RDF na Interoperabilidade entre Domínios na Web

Domingos S. A. Santos e Ulrich Schiel
Universidade Federal de Campina Grande (UFCG)
Caixa Postal 10.106 – CEP 58.109-970 - Campina Grande – PB
domingos@netdados.com.br, ulrich@dsc.ufpb.br

Abstract

This work is a study of the applicability of the Resource Description Framework (RDF) on the interoperability of data between different domains. The specific objective is to present a strategy for the application of the RDF, demonstrating it through a case study. In this work, the interoperability of two domains (Classified Ads and Notarial Services) is demonstrated through web applications.

Resumo

Este trabalho trata da aplicabilidade da tecnologia Resource Description Framework – RDF na interoperabilidade entre diferentes domínios. O objetivo específico é apresentar uma estratégia para promover a interoperabilidade entre domínios na Web. Sua aplicabilidade é demonstrada por um estudo de caso com dois domínios: Anúncios Classificados e Serviços de Cartórios.

1. Introdução

O resultado da pesquisa da *ACNielsen eRatings.com* (<www.eratings.com>), referente ao quarto trimestre de 2001, demonstra a existência de 498 milhões de pessoas com acesso à internet em residências no mundo. Também observa um crescimento de 24% entre o terceiro e quarto semestre do mesmo ano. Com esta penetração na sociedade, evidencia-se uma demanda crescente por serviços na Web. Este processo tende a impulsionar a oferta de aplicações Web, provocando o desenvolvimento de tecnologias que atendam às necessidades de usuários, desenvolvedores e infraestrutura existente.

Uma destas tecnologias, *Resource Description Framework (RDF),* permite a identificação de conteúdo semântico em documentos na Web. Além desta característica, RDF possibilita a interoperabilidade entre domínios através da adoção de esquemas de dados públicos. Portanto, é uma tecnologia que pode aperfeiçoar os serviços disponíveis na Web e pode demandar outros derivados de sua utilização.

Baseando-se no cenário descrito, este artigo discute a aplicação do *Resource Description Framework* à interoperabilidade de domínios na Web. Apresenta-se uma estratégia para sua aplicação, considerando os recursos de software e os padrões para a Web disponíveis. Por fim, um estudo de caso ilustra sua aplicabilidade.

2. Contextualização

A Web é um ambiente composto por elementos das mais diversas tecnologias e, devido a sua abrangência universal e dependência da infraestrutura de hardware, requer que seus componentes de software sejam modulares. Neste contexto, a colaboração entre seus elementos é vital para que o conjunto produza os resultados esperados. Portanto, A Web é um ambiente em que a interoperabilidade é uma característica chave para maximizar os recursos disponíveis. Nessse ambiente, o desafio é vencer as barreiras da estrutura, sintaxe e semântica entre as fontes de informação envolvidas no processo. Assim, quanto mais abrangentes forem as tecnologias empregadas, mais recursos estarão disponíveis aos componentes envolvidos na solução de interoperabilidade.

Em [13], é feita uma analise do evento *Interoperability Summit,* sintetizada na frase: "*Interoperate or Evaporate*". Isto exemplifica o fato de que já não se discute mais a necessidade ou não da interoperabilidade. Discute-se agora como aprimorar os mecanismos de interoperabilidade a fim de que estes colaborem entre si e convirjam para uma padronização única.

A *World Wide Web* - WWW - foi inicialmente projetada visando exclusivamente à interação homem-dados [4]. Atendendo a esta finalidade, padronizou-se a linguagem HTML (*Hypertext Markup Language*). Porém, com sua ampla utilização e o aumento em grande escala do conteúdo disponível na Web, sentiu-se a necessidade da informação também ser processada pela máquina, a fim de que a interação homem-dados fosse facilitada e otimizada.

Para solucionar o problema, diversas soluções foram propostas e cada uma atendia a uma necessidade específica. Um importante fato desta fase foi a padronização da linguagem XML (eXtensible Markup Language), que permite aos usuários a definição de marcas para indicar estrutura no conteúdo [1]. Baseando-se nos resultados obtidos e na experiência com o armazenamento e consulta de dados convencionais, constatou-se que a utilização de metadados na Web era uma das melhores alternativas para possibilitar o processamento inteligente do conteúdo disponibilizado, pois metadados podem ser extensíveis e distribuídos na rede. O *Resource Description Framework* (RDF) é uma destas soluções baseadas em metadados que aplica a linguagem XML para tal fim.

A padronização RDF estabelece um modelo e sintaxe para representar, codificar e transmitir metadados, com o objetivo de maximizar a interoperabilidade de dados de fontes heterogêneas na WEB [4]. Outro objetivo é tornar possível a especificação de semântica para base de dados em XML.

O modelo de dados do *Resource Description Framework* é caracterizado por prever três tipos de elementos [4]:

- **Recursos** – tudo descrito por expressões RDF e identificado por um URI (*Uniform Resource Identifier*). Recursos podem ser páginas Web ou partes dela, a exemplo de figuras;
- **Propriedades** – são aspectos específicos, características, atribuitos ou relações utilizadas para descrever recursos;
- **Sentenças** – informação estruturada composta de **sujeito** (recurso), **predicado** (propriedade) e **objeto** (valor da propriedade) e descrita nesta ordem. O objeto pode ser outro recurso ou um dado primitivo como uma *string*.

Como exemplo, podemos citar a sentença "**a figura quadro3.jpg foi criada por Aloysio Novis**". Expressando-a através da sintaxe RDF e utilizando o namespace "dc" (Dublin Core Element Set), obtém-se o documento da figura 1.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.estacio.br/universidarte/galeria/quadro3.jpg">
    <dc:creator>Aloysio Novis</dc:creator>
  </rdf:Description>
</rdf:RDF>
```

Figura 1 - Sentença "a figura quadro3.jpg foi criada por Aloysio Novis" expressa na sintaxe RDF.

Graficamente, a mesma sentença pode ser expressa através de arcos (propriedades) e nós (recursos ou objetos), como na figura 2.

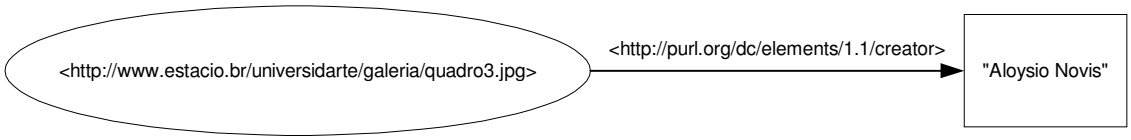

Figura 2. Sentença da figura 1 expressa graficamente.

Alternativamente, há outro meio de expressar sentenças RDF - Notation 3 (N3) - que consiste em listar os três elementos na ordem: sujeito, predicado e objeto (tríplas RDF).

A padronização RDF pelo W3C (World Wide Web Consortium) é dividida em duas especificações. A primeira dedica-se ao modelo e sintaxe RDF e utiliza-se do prefixo 'rdf' para referir-se ao espaço de nomes associado a esta padronização, identificado pela URI <http://www.w3.org/1999/02/22-rdf-syntax-ns#> e documentada na URL <www.w3.org/TR/REC-rdf-syntax/>. A segunda dedica-se a padronização de esquemas e normatiza como o RDF deve ser usado para a descrição de vocabulários. Também define um vocabulário básico para este propósito, bem como um mecanismo que permite a

extensibilidade do modelo. Este vocabulário define recursos e propriedades em um sistema tipado e é definido em um espaço de nomes denominado 'rdfs', identificado pelo URI <http://www.w3.org/2000/01/rdf-schema#> e documentada na URL <www.w3.org/TR/rdf-schema/>.

A escolha da linguagem RDF no processo de interoperabilidade entre domínios na Web, objeto deste trabalho, foi motivada pelos fatores descritos a seguir:

- é baseada na linguagem XML, padrão praticamente consolidado no mercado de soluções para a Web;
- sua aplicação independe de plataforma, fornecedor, linguagem de programação ou qualquer tecnologia proprietária;
- é elemento-chave na arquitetura proposta para a Web Semântica, vista por muitos pesquisadores como a "Web do Futuro";
- os recursos de sua especificação atendem desde a interoperabilidade sintática até a semântica, sendo mais adequado à interoperabilidade estrutural.
- é extensível, podendo utilizar-se de outras, através dos respectivos espaços de nomes, como XML *Schema* (<www.w3.org/XML/Schema>) e DAML+OIL (<www.w3.org/TR/daml+oil-reference>).

3. Trabalhos relacionados

Sobre a aplicação do Resource Description Framework, há várias abordagens em campos específicos de aplicação. Em [14], apresenta uma proposta para integração de informações em ambientes científicos com RDF. Difere do presente trabalho pelo fato de estar centrado na integração de dados e não na interoperabilidade destes, além da aplicabilidade restringir-se a aplicações científicas. Em [16], também é proposta uma arquitetura de integração com RDF, mas muito complexa e com contexto de aplicação o Projeto HERA [12].

Outros trabalhos são mais teóricos e genéricos, embora sejam bem mais direcionados à aplicabilidade do RDF. Um deles [11], apresenta uma visão geral de como RDF pode ser aplicado para a construção da Web Semântica, mas não define uma estratégia sistematizada. Em [9], discute-se sobre as tecnologias de suporte a ontologias e domínios de conhecimento na Web Semântica, como UML, XML, RDF e Java, de forma genérica, sem vinculação com aplicações reais. Em um trabalho mais prático [2], é feito um estudo de caso com RDF para a manipulação de dados em base relacional, caso mais comum nas aplicações comerciais, porém não centrado em um processo de interoperabilidade .

A definição de esquemas é outro ponto fundamental do assunto discutido. Neste aspecto, [6] traça uma estratégia para a geração de esquemas XML através da UML, embora não trate de esquemas RDF. Nesta mesma linha, este artigo apresenta estratégias semelhantes especificamente indicadas para esquemas RDF.

4. Uma estratégia prática para a interoperabilidade de domínios na Web com RDF

A fim de sistematizar as etapas envolvidas na preparação do ambiente de interoperabilidade, apresentamos uma estratégia composta de quatro fases: atualização do modelo dos domínios, definição ou escolha do esquema, atualização ou definição da base de dados e implementação do processo de interoperabilidade.

4.1. A atualização do modelo

Sob o ponto de vista da modelagem e requisitos do negócio, possíveis situações do ambiente de início do projeto devem ser analisadas a fim de que o planejamento das fases seja efetuado com o aproveitamento dos recursos existentes. Um ambiente prévio de um projeto de interoperabilidade pode ser caracterizado por:

a) existir suporte a aplicações Web;

b) existir aplicações sem suporte a Web;

c) inexistência de aplicações.

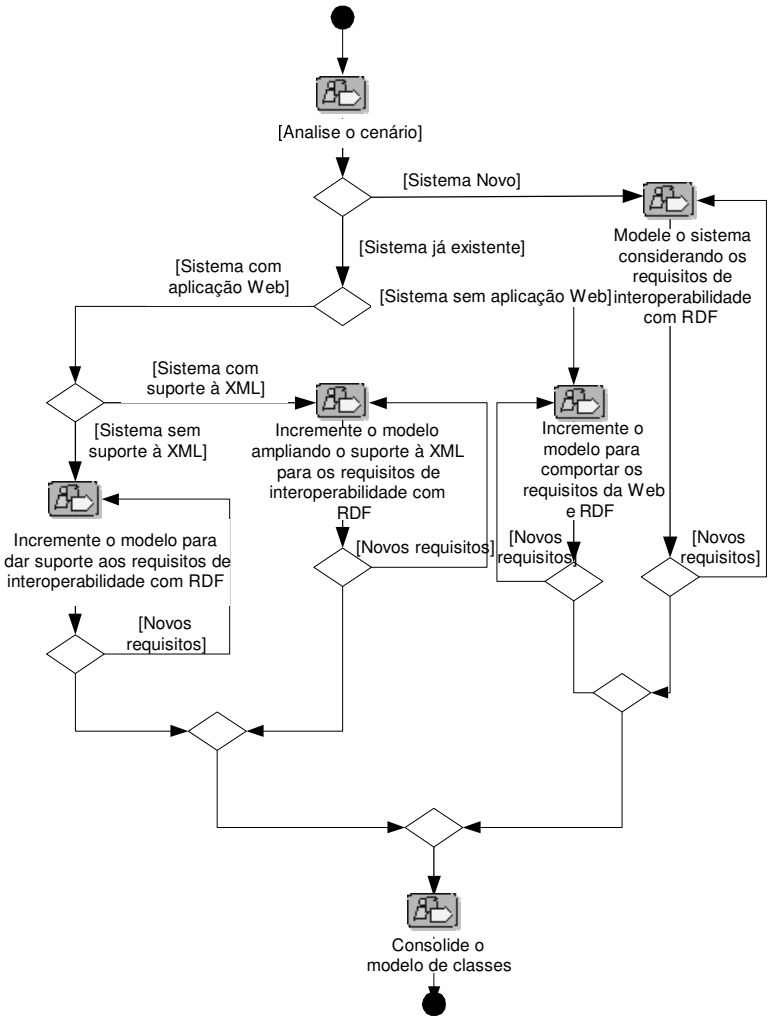

Figura 3 – Diagrama de atividades relacionadas à atualização do modelo

O primeiro ambiente, item "a", pode ter suporte à linguagem XML ou não. No primeiro caso, a implementação de interoperabilidade através do RDF é facilitada pela similaridade de conceitos entre estas tecnologias, pois o RDF é uma aplicação XML. Neste caso, as iterações adicionais do processo de desenvolvimento terão como objetivo básico promover a migração de esquemas e aplicações XML para o RDF. No segundo caso, as iterações adicionais terão a finalidade de acrescentar ao sistema o completo suporte ao Resource Description Framework.

Quanto ao item "b", as iterações adicionais devem acrescentar ao sistema o suporte à Web e ao Resource Description Framework conjuntamente.

Para o caso do início de um projeto totalmente novo, item “c”, em todo o processo de desenvolvimento as atividades deverão estar integradas para atender ao conjunto de requisitos: aplicação com funcionalidades dentro e for a da Web com interoparbilidade via RDF.

A figura 3 ilustra as alternativas de atividades inerentes ao aspecto aqui abordado.

4.2. Definição ou escolha do esquema RDF

A partir do modelo de classes atualizado do sistema, pode-se iniciar a definição do esquema RDF do domínio. Esta seqüência é conveniente porque, com a modelagem das classes, já se dispõe de um pseudo-esquema dos dados, restando apenas seu correto mapeamento para o RDF, caso não exista um esquema RDF adequado para ser reutilizado no domínio da aplicação.

A seqüência citada no parágrafo anterior, ou seja, atualizar o modelo de classes e posteriormente definir ou escolher o esquema RDF, pode não ser conveniente quando o objetivo é adotar o esquema como padrão para o domínio considerado. Nesse tipo de esquema, a modelagem deve ser feita para todo o conjunto de funcionalidades possíveis no domínio, pois o esquema deve ser o mais genérico possível. Portanto, para este caso específico, a definição ou escolha do esquema deve ser feita simultaneamente com a modelagem, pois o modelo do sistema será um subconjunto do modelo do domínio.

O processo inicia-se com a pesquisa de esquemas já existentes a fim de que se possa reutilizá-los no domínio. A busca deve ser feita em repositórios públicos e não se deve limitar aos definidos para RDF. Outras linguagens podem atender à necessidade do projeto promovendo-se as conversões necessárias (DTD, XML Schema, DAML, OIL, dentre outras).

A seguir, dois caminhos podem ser seguidos: O primeiro refere-se à possibilidade de não existir esquema do domínio. Neste caso é necessário criar o esquema diretamente a partir do modelo de classes já obtido. Para isto, mapeia-se os elementos do modelo UML para RDF ou XML, conforme discutido em [3, 6, 7, 9]. O segundo caminho refere-se à possibilidade de existirem esquemas aplicáveis ao domínio. Nesta alternativa, há duas possíveis situações. A primeira é a existência de um esquema na própria sintaxe RDF e, neste caso, o processo é facilitado, pois somente é necessário utiliza-lo na representação das instâncias de dados do sistema. A segunda é a existência de esquemas em outras especificações, como XML Schema e DTD. Neste caso, é preciso converter o esquema para RDF, baseando-se nas semelhanças estruturais entre eles, conforme os conceitos e métodos descritos em [6, 3, 8, 10].

No processo de definição/construção do esquema para o domínio, é conveniente analisar a possibilidade de combinar outras linguagens (XML Schema, DAML, OIL, etc) a fim de que os requisitos da modelagem do sistema sejam atendidos. Caso essa deficiência seja atendida por outra linguagem, é conveniente adotá-la, incorporando-a ao esquema com o respectivo espaço de nomes, pois a especificação do esquema RDF é extensível.

Definido o esquema, é preciso submetê-lo a um serviço de validação. Com isto, garante-se a consistência do esquema com a sintaxe RDF e, por conseqüência, com o componente que manipula o modelo (API, middleware ou SGBD com suporte nativo a RDF). Após esta etapa, o esquema pode ser publicado e referenciado através do espaço de nomes e utilizado para representar suas instâncias. Esta representação também deve ser validada, pois também está relacionada com os serviços da API RDF ou mecanismo equivalente utilizado pela aplicação para manipular o modelos de dados RDF. Com isto, o sistema está apto a interagir com os dados, conforme é detalhado na próxima seção.

A figura 4 ilustra o conjunto de atividades analisadas nesta seção.

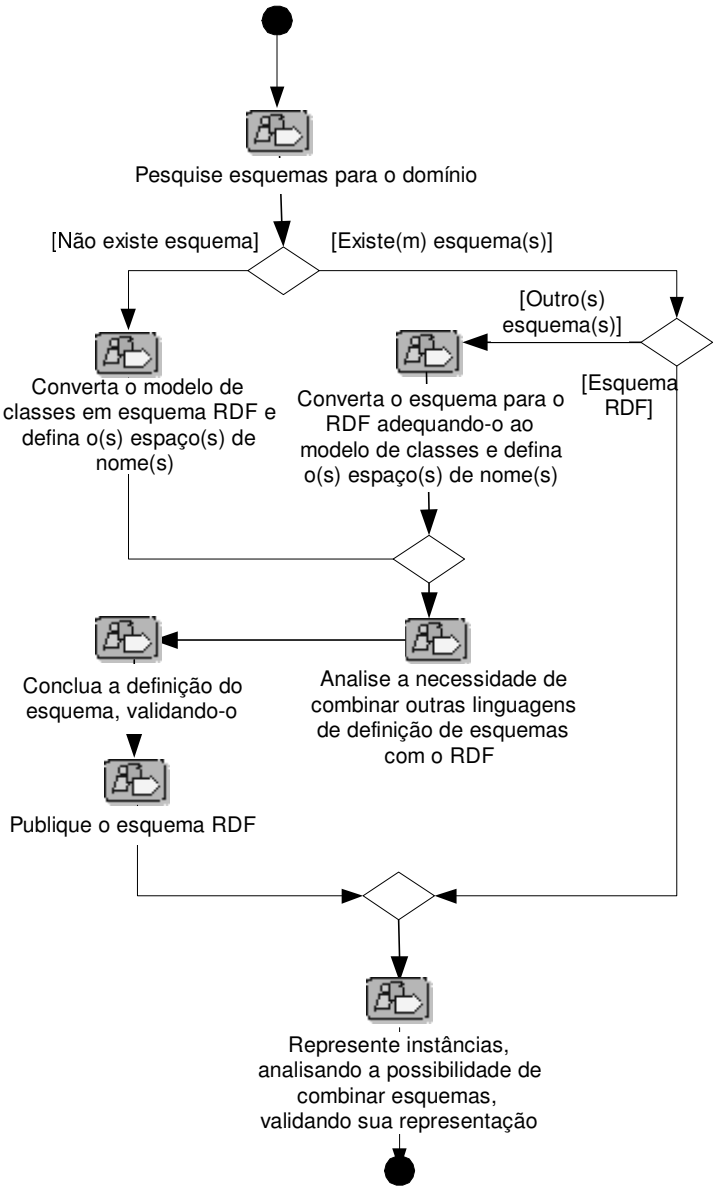

Figura 4 – Diagrama de atividades relacionadas à definição do esquema

4.3. Atualização ou definição da base de dados

Sob o ponto de vista da base de dados, duas situações iniciais possíveis podem ser analisadas ao iniciar um projeto. Uma delas é a existência prévia de um SGBD em uso. A outra, trata-se do início de um sistema novo em que um SGBD deve ser escolhido. Cada situação tem suas peculiaridades que serão analisadas a seguir.

Para a situação do SGBD já existente, caso mais real, provavelmente não há suporte nativo ao RDF. Neste caso, seria útil um middleware RDF para atuar entre o SGBD e as aplicações. Isto traz algumas vantagens: libera tanto a aplicação quanto o SGBD da manipulação do modelo de dados RDF; as atualizações na aplicação e no SGBD não afetam o middleware e a atualização ou substituição do middleware não afeta as aplicações e o SGBD, respeitando-se as regras de conectividade e troca de dados.

Caso decida-se pela não utilização do middleware junto ao SGBD, as tarefas inerentes à manipulação de dados RDF ficarão na aplicação e serão tratadas pela API RDF adotada. Nesta situação, a aplicação deve possuir mecanismos para compatibilizar a conectividade entre o SGBD e a API RDF. A figura 5 ilustra a situação sem middleware, caso mais comum em função da disponibilidade de ferramentas estáveis para esta função.

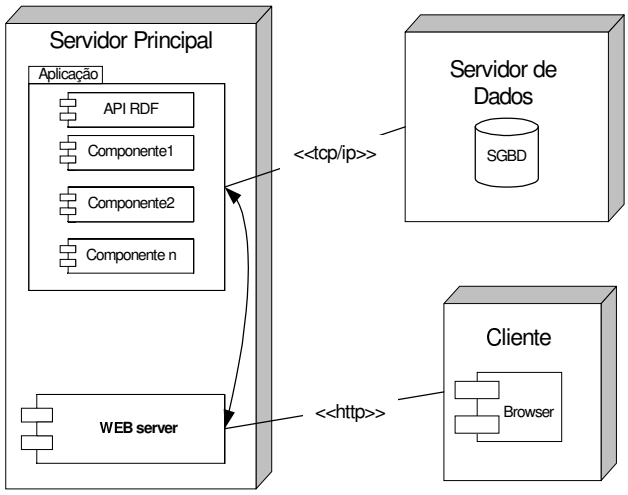

Figura 5- Diagrama de componentes para aplicação sem middleware RDF

Uma análise semelhante pode ser feita para o caso do sistema novo, situação caracterizada pela necessidade da escolha de um SGDB. Esta escolha deve ser compatibilizada com a arquitetura definida para o sistema e a opção por um SGBD com suporte nativo ao RDF deve ser considerada. Neste aspecto, produtos com esta finalidade ainda estão na fase das versões iniciais.

4.4. As aplicações e a interoperabilidade através de RDF

A interoperabilidade de aplicações através de RDF está relacionada com o ambiente do SGBD e com componentes específicos para processamento dos modelos RDF. O conjunto destes componentes será determinado pela configuração escolhida para o SGBD (suporte RDF nativo, middleware ou API junto à aplicação).

Independentemente desta escolha, é necessário estabelecer os mecanismos de conectividade entre o SGDB/middleware e os componentes das aplicações (ODBC, JDBC, JDO, driver nativo, etc) a fim de compatibilizá-los. Caso a opção escolhida tenha sido por não utilizar middleware ou SGBD com suporte nativo ao RDF, a aplicação necessitará de uma API para fornecer operações para manipulação do modelo RDF como: importação, exportação, consulta, e tradução. Neste caso, os componentes da aplicação utilizarão as operações da API para processar dados RDF. Na situação oposta, os componentes utilizarão os serviços do middleware ou SGBD com suporte nativo para atender tais serviços.

Para o ambiente de componentes, além da API já citada, outros elementos são necessários a fim de integrar o conjunto de recursos exigidos. Um deles é o processador XSLT necessário para transformar informações de um modelo RDF para os diversos formatos de apresentação de dados exigidos pela aplicação, principalmente XHTML, ou fazer a conversão contrária.

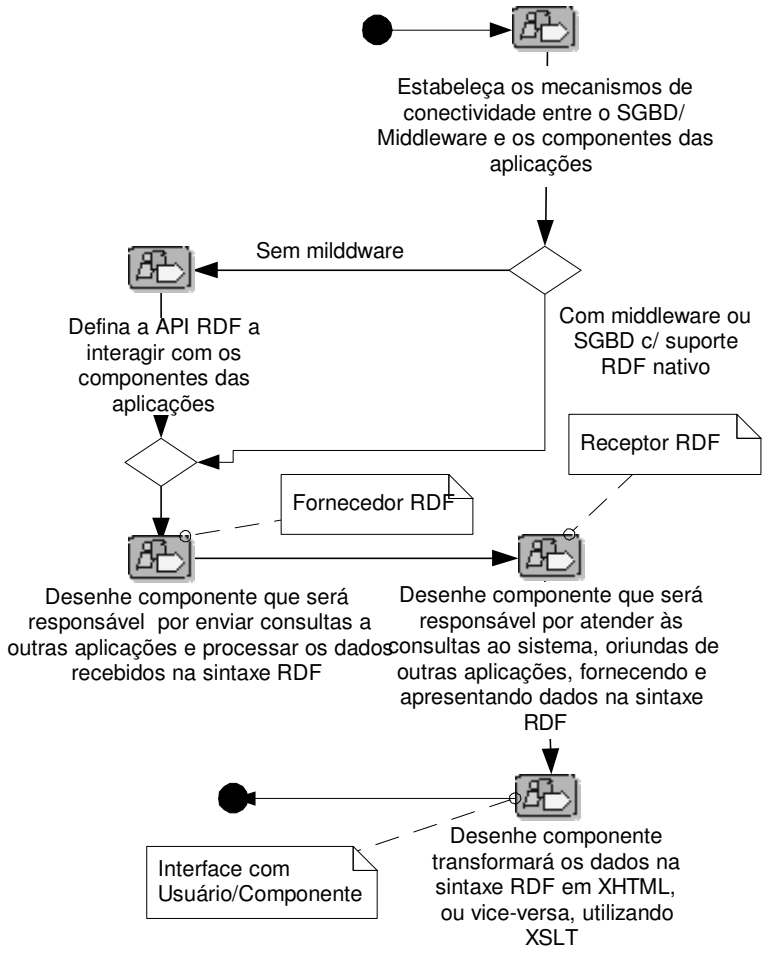

Figura 6 – Diagrama de atividades relacionadas à implementação da interoperabilidade

Outro componente essencial é um fornecedor de dados RDF, cuja finalidade é atender todas as consultas solicitadas ao sistema referente a informações do modelo RDF. Estas consultas podem ter origem na própria aplicação ou, principalmente, de outra de domínio diferente. Além deste, outro essencial é um receptor de dados RDF, cuja finalidade é receber informações de um modelo RDF, da própria aplicação ou de outra de domínio diferente, apresentando os dados de forma conveniente a outro componente do processo.

Por fim, é relevante observar que, para toda informação do domínio publicada em XHTML, a mesma deve também ser publicada em RDF de acordo com o esquema específico. Isto é feito incluindo o código RDF no cabeçalho do documento XHTML. Deste modo, a aplicação poderá ser fornecedora de dados para qualquer outra que conheça o esquema RDF do domínio.

O conjunto de atividades analisadas nesta seção é ilustrado através da figura 6.

Quanto ao método de implementação da interoperabilidade entre as aplicações, a abordagem sugerida neste trabalho é bem simples. São utilizados basicamente os elementos da linguagem XHTML, principalmente formulários e as características da *tag* <head> de embutir sintaxe RDF, além dos componentes que fornecem operações para manipulação do modelo RDF e qualquer tecnologia para geração dinâmica de páginas (JSP, ASP, PHP, dentre outras). Tomando, por exemplo, o caso em que uma aplicação A de um domínio consulta várias aplicações Ni de outro domínio. Nesse exemplo, a arquitetura das aplicações é composta por uma API RDF e pelos componentes descritos anteriormente. O processo de interoperabilidade ocorre conforme descrito a seguir:

a) Na aplicação A:

- usuário faz uma determinada consulta através de um formulário XHTML e este aciona o componente fornecedor RDF caracterizado por uma página dinâmica;

- os parâmetros obtidos através do formulário são enviados para o componente fornecedor RDF nas aplicações Ni de outro domínio. Isto é realizado pelo componente fornecedor RDF da aplicação A, através da leitura das URL cadastradas das aplicações Ni, com os parâmteros necessários (URL?parâmetros);

b) Em cada aplicação Ni:

- o fornecedor RDF faz a consulta ao SGBD;
- o SGBD fornece o resultado da consulta;
- o fornecedor RDF submete os dados recebidos do SGBD para a API RDF gerar um modelo RDF;
- a API RDF fornece as triplas (sujeito, predicado, objeto) do modelo RDF;
- o fornecedor RDF submete o modelo RDF ao processador XSLT;
- o processador XSLT transforma a apresentação dos dados do modelo RDF no formato conveniente (XHTML), embutindo o modelo RDF na *tag* <head>;

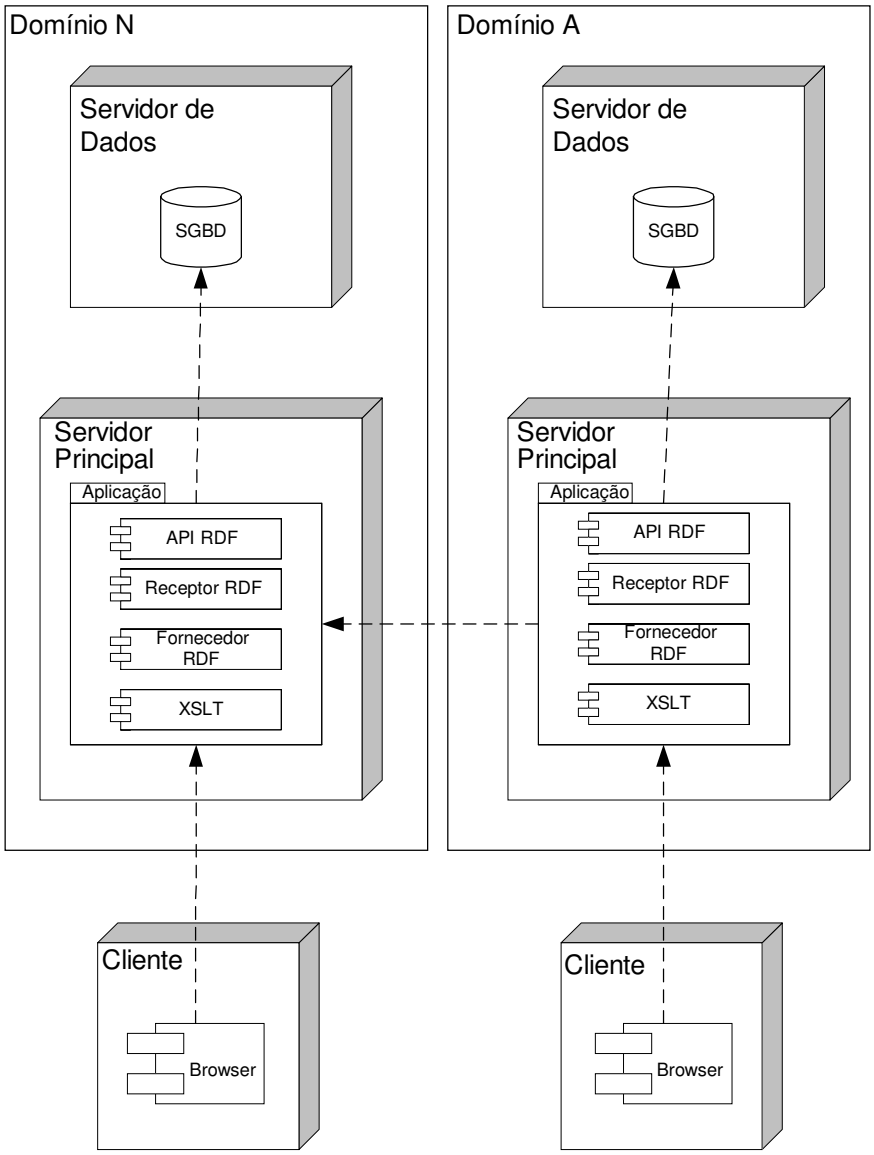

Figura 7 – Componentes no processo de interoperabilidade

c) Na aplicação A:

- o fornecedor RDF obtêm cada página XHTML obtida nas consultas e submete-as ao receptor RDF;
- o receptor RDF lê cada página XHTML com o auxílio de uma API RDF. A API interpreta o código RDF, fornece todas as triplas (sujeito, predicado, objeto) do modelo ou um subconjunto, através de uma consulta específica;
- o receptor RDF submete o resultado obtido ao processador XSLT;

- o processador XSLT transforma a apresentação dos dados do modelo RDF no formato conveniente (XHTML), embutindo o modelo RDF na **tag** <head> e exibe o resultado;

A figura 7 mostra os componentes no processo de interoperabilidade para os domínios citados no exemplo.

5. Estudo de Caso

Para ilustrar a estratégia aqui apresentada com um exemplo concreto, foi realizado um estudo de caso com dois domínios. O primeiro refere-se a um serviço de classificados on-line e o segundo à serviços de cartórios. Em ambos, foram desenvolvidas aplicações que, através do RDF, implementam a interoperabilidade entre os domínios.

No processo foram tomadas as seguintes decisões quanto ao conjunto de tecnologias usadas:

- Metodologia de desenvolvimento: RUP (Rational Unified Process), por estar bem relacionado com a UML (Unified Modeling Language), já padrão na área de desenvolvimento de software, e ser adaptável ao tamanho e características do projeto. Como ferramenta de modelagem: Rational Rose (<www.rational.com/products/rose>) ; por possuir um conjunto de funcionalidades que facilitam a interoperabilidade do modelo com outras aplicações;
- Linguagem para geração de páginas dinâmicas: JSP (Java Server Pages), por ser uma tecnologia independente de plataforma;
- API RDF: Jena (<www.hpl.hp.com/semweb/jena-top.html>); por ser uma API em Java, compatível com a linguagem de geração de páginas dinâmicas, e porque já é bem utilizada em ferramentas RDF;
- SGBD: PostgreSQL (<www.postgresql.org/>); por ser um SGBD independente de plataforma e software de código aberto com um rico conjunto de funcionalidades equivalentes às disponíveis nos principais produtos comerciais;
- Ferramentas para criação dos esquemas RDF: Protégé 2000 (<www.xmlspy.com/>), XML Spy (<www.xmlspy.com/>), Xpetal (<www.langdale.com.au/styler/xpetal/>) e Semtalk (<www.semtalk.com/>), por possuírem bons recursos de importação e/ou exportação de modelos e esquemas RDF.

Nas duas aplicações do estudo não havia sistema legado, portanto eram aplicações novas. Aplicando a estratégia apresentada, era necessário modelar completamente o sistema, incluindo as funcionalidades para a Web e para o processo de interoperabilidade com RDF. Aliado a este fato, desejava-se adotar os esquemas do domínio como padrão. Portanto, foi preciso modelar complemente os domínios e, de um subconjunto de cada um deles, contruiu-se as aplicações.

Para o domínio "Serviços de Cartórios", foi preciso criar o esquema completamente por não existir equivalente. Contruiu-se um modelo UML do domínio e este foi convertido em esquema RDF com a ferramenta Xpetal citada. Neste domínio, foram desenvolvidas duas aplicações Web (Cartório 1 e Cartório 2), ambas disponibilizando consultas a dados

de cartórios e produzindo o resultado em páginas XHTML com dados também expressos na sintaxe RDF.

Para o domínio "classificados on-line" existia um DTD criado pelo NAA Classified Advertising Standards Task Force (Newspaper Association of America). Como forma de reutilizar este esquema, o DTD convertido em um modelo UML e este em esquema RDF. Neste domínio, foi desenvolvida uma aplicação Web de consulta a classificados on-line em que se disponibiliza ao usuário a opção de pesquisa a dados de cartórios, dependendo do contexto da consulta.

Exemplificando a funcionalidades das aplicações, em um anúncio sobre a venda de imóveis, pode-se consultar dados referentes ao registro em cartório do imóvel de interesse. O usuário faz sua consulta nos classificados, o sistema busca os dados no SGBD e exibe o resultado em XHTML e RDF. Junto com o resultado, é disponibilizado um link para consulta a detalhes do imóvel referente ao seu registro em cartório. O usuário aciona este link e o sistema dispara uma outra consulta às aplicações do domínio "serviços de cartório" que estão cadastradas no sistema e, portanto, de esquema RDF conhecido das aplicações. Este processo é realizado através da leitura das respectivas URL de consultas nas aplicações do domínio "serviços de cartórios" (Cartório 1 e Cartório 2, conforme figura 8).

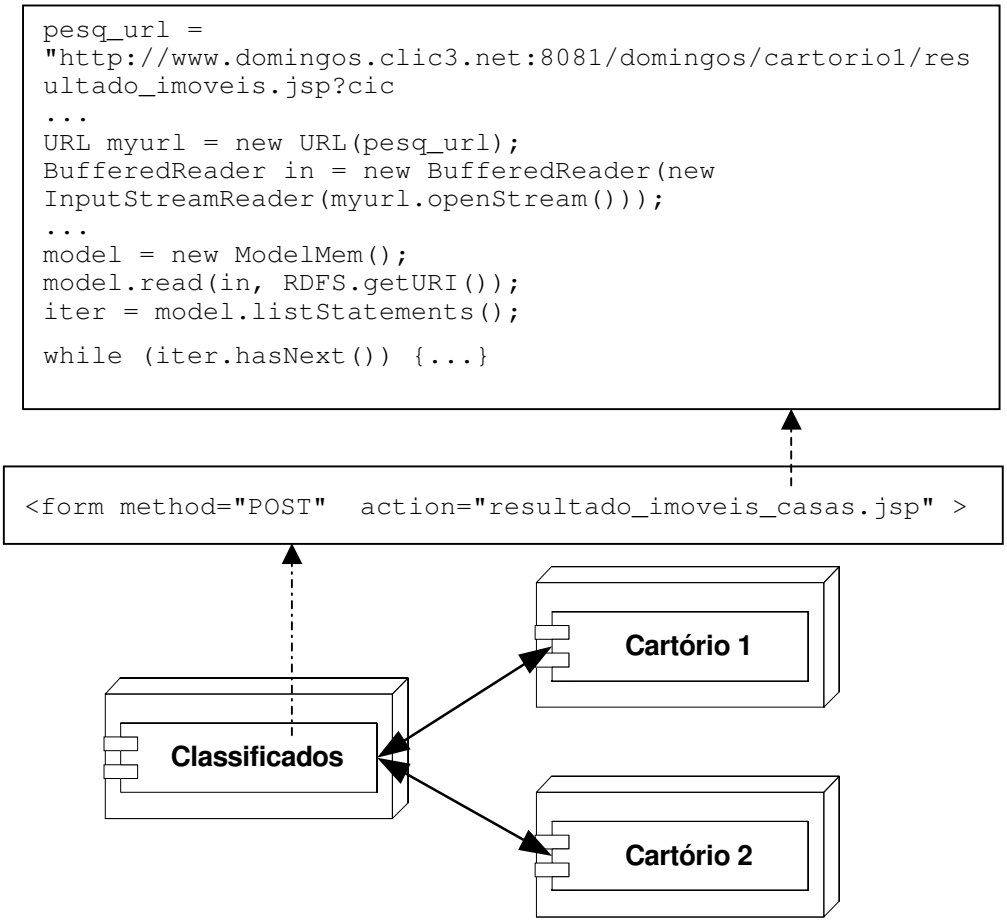

Figura 8 – Interação entre as aplicações no estudo de caso

Durante a consulta, o componente receptor RDF fica aguardando as páginas de resultados do outro domínio. Em cada uma destas aplicações é realizada a consulta ao seu respectivo SGBD que retorna os dados ao componente fornecedor RDF. Este exibe uma página de

resultado com as informações em XHTML e RDF de forma a tornar possível a interoperabilidade pelas aplicações que conhecem o esquema. Esta página é processada pelo componente receptor RDF da aplicação Classificados on-line que utiliza a API RDF Jena, para ler as informações do modelo, e do componente processador XSLT, para converter tais dados em HTML. Por fim, os resultados oriundos das aplicações de serviços de cartórios são exibidos ao usuário da aplicação de Classificados on-line, de modo transparente, como se os dados fossem consultados no próprio site.

O processo está ilustrado na figura 9 e uma versão funcional está disponível em http://www.domingos.clic3.net.

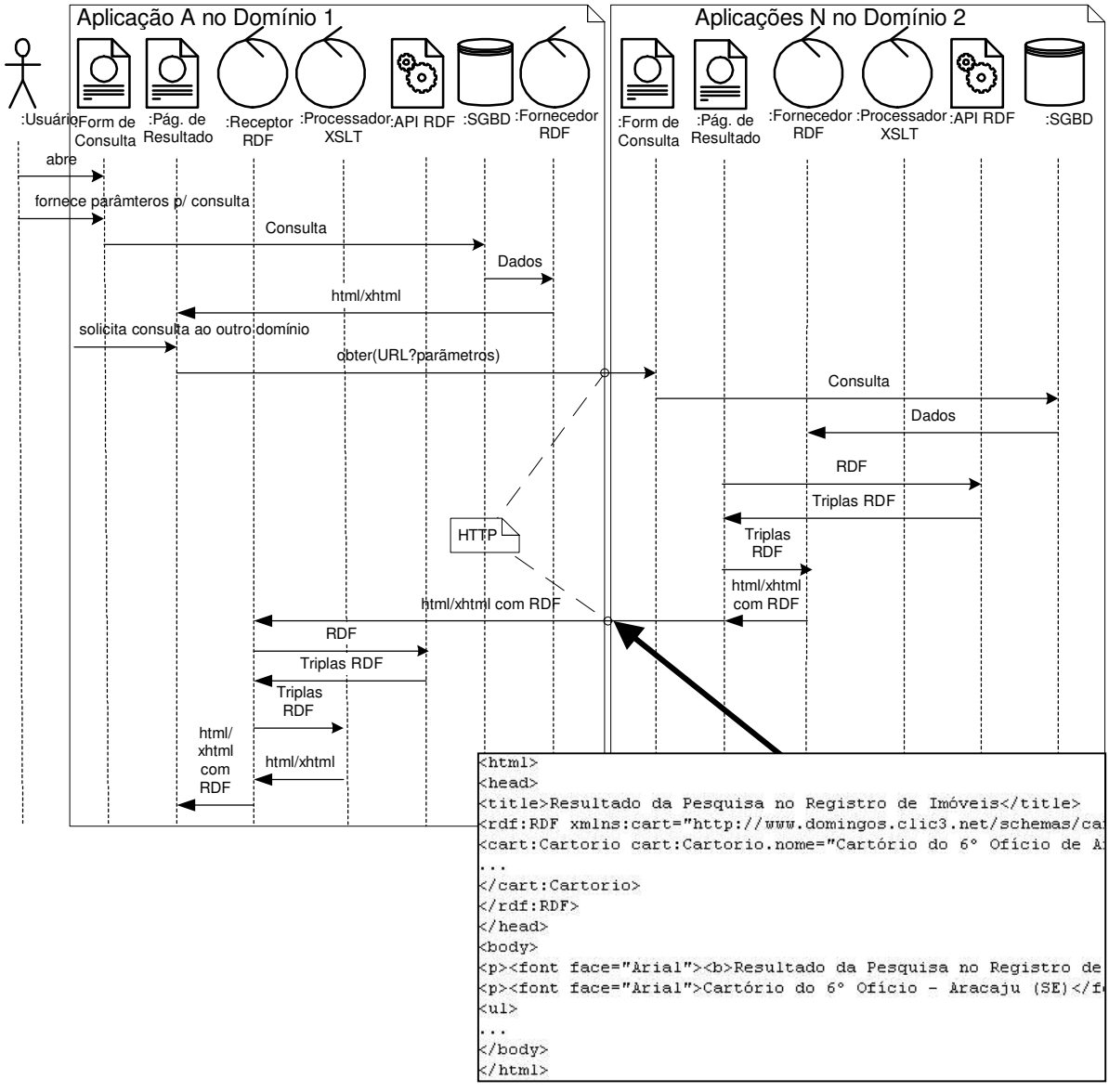

Figura 9 – Diagrama de seqüência do processo de interoperabilidade no estudo de caso

6. Conclusões

A estratégia apresentada possui o foco na praticidade e simplicidade. Praticidade porque foi baseada no estudo de caso citado e, conseqüentemente, demonstra sua aplicabilidade para casos reais, além de não promover replicação de dados, pois utiliza-se de geração dinâmica das informações. A simplicidade é evidenciada porque utiliza recursos já existentes e não complexos do universo de tecnologias aplicadas ao desenvolvimento de soluções para a Web, como UML (Unified Modeling Language), HTTP (Hypertext Transfer Protocol), formulários XHTML e linguagens para geração de páginas dinâmicas. Portanto, procura atender às situações mais convencionais de interoperabilidade na Web.

Quatro pontos foram fundamentais na discussão sobre a estratégia apresentada:

- A modelagem ou remodelagem do sistema para atender os requisitos do processo de interoperabilidade;
- A definição ou criação dos esquemas dos domínios envolvidos no processo, a fim de que o conjunto de dados e metadados seja consistente com a especificação RDF e com os requisitos do domínio;
- A estratégia de alocação do elemento fornecedor de operações para manipulação do modelo RDF, seja API RDF, middleware ou SGBD, por influenciar a arquitetura dos componentes do sistema;
- A forma de implementar o mecanismo de interoperabilidade, por envolver tecnologias já consolidadas e simples, integradas para aplicar Resource Description Framework de modo a atender aos requisitos das aplicações.

Alguns aspectos não foram considerados na abordagem e, para aprimorá-la, alguns trabalhos futuros são sugeridos. A questão da segurança dos dados deve ser considerada para dar confiabilidade ao processo. Outro aprimoramento é a adaptação da estratégia na implementação de Web Services, aplicabilidade em evidência no cenário atual, com a utilização da WSDL (Web Service Description Language), UDDI (Universal Description, Discovery, and Integration) e SOAP (Simple Object Access Protocol). Além desses, a estratégia também pode ser enriquecida com outras linguagens para definição de esquemas, como DAML+OIL, e aliadas a ontologias, a fim de dar suporte à interoperabilidade semântica.

Referências

1. ABITEBOUL, Serge, BUNEMAN e Peter, SUCIU, Dan. **Gerenciando dados na WEB**. Tradução de Mônica Cardia, Rio de Janeiro, Campus, 2000. 251p., ISBN 85-352-0648-5
2. AHMED, Kal et al..**XML Professional Meta Data**. Acocks Green, Birmingham - UK. Wrox Press Ltd. 2000. 567p.

3. BRAGANHOLO, Vanessa; HEUSER, Carlos. **XML Schema, RDF(S) e UML: uma comparaçao**. In: IDEAS 2001 - 4th Iberoamerican Workshop on Requirements Engineering and Software Environments ,Costa Rica, 2001. page 78-90.
4. BRICKLEY, Dan et al. **RDF Specifications**. Lincon, NE - USA. iUniverse.com, Inc. 2000, 120 p., ISBN 0-595-13230-8
5. BRICKLEY, Dan, MILLER, Libby. RDF, SQL and the Semantic Web - a case study. Institute for Learning and Research Technology (ILRT) - University of Bristol - UK, 2000. Disponível em <iltr.org/discovery/2000/10/swsql>. Acesso em 13/10/2000.
6. CARLSON, David. **Modeling XML applications with UML**: practical e-business applications. Upper Saddle River, NJ - USA. Addison-Wesley. 2001. 333p.ISBN 0-201-70915-5.
7. CHANG, Walter W. **Discussion of the Relationship Between RDF-Schema and UML**. W3C note, 1998. Disponível em www.w3.org/TR/NOTE-rdf-uml/. Acesso em 12/08/2000.
8. DAML.ORG. **Language Feature Comparison**. DARPA Agent Markup Language (DAML) Program. Dez, 2001. Disponivel em <www.daml.org/language/features.html>. Acesso em 23/01/2002.
9. CRANEFIELD, Stephen. **UML and The Semantic Web**. International Semantic Web Working Symposium (SWWS). Stanford University, California, USA, 2001. Disponivel em <www.semanticweb.org/SWWS/program/full/paper1.pdf>. Acesso em 25/01/2002.
10. GIL, Yolanda, RATNAKAR, Varun. **Makup Languages: Comparison and Examples**. Proceedings of the 15 th International FLAIRS Conference. Florida, USA, 2002. Disponivel em <trellis.semanticweb.org/expect/web/ semanticweb/comparison.html>. Acesso em 13/02/2002.
11. HJELM, Johan - **Creating the Semantic Web with RDF - Professional Developer's Guide Series**. New York - USA. Wiley Computer Publishing. 2001. 277p.
12. HOUBEN, Geert-Jan. **HERA: Automatically Generating Hypermedia Front-Ends for Ad Hoc Data from Heterogeneous and Legacy Information Systems**. In Engineering Federated Information Systems, pages 81-88. Aka and IOS Press, 2000. Disponível em <wwwis.win.tue.nl/~houben/respub/efis2000.pdf>. Acesso em 03/11/2001.
13. KOTOK, Alan. **Interoperate or Evaportate**. XML.Com. O' Reilly & Associates, Inc. Dez. 2001. Disponível em: <www.xml.com/pub/a/2001/12/12/kotok.html>. Acesso em: 15.03.2002.
14. MARINO, Maria Teresa.**Integração de Informações em Ambientes Científicos na Web: Uma Abordagem Baseada na Arquitetura RDF**. Rio de Janeiro: UFRJ/IM/NCE, 2001.122p.
15. MOHR, Stephen. **Designing Distributed Applications**. Acocks Green, Birmingham - UK. Wrox Press Ltd. 2000.
16. VDOJVJAK, Richard, HOUBEN, Geert-Jan. **RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources**. Proc. of the Intl. Workshop on

Information Integration on the Web – WIIW'2001. Rio de Janeiro - Brasil. E. Simon e A Tanaka.(eds.). UNIRIO, 2001. Páginas 51 a 57.