a 500.00 (leia-se ter o Tempo de Resposta menor ou igual a 500 milissegundos).
- Possuem o valor da propriedade *hasCostContentValue* maior ou igual a 1.00 (leia-se ter o Custo maior ou igual a 1 unidade monetária).
- Possuem o valor da propriedade *hasAvailabilityContentValue* maior ou igual a 98.00 (leia-se ter a Disponibilidade maior ou igual a 98%).

O mesmo processo de equivalência também acontece para as classes *QoSSilver* e *QoSBronze*, diferindo os intervalos e valores impostos em cada propriedade de dado.

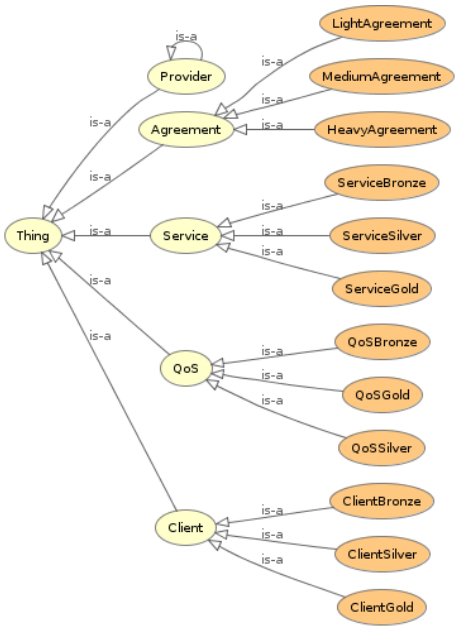

Figura 3. Hierarquia de Classes da ontologia.

Uma vez criada e consistente, a ontologia pode ser utilizada como base de conhecimento para buscas semânticas por informações. Diversos indivíduos de cada classe podem ser criados para representar o mundo real. Por exemplo, determinado provedor é representado por um indivíduo da classe *Provider*, os seus serviços são indivíduos da classe *Service* e alguns indivíduos da classe QoS representam as suas qualidades.

5 Configuração do Ambiente

Como parte do processo de avaliação de desempenho, é pertinente detalhar os elementos de *hardware* e *software* utilizados nos experimentos. Tais informações são úteis, pois facilita a reprodução do ambiente utilizado durante os experimentos. Neste contexto, na Tabela 1 é possível visualizar a infraestrutura computacional utilizada na avaliação de desempenho do módulo.

Tabela 1. Elementos de Hardware

Componente	Quantidade	Configuração
Provedores de Serviços	**5**	**Intel QuadCore Q6000 (2.4GHz) 2GB de RAM HD 120GB, 7200RPM**
Clientes	**3**	**Intel QuadCore Q9400(2.66GHz) 4GB de RAM HD 500GB, 7200RPM**
UDOnt-Q e UDDI	**1**	**Intel QuadCore Q9400(2.66GHz) 8GB de RAM HD 320GB, 7200RPM**

Complementando os elementos de *hardware* descritos anteriormente na Tabela 1, a Tabela 2 lista os principais

elementos de *software* utilizados no desenvolvimento e também na avaliação de desempenho do módulo.

Tabela 2. Elementos de Software

Elemento	Descrição	Versão
Linux Ubuntu Server	Sistema Operacional	10.04
Apache Web Server	Servidor Web da Apache	2.2.14
Apache Tomcat	Container de Servlets	6.0.26
Apache Axis2	Motor (engine) de processamento SOAP	1.4.1
jUDDI	UDDI da OASIS desenvolvida pela Apache (em Java)	0.9rc4
MySQL Server	Sistema Gerenciador de Bancos de Dados (SGBD)	5.1.41-3 ubuntu12.8
Pellet	Máquina de inferência semântica	2.2.2
Jena	Framework que fornece uma API para a manipulação de ontologias	2.6.3
Log4J	Biblioteca desenvolvida pela Apache que fornece uma API para o registro de logs	1.2.16

6 Planejamento dos Experimentos

Segundo Jain [17], das diversas técnicas existentes para realizar o planejamento de experimentos as mais utilizadas são: o planejamento simples, o planejamento fatorial completo e o planejamento fatorial parcial. A técnica adotada neste projeto foi o planejamento completo que utiliza todas as combinações possíveis de todos os fatores e níveis escolhidos para a avaliação. Um fator se refere a uma condição imposta nos experimentos a fim de ser analisada e os níveis são as variações propostas a um determinado fator. Esta técnica é mais dispendiosa que as demais, no entanto, ela permite que os fatores sejam avaliados, sendo assim possível identificar as interações e influências entre eles.

Um fato que ocorreu durante a execução dos experimentos do módulo UDOnt-Q foi uma característica interessante no comportamento do processo de classificação e realização (inferência) da ontologia por parte do *reasoner* Pellet. Foram adicionados mais indivíduos (*Web Services*) na ontologia e, consequentemente, maiores foram os tempos para o processo de inferência. Um aumento já era esperado, uma vez que há mais informações a serem analisadas, porém, a intensidade do aumento não parecia seguir um comportamento linear. Assim, decidiu-se avaliar também esta etapa, realizando uma comparação com uma ontologia clássica e analisando o seu comportamento com vários indivíduos. A ontologia Pizza [7] foi escolhida e o resultado da avaliação é apresentado na próxima seção. Dessa forma, foram criados três **Planejamentos de Experimentos (PE)**:

- PE para a avaliação do processo de classificação e realização (inferência) da ontologia de *Web Services* proposta, utilizando o processo de inicialização do módulo e com a configuração padrão do *reasoner* Pellet (**PE-Ontologia**).
- PE para avaliação do processo de classificação e realização (inferência) da ontologia Pizza utilizando o processo de inicialização do módulo com a configuração padrão do *reasoner* Pellet (**PE-Pizza**).
- PE para a avaliação do módulo UDOnt-Q no processo de busca por *Web Services* com QoS (**PE-Módulo**).

No **PE-Ontologia** a variável de resposta analisada é o tempo de resposta para que a ontologia ficasse disponível para utilização, ou seja, é o tempo gasto para que o módulo classifique, faça a inferência e carregue na memória do servidor a ontologia processada (inferida). Os experimentos dessa avaliação consideram o fator "Número de Serviços" com os níveis: 300, 600, 900 e 1200 serviços. Nessa ontologia, os serviços estão divididos entre os cinco provedores e são classificados em: *Gold*, *Silver* e *Bronze*.

O **PE-Pizza** é semelhante ao **PE-Ontologia**, porém a quantidade de indivíduos é diferente. Para a sua avaliação é considerado o fator "Número de Pizzas" com os níveis: 200, 400, 600 e 1200 pizzas. Nessa ontologia, as pizzas estão agrupadas em pizzas de alta (*HighCaloriePizza*) e baixa caloria (*LowCaloriePizza*).

No **PE-Módulo** a variável de resposta analisada é o tempo de resposta (tempo gasto pelo módulo para encontrar o melhor serviço por meio da busca semântica na ontologia e posteriormente consultar o registro UDDI para recuperar a WSDL), ou seja, executar os passos do item 2 até o item 7 da Figura 2. Neste ponto, a ontologia já foi processada previamente na inicialização do servidor e estava pronta para ser utilizada. A Tabela 3 lista os fatores e seus respectivos níveis associados para a avaliação deste experimento.

Tabela 3. Fator e níveis do PE-Módulo

Fator	Níveis	Descrição
Número de Serviços (Fator A)	300, 600 e 1200	N. de Serviços (indivíduos) na ontologia classificados em: *Gold*, *Silver* e *Bronze*.
Número de Clientes (Fator B)	15 e 30	N. de Clientes (indivíduos) na ontologia. Número de requisições feitas concorrentemente ao módulo. São classificados em: *Gold*, *Silver* e *Bronze*.
Algoritmo de Busca (Fator C)	*OntAlgorithmObject* e *OntAlgorithmSPARQL*	Os dois principais algoritmos de busca com semântica foram avaliados.

7 Avaliação de Desempenho

As avaliações de desempenho dos planejamentos **PE-Ontologia** e **PE-Pizza** são simples, pois há apenas um fator a ser considerado. O objetivo é identificar qual é o comportamento dos tempos de resposta conforme o número de indivíduos aumenta na ontologia. Portanto, cada nível do fator corresponde a um experimento isolado. Dessa

forma, para o **PE-Ontologia** e o **PE-Pizza** foram realizados quatro experimentos e cada um deles foi repetido 10 vezes, obtendo-se um total de 40 testes para o **PE-Ontologia** e para o **PE-Pizza**. Os resultados obtidos e a análise do comportamento destas avaliações serão apresentadas na próxima seção.

Por outro lado, a avaliação de desempenho do planejamento **PE-Módulo** é mais complexa que as anteriores. Conforme mostra a Tabela 5, há três fatores (A, B e C) que variam em dois níveis, ficando de acordo com fatorial completo 2^k que limita que cada fator (k) tenha no máximo 2 níveis. Cada variação corresponde a um novo experimento que deve ser analisado. Dessa forma, o projeto fatorial completo para a avaliação do módulo é de 2^3 possíveis combinações que são listadas na Tabela 4. Os experimentos desta avaliação foram replicados 30 vezes para cada combinação possível.

7.1 Análise dos Resultados Obtidos

Os resultados coletados durante os experimentos foram analisados e são apresentados nesta subseção. Com relação à análise comportamental do processo de inferência das ontologias, pode-se observar nas Figuras 4 e 5 que o tempo gasto com este processo em ambas as ontologias aumentou conforme crescia o número de indivíduos (Serviços ou Pizzas) cadastrados.

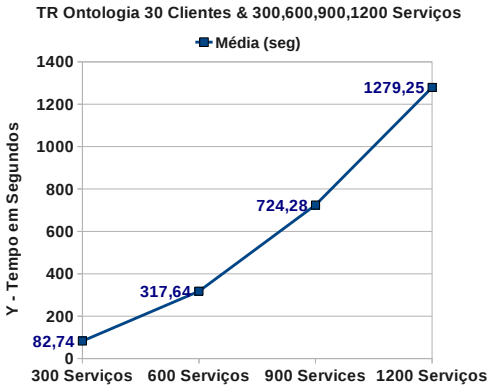

Figura 4. Inferência da Ontologia com 30 Clientes e 300, 600, 900 e 1200 Serviços (PE-Ontologia).

Estes resultados mostram um avanço progressivo no aumento do tempo de resposta (TR) do módulo durante o processo de inferência. As ontologias não podem ser comparadas, pois existem diferentes números de triplas RDF, classes, indivíduos, propriedades, etc. Por exemplo, na ontologia Pizza (Figura 5) não existem elementos como: Clientes, Provedores, QoS, etc. Essa diferença faz com que a ontologia proposta tenha um tempo maior, quando comparada a ontologia Pizza, para o processo de inferência. Além

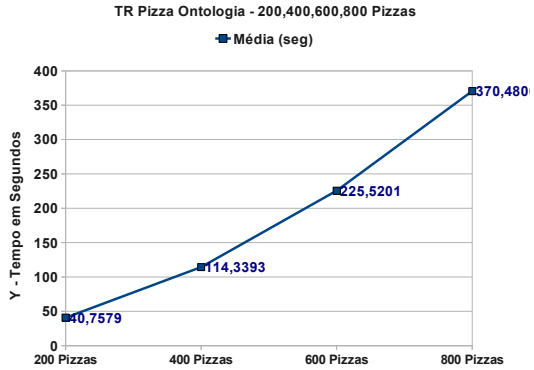

Figura 5. Inferência da Ontologia com 200, 400, 600 e 800 Pizzas (PE-Pizza).

disso, essa diferença na ordem de grandeza do tempo pode ser justificada pelo fato da ontologia proposta possuir duas restrições de *Equivalência de Classes* para cada subclasse de QoS, enquanto o exemplo da ontologia Pizza possui apenas duas restrições, as quais definem se uma pizza é de alta ou baixa caloria. Apesar dessa diferença de tempo, o foco está na análise do comportamento do tempo em cada ontologia, nota-se que para ambas esse aumento não é linear, conforme exibe a Figura 6, e proporciona um aumento agressivo quando a ontologia chega a um alto número de indivíduos.

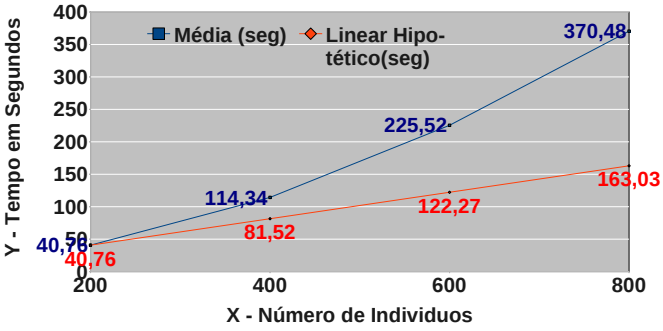

Figura 6. Tempo de inferência da Ontologia Pizza X Comportamento Linear Hipotético.

A Figura 6 compara o comportamento dos tempos de resposta para a inferência da ontologia Pizza com uma linha hipotética que representaria um comportamento linear. Apesar do alto tempo de resposta para o processo de inferência executado na inicialização do módulo, os resultados posteriores durante a busca dos serviços com QoS obtiveram tempos de resposta interessantes. Nestes experimentos o processo de monitoração dos provedores (*QoSInformer*) estava desativado.

O gráfico da Figura 7 mostra a variação no tempo de res-

posta quando o número de clientes concorrentes é dobrado (de 15 para 30) em cenários com 300 e 1200 serviços.

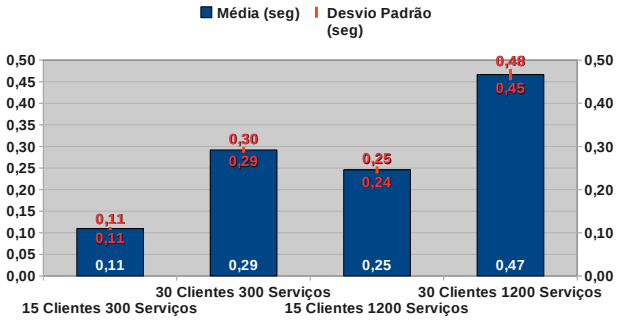

Figura 7. Variação no TR de 15 para 30 Clientes com 300 e 1200 Serviços utilizando o OntAlgorithmObject.

Com a análise da Figura 7, observa-se que ao dobrar o número de clientes concorrentes, o tempo de resposta também apresenta um aumento. Este resultado já era esperado, pois ao aumentar o número de requisições, aumenta-se também a carga de trabalho que o módulo precisa processar. O mesmo ocorre quando há uma variação de 300 para 1200 serviços, pois aumenta a quantidade de informações que o módulo precisa analisar.

O gráfico da Figura 8 mostra a variação no tempo de resposta quando, o número de clientes concorrentes é fixo (30 clientes) e ocorre a troca no algoritmo de busca (*OntAlgorithm**Object*** x *OntAlgorithm**SPARQL***) em cenários com 300 e 1200 serviços.

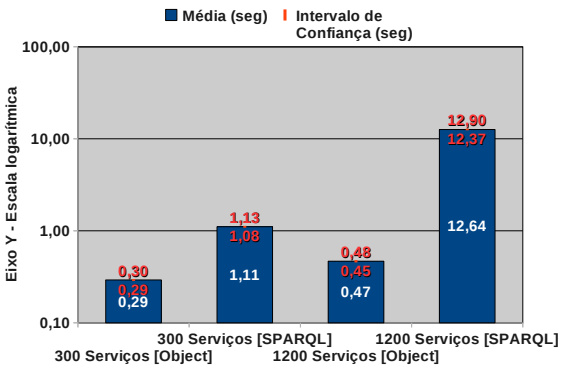

Figura 8. Variação no TR devido a troca de algoritmos com 30 clientes e 300 ou 1200 Serviços.

A análise da Figura 8 mostra um aumento considerável quando se alterna o algoritmo de busca. O algoritmo *OntAlgorithmObject* é mais eficiente do que o *OntAlgorithmSPARQL*. A razão para este resultado é que o *OntAlgorithmObject* teve seu desenvolvimento baseado especificamente para a ontologia do UDOnt-Q. Por outro lado, o *OntAlgorithmSPARQL*, além de exigir um pacote adicional para o seu funcionamento, ele permite que sejam executadas diversas consultas independente da ontologia trabalhada, ou seja, ele é mais flexível as alterações na ontologia.

Tabela 4. Experimentos, Fatores, Níveis e Média dos Resultados

Exp.	A (N. Serviços)	B (N. Clientes)	C (Algoritmo)	Média TR
1	300	15	*Object*	0,10972
2	300	15	*SPARQL*	0,42332
3	300	30	*Object*	0,29207
4	300	30	*SPARQL*	1,10697
5	1200	15	*Object*	0,24566
6	1200	15	*SPARQL*	5,95906
7	1200	30	*Object*	0,46665
8	1200	30	*SPARQL*	12,63800

A influência dos fatores ilustrada na Figura 9, informa que o fator Algoritmo (C) exerceu a maior influência (32,14%) nos experimentos realizados. O segundo fator de maior influência (26,84%) foi o Número de Serviços (A) e a terceira maior influência foi a combinação dos fatores AC (24,96%). O fator Número de Clientes (B) ficou em quarto lugar (5,36%), seguindo pelas combinações BC (4,31%), AB (3,24%) e ABC (3,15%) respectivamente.

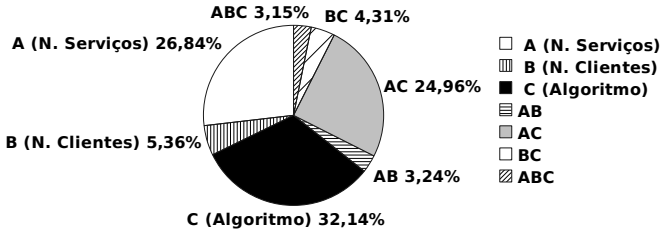

Figura 9. Influência dos Fatores.

8 Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo sobre o emprego de Web Semântica e uma Ontologia aplicada a descoberta de *Web Services* com QoS. Há diversas ferramentas e abordagens de utilização de Web Semântica. Contudo, muitas delas ainda estão em fase de amadurecimento. Neste trabalho foram utilizadas algumas dessas técnicas e ferramentas para a construção de uma ontologia que representasse o contexto dos *Web Services* com QoS.

Notou-se que a proposta de utilizar restrições de equivalência de classes eleva o tempo de inferência quando há muitos indivíduos cadastrados. Logo, em sistemas que necessitam de alterações constantes na ontologia e a mesma precisa ser inferida, tal proposta pode se tornar inviável. Contudo, os resultados dos tempos de busca pelo serviço adequado são, com o algoritmo ***OntAlgorithmObject***, aceitáveis e interessantes.

Em trabalhos futuros será feita uma análise mais detalhada nos códigos e bibliotecas do módulo, Jena e Pellet a fim de encontrar a razão do comportamento do algoritmo ***OntAlgorithmSPARQL***. Comparações com outras propostas na literatura também estão previstas. Quanto a análise do processo de inferência, pretende-se utilizar outro *reasoner* em experimentos futuros, pois notou-se que o Pellet ocupava apenas um núcleo dos quatro disponíveis pela CPU durante esse processo. Além disso, o módulo UDOnt-Q será incluindo na arquitetura WSARCH [6], para que novos experimentos sejam realizados, por exemplo, a criação de réplicas da ontologia dividindo-se os números de serviços (indivíduos) entre elas. Dessa forma, o processo de inferência poderá ser executado em paralelo.

Agradecimentos

Os autores agradecem pelo apoio financeiro da FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo).

Referências

[1] U. Aguilera, J. Abaitua, J. Diaz, D. Bujan, and D. Lopez de Ipina. A semantic matching algorithm for discovery in uddi. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 751 –758, 2007.

[2] Y. Aklouf and E. Rezig. An ontological approach for dynamic functionality-based web services discovery using expert systems. In *Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the*, pages 187 –192, 2009.

[3] J. P. A. Almeida and G. Guizzardi. A semantic foundation for role-related concepts in enterprise modelling. In *EDOC '08: Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 31–40, Washington, DC, USA, 2008. IEEE Computer Society.

[4] S. J. Carase. *Uma ontologia funcional de reputação para agentes.* Dissertação de mestrado em engenharia elétrica, Escola Politécnica da Universidade de São Paulo - USP, São Paulo, SP, 2005.

[5] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. *In.Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters*, 2004.

[6] J. C. Estrella, M. J. Santana, and R. H. C. Santana. *WSARCH: An Architecture for Web Services Provisioning with QoS Support: Performance Challenges*. VDM Verlag Dr. Muller, 2011.

[7] K. Holger, A. Rector, R. Stevens, C. Wroe, S. Jupp, G. Moulton, and N. Drummond. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.2. 2009.

[8] S. Isotani, R. Mizoguchi, I. I. Bittencourt, and E. d. B. Costa. Estado da arte em web semântica e web 2.0: Potencialidades e tendências da nova geração de ambientes de ensino na internet. *Revista Brasileira de Informática na Educação*, 17:30–42, 2009.

[9] 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.

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

[11] L. A. F. Martimiano. *Sobre a estruturação de informação em sistemas de segurança computacional: o uso de ontologia*. Tese de doutorado em ciências de computação e matemática computacional, ICMC-USP, São Carlos, SP, 2006.

[12] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services, 2004. Disponível em: http://www.w3.org/Submission/OWL-S/. Último acesso: 14/08/2009.

[13] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: Design, implementation and experience. *Parallel Computing*, 30:2004, 2003.

[14] L. H. V. Nakamura, J. C. Estrella, M. J. Santana, and R. H. C. Santana. Using semantic web for selection of web services with qos. In *Proceedings of the 17th Brazilian Symposium on Multimedia and the web*, WebMedia '11, New York, NY, USA, 2011. ACM.

[15] I. V. Papaioannou, D. T. Tsesmetzis, I. G. Roussaki, and M. E. Anagnostou. A qos ontology language for web-services. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)*, 2006.

[16] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, 2008. Disponível em: http://www.w3.org/TR/rdf-sparql-query/. Último acesso: 05/03/2011.

[17] J. Raj. *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1991.

[18] E. Sirin and B. Parsia. Sparql-dl: Sparql query for owl-dl. In *In 3rd OWL Experiences and Directions Workshop (OWLED-2007)*, 2007.

[19] X. V. Tran. Ws-qosonto: A qos ontology for web services. In *In: 2008 IEEE International Symposium on Service-Oriented System Engineering*. IEEE International Symposium, 2008.

[20] W3C. Web ontology language overview, 2004. Disponível em: http://www.w3.org/TR/owl-features/ Último acesso: 21/04/2009.

[21] 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.

[22] 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.

[23] L. Yuan-sheng, H. Xiao, W. Yu, and S. Si-xin. Research on a web services discovery model framework. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1 –4, 2010.