OWLPref: Uma Representação Declarativa de Preferências para Web Semântica

Leonardo Ayres, Vasco Furtado

Mestrado em Informática Aplicada – Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Bloco J, Sala 30 60.811-905 – Fortaleza – CE – Brasil
leoayres@gmail.com, vasco@unifor.br

Abstract. *Preference information filtering is a required feature of multiagent systems if they want access a database. However, few studies have paid attention to create a generic representation of preference for agent consumption. In this paper, we describe our proposal for representing preferences in a declarative, domain-independent and machine-interpretable way in OWL. The OWLPref aims at being used for knowledge sharing between agents allowing the representation of partial order between attributes classes and attributes-values. In the article, we also describe the API that translates OWLPref in SPARQL queries. We exemplify the use of the ontology and of the API in the development of multiagent application in the semantic web context.*

Resumo. *Filtrar informações baseadas em preferências é algo comum em aplicações envolvendo agentes na Web, mas poucos trabalhos têm estudado a questão da representação genérica de preferências para uso por esses agentes. Nosso objetivo neste artigo é descrever nossa proposta de representação de preferências de maneira declarativa, independente de domínio e interpretável por máquinas usando OWL. OWLPref será útil para o compartilhamento de conhecimento entre agentes de software, permitindo a representação parcial de ordem entre atributos, classes e valores de atributos. Neste artigo descrevemos a interface de software que mapeia OWLPref em consultas SPARQL e a exemplificamos em uma aplicação na Web Semântica.*

1. Introdução

O desenvolvimento de agentes de software colaborativos que resolvem problemas na Web está tendo um ganho de popularidade. Filtrar informações é algo comum nesse caso e torna-se cada vez mais relevante principalmente em problemas de síntese, tais como *scheduling*, *configuration* e *design* [Junker 2006]. Uma forma tradicional de filtrar informações, originalmente estudada no domínio de banco de dados [Kießling 2002], é a seleção de informações que satisfazem parcialmente um conjunto inicial de restrições. Isto é, ao invés de uma solução que atenda exatamente as restrições, a melhor solução possível é aceitável. Nesses casos, as restrições são chamadas de preferências ou *soft-restrições.*

Poucos trabalhos têm estudado a questão da representação genérica de preferências para uso por agentes na Web. O trabalho mais notório nessa direção foi recentemente proposto por [Siberski 2006] o qual estendeu SPARQL [Prud'hommeaux 2006] para

consultar bases RDF [Hayes 2004] levando em consideração as preferências de usuário. A relevância deste trabalho não o faz definitivo quando se pensa em preferências sob o ponto de vista da representação do conhecimento para seu compartilhamento e padronização. Neste sentido a adoção de ontologias constitui-se um fator determinante para prover interoperabilidade semântica e fácil acesso a informações em ambientes abertos. Nosso objetivo neste artigo é descrever nossa proposta nesta direção que vai além de um método para consultar bases RDF ou OWL [Patel-Schneider 2004] levando-se em consideração as preferências. Temos como foco principal prover uma representação para preferências de maneira declarativa, independente de domínio e interpretável por máquinas. Por esta razão estamos propondo uma ontologia de preferências, denominada *OWLPref*, que será útil para o compartilhamento de conhecimento entre agentes de software, permitindo a representação parcial de ordem entre atributos, classes e valores de atributos.

A partir dessa representação, nós desenvolvemos ainda uma interface de software (aqui chamada de *OWLPref* API) que mapeia *OWLPref* em consultas a uma base RDF. A execução dessas consultas retorna um conjunto de triplas candidatas a atender as preferências especificadas. O uso desta API traz ainda benefícios em termos de engenharia de software, pois as consultas ao banco RDF são automaticamente geradas - o que isenta o desenvolvedor dos agentes de software de fazê-las. Neste artigo descrevemos *OWLPref*, sua API e exemplificamos como ela tem sido usada no desenvolvimento de aplicações multiagentes no contexto da Web semântica.

2. Trabalhos relacionados

Existem duas formas de representar preferências: abordagem numérica ou de ordem total e abordagem simbólica ou ordem parcial. Na abordagem numérica, funções de utilidade calculam o grau/*rank* das alternativas e o de maior valor calculado é selecionado como solução. A representação numérica de preferências possui alguns problemas que se tornam determinantes quando se fala em compartilhamento de conhecimento. A determinação da função de utilidade e sua descrição são difíceis e requerem um grande esforço, pois descrever a preferência de todas as alternativas muitas vezes é inviável [Doyle 1994], [Ha 1999], [Faltings 2004]. Além disso, em muitas situações reais o ótimo global não é crucial [Brafman 1997] [Boutilier 2004].

Já o enfoque qualitativo ou simbólico permite a explicitação de uma ordem parcial sobre as preferências o que facilita a compreensão e a explicação das mesmas [Brafman 1997],[Rossi 2004],[Dubois 2001]. A representação de preferências parciais vem sendo estudada no contexto de banco de dados e recuperação de informações [Kießling 2002], [Chomicki 2003] onde são usadas representações baseadas em atributos/valores. [Brafman 1997],[Boutilier 2004] estudam preferências condicionais também com o uso de uma representação baseada em atributos e valores.

3. Descrição da Ontologia *OWLPref*

Nenhum trabalho até então tinha buscado representar genericamente preferências em lógica descritiva. Desta forma não aproveitam seu poder de representação e de raciocínio para fins de aplicações, principalmente, no contexto da Web Semântica. Nosso trabalho segue essa direção e por isso decidimos usar OWL como linguagem, visto que se tornou padrão para a representação de conhecimento na Web e de possuir

potência representacional suficiente para a definição das preferências que consideramos importantes. Além disso, OWL permite descrever conceitos de domínios e a relação entre os mesmos através de um conjunto de operadores (interseção, união, complemento, propriedades transitivas e inversas, etc.).

Os principais conceitos da ontologia de preferências *OWLPref* são descritos na Figura 1 e visam representar preferências simples, compostas e condicionais. Todos estes conceitos foram formalizados em OWL utilizando uma sintaxe baseada em RDF/XML[1]. As Preferências Simples referem-se aos atributos (prefira *Preço* a *Cor*), às classes (prefira *Notebook* a *Desktop* – significando que todas as instâncias da classe *Notebook* são preferidas em relação às instâncias da classe *Desktop*), ou aos valores de atributos (prefira *discos rígidos com modelo SATA* aos *discos rígidos com modelo IDE*). Como a ontologia refere-se a conceitos OWL, classes referem-se ao conceito *owl:Class*, enquanto que atributos referem-se às propriedades de objetos bem como tipos de dados (*owl:ObjectProperty* e *owl:DatatypeProperty*, respectivamente). Por último, preferência por valores de atributos são os possíveis valores que *owl:ObjectProperty* e *owl:DatatypeProperty* podem assumir. No caso de preferências por valores de atributos (*AttributeValuePreference* da Figura 1), criamos os seguintes conceitos:

- Preferência Máxima ou Mínima – representa a preferência pela maximização/minimização de um valor de uma propriedade. Por exemplo: prefira *o processador mais veloz* ou prefira *o mouse mais barato.*
- Preferência Aproximada – representa a preferência por um valor que é próximo a um valor de referência. Por exemplo: prefira *alterar a freqüência do relógio o menor possível (*em referência à freqüência atual).

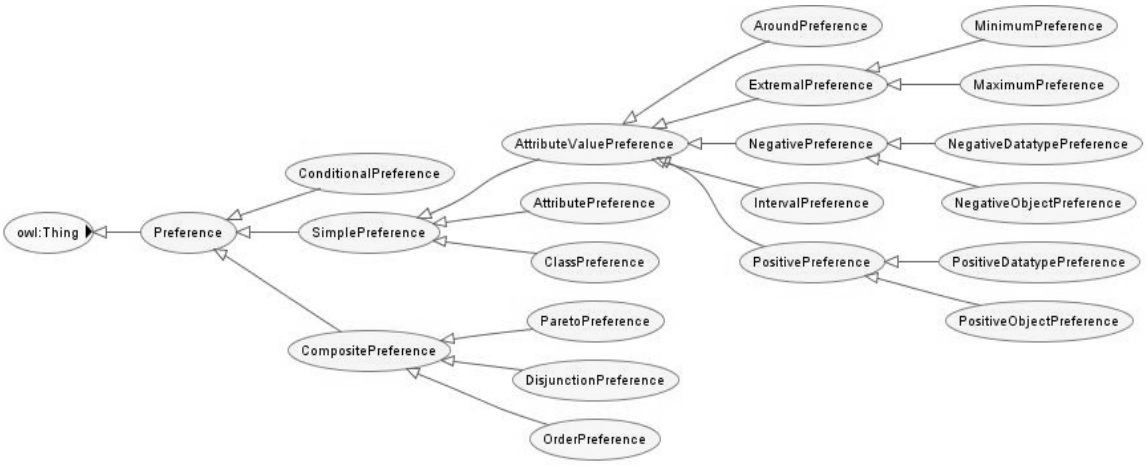

Figura 1. Ontologia de Preferências - *OWLPref*

- Preferência Positiva ou Negativa – nesse caso é especificado o valor que é preferido (ou não) em relação aos demais. Por exemplo: prefira *Carros azuis* ou prefira *Carros que não sejam vermelhos*.

[1] Disponível em http://unifor.s41.eatj.com/ontologies/preferences.owl

- Preferência Intercalada – semelhante à Preferência Aproximada, a Intercalada representa uma preferência por uma faixa ou intervalo de valores. Exemplo: prefira *Livros entre R$10,00 e R$30,00.*

Preferências compostas permitem a combinação de preferências para que se represente ordem (prefira *desktop, tablet, handhelp* – nessa ordem), disjunções (prefira *teclado qwerty keyboard ou mouse de três botões*) e pareto-optimal (prefira *preço e cor de maneira igual*).

3.1 Domínio de preferências

Preferências variam sempre de acordo com o domínio em que está contida. Por exemplo, usuário A ter a cor preferida como AZUL. Nesse caso, bastaria modelar uma Preferência Positiva na propriedade COR com o valor AZUL. Porém, no domínio de carros, a cor preferida dele pode ser outra – vermelha, por exemplo. Assim, nesse caso, a classe Carro deve ser especificada para restringir os domínios. *OWLPref* permite essa representação através da propriedade *hasOWLClass* que define a classe OWL, e, consequentemente, o domínio que a preferência se aplica. Na Figura 2, pode-se visualizar os dois exemplos citados.

```
<PositiveObjectPreference rdf:ID= "Pref_Cor_Azul">
   <hasOWLProperty rdf:resource= "http://prop.owl#Cor" />
   <hasValueObjectProperty rdf:resource="http://cor.owl#Azul">
</PositiveObjectPreference>
```

```
<PositiveObjectPreference rdf:ID= "Pref_Carro_Cor_Azul">
        <hasOWLClass rdf:resource= "http://carros.owl#Carro" />
        <hasOWLProperty rdf:resource= "http://prop.owl#Cor" />
         <hasValueObjectProperty rdf:resource="http://cor.owl#Azul">
</PositiveObjectPreference>
```

Figura 2. Preferência Positiva sem restrição de domínio e com domínio restrito (Carros)

3.2 Combinação de Preferências

Com a universalidade de domínios e atributos existentes, aumenta cada vez mais a quantidade de preferências específicas. Com isso, é necessário criar mecanismos que representem relacionamentos entre essas preferências: seja do tipo hierárquico ou não hierárquico.

Um relacionamento hierárquico, como próprio nome diz, define hierarquias ou prioridades entre as preferências. Ou seja, Pref 1 tem mais prioridade que Pref 2 que por sua vez tem mais prioridade que Pref n, e assim sucessivamente. Por exemplo, Preferência por menor *Preço* pode ser maior que preferência por *Pagamento com Cartão de Crédito*. Para representação desse tipo de hierarquia entre preferência utilizamos a Preferência Composta do tipo Ordenada. Nessa preferência temos uma lista de preferências ordenada pela importância: a primeira da lista tem maior importância enquanto que a última tem a menor importância em relação às outras da lista.

Já em um relacionamento não hierárquico, as preferências possuem entre si o mesmo grau de importância. Podem ser divididas em: Preferência Composta Disjuntiva e a Preferência Composta Conjuntiva (também conhecida como Cumulativa ou de Pareto-optimal [Kießling 2001]). Em ambas, todas as preferências são igualmente importantes. Porém, na Disjuntiva, é necessário apenas que uma das preferências seja atendida, enquanto que na Pareto todas as preferências precisam ser atendidas. Por exemplo, suponha essas duas preferências: *Prefira Carro Zero km* e *Prefira Carro Branco*. Se para o agente B, tanto faz o carro ser zero km "*ou*" branco, então essa preferência deve

ser representada como Disjuntiva. Agora, se pro agente A, o ideal é que o carro seja zero km "*e"* branco, então nesse caso uma preferência Pareto deve ser utilizada.

3.3 Preferências e OWL

O fato de *OWLPref* ser em OWL permite que as características e funcionalidades de OWL possam ser utilizadas para uma melhor representação de preferências mais elaboradas. Por exemplo, vamos supor uma ontologia que represente três classes (*Pessoa*, *Linguagem* e *Java*) e duas propriedades relativas às duas primeiras classes ("*programa*" e "*tipo*"). Suponha que exista uma preferência do tipo: "*Prefiro pessoas que programem em Linguagem Java"*. Para este caso podemos utilizar o conceito de restrição de OWL criando uma interseção (*owl:intersection*) entre a classe *Linguagem* com uma restrição (*owl:Restriction)* na propriedade *tipo* que obriga que todos os valores sejam da classe *JAVA.* Na Figura 3, temos a representação desse exemplo onde o quadro branco da direita contém a interseção definida e a inclusão do mesmo como valor de uma propriedade da preferência positiva. É nesse momento que combinamos o poder de representação já existente em OWL com a complementação de nossa ontologia para representar preferências mais adequadamente.

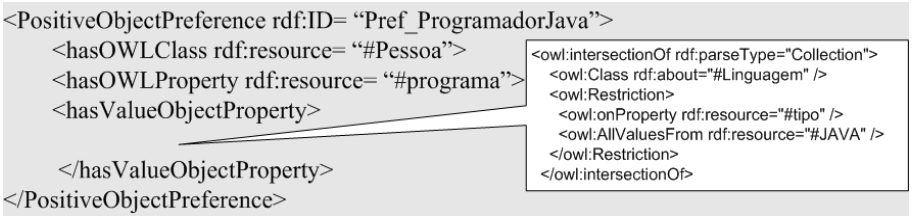

Figura 3. Combinação de representação em OWL e OWLPref

3.4. Representação de Preferências Condicionais

A representação de preferências que variam de acordo com o contexto é possível através de preferências condicionais. Por exemplo, *se localização for Brasil, prefira processadores Pentium.* O conceito de Preferência Condicional é composto de duas propriedades: *onCondition* e *hasPreference.* A primeira propriedade permite a representação das condições em que a preferência se aplica. Já a segunda propriedade, *hasPreference,* refere-se à preferência propriamente dita (simples ou composta) que será aplicada caso a condição, expressada na propriedade *onCondition,* seja verdadeira. As expressões condicionais, no caso, são representadas em SWRL [Horrocks 2004] (*swrl:expression)* com o intuito de aproveitar o poder de expressividade de condições dessa linguagem de representação de regra e manter-se dentro do quadro de linguagens de representação semântica para Web.

4. Uma Interface de software para uso de *OWLPref*

Somente a representação da ontologia não seria suficiente para permitir que agentes pudessem trabalhar com preferências na Web Semântica. Isso exigiria o desenvolvimento de um raciocinador que interpretasse os conceitos da ontologia e aplicasse as preferências especificadas sobre os dados. Por esta razão, desenvolvemos *OWLPref* API para realizar o mapeamento dos conceitos de *OWLPref* para sentenças SPARQL *Preference.* Ela foi desenvolvida em Java e é baseada em JENA [Jena 2006]. Basicamente, foram criadas classes Java representando todos os conceitos de *OWLPref*

e através de métodos de manipulação do JENA foram relacionadas aos conceitos de *OWLPref*. O seu fluxo de funcionamento é simples e se baseia praticamente em três etapas: identificação, geração e execução de consultas.

Na fase de identificação da preferência, *OWLPref* API efetua a leitura da preferência em *OWLPref* e faz a associação do conceito *OWLPref* identificado com a classe Java equivalente da API. Após a identificação do conceito e, consequentemente, seu mapeamento à classe representante da preferência, a API possui condições de fazer a geração da consulta relativa à preferência identificada. Isso porque, dependendo do tipo de preferência, uma consulta pode ter formas diferentes de geração e necessitar (ou não) de determinados valores. Por exemplo, uma consulta para uma preferência positiva necessita identificar a propriedade e o valor da mesma, porém uma consulta para uma preferência extrema (máxima ou mínima) necessita apenas da identificação da propriedade visto que o objetivo é a maximização/minimização do valor da propriedade.

O motivo da escolha de SPARQL *Preference* [Siberski 2006] como linguagem de consulta é que SPARQL é atualmente a mais importante e utilizada linguagem para consultas de dados semânticos. Além disso, ao usarmos uma linguagem já existente podemos nos concentrar mais na modelagem da ontologia e utilizar o motor de inferência do próprio SPARQL para a execução das consultas. De qualquer forma, implementações de outras linguagens de consulta podem ser desenvolvidas de forma similar ao que fizemos para SPARQL. Para isso, basta que as interfaces de geração e execução da linguagem de consulta desejada sejam implementadas e adicionadas à *OWLPref* API. O mapeamento entre a preferência definida em *OWLPref* e a sentença SPARQL *Preference* correspondente é realizado de acordo com cada tipo de preferência. Para cada conceito definimos a correspondente sentença em SPARQL *Preference*. Uma tabela de correspondência foi criada contendo todas as possíveis transformações. A partir dessas informações a API tem condições de gerar as sentenças SPARQL corretamente[2].

Figura 4. Exemplo de mapeamento do OWLPref para SPARQL Preference

Na Figura 4, temos exemplo do mapeamento de uma preferência positiva para uma sentença SPARQL *Preference* equivalente. Os itens destacados são as variáveis das propriedades que serão mapeadas da representação da ontologia de preferência para a sentença SPARQL *Preference*. No caso, a API verifica que *hasOWLClass* e seu valor (linha 2 do lado esquerdo) equivalem a uma cláusula *rdf:type* no SPARQL. O mesmo processo se repete com as outras propriedades destacadas.

Por fim, a fase de execução de consultas utiliza o motor de inferência de SPARQL. Para execução propriamente dita, a API fornece a classe *PreferenceApplier* que aplica

[2] As sentenças geradas puderam ser validadas através do site http://prefs.l3s.uni-hannover.de/ disponibilizado pelos autores do SPARQL *Preference*

preferências *OWLPref* nos dados especificados. Ela funciona como uma espécie de "filtro" onde são retornadas, dentro desta base de dados, todas as instâncias que atendem as preferências identificadas pela API. Caso nenhuma instância for compatível com preferência repassada, então a preferência é simplesmente ignorada.

5. Estudo de Caso

OWLPref foi utilizada em um sistema de configuração de serviços Web semânticos [Fernandes 2006]. Nessa proposta, uma abordagem multiagente foi utilizada para realizar um processo de montagem e configuração de um computador através de um pedido de um cliente. Resumidamente, o fluxo do processo é o seguinte: um cliente faz um pedido através do telefone ou da Web; o pedido do cliente contém informações como: características especiais desejadas, preferências sobre as peças, preferências sobre pagamento, dados pessoais, etc.; depois disso, o pedido é analisado para que a configuração do computador seja produzida, levando em consideração as informações de fábricas espalhadas pelo mundo além das informações repassadas pelo cliente. A configuração final é apresentada ao usuário para que o mesmo confirme o pedido e o computador seja produzido. Nessa aplicação, agentes interagem para cooperativamente encontrar propostas de peças que atendam às restrições e preferências do cliente até a montagem final do produto. Quando um agente solicita proposta de peça para outro agente fornecedor, é necessário que este envie a preferência do usuário modelada em *OWLPref*. Com isso, o agente fornecedor propõe as peças que mais se adequam às preferências repassadas. Por exemplo: suponha que um agente *personalAgent1* requisita ao agente fornecedor *personalAgent2* proposta de disco rígidos e informa que prefere aqueles com preço menor que $300,00. Para este fim, uma preferência *PositiveDatatypePreference* deve ser modelada da seguinte forma: *hasOWLClass* - como se refere à classe OWL que a preferência se aplica, deve apontar para o conceito *HardDisk* da ontologia de domínio de computadores. *hasOWLProperty* - refere-se a propriedade da classe OWL em que a preferência se aplica. No caso, aponta para a propriedade *Price* da mesma ontologia de domínio. *hasOperator* - define o operador que é aplicado na representação da preferência. Como no exemplo são os discos rígidos que custam menos que $300,00 então deve-se utilizar o operador *lessThan*. *hasValueDatatypeProperty* - define o valor da propriedade OWL referenciada no *hasOWLProperty*. No caso do exemplo, 300,00. Na Figura 5, mostramos a preferência acima definida no formato RDF/XML-ABBREV que foi salva como *myPreferencesAboutHDs.owl* .

```
<PositiveDatatypePreference rdf:ID= "Preferencia_1">
        <hasOWLClass rdf:resource= "http://mia.unifor.br/computer.owl#HardDisk" />
        <hasOWLProperty rdf:resource= "http://mia.unifor.br/computer.owl#price" />
        <hasOperator rdf:resource="&swrl;lessThan" />
        <hasValueDatatypeProperty rdf:datatype="&xsd;float">300</hasValueDatatypeProperty>
</PositiveDatatypePreference>
```

Figura 5. Preferência por discos rígidos com preço menor que $300,00

Nesta aplicação a comunicação entre os agentes é feita por envio de mensagens FIPA/ACL[3]. No lado esquerdo da Figura 6, temos o formato da mensagem que é enviada ao *personalAgent2*, sendo destacado o tipo de peça do qual se necessita e qual a preferência do usuário. Internamente o *personalAgent2* processa a mensagem e propõe um disco rígido com base na preferência informada pelo *personalAgent1*. Abaixo temos o trecho do código utilizado pelo *personalAgent2* para aplicar a preferência utilizando a API do *OWLPref.*

```
OWLIndividualList resultData = PreferenceApplier.applyPreference(
URI.create(http://mia.unifor.br/prefs/myPreferencesAboutHDs.owl),
URI.create(http://mia.unifor.br/supplier1/database.owl)));
```

Neste trecho a URI das preferências sobre a peça requisitada e a URI da base de dados OWL em que as preferências serão aplicadas são passadas como argumento para a classe *PreferenceApplier*. A base consultada pelo *personalAgent2* possui oito discos rígidos com características específicas e estão na Tabela 1.

```
(REQUEST
  :sender(agent-identifier :name personalAgent1@mia.unifor.br)
  :receiver(set (agent-identifier :name personalAgent2@mia.unifor.br
:addresses (sequence http://grid10:7778/acc)) )
  :content "(propose http://mia.unifor.br/computer.owl#HardDisk),
            (preferences http://prefs/hd/myPreferencesAboutHDs.owl)"
  :language fipa-sl
  :ontology configuration-ontology
  :protocol fipa-request
  :conversation-id PSMOWLS-MIA-1315817452067
)
```

```
(INFORM
  :sender(agent-identifier :name personalAgent2@mia.unifor.br)
  :receiver(set    (agent-identifier :name personalAgent1@mia.unifor.br :addresses
(sequence http://grid10:7778/acc)))
  :content "((response
              (proposeProcess :input-string
                 (propose http://mia.unifor.br/computer.owl#HardDisk),
                 (preferences http://mia.unifor.br/prefs/myPreferencesAboutHDs.owl)
              proposeProcess :result (http://result.owl))
  ))"
  :language fipa-sl
  :ontology configuration-ontology
  :protocol fipa-request
  :conversation-id PSMOWLS-MIA-1210317290357
)
```

Figura 6. Mensagens trocadas entre dois agentes: o 1º requisita proposta de HD com preferência e o 2º devolve dados baseados nas preferências

Tabela 1. Instâncias HardDisk consultadas pelo *personalAgent2*

Id	hd:Price	hd:Model	hd:storageCapacity
<#HardDisk_1>	300	SATA	250
<#HardDisk_2>	250	SATA	300
<#HardDisk_3>	100	IDE	150
<#HardDisk_4>	100	IDE	200
<#HardDisk_5>	100	IDE	250
<#HardDisk_6>	150	IDE	300
<#HardDisk_7>	850	SCSI	500
<#HardDisk_8>	900	SCSI	550

O módulo gerador de consultas gera uma consulta SPARQL incluindo a preferência *PREFERRING (?price < 300)*[4] e definindo o *rdf:type* como *#HardDisk.* A consulta e os dados resultantes podem ser conferidos na Figura 7.

Após isso, *PreferenceApplier* retorna a lista de instâncias OWL contendo (se houver) os discos rígidos cujo valor seja inferior à $300,00. Essa lista é persistida em um arquivo

[3] Mais informações sobre FIPA/ACL bem como especificações podem ser encontradas em http://www.fipa.org/repository/aclspecs.html

[4] SPARQL *Preference* estendeu SPARQL incluindo diversos modificadores como: *LOWEST, CASCADE, HIGHEST*, etc. No caso, *PREFERRING* também se trata de um operador incluído na extensão. Mais informações sobre a sintaxe do SPARQL Preference podem ser encontradas em [Siberski 2006].

OWL (no caso *result.owl*) e enviada ao agente que demandou a proposta de peças. A mensagem de resposta ao agente demandante pode ser visualizada no lado direito da Figura 6. Caso a preferência fosse por discos rígidos mais baratos (preferência mínima no atributo preço), então a API iria gerar uma consulta com a cláusula *PREFERRING LOWEST (?price)* que retornaria os #HardDisk_3, #HardDisk_4 e #HardDisk_5.

```
 1 PREFIX  hd:    <http://mia.unifor.br/computer.owl#>
 2 PREFIX  rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3 SELECT  *
 4 WHERE
 5   { ?hd  hd:price            ?price ;
 6          hd:model            ?model ;
 7          hd:storageCapacity  ?storageCapacity ;
 8          rdf:type            hd:HardDisk .
 9   }
10 PREFERRING ( ?price < 300 )
```

hd	price	model	storageCapacity
<file:///srv/www/vhosts/test/cgi-bin/d246003.n3#HardDisk_6>	150	"IDE"	300
<file:///srv/www/vhosts/test/cgi-bin/d246003.n3#HardDisk_5>	100	"IDE"	250
<file:///srv/www/vhosts/test/cgi-bin/d246003.n3#HardDisk_4>	100	"IDE"	200
<file:///srv/www/vhosts/test/cgi-bin/d246003.n3#HardDisk_3>	100	"IDE"	150
<file:///srv/www/vhosts/test/cgi-bin/d246003.n3#HardDisk_2>	250	"SATA"	300

Figura 7. Consulta gerada representando discos rígidos com preço inferior a $300,00

6. Conclusão

Nesse artigo, nós apresentamos uma ontologia que visa facilitar a representação de preferências para utilização em aplicações na Web Semântica. Demonstramos como integrar *OWLPref* com propriedades OWL bem como combinar preferências entre si com o objetivo de ampliar ao máximo o poder de representação.

Inúmeras aplicações relacionadas a Web Semântica podem fazer uso de *OWLPref* tais como: personalização de conteúdos – disponibilizar conteúdo mais apropriado ao usuário (Exemplo: Prefiro Noticiário de Esportes mais que Novelas.), otimização de buscas – *search-engines* semânticas podem captar as preferências do usuário antes de retornar os resultados, grids semânticos – preferências por máquinas ou processadores específicos, etc. Além disso, outras linguagens que estendem OWL também podem ser utilizadas como insumo para modelagem de preferências como OWL-S– extensão de OWL para descrição de serviços Web que auxilia a automatização da localização e composição de serviços [Martin 2006]. *OWLPref* pode ser utilizado para definir os serviços preferidos de certos agentes que realizam composições de serviços. A definição de um serviço Web que disponibilize as funcionalidades da classe *PreferenceApplier* da API e elimine a necessidade de inclusão do código que instância a API dentro da estrutura interna do agente é objeto de nossos estudos em andamento.

7. Referências

Boutilier C., Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Preference-based constrained optimization with CP-nets. Computational Intelligence, 20:137.157, 2004.

Brafman R. and Tennenholtz M.; Modeling agents as qualitative decision makers. Artif. Intell., 94(1-2):217.268, 1997.

Chomicki, J.; Preference formulas in relational queries. ACM Trans. DB Syst.28. 2003.

Doyle, J. and M. Wellman. Representing preferences as ceteris paribus comparatives. In AAAI Spring Sym. on Decision-Theoretic Planning, 1994.

Dubois, D., Prade, H., and Sabbadin, R.Decision-theoretic foundations of qualitative possibility theory. European Journal of Operational Research, 128:459–78. 2001.

Faltings, B.; Marc Torrens, and Pearl Pu. Solution generation with qualitative models of preferences. In Computational Intelligence, pages 246–263(18). ACM, 2004.

Fernandes, G. Ayres, L. Oliveira, J., Furtado, V.: An Agent-based Approach for Explaining Web Service Composition via Problem Solving Methods in OWL-S. Business Agents and Web Services - AAMAS, workshop, 2006

Hayes, P.: RDF Semantics (2004) W3C recommendation. Disponível em http://www.w3.org/TR/rdf-mt/. Acessado em Abril/2007.

Horrocks, I.; P. F. Patel-Schneider; H. Boley; S. Tabet; B. Grosof and M. Dean. SWRL: A semantic web rule language combining owl and ruleml. W3C Member Submission, May 2004. Disponivel em http://www.w3.org/Submission/SWRL/.

Jena Semantic Web Framework for Java Web Site. HP Labs Semantic Web Programme. http://jena.sourceforge.net/ . Acessado em Março/2006.

Junker, Ulrich. Preference-based problem solving for Constraint Programming. In Preferences:Specification,Inference,Applications,Dagstuhl SeminarProc.04271 2006.

Kießling, W.; Foundations of preferences in database systems. In: Proc. of the 28th Inter. Conference on Very Large Data Base, 311-322. Hong Kong, China, 2002.

Kießling, W.; B. Hafenrichter; S. Fischer; and S. Holland.; Preference XPATH - A Query Language for ECommerce. In Proc. 5. Intern. Tagung WirtschaftsInformatik, Augsburg, Germany, Sept 2001.

Martin, D.; et al. OWL-S: Semantic Markup for Web Services. W3C Submission. Disponível em http://www.w3.org/Submission/OWL-S/. Acessado em Abril/2006

McIlraith, S., T. C. Son and H. Zeng.; Semantic Web Services. IEEE Intelligent Systems, Special Issue on the Semantic Web, 16(2):46—53, March/April, 2001.

Paolucci, M.; Kawamura, T.; Payne, T., and Sycara, K. Semantic Matching of Web Services Capabilities. In Proceedings of the First International Semantic Web Conference (ISWC). LNCS. Berlin: Springer-Verlag. 2002.

Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax (2004) W3C Recommendation. Disponível em http://www.w3.org/TR/owl-semantics/

Prud'hommeaux, Eric; Andy Seaborne; SPARQL Query Language for RDF. W3C Candidate Recommendation, http://www.w3.org/TR/rdf-sparql-query/. 2006

Rossi, F.; Kristen Brent Venable, Toby Walsh.; mCP nets: representing and reasoning with preferences of multiple agents. In AAAI'04, San Jose, CA, USA, July 2004.

Siberski, W.; Pan, Jeff and Uwe Thaden. Querying the Semantic Web with Preferences. In Proc of the 5th Intern. Semantic Web Conference (ISWC2006), Athens,USA, 2006.