6
Estudo de caso

O estudo de caso implementado visa avaliar o quanto a utilização de ontologias pode auxiliar no desenvolvimento de aplicações B2C.

A aplicação que foi implementada possui duas funcionalidades principais: auxiliar a busca de determinado produto em várias lojas, e recomendar produtos a um determinado perfil de usuário.

A busca de produtos por meio do sistema é feita em várias lojas. O consumidor entra com as características desejadas do produto em cada campo específico e o sistema retorna todos os produtos que satisfazem a essas características. Por exemplo, se o consumidor deseja comprar um livro de título "Harry Potter" e de preço inferior a 40 reais, basta que ele entre com essas informações nos campos específicos da busca por livros, tendo como resultado a lista de livros que estão de acordo com a pesquisa nas diversas lojas virtuais.

O sistema de recomendação permite ao consumidor entrar com um perfil de pessoa e obter todos os produtos que são indicados para aquele perfil de pessoa. O sistema de recomendação implementado faz recomendações por segmentos funcionando da seguinte forma:

- O consumidor entra com um perfil de pessoa. Esse perfil é composto de informações como: sexo, idade, se a pessoa é casada, se a pessoa tem filhos, etc. Na concepção do sistema a idéia é que o perfil da pessoa que será presenteada seja inserido explicitamente.
- Uma vez inserido o perfil da pessoa no sistema, esse compara com os segmentos de consumo previamente inseridos e enquadra o perfil inserido em um desses estereótipos.
- Dado que o sistema descobriu a que segmento o perfil pertence ele retorna todos os produtos que sejam indicados para o segmento do perfil.

No apêndice A são mostradas as telas do sistema, onde podemos mais facilmente ver as funcionalidades acima descritas.

Para o desenvolvimento da aplicação foi criada uma ontologia de produto genérico e outras ontologias de produtos específicos, que no caso desse trabalho foram livros e CDs. Criamos também uma ontologia de lojas, utilizada para relacionar o produto a loja que o vende. Na seção seguinte falamos a respeito do desenvolvimento dessas ontologias. Após isso falamos com mais detalhes da aplicação implementada, e por fim analisamos os resultados apurados por meio dessa aplicação.

6.1. Ontologia proposta para a aplicação

A ontologia proposta visa permitir que as lojas on-line descrevam seus produtos seguindo essa ontologia, facilitando assim o desenvolvimento de sistemas de comparação de produtos.

A ontologia usada na aplicação como um todo é composta das seguintes partes:

- Uma ontologia que descreve as informações comuns a todas as lojas dos produtos.
- Uma ontologia de alto nível que descreve as informações comuns a todos os produtos. Essa pode ser classificada como uma ontologia de domínio, pois está descrevendo um domínio genérico, que no caso é o domínio de produtos.
- Uma ontologia de que estende a ontologia de alto nível descrevendo as informações específicas de cada produto. As ontologias especificamente desenvolvidas descrevem Livros e CDs. Essas podem ser classificadas como ontologias de aplicação, pois podem ser utilizadas dentro das aplicações específicas, não sendo tão genérica quanto à ontologia de produtos.

A especificação da ontologia foi desenvolvida usando a metodologia METHONTOLOGY mostrada na seção 4.7.3. O principal atrativo dessa metodologia é o fato de ela propor um conjunto bem definido de artefatos que devem ser gerados. Isso se mostrou importante por duas razões: facilitar a especificação precisa da ontologia, e gerar documentação de alto nível, ou seja, não ter somente o código final da ontologia como única documentação. O

apêndice B reúne os documentos gerados na especificação da ontologia desenvolvida para o estudo de caso.

A codificação da ontologia foi feita utilizando a linguagem DAML+OIL. Essa linguagem foi escolhida devido aos seguintes pontos principais:

- A linguagem provê uma capacidade satisfatória de expressar ontologias.
- A linguagem tem algumas primitivas que permitem executar inferências. Isso é interessante, pois assim podemos descobrir novas informações que não estão descritas explicitamente nas instâncias dos produtos.
- Existe uma quantidade razoável de ferramentas que auxiliam o desenvolvimento de ontologias nessa linguagem. Como exemplos de ferramentas que foram utilizadas nesse trabalho nós temos: o ontoEdit (http://www.ontoprise.de/) e o Oiled (http://oiled.man.ac.uk/) que são ambientes de desenvolvimento de ontologias, e JESS (http://herzberg.ca.sandia.gov/jess/) que é uma máquina de inferência que possui uma API para trabalhar com DAML+OIL (DAMLJessKB (Kopena, 2001)).

Inicialmente testamos os dois ambientes para desenvolvimento de ontologias (ontoEdit e Oiled), porém ao final preferimos não utilizar nenhum dos dois. O principal fator que nos fez desistir de utilizá-los foi o fato de a API da máquina de inferência utilizada na aplicação (DAMLKJessKB) não fornecer suporte para muitas das construções geradas por esses ambientes. Essa questão será abordada mais à frente no capítulo sobre máquinas de inferência.

Utilizamos o *framework* 2BuyNet (Fortunato, 1999) como principal fonte de referência para o desenvolvimento da ontologia que descreve os conceitos comuns a todos os produtos, e também a ontologia que descreve os conceitos comuns a todas as lojas. Na seção seguinte falaremos mais sobre esse *framework*.

Para o desenvolvimento das ontologias que descrevem conceitos específicos de livros e CDs foram utilizadas como fontes de referência às descrições dos produtos feitas nas seguintes lojas virtuais:

- Livraria cultura (http://www.livrariacultura.com.br/)
- Amazon (http://www.amazon.com/)
- Fanshop do Paulo Coelho (http://www.fanshop.com.br/paulocoelho/)

- Livraria dirigida (http://www.livrariadirigida.com.br)

Essas duas últimas lojas on-line são instâncias de lojas geradas pelo *framework* 2BuyNet. É importante salientar que o objetivo aqui não é fazer uma descrição muito elaborada desses produtos, e sim uma descrição que permita avaliar a utilidade de ontologias dentro de B2C.

Além da criação das ontologias, criamos também instâncias que seguem as ontologias desenvolvidas, uma vez que nenhuma loja on-line fornece a descrição de seus produtos seguindo uma ontologia.

6.1.1. 2BuyNet (Fortunato, 1999)

O 2BuyNet (Fortunato, 1999) é um gerador de lojas virtuais. Ele é um *framework* para geração de lojas que possibilita o gerenciamento da loja virtual, e a criação da loja propriamente dita.

No gerenciamento da loja é possível mudar o *design* da loja, inserir produtos, etc. Uma vez concluída a modificação da loja, existe a opção de gerar loja, sendo que essa faz a "sincronização" das modificações feitas na loja virtual publicada.

Internamente, o 2BuyNet é composto de um conjunto de *templates* básicos sendo que esses estão cheios de marcadores especiais (*tags*) do 2BuyNet que possuem o seguinte formato "<*2BN_...*>".

Cada vez que é utilizada a opção gerar loja virtual, essas *tags* são substituídas por valores específicos da loja que está sendo criada (por ex.: nome da loja, produtos da loja, etc.). Cada *template* pode gerar mais de uma página HTML. Por exemplo, um *template* para produto pode ser usado para gerar páginas de vários produtos.

Quando se deseja modificar o *design* de uma página, o que se faz é mudar a posição das *tags* do *template* básico.

A figura 22 ilustra o funcionamento do 2BuyNet.

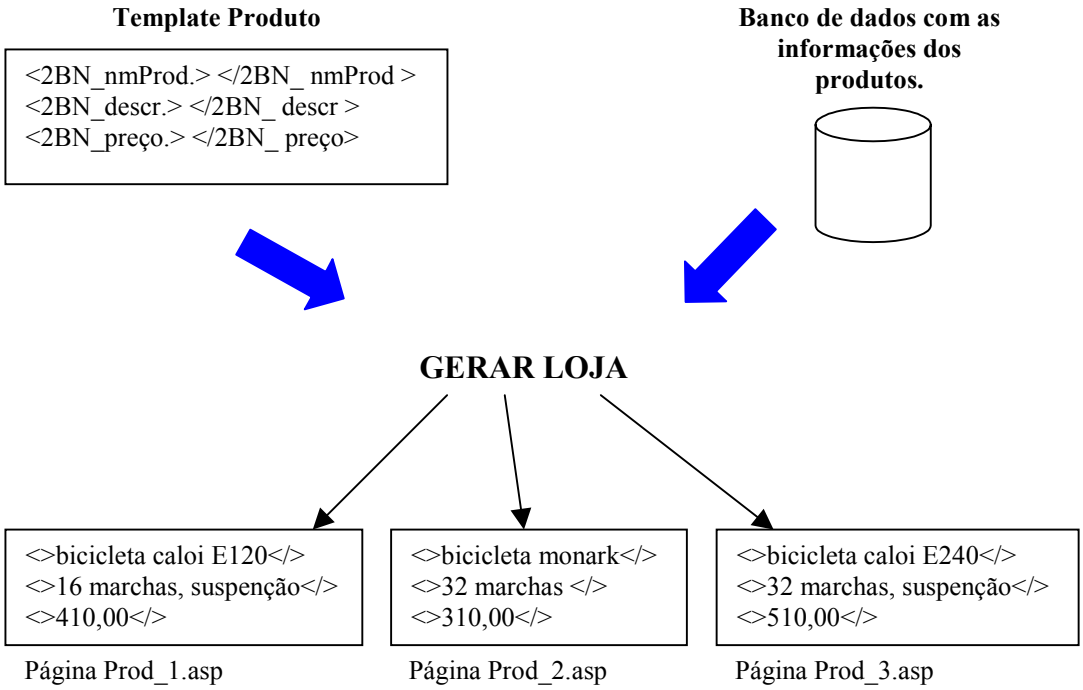

Figura 22 – Funcionamento do 2BuyNet.

Para iniciarmos o desenvolvimento das ontologias utilizadas nesse estudo de caso, utilizamos algumas das informações existentes nas descrições de produtos e lojas do *framework* 2BuyNet.

A descrição dos produtos feita dentro do *framework* é composta por alguns campos comuns a todos os produtos e alguns campos livres para a descrição específica do produto a ser vendido. Entre os campos comuns utilizados temos: o preço e peso dos produtos.

O *framework* também possui informações específicas da loja que vende o produto. Essas informações são: o nome da loja, o endereço da loja, as formas de pagamento aceitas pela loja, a unidade monetária padrão da loja.

Um trabalho futuro interessante a ser desenvolvido a partir desse estudo de caso, é a modificação do *framework* 2BuyNet de forma que ele possa gerar além das páginas HTML que descrevem os produtos, também documentos RDF que descrevem os produtos e suas lojas, conforme alguma ontologia pré-estabelecida.

6.2. Aplicação

A aplicação desenvolvida visa permitir ao usuário fazer buscas e comparações entre produtos provenientes de várias lojas. Além disso, a aplicação também permite ao usuário obter uma lista de produtos indicados a partir do perfil da pessoa que receberá o produto.

As principais tarefas realizadas pela aplicação são as seguintes:

- Carrega as ontologias dos produtos e das lojas dentro da máquina de inferência. Carrega também outras ontologias auxiliares (ontologia de pessoa e ontologia de gênero do produto).
- Carrega as descrições dos produtos de cada uma das lojas dentro da máquina de inferência. Essas descrições são instâncias que seguem as ontologias definidas.
- Uma vez que todas as ontologias e as instâncias foram carregadas dentro da máquina de inferência, a aplicação fica a espera de requisições do usuário. Essas requisições podem ser feitas para realizar buscas de produtos que estejam dentro da máquina de inferência, e também fazer recomendações de produtos que estejam carregados na máquina de inferência, dado um perfil de pessoa.

A figura 23 apresenta a arquitetura da aplicação.

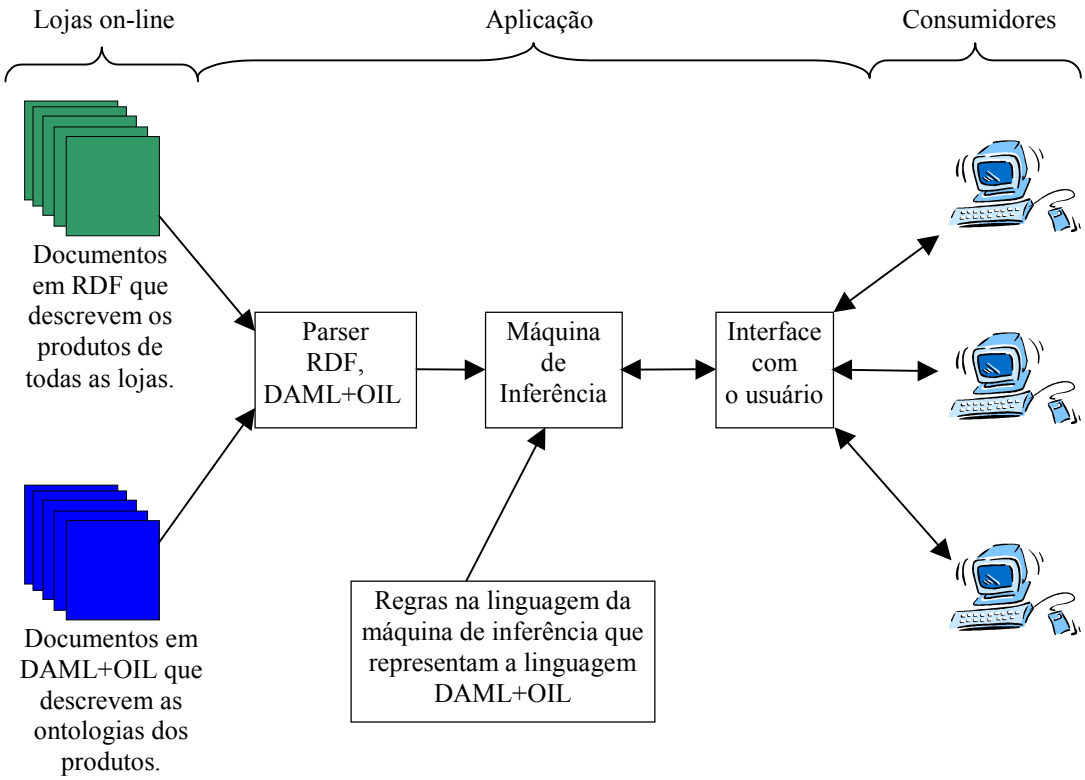

Figura 23 – Arquitetura da aplicação.

A aplicação foi toda desenvolvida utilizando a linguagem Java e quatro pacotes adicionais: o pacote da máquina de inferência JESS, o pacote DAMLJessKB que é utilizada para fazer a interface entre a máquina de inferência e a linguagem DAML+OIL, o pacote RDF-API que é utilizado por DAMLJessKB para fazer o *parser* dos documentos RDF, e por fim o pacote Servlet que é utilizada para o desenvolvimento do servidor.

No apêndice C estão os diagramas de classe que formam a aplicação, enquanto no apêndice D estão os diagramas de seqüência da aplicação.

6.2.1. Máquina de inferência

A máquina de inferência JESS (*Java Expert System Shell*) foi utilizada para desenvolver a implementação desse trabalho. JESS foi desenvolvida por Ernest Friedman-Hill no Sandia National Laboratories como um projeto de pesquisa interno do laboratório. Essa máquina de inferência é de uso livre para a comunidade acadêmica, e ela pode ser obtida no seguinte endereço eletrônico (http://herzberg.ca.sandia.gov/jess/). A versão 6.0 foi utilizada para o desenvolvimento da aplicação. Não é acessível o código fonte da máquina de inferência, sendo somente acessível um arquivo de extensão "*.jar*" que possui a biblioteca da máquina de inferência pré-compilada.

Essa máquina de inferência possui as mesmas funcionalidades básicas providas por uma máquina de inferência como Prolog. A vantagem da utilização de JESS é que, por ser uma biblioteca Java, ela permite que dentro de um programa Java seja executada a máquina de inferência, fornecendo assim uma interface entre a aplicação e a máquina de inferência.

A máquina de inferência sozinha não tem a capacidade de realizar inferências dado somente um conjunto de declarações RDF (ontologias em DAML+OIL e instâncias das ontologias em RDF). Para que isso seja possível é necessário adicionar as regras semânticas (axiomas) de RDF, RDF Schema e DAML+OIL (Fikes & McGuinness, 2001) que fornecem a semântica a essas linguagens. Esses axiomas devem ser escritos na linguagem utilizada pela máquina de inferência.

Além da máquina de inferência, foi utilizado também o pacote DAMLJessKB (Kopena, 2001) desenvolvido para fazer a interface entre JESS e DAML+OIL. Ele adiciona parte da semântica existente em DAML+OIL na máquina de inferência JESS. A versão 010817b foi utilizada para o desenvolvimento da aplicação.

Através desse DAMLJessKb é possível a execução das seguintes tarefas:

- Ler um documento DAML+OIL e colocar as declarações feitas nesse documento dentro da máquina de inferência. DAMLJessKB utiliza para leitura do documento o *parser* RDF-API.
- Adicionar alguns axiomas da linguagem RDF Schema e DAML+OIL na máquina de inferência, permitindo assim a máquina inferir novas declarações dado um conjunto de declarações DAML+OIL. As regras adicionadas por essa API não fornecem todos os axiomas presentes nas linguagens RDF, RDF Schema e DAML+OIL (Fikes & McGuinness, 2001). A lista de axiomas cobertos por DAMLJessKB pode ser obtida em (Kopena, 2001).

A distribuição de DAMLJessKB fornece o código fonte do mesmo, sendo possível assim modificá-lo e recompilá-lo. Algumas pequenas modificações foram feitas na API a fim de adequá-la melhor à máquina de inferência JESS.

O pacote DAMLJessKB carrega os axiomas da linguagem DAML+OIL dentro da máquina de inferência JESS. Isso faz com que a máquina de inferência, ao receber uma declaração RDF que ative um dos seus axiomas, gere outras

declarações RDF, ou seja, a máquina realiza inferências a partir de declarações RDF. A figura 24 exemplifica o processo de inferência utilizando a máquina JESS.

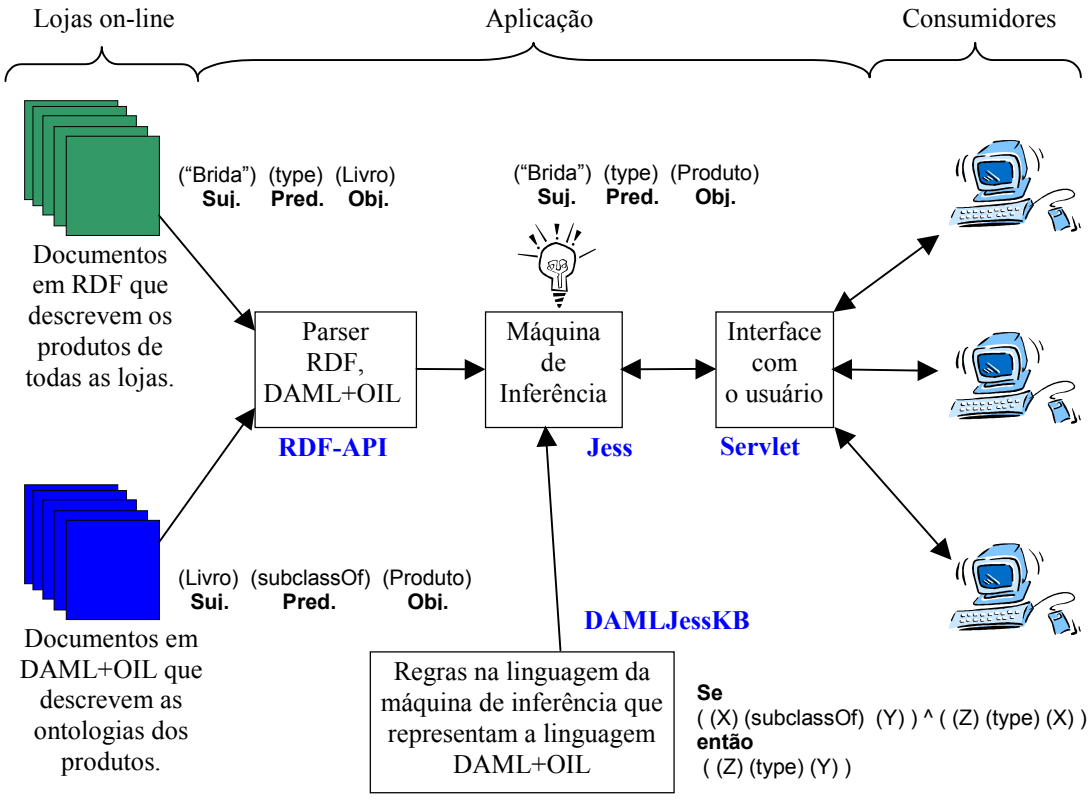

Figura 24 – Exemplo de inferência.

6.2.2. Vantagens obtidas com o uso de ontologias

Para a busca e comparação de informações de produtos entre diversas lojas virtuais, a utilização de ontologias mostrou as seguintes vantagens principais: a possibilidade de padronização da descrição de uma família de produtos (não é exclusivo da solução de ontologias), e a capacidade de fazer inferências sobre as informações que estão de acordo com as ontologias.

Há outros benefícios menores que aparecem com o uso de ontologias, porém esses não são exclusivos desse tipo de solução. São eles: os produtos são descritos de uma forma estruturada, e a separação das *tags* de *layout* da informação útil. Essas duas vantagens advêm do uso de XML e RDF para o suporte na troca de informações.

A seguir falarei com mais detalhes sobre essas principais vantagens.

6.2.2.1. A descrição dos produtos de forma estruturada, e a separação das tags *de* layout *da informação útil*

Muitas lojas *on-line* têm como única forma de fornecer a descrição de seus produtos através de documentos HTML. O código a seguir apresenta um exemplo de documento HTML retirado do *site* da Livraria Dirigida.

```
<html>
<head>
<title>Livraria Dirigida - &lt;2BN_PROD ID='NOME'&gt;&lt;/2BN_PROD&gt;</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" leftmargin=0 topmargin=0 link="#006666" vlink="#009933">
<table width="100%" border="0" cellspacing="0" cellpadding="2">
 <tr bgcolor="#D3A87D">
  <td>
   <center>
    <font face="Verdana, Arial, Helvetica, sans-serif" size="3"><font color="#000000"><b>Estação Carandiru</b></font></font>
   </center>
  </td>
 </tr>
 <tr>
  <td>
   <center>
    <img src="produtos/28078.jpg" width="145" height="190" vspace="4">
   </center>
  </td>
 </tr>
 <tr>
  <td>
   <center>
    <font face="Arial, Helvetica, sans-serif" size="2">
    </font>
   </center>
   <center>
    <font face="Arial, Helvetica, sans-serif" size="2">
     <table border="0" cellspacing="0" cellpadding="0">
      <tr>
       <td>
        <font face="Arial, Helvetica, sans-serif" size="2">
         O médico Drauzio Varella relata dez anos de atendimento voluntário na Casa de Detenção de São Paulo, o maior presídio do Brasil, e mostra como um código penal não-escrito organiza o comportamento da população carcerária.
<br><br>
        </font>
        <FONT face="Arial, Helvetica, sans-serif" size=2 color=#000000>Número de páginas: 368
         <BR>Nosso Preço:
        </FONT>
        <FONT face="Arial, Helvetica, sans-serif" size=2 color=#FF0000>R$28.50</FONT>
        <BR> 
       </td>
      </tr>
     </table>
    </font>
    <table cellspacing="0" cellpadding="0">
     <tr>
      <td valign="MIDDLE" bgcolor="#B2E8C7">
       <a href=javascript:window.close()>
        <center>
         <IMG src="imagens/botao_fecharjanela.gif" border="0">
        </center>
       </a>
      </td>
     </tr>
    </table>
   </center>
  </td>
 </tr>
</table>
```

```
</body>
</html>
```

Código 17 – Documento HTML que descreve um livro da Livraria Dirigida.

Figura 25 - Código 17 apresentado pelo Internet Explorer.

Conforme discutido anteriormente, documentos HTML têm duas características extremamente negativas no que se refere à extração de informações por máquinas:

- A informação em um documento HTML é estruturada de forma a facilitar o trabalho do *browser* para efetuar a apresentação da mesma. Dessa forma documentos HTML que descrevem livros, carros ou brinquedos terão a mesma estrutura. Podemos ver no código 17 que não há nenhuma estrutura específica na descrição da informação do livro. Sendo assim se um agente de software desejar obter o preço do livro, esse teria que utilizar alguma técnica de extração de informações em linguagem natural (linguagem não estruturada) para conseguir essa informação.
- Há uma mistura de *tags* de *layout* utilizadas na apresentação do documento HTML pelo *browser* com o conteúdo propriamente dito, que no nosso caso são informações sobre produtos. Isso dificulta muito aos agentes de software extraírem informação, pois essas *tags* "poluem" o documento. Qualquer mudança no *layout* das páginas exigirá que o agente que extrai o conteúdo dessas seja modificado.

Com a utilização de ontologias para a descrição dos produtos, o problema de falta de estrutura nas instâncias desaparece, pois as ontologias fornecem uma estrutura para todas as instâncias de produtos que estão de acordo com ela. Isso porém não é obtido unicamente com o uso dessa técnica. Utilizando técnicas mais simples pode ser obtido o mesmo ganho. Por exemplo, o uso da linguagem XML com DTD ou XML Schema, que não são linguagem para descrição de ontologias, fornecem o mesmo tipo de resultado.

Já o problema de mistura de *tags* de *layout* com a informação útil não é necessariamente resolvido com o uso de ontologia. Isso dependerá da linguagem específica que é utilizada para criação de ontologias e suas instâncias.

A linguagem SHOE melhora o problema, pois a descrição do produto em SHOE se encontra separada da descrição HTML do produto. Porém essa descrição SHOE ainda está vinculada a um documento HTML, pois a descrição do produto em SHOE se encontra no mesmo documento que a descrição HTML do produto, porém não de forma intercalada.

Já em linguagens como RDF Schema e DAML+OIL há uma separação completa entre os documentos HTML e documentos de ontologias. Isso se deve principalmente ao uso de XML em camadas inferiores na pilha de padrões utilizados: **XML** => RDF => RDF Schema => DAML+OIL.

O código a seguir apresenta a mesma descrição de livro apresentada acima (código 17) em linguagem RDF; a descrição da ontologia do produto livro está na linguagem DAML+OIL.

```
<?xml version="1.0"  encoding='ISO-8859-1'?>
<rdf:RDF
        xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:plvr="http://usuarios.uninet.com.br/~chicao/ontologiaLivro.daml#">

        <plvr:Livro rdf:ID="http://www.livrariadirigida.com.br/DEP28000.asp#28005">
                <plvr:autor>DRAUZIO VARELLA</plvr:autor>
                <plvr:titulo>ESTAÇAO CARANDIRU</plvr:titulo>
                <plvr:editora>COMPANHIA DAS LETRAS</plvr:editora>
                <plvr:numeroPaginas>368</plvr:numeroPaginas>
                <plvr:idioma>PORTUGUES</plvr:idioma>
                <plvr:ISBN>8571648972</plvr:ISBN>
                <plvr:generoLiterario  rdf:resource="http://usuarios.uninet.com.br/~chicao/ontologiaGeneroLiterario.
daml#generoLiteraturaBrasileira"/>
                <plvr:generoLiterario  rdf:resource="http://usuarios.uninet.com.br/~chicao/ontologiaGeneroLiterario.
daml#generoContosCronicas"/>
                <plvr:preco>28.50</plvr:preco>
                <plvr:loja rdf:resource="http://www.livrariadirigida.com.br"/>
        </plvr:Livro>
</rdf:RDF>
```

Código 18 – Documento RDF que descreve o livro da Livraria Dirigida.

6.2.2.2. Padronização da descrição de uma família de produtos

A descrição de um mesmo tipo de produto pode variar muito entre as diversas lojas on-line. Por exemplo, no campo de livros, o *site* da Amazon apresenta uma descrição muito mais completa do que os outros *sites* utilizados no trabalho (Livraria Dirigida, FanShop do Paulo Coelho, Livraria Cultura). Essa variação da descrição dificulta muito o trabalho de comparação entre produtos de mais de um *site*, pois além de não existir uma padronização nos nomes dos atributos, a falta de atributos chave em uma descrição pode ser problemática (por exemplo: a falta do atributo ISBN na descrição de um livro).

Com a criação de uma ontologia para descrição de um tipo de produto, todas as instâncias de produtos que sejam desse tipo devem seguir o formato estipulado. Caso um tipo de produto seja descrito em duas lojas com ontologias diferentes, pode-se fazer um mapeamento entre as ontologias criando uma ontologia de mapeamento. A figura 26 ilustra a questão.

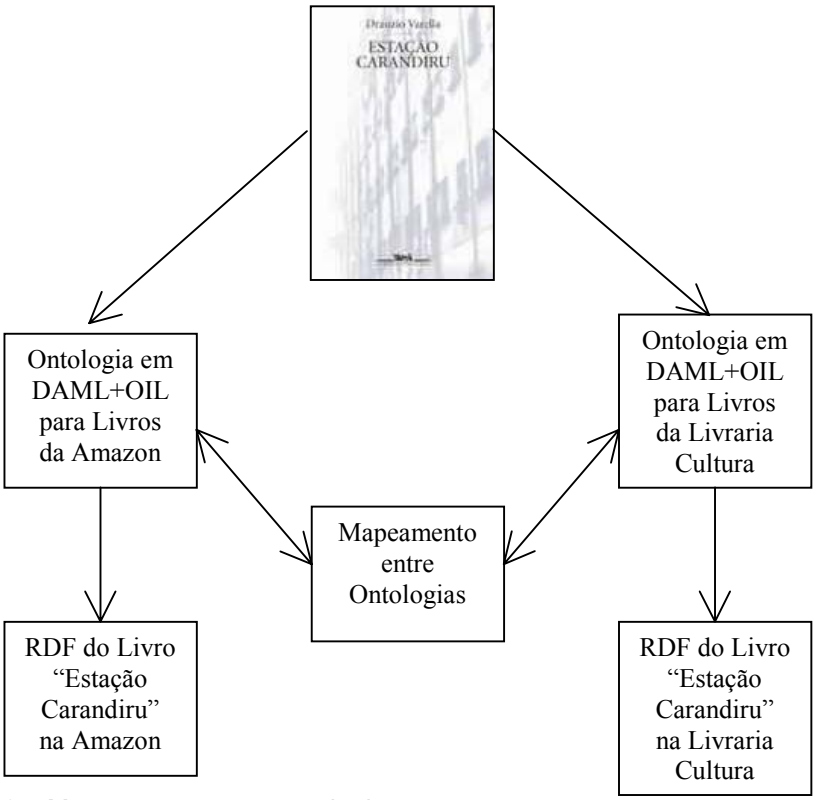

Figura 26 – Mapeamento entre ontologias.

A ontologia de mapeamento faz o relacionamento entre ontologias diferentes que descrevem uma mesma conceitualização. Uma ontologia de mapeamento em DAML+OIL fará a equivalência entre classes e relacionamentos existentes nas duas ontologias. O vocabulário de DAML+OIL fornece primitivas para essas relações de equivalência (*daml:sameClassAs*, *daml:samePropertyAs*, *daml:equivalentTo*).

Apesar de existir a capacidade de fazer mapeamentos entre ontologias, nem sempre é fácil conseguir o mesmo. Em alguns casos esse mapeamento pode ser impossível, pois as ontologias partem de conceitualizações diferentes (não adianta tentar mapear uma maçã em uma laranja).

No trabalho implementado foi desenvolvida uma ontologia básica de produtos que é estendida para os vários produtos específicos (Livros e CDs). As lojas ao descreverem seus produtos utilizam a ontologia específica do produto que fornece. Isso permite que o programa dê o mesmo tratamento para as informações provenientes dos diversos *sites*, pois elas seguem a mesma ontologia.

Esse tipo de solução poderia, até certo ponto, ser obtida com o uso de DTD ou XML Schema, pois seria possível padronizar descrições de livros usando essas linguagens. Porém a incapacidade de utilizar herança em DTD e XML Schema faz

com que a solução com ontologia tenha um melhor resultado. Por exemplo, se quisermos definir que a classe livro é subclasse da classe produto, não poderíamos fazer isso em nenhuma das duas linguagens. Já em linguagens que definem ontologias (RDF Schema, SHOE e DAML+OIL), isso é facilmente conseguido.

6.2.2.3. A capacidade de fazer inferências sobre as informações que seguem a ontologia

A vantagem mais importante do uso de ontologia está na possibilidade de conseguirmos fazer inferências sobre as informações que estão de acordo com a mesma, pois nela além da estrutura da informação também existe a semântica dessa informação.

No trabalho implementado, a utilização de ontologias associadas a uma máquina de inferência permitiu obter as seguintes vantagens:

- Como as classes de *Livro* e *CD* são subclasses de *Produto*, ao carregar as instâncias de *Livro* e *CD* na máquina de inferência, ela automaticamente infere que tanto os livros como os CDs são instâncias também da classe *Produto*. Dessa forma, é possível obter da máquina de inferência quais os produtos que têm preço menor que um determinado valor, pois ela sabe que tanto *Livro* como *CD* são também produtos. A figura 27 ilustra o relacionamento entre as classes *Livro*, *CD* e *Produto*.
- A propriedade *ehRelacionado* que a ontologia define para a classe de produtos permite que se diga que um produto é relacionado a outro produto, ou seja, possui afinidade com outro produto. Nessa propriedade, se um produto X tem como produto afim o produto Y e o produto Y tem como afim o produto Z, então o produto X também tem como afim o produto Z. Essa relação indica uma relação semântica de transitividade da propriedade. Uma vez definido isso na ontologia, pode-se obter diretamente da máquina de inferência essa informação sem ter que caminhar por todos os produtos relacionados. O código 19 mostra a definição da propriedade *ehRelacionado* na ontologia de produto.

- A classe *Pessoa* possui várias restrições que indicam a que segmento uma pessoa pertence dadas suas informações. Isso permite que a máquina de inferência retorne em que segmentos o perfil dessa pessoa se encontra. O código 20 mostra a definição da classe *Pessoa* e das classes *Homem* e *Mulher* que são restrições dessa classe.

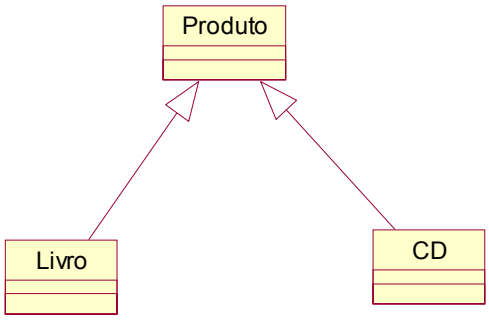

Figura 27 – Relacionamento entre livro, CD e produto.

```
<!-- Define o Slot (Atributo) ehRelacionado -->
<daml:ObjectProperty rdf:ID="ehRelacionado">
        <rdfs:comment xml:lang="pt">Indica relacionamento (afinidade) entre produtos.</rdfs:comment>
        <rdfs:label xml:lang="pt">Eh relacionado</rdfs:label>
        <rdfs:domain rdf:resource = "Produto"/>
        <rdfs:range rdf:resource = "Produto"/>
</daml:ObjectProperty>
<daml:TransitiveProperty rdf:about="ehRelacionado"/>
```

Código 19 – Definição da propriedade ehRelacionaodo na ontologia de produto.

```
<!-- Define a classe Pessoa -->
<daml:Class rdf:ID="Pessoa">
        <rdfs:label>Pessoa</rdfs:label>
</daml:Class>

<!-- Define a restricao Mulher -->
<daml:Class rdf:ID="Mulher">
        <rdfs:label>Mulher</rdfs:label>
        <daml:sameClassAs>
                <daml:Restriction>
                        <daml:onProperty rdf:resource="sexoPessoa"/>
                        <daml:hasValue>feminino</daml:hasValue>
                </daml:Restriction>
        </daml:sameClassAs>
</daml:Class>

<!-- Define a restricao Homem -->
<daml:Class rdf:ID="Homem">
        <rdfs:label>Homem</rdfs:label>
        <daml:sameClassAs>
                <daml:Restriction>
                        <daml:onProperty rdf:resource="sexoPessoa"/>
                        <daml:hasValue>masculino</daml:hasValue>
                </daml:Restriction>
        </daml:sameClassAs>
</daml:Class>
```

Código 20 – Documento RDF que descreve o livro da Livraria Dirigida.

6.2.3. Dificuldades encontradas

Alguns problemas foram constatados durante o desenvolvimento desse estudo de caso. A seguir mencionamos os principais.

6.2.3.1. Desempenho da aplicação

Uma característica observada na aplicação é a lentidão do processo de carga das ontologias dentro da máquina de inferência, e também a lentidão do processo de resposta da máquina de inferência a uma pergunta requerida pela aplicação.

O problema do tempo de resposta da máquina de inferência a uma pergunta requerida pela aplicação foi resolvido através da criação de uma cópia dos fatos (triplas RDF) existentes dentro da máquina de inferência para uma classe dentro da aplicação responsável por fazer a interface entre a aplicação e a máquina de inferência (classe *BSC_BaseConhecimento*). Uma vez que todas as declarações RDF são carregadas dentro da máquina de inferência, e todas as inferências são realizadas pela mesma, é feita uma cópia de todas as declarações RDF existentes na máquina de inferência (declarações RDF explícitas + declarações RDF inferidas) para uma estrutura dentro da classe *BSC_BaseConhecimento*. Dessa forma, conseguimos melhorar o tempo de resposta. A estrutura é formada por um vetor e duas *hashtables*.

O problema do tempo de carga das ontologias dentro da máquina de inferência não foi grande para a nossa aplicação devido à pequena quantidade de instâncias de produtos descritos para teste (38 livros e 4 CDs), porém para um número maior de instâncias isso pode ser um problema mais sério. Uma forma de minimizar esse problema é fazer uma mudança na arquitetura da aplicação.

Atualmente a arquitetura da aplicação é centralizada, ou seja, todas as instâncias de produtos das diversas lojas são carregadas em uma única máquina de inferência (figura 23). Uma alternativa seria uma arquitetura distribuída, onde podemos ter uma máquina de inferência para cada loja on-line. Para responder uma pergunta nessa arquitetura, o agente de compra teria que enviar a pergunta para máquina de inferência de cada loja on-line e após isso combinar as respostas. Como as várias máquinas de inferência possuem as mesmas ontologias de produtos, a combinação dessas respostas é fácil, pois todas seguem a mesma ontologia. A figura 28 apresenta a arquitetura proposta.

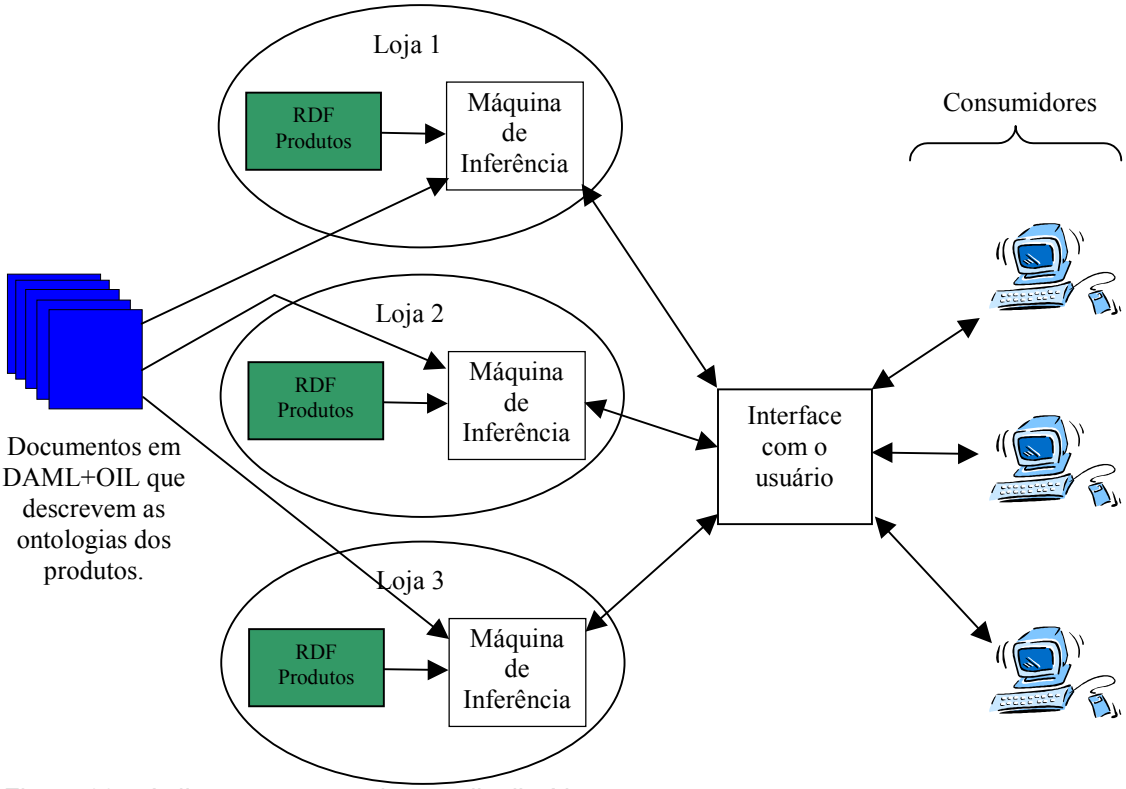

Figura 28 – Aplicação com arquitetura distribuída.

6.2.3.2. Expressividade da máquina de inferência

Apesar de a linguagem DAML+OIL possuir um vocabulário básico que fornece uma capacidade muito boa de expressão de ontologias, conforme pôde ser visto na seção 5.6, não podemos contar com toda essa capacidade devido a restrições da ferramenta utilizada na implementação.

O pacote DAMLJessKB não fornece todos os axiomas das linguagens RDF Schema e DAML+OIL (Fikes & McGuinness, 2001). Dessa forma, ficamos limitados a conseguir fazer inferências somente com os axiomas presentes no pacote, ou seja, nem toda a semântica presente no vocabulário básico de DAML+OIL é coberta pelos axiomas do pacote. Os axiomas cobertos podem ser vistos em (Kopena, 2001). Isso porém não nos impede de utilizar todo o vocabulário básico de DAML+OIL para descrever as ontologias.

As construções do vocabulário DAML+OIL que não são cobertas pelo pacote são colocadas na máquina de inferência da mesma forma que as construções que são cobertas. A única diferença entre a construção coberta e a não coberta é que a coberta possui axiomas que podem gerar inferências conforme a

semântica da construção, enquanto a construção que não é coberta não pode gerar inferências.

Algumas das construções utilizadas na descrição das ontologias que não são cobertas pelo pacote são: *rdfs:domain*, *rdfs:range*, *daml:oneOf*, *daml:cardinality*. Porém, há outras construções que foram utilizadas na descrição das ontologias que geram inferências importantes para a aplicação: *daml:subClassOf*, *daml:TransitiveProperty*, *rdfs:subPropertyOf*, *daml:hasValue*.

Para resolver essa questão é necessário adicionar os demais axiomas das linguagens RDF Schema e DAML+OIL (Fikes & McGuinness, 2001) no pacote DAMLJessKB. Recentemente uma nova versão mais abrangente desse pacote foi lançada, sendo que essa nova versão cobre mais construções das linguagens.