Sérgio José Rossini Ferreira

Serviços Web Semânticos

Orientadora
Regina Maria Maciel Braga Villela

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Juiz de Fora

Julho de 2008

Monografia submetida ao corpo docente do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como parte integrante dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação.

Profa. Regina Maria Maciel Braga Villela
D. Sc., COPPE/UFRJ
Orientadora

Profa. Fernanda Claudia Alves Campos
D. Sc., COPPE/UFRJ

Prof. Tarcísio de Souza Lima
M. Sc., PUC-Rio

Sumário

Lista de Figuras

Listagens

Resumo

O crescimento da internet possibilitou um grande conjunto de aplicações baseadas em sua infra-estrutura. Atualmente ela não apresenta somente informações estáticas para consulta mas também várias opções de serviços que são relacionados entre consumidores e empresas, sendo que os serviços web têm recebido um maior destaque. Além dos serviços web serem os facilitadores de SOA - a arquitetura orientada a serviços, eles possibilitam agregar ainda maior valor à internet quando são colocados em conjunto com os conceitos da Web Semântica. Este trabalho têm por finalidade apresentar os conceitos e as tecnologias que envolvem os serviços web semânticos. Para isso são contextualizados e conceitualizados os serviços web, apresentadas as ontologias para sistemas de software e identificados os papéis das anotações semânticas em serviços web. Além disso, são apresentadas as principais abordagens para o desenvolvimento dos serviços web semânticos. Finalmente uma das tecnologias é selecionada e utilizada para realizar as anotações semânticas em um serviço web proposto.

1 Introdução

O desenvolvimento da Web tem se apresentado como fator importante para o compartilhamento de informações e criação de novas oportunidades de negócios, onde a distância entre as empresas não é mais vista como fator de dificuldade.

Tipos de negócios como *e-commerce* e *business-to-bussines* aproveitam a infraestrutura disponibilizada pela Web para realizar transações entre empresas e usuários comuns. Dessa forma o papel da internet deixa de ser somente o de repositório de informações e passa a ter um maior significado.

Uma transação de negócio realizada entre empresas pode utilizar diferentes tecnologias. Essas tecnologias buscam eliminar dificuldade de comunicação que possa acontecer entre sistemas que sejam incompatíveis. A Arquitetura Orientada à Serviços - SOA, pode ser definida como uma arquitetura de software que relaciona componentes de um sistema em um ambiente distribuído, onde tecnologias distintas podem ser facilmente conectadas. Os componentes são disponibilizados como serviços que podem ser acessados dinamicamente através de uma rede. SOA dá ênfase aos serviços web, que podem ser disponibilizados pela Web através de protocolo HTTP e são utilizados atualmente para vários tipos de operações como transportar dados e trocar mensagens.

A Web Semântica defende que os recursos disponíveis na Web não sejam acessados apenas por palavras-chaves, mas principalmente por significado do conteúdo. Essa tarefa pode ser realizada a partir da descrição de sites web com o uso de ontologias, descritas por linguagem como OWL, baseada em XML.

Os principais recursos da Web são aqueles que disponibilizam serviços, e não apenas conteúdo estático. Para fazer uso de um serviço web, um agente de software precisa que sua descrição seja feita numa linguagem que possibilite processamento de máquina e ainda que seja descrito com informações de significado das suas operações e dos seus dados, ou seja, anotações semânticas. Dessa forma, surge o estudo de serviços web semânticos.

1.1 Motivação

Os serviços web têm desempenhado papel importante nas organizações atuais visto que o mercado globalizado é dinâmico e fusões acontecem à todo momento. Nesse contexto, a necessidade de criar novos processos de negócio a partir da conexão de sistemas legados, plataformas e tecnologias diferentes pode se tornar uma tarefa custosa uma vez que, de cada lado, as empresas disponibilizam o acesso via serviços web.

A utilização de serviços web semânticos bem definidos e descritos em conjunto com plataformas de inferência e execução, possibilita agilizar a tarefa de descoberta, invocação, execução e monitoração de serviços.

1.2 Objetivo

O objetivo deste trabalho é estudar o conceito de serviços web semânticos, o processo de anotação semântica e as propostas de implementação. Para ilustrar o estudo realizado, uma das propostas é utilizada para o desenvolvimento de um exemplo.

1.3 Organização do trabalho

O trabalho é iniciado com a apresentação do conceito de serviço web no capítulo 2, onde ainda são citadas as tecnologias envolvidas e desenvolvido um exemplo utilizando a linguagem Java.

Em seguida, no capítulo 3, as ontologias são definidas no contexto de sistemas de software, uma ferramenta para desenvolvimento é apresentada e utilizada para criação de uma simples ontologia.

Já o capítulo 4 conceitua os serviços web semânticos, como as informações semânticas podem ser definidas e realiza o estudo de um processo de anotação semântica.

Finalmente, no capítulo 5, são apresentadas as propostas para desenvolvimento de serviços web semânticos sendo uma delas escolhida para a implementação de um exemplo.

2 *Serviços web*

2.1 Introdução

A idéia inicial para o desenvolvimento da Web pode ser definida como a publicação de informações de forma segura e de fácil acesso a qualquer tipo de público ou, em outras palavras, uma base universal de dados contendo qualquer tipo de informação (CARDOSO, 2007).

Além de disponibilizar as informações para qualquer tipo de usuário, essa infra-estrutura deveria possibilitar o seu fácil acesso. Para isso, as informações devem fazer referência entre si, ou como no jargão da informática, *linkadas* umas às outras, possibilitando uma navegação entre o que existe disponível e conseqüentemente acelerando sua descoberta.

Com essa infra-estrutura desenvolvida e acessível aos vários tipos de usuários, organizações logo verificaram a importância de usar essa tecnologia para gerenciar, organizar e distribuir seus dados internos e informações para seus clientes e parceiros. Assim, iniciaram a implementação de soluções *business-to-costumer* e *e-business*, mas como a infra-estrutura disponibilizada ainda não era suficiente para realizar transações comerciais e disponibilizar serviços relacionados, funcionalidades adicionais tornaram-se necessárias para garantir troca de informações conduzidas de maneira segura.

Nesse contexto, o desenvolvimento da infra-estrutura da Web, acompanhou a geração de novos protocolos e tecnologias que possibilitaram garantir transações de forma segura a partir da Internet. A criação de um novo protocolo ou nova tecnologia acontecia sempre a partir da limitação encontrada naquela que era utilizada anteriormente. Houve então a evolução da utilização para negócios na Web (figura 1) (CARDOSO, 2007).

O protocolo SSL (*Secure Socket Layer*) foi desenvolvido pela Netscape para transmissão de documentos privados na Internet. As soluções EAI (*Enterprise Application Integration*) eram e ainda são utilizadas para garantir a integração entre sistemas distribuídos e incompatíveis, permitindo negócios entre empresas. Com o advento do XML (*Extensible Markup Language*) (BRAY *et al.*, 2006), foram desenvolvidas aplicações B2B (*business-to-business*) com

um nível mais alto de integração, pois a utilização do XML possibilita a publicação de dados em múltiplos formatos. A infra-estrutura das aplicações B2B foi direcionada de forma que garantisse eficiência nos processos de negócio com parceiros das organizações. Apesar de explorar as vantagens oferecidas pelo XML na construção de aplicações B2B, as organizações verificaram nesses sistemas pouca flexibilidade e adaptação dinâmica limitada, ou seja, as aplicações distribuídas que utilizam plataformas diferentes continuavam oferecendo dificuldade para integração. A integração ainda é uma das maiores preocupações dos gerentes de tecnologia da informação.

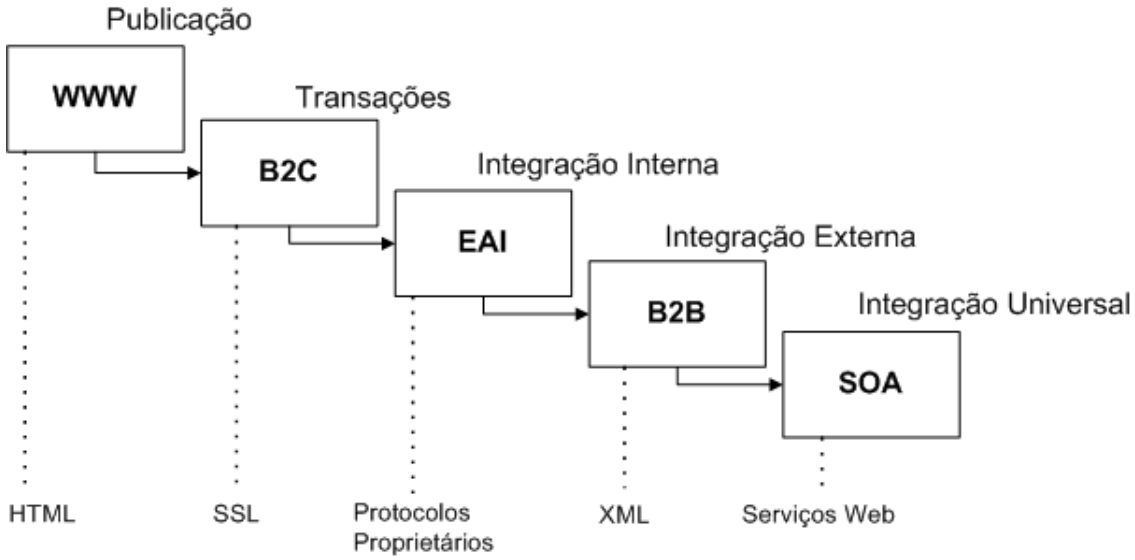

Figura 1: A evolução da utilização para negócios na *Web* (CARDOSO, 2007)

Dessa forma, surgiu a Arquitetura Orientada a Serviços ou em inglês *Service Oriented Architecture* (SOA), que introduziu um método de projetar, desenvolver e gerenciar partes discretas da lógica computacional, os *building blocks* (TAURION, 2007) ou serviços, através da Web, facilitando a integração entre diferentes plataformas e possibilitando o desenvolvimento de aplicações distribuídas. A implementação mais importante do princípio de SOA utiliza a tecnologia XML e os serviços web como fundamento tecnológico.

2.2 A Arquitetura Orientada a Serviços (SOA)

A *arquitetura orientada a serviços* descreve uma abordagem que facilita o desenvolvimento e a composição de serviços modulares que podem ser facilmente integrados e reusados para criar aplicações distribuídas (CARDOSO, 2007). Isto quer dizer que SOA se propõe a dissolver as aplicações em serviços que podem ser infinitamente rearranjados. Assim, no âmbito organizacional, criar ou transformar um processo de negócio significa rearrumar os serviços que

suportam as atividades desse processo estabelecendo então maior agilidade em escrever novas aplicações (TAURION, 2007). SOA oferece escalabilidade, baixo-acoplamento, interoperabilidade, descoberta, abstração e padronização como atributos e por esse motivo alcançou maior sucesso em comparação os seus predecessores.

De acordo com o W3C (*World Wide Web Consortium*), SOA é definida como um conjunto de componentes que podem ser invocados, sendo que a descrição das interfaces desses componentes podem ser facilmente publicadas e encontradas. Estes componentes são desenvolvidos como serviços independentes e devem ser acessados de maneira padronizada.

Existem três ações comuns associadas a um serviço na arquitetura SOA – descoberta, requisição e resposta. A descoberta é o processo de encontrar o serviço que fornece a funcionalidade necessária para a aplicação que se deseja construir. A requisição define a entrada de dados para o serviço enquanto a resposta define a sua saída de dados. Dessa forma, são identificados três atores: o cliente, o provedor e o registro de serviços, como podem ser verificados na figura 2.

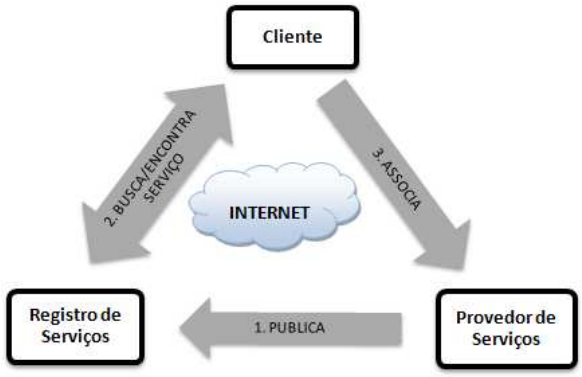

Figura 2: Visão geral da arquitetura SOA (MACHADO, 2006)

Uma aplicação que utilize os conceitos de SOA pode ser desenvolvida utilizando tecnologias como Java RMI (SUN, 2003a), DCOM (HORSTMANN; KIRTLAND, 1997), CORBA (OMG, 2008a) e serviços web.

2.3 Serviços web

De uma maneira geral, os serviços web podem ser observados como facilitadores de comunicação entre aplicações que residem em múltiplas plataformas, usando diferentes modelos de objetos (AMORIM, 2004) e baseados em linguagens diferentes. Possibilitam ainda o desenvolvimento de novas aplicações, assim como o desenvolvimento baseado em componentes (BRAGA;

WERNER, 2000) através de um conjunto de conceitos de interoperabilidade como a XML, o protocolo SOAP (*Simple Objetct Access Protocol*), o WSDL (*Web Service Definition Language*) e o UDDI (*Universal description, discovery and integration*). Os conceitos envolvidos serão tratados nas próximas subseções.

2.3.1 Definição

Os *serviços web* foram concebidos com o objetivo de desenvolver uma arquitetura onde diversos protocolos permitissem a interoperabilidade entre aplicações e sistemas, de plataformas, ambientes e arquiteturas diferentes (GOMES, 2005). Assim, a definição formal adotada pela W3C diz que serviço web é um sistema de programas desenhados para suportar a interação máquina a máquina através de uma rede, utilizando protocolos padronizados. Outros sistemas interagem com o serviço web através de mensagens SOAP, normalmente transmitidas utilizando o protocolo HTTP em conjunto com outros padrões utilizados na Web. De acordo com Cardoso (2007), o serviço web é um componente de software capaz de ser invocado através da Web via mensagem XML que segue a padronização de SOAP. Este componente de software pode disponibilizar uma ou mais operações para execução de ações úteis em nome do cliente.

É importante destacar que a escolha de serviços web para implementação de aplicações SOA deve-se ao fato de que a sua construção utiliza os conceitos por trás da própria arquitetura SOA para efetivar a sua execução. Os atores cliente, provedor de serviços e registro de serviços envolvem as operações de pesquisa, publicação e interação. Essa interação ocorre utilizando as mensagens SOAP já citadas.

2.3.2 SOAP – *Simple Object Access Protocol*

Esse protocolo padroniza a definição de tipos e formatos da mensagem XML que podem ser trocadas entre pontos de um ambiente distribuído e descentralizado.

Um dos principais objetivos do SOAP é ser o protocolo de comunicação que pode ser utilizado em aplicações distintas desenvolvidas a partir de diferentes linguagens de programação, sistemas operacionais e plataformas.

A especificação atual define um esqueleto que pode ser demonstrado a partir da listagem 2.1. A seção Envelope define o *namespace* da especificação SOAP e o estilo de codificação utilizado na criação da mensagem. A seção Header é opcional e contém informação adicional sobre a mensagem. A seção Body contém os dados a serem transferidos.

Listagem 2.1: Estrutura de uma mensagem SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:tsufjf="http://localhost:8080/TUFJFWeb/soap/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soapenv:Header>
                <!-- O elemento HEADER contém informações específicas como
                    autenticação -->
        </soapenv:Header>
  <soapenv:Body>
    <tsufjf:insereAluno soapenv:encodingStyle="http://schemas.xmlsoap.org/
        soap/encoding/">
      <tsufjf:nome>Sergio Rossini</tsufjf:nome>
      <tsufjf:senha>password</tsufjf:senha>
      <tsufjf:endereco>Rua Americo Lobo 559</tsufjf:endereco>
    </tsufjf:insereAluno>
        <soapenv:Fault>
                        <!-- O elemento FAULT contém os erros que podem
                            ocorrer. -->
        </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

2.3.3 WSDL – *Web Service Description Language*

O provedor de serviços é a entidade que cria o serviço web, disponibilizando-o para um cliente que necessite de sua utilização. De forma geral, uma funcionalidade implementada em um software a ser disponibilizada deve ser formatada como um serviço web, sendo descrita num formato padrão – tarefa identificada como descrição do serviço. O padrão utilizado para a descrição do serviço é a WSDL.

WSDL é um padrão W3C que utiliza a linguagem XML para especificar a interface de um serviço web. Este padrão possibilita separar a descrição abstrata de uma funcionalidade oferecida pelo serviço de seus detalhes de implementação definindo assim interfaces a serem oferecidas aos clientes. A definição da interface é realizada pelo elemento portType na versão 1.x ou elemento interface na versão 2.0 da especificação do WSDL e oferecem a assinatura de todas as operações disponíveis, incluindo nome da operação, dados de entrada, dados de saída e falhas.

A descrição do serviço, a partir de um documento WSDL, ainda inclui os detalhes referentes à definição de tipos de dados – através do elemento types; os formatos das mensagens

de entrada e saída – através do elemento message e o mapeamento de protocolos, a partir dos elementos *binding* e *service*.

A figura 3 mostra a estrutura de um documento WSDL e a listagem 2.2 exemplifica um documento WSDL para um serviço de inclusão de aluno.

Figura 3: A estrutura de um documento WSDL, versão 1.x (AMORIM, 2004)

Listagem 2.2: Documento WSDL para um serviço de inclusão de aluno

```
<wsdl:definitions targetNamespace="http://localhost:8080/TSUFJFWeb/service/
    "
     xmlns:apachesoap="http://xml.apache.org/xml-soap"
     xmlns:impl="http://localhost:8080/TSUFJFWeb/service/"
     xmlns:intf="http://localhost:8080/TSUFJFWeb/service/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
     xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4 Built on Apr 22, 2006 (06
    :55:48 PDT)-->
<wsdl:types>
    <schema elementFormDefault="qualified"
        targetNamespace="http://localhost:8080/TSUFJFWeb/service/"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="insereAluno">
            <complexType>
                <sequence>
                    <element name="nome" type="xsd:string"/>
                    <element name="senha" type="xsd:string"/>
                    <element name="endereco" type="xsd:string"/>
                </sequence>
            </complexType>
        </element>
        <element name="insereAlunoResponse">
            <complexType/>
        </element>
    </schema>
</wsdl:types>

<wsdl:message name="insereAlunoRequest">
    <wsdl:part element="impl:insereAluno" name="parameters"/>
</wsdl:message>

<wsdl:message name="insereAlunoResponse">
    <wsdl:part element="impl:insereAlunoResponse" name="parameters"/>
</wsdl:message>

<wsdl:portType name="TSUFJFService">
    <wsdl:operation name="insereAluno">
        <wsdl:input message="impl:insereAlunoRequest" name="
            insereAlunoRequest"/>
```

```
        <wsdl:output message="impl:insereAlunoResponse" name="
            insereAlunoResponse"/>
    </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="TSUFJFServiceSoapBinding" type="impl:TSUFJFService">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.
        org/soap/http"/>
        <wsdl:operation name="insereAluno">
            <wsdlsoap:operation soapAction=""/>
            <wsdl:input name="insereAlunoRequest">
                <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="insereAlunoResponse">
                <wsdlsoap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="TSUFJFServiceService">
        <wsdl:port binding="impl:TSUFJFServiceSoapBinding"
            name="TSUFJFService">
            <wsdlsoap:address
                location="http://localhost:8080/TUFJFWeb/soap/" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

2.3.4 UDDI – *Universal Description, Discovery and Integration*

O registro de serviços é a entidade que representa a localização central onde o provedor de serviços pode relacionar os seus serviços web, possibilitando assim a pesquisa e descoberta desses serviços. Esse papel é desempenhado pela UDDI, que consiste em um serviço estruturado na forma de repositórios para nomeação e localização de serviços web (BELLWOOD, 2002). Uma das vantagens que a especificação da UDDI disponibiliza é fornecer escalabilidade e facilidade para a descoberta de serviços web (CARDOSO, 2007). Portanto, o mecanismo de descoberta deve ser escalado na amplitude da Web fornecendo assim uma maneira eficiente para descoberta de serviços web relevantes dentre dezenas e centenas deles.

Como podem existir diversos repositórios UDDI, estes devem ser sincronizados para que seus conteúdos sejam os mesmos. Dessa forma, um serviço web publicado num repositório é automaticamente replicado para os outros. Mas essa sincronização não é obrigatória, uma vez que existem repositórios UDDI privados para proteger o acesso externo a serviços web de determinadas organizações.

2.4 Desenvolvimento de serviços web

O desenvolvimento de serviços web é similar ao desenvolvimento de qualquer outro software. A diferença existente é que a especificação das interfaces e o suporte das ferramentas envolvidas para esse desenvolvimento assumem maior importância. Cardoso (2007) propõe oito passos gerais para a criação de um serviço web utilizando tecnologia Java.

1. Criação do diagrama de classes;
2. Geração do código Java a partir do diagrama de classes, quando possível;
3. Inclusão de *WebServices Annotations* (SUN, 2005) no código Java;
4. Geração do documento WSDL a partir das anotações inseridas;
5. Implementação dos métodos;
6. Efetuar o deploy da aplicação web;
7. Executar o teste do serviço;
8. Publicação do serviço.

Com o objetivo de exemplificar o processo de criação de serviços web e validar o estudo teórico proposto neste trabalho, propõe-se o desenvolvimento de uma pequena aplicação que será evoluída ao longo dos capítulos. Pretende-se dessa forma, chegar ao desenvolvimento de serviços web semânticos.

O serviço web exemplo a ser criado nesse trabalho demonstra a inclusão de um novo aluno no banco de dados do sistema TSUFJFManager. O TSUFJFManager é uma aplicação Java que utiliza tecnologia Java RMI para operações simples em banco de dados de forma distribuída. Como sua proposta inicial não era uma aplicação Web, a aplicação TSUFJFManager utiliza também a API SWING (SUN, 2003b) para criação de interface com usuários do sistema.

Dentro do projeto TSUFJFServer foi criada a classe de serviço br.org.ufjf.service. TSUFJFService, com respectivas *annotations* citadas na etapa 2. A listagem 2.3 ilustra essa classe.

Listagem 2.3: A classe de serviço TSUFJFService

```
package br.org.ufjf.service;

import javax.jws.WebMethod;
import javax.jws.WebService;

import br.org.ufjf.business.ServiceDAO;
```

```
import br.org.ufjf.business.ServiceDAOFactory;
import br.org.ufjf.valueobject.AlunoVO;

@WebService
public class TSUFJFService {

   private static ServiceDAO service = ServiceDAOFactory.getInstance();

   @WebMethod
   public void insereAluno(String nome, String senha, String endereco) {
        AlunoVO alunoVO = new AlunoVO(nome, senha, endereco);
                try {
                        service.incluiAluno(alunoVO);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

Essa classe de serviço, encapsula o método insereAluno, que recebe três parâmetros do tipo String – nome, senha e endereço. Em seguida, utiliza-os para invocar o construtor da classe AlunoVO. O atributo ServiceDAO dessa classe de serviço é uma interface que define todos os métodos de acesso ao banco de dados. O padrão singleton (GAMMA; HELM; JOHNSON, 2000) é utilizado na classe ServiceDAOFactory para retornar a instância única da classe que implementa a interface citada. Dessa forma, é invocado o método insereAluno que faz o acesso ao banco de dados.

Com a classe de serviço devidamente implementada, foi possível gerar o serviço web referente ao método citado, pois as *annotations* da etapa anterior indicam ao compilador Java que o documento WSDL pode ser gerado em tempo de compilação. O documento WSDL gerado nessa etapa pode ser conferido na listagem 2.2.

A implementação do serviço cliente então deve ser definida, como pode ser verificado na listagem 2.4.

Listagem 2.4: Implementação do cliente para acesso ao serviço web

```
package br.org.ufjf.service.client;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

public class TSUFJFClient {

        public void insereAluno(String nome,
                                String senha,
                                String endereco){
                try {
                        String urlWS =
            "http://localhost:8080/TSUFJFWeb/services/TSUFJFService";
                        Object[] params =
```

```
                    {new String("Sérgio"),
                     new String("pass"),
                     new String("Rua Américo Lobo, 559")};
                           Service service = new Service();
                           Call call = (Call) service.createCall();
                           call.setTargetEndpointAddress(urlWS);
                           call.setOperationName("insereAluno");
                           call.invoke(params);
                    } catch (Exception e) {
                           e.printStackTrace();
                    }
             }
}
```

A desvantagem de executar a aplicação cliente dessa maneira é que os métodos do serviço web não são conhecidos, ou seja, a interface que define os métodos disponibilizados não existe, o que prejudica o desenvolvimento da aplicação. Uma maneira de conseguir agilidade e segurança, conhecendo a interface que define os métodos disponibilizados pelo serviço web é a utilização da classe WSDL2Java (APACHE, 2006). Ao efetuar a leitura de um arquivo WSDL, o WSDL2Java gera os arquivos necessários para que a aplicação cliente conheça todos os métodos disponibilizados e possa instanciar classes para localização do serviço web. Ao executar a geração do cliente com essa classe, os seguintes arquivos são criados:

- A interface TSUFJFService define o método publicado no serviço web;
- A interface TSUFJFServiceService define os métodos de localização do serviço web e é implementada pela classe TSUFJFServiceServiceLocator. No locator fica configurada a URL para o acesso ao serviço web.
- A classe TSUFJFServiceSoapBindingStub é o *stub*, ou seja, a classe que executa o acesso proxy a partir da classe TSUFJFServiceProxy entre a aplicação cliente e o serviço web, processo chamado de *binding*. Ele traduz a comunicação entre o Java e o serviço web, que pode estar implementado em outra linguagem de programação.

Assim, a aplicação cliente passa a conhecer a interface que define o serviço web e o desenvolvimento torna-se claro e simplificado, como disponível na listagem 2.5.

Listagem 2.5: Implementação do cliente com conhecimento da interface definida a partir do serviço web

```
package br.org.ufjf.service.client;

import br.org.ufjf.service.TSUFJFService;
import br.org.ufjf.service.TSUFJFServiceService;
import br.org.ufjf.service.TSUFJFServiceServiceLocator;

public class TSUFJFClientMain2 {
```

```
        public static void main(String[] args) {
                try {
                        TSUFJFServiceService serviceLocator = new
                        TSUFJFServiceServiceLocator();
                        TSUFJFService service =
                              serviceLocator.getTSUFJFService();
                        service.insereAluno("Sergio", "password", "Rua
                              Américo Lobo, 559");
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

O *locator* possui o método getTSUFJFService e retorna um objeto que implementa a interface TSUFJFService, que nesse projeto é a interface que define os métodos disponibilizados pelo serviço web. Assim, a invocação do método torna-se transparente.

O registro do serviço web criado e implementado deve ser feito utilizando UDDI (BELLWOOD, 2002).

2.5 Considerações finais

A evolução das tecnologias possibilitou que as transações eletrônicas entre empresas e clients, como *business-to-business* e *e-commerce*, assumissem uma posição importante na economia atual, pois são direcionadas a alavancar transformações nas estratégias de negócio.

A existência de várias tecnologias diferentes convivendo entre si resulta em grande parcela de esforços e investimentos destinados às atividades de manutenção e integração. Aplicações "departamentalizadas" e embaralhadas em diversas gerações de tecnologias não interoperáveis e conflitantes constituem um grande empecilho para a maioria das organizações que necessita realizar ações estratégicas.

A adoção da arquitetura SOA utilizando serviços web possibilita então agilizar o desenvolvimento de novas aplicações ou rearranjar aquelas existentes, minimizando a parcela de esforços e investimentos citados, buscando os serviços ou componentes já construídos.

Sendo assim, o estudo dos serviços web ganha destaque, sendo necessária sua contextualização, padronização e implementação.

3 *Ontologias para sistemas de software*

3.1 Introdução

Os sistemas de software visam resolver ou minimizar problemas das mais diversas áreas de atuação, seja comercial, empresarial ou área de saúde. Dessa forma, o seu desenvolvimento demanda necessidade de conhecimento prévio do problema, dos conceitos envolvidos, da relação existente entre esses conceitos e ainda uma metodologia prática para essa construção.

A literatura geralmente define os conceitos envolvidos no problema e as relações existentes entre esses conceitos como domínio (KEAN, 1998). À metodologia prática de desenvolvimento de software e seus respectivos processos, dá-se o nome de Engenharia de software.

Diversos problemas encontrados durante a construção de software são conhecidos pela comunidade envolvida. Desde a definição mal formada do problema que se quer resolver, dada pelos interessados na sua resolução até o entendimento desse problema, e dos conceitos envolvidos pelos responsáveis durante o desenvolvimento do software são alguns dos problemas que podem ser citados. Nota-se aqui que os dois conceitos citados no parágrafo anterior estão relacionados.

Além dos problemas encontrados no desenvolvimento de software, Davies, Studer e Warren (2006) mencionam que o volume de informações na Web triplicou entre os anos 2000 e 2003. Sabendo que muitas aplicações atualmente são construídas para funcionar na Web e que essas aplicações normalmente disponibilizam informações de forma pública, a criação de técnicas que sejam capazes de organizá-las torna-se necessária.

O estudo de Ontologia então foi introduzido à área de Computação para que pessoas, organizações e sistemas de *softwares* possam se comunicar, utilizando um conhecimento compartilhado de conceitos e domínios, do inglês *shared understanding*. Uma ontologia é uma especificação explícita e formal da *conceitualização* de um domínio de interesse (DAVIES; STUDER; WARREN, 2006). Logo, utilizando ontologias, minimizam-se possíveis falhas de comunicação e facilita-se a identificação de requisitos e definição da especificação de sistemas (USCHOLD;

GRUNINGER, 1996). Como o entendimento do domínio pode ser utilizado como um *framework* de unificação – do inglês *unifying framework*, para diferentes pontos de visão consegue-se estabelecer:

- Comunicação entre pessoas com diferentes necessidades e pontos de vista definidos em seus diferentes contextos e
- Interoperabilidade entre sistemas de software, através da tradução de diferentes modelagens, métodos, paradigmas, linguagens de programação e ferramentas.

Ainda na área da Computação, especificamente com relação à Web, o uso de ontologias faz parte central de todas as aplicações de Web Semântica (W3C, 2001). A idéia fundamental por trás da Web Semântica descreve que para efetiva implementação, o gerenciamento de informações e a integração de aplicações devem dispor de dados relacionados e processos descritos e gerenciados semanticamente, sendo esses dados associados a uma descrição capaz de ser processada por uma máquina (DAVIES; STUDER; WARREN, 2006).

3.1.1 Conceito

O termo ontologia é proveniente da Filosofia, sendo empregado como a representação da existência através de uma explicação sistemática (LUSTOSA; FAGUNDES; BRITO, 2003). Dessa forma, uma ontologia necessariamente incorpora alguma parte da visão real do mundo com referência a um dado domínio. A visão real do mundo é concebida muitas vezes como um conjunto de conceitos, por exemplo, entidades, atributos e processos, suas definições e suas relações. Define-se dessa forma a conceitualização (USCHOLD; GRUNINGER, 1996). Como a conceitualização de uma parte do mundo real pode existir dentro de uma estrutura de software ou idealizada por alguém, ela pode ser citada como implícita.

Dessa forma, as ontologias representam acordos sobre conceitualizações compartilhadas. As conceitualizações compartilhadas incluem *frameworks* conceituais para modelagem de conhecimento de domínio; protocolos de conteúdo específico para comunicação entre agentes interoperacionais; e acordos entre a representação de teorias particulares de domínio. No contexto de compartilhamento do conhecimento, ontologias são especificadas na forma de definições de vocabulário representacional. Uma ilustração para esse conceito pode ser a representação de uma hierarquia de tipos, especificando classes e suas relações, como exemplo prático, a estrutura relacional de um banco de dados.

As ontologias então podem ser utilizadas para organizar conhecimento de forma estruturada em várias áreas, desde a filosofia, passando pelo gerenciamento do conhecimento e chegando à Web Semântica. Davies, Studer e Warren (2006) referem-se a uma ontologia como um grafo ou rede semântica, consistindo de:

1. Um conjunto de conceitos ou classes – vértices do grafo;
2. Um conjunto de relacionamentos, conectando os conceitos – ligações do grafo;
3. Um conjunto de instâncias associadas a um conceito particular – registro de dados associados a conceitos ou relações.

3.1.1.1 Elementos da ontologia

As ontologias podem ser representadas de diferentes maneiras quando desenvolvidas ou processadas por sistemas de software. Ao utilizar uma ferramenta, os responsáveis pela construção da ontologia – ou seja, os engenheiros do conhecimento, devem ter a possibilidade de representá-la de forma gráfica ou através de uma visualização formal. Para armazenar ou transferir a ontologia, a mesma pode ser representada através de uma linguagem específica para ontologias, o que possibilita seu processamento efetuado por uma máquina (STUDER; GRIMM; ABECKER, 2007). A representação gráfica mais usual segue modelos de redes semânticas, assim como um grafo, indo ao encontro do conceito apresentado no parágrafo anterior. A figura 4 representa a rede semântica para um contexto aluno, professor e instituição em um pequeno cenário dentro de outro bem mais amplo – para o caso de uma universidade.

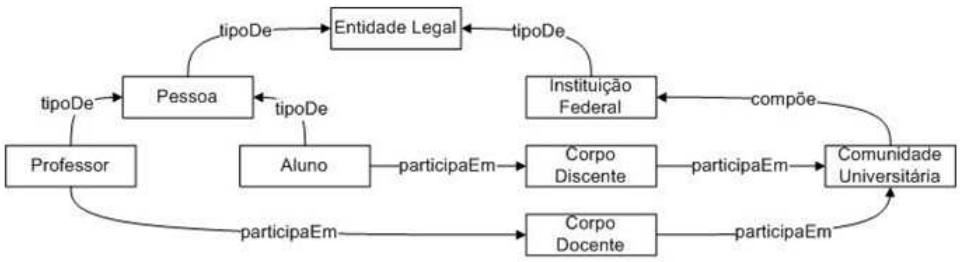

Figura 4: A rede semântica para o domínio de relacionamento entre aluno, professor, comunidade universitária e instituição

Com uma rede semântica pode-se então identificar os elementos que fazem parte de uma ontologia. Tecnicamente, os elementos principais de uma ontologia são as classes ou conceitos, as relações e as instâncias (STUDER; GRIMM; ABECKER, 2007).

Uma classe pode ser vista como um conjunto de elementos. Objetos individuais que pertencem a uma classe são referenciados como instâncias dessa classe. Um importante uso das classes é impor restrição para o que é declarado, assim como nas linguagens de programação. Dessa forma, em sistemas baseados no conhecimento, evitam-se declarações como:

- “Instituição Federal” é um “Aluno” ou
- “Aluno” participa em “Corpo Docente”.

A partir das classes, os relacionamentos entre elas podem ser estabelecidos. Com o relacionamento “tipoDe” entre Aluno e Pessoa, pode-se afirmar que Aluno é um tipo de Pessoa e dessa forma, afirma-se que é uma subclasse de Pessoa. De modo inverso, pode-se afirmar que Pessoa é uma superclasse de Aluno. O mesmo pode ser afirmado para o relacionamento entre as classes Pessoa e Professor. O relacionamento de subclasse e superclasse define uma hierarquia entre as classes. De forma geral, se Aluno é uma subclasse de Pessoa então toda instância de Aluno é também uma instância de Pessoa. A organização de classes em forma hierárquica tem um significado prático muito significativo, pois possibilita eliminar redundância na representação, o que causaria um efeito negativo para a manutenção (CARDOSO, 2007).

3.1.1.2 Utilização de ontologias

Uschold e Gruninger (1996) identificam três categorias principais para o uso de ontologias: comunicação, interoperabilidade e engenharia de sistemas. Para cada categoria, outras definições podem ser importantes, como a natureza do software, quais os usuários finais e quanto geral o domínio é. A figura 5 identifica as três categorias para o uso de ontologias propostas.

Figura 5: As três categorias identificadas para o uso de ontologias (USCHOLD; GRUNINGER, 1996)

3.1.2 Construção de ontologia

Conforme Davies, Studer e Warren (2006), o processo de construção de uma ontologia pode demandar um grande consumo de tempo, sendo necessária ainda a inclusão de profissionais experientes e que ainda tenham conhecimento do domínio de interesse.

Uschold e Gruninger (1996) propõem uma estrutura de metodologia para a construção de ontologias baseada na engenharia de conhecimento. A proposta engloba os seguintes passos: identificação do propósito e do escopo; construção da ontologia a partir da captura, codificação e integração com outras existentes; execução da avaliação e documentação da ontologia e diretrizes para o desenvolvimento.

A identificação do propósito e do escopo da ontologia explicita a causa da construção e quais são os interesses de uso. Essa etapa provê uma meta bem definida para a construção da ontologia.

Capturar ontologia sintetiza o processo de conceitualização da área de interesse e significa identificar os conceitos chave e relacionamentos no domínio de interesse, produzir definições textuais precisas, sem ambigüidades para os conceitos e relacionamentos citados e identificar outros termos que possam fazer referência aos conceitos e relacionamentos definidos. Em seguida, é necessário realizar uma representação dos conceitos e relacionamentos definidos anteriormente, realizando-se a escolha de alguma linguagem formal – formando assim a codificação da ontologia.

Durante as fases de captura e codificação, outras ontologias que englobem todo ou parte da área de interesse podem ser identificadas e dessa forma, torna-se necessário avaliar como elas serão utilizadas. Uschold e Gruninger (1996) identificam essa etapa como sendo um dos maiores desafios na construção de ontologias visto que não são claros os conceitos similares óbvios definidos em ontologias existentes e como esses conceitos podem ser adaptados e reutilizados.

A importância e contribuição da documentação da ontologia são identificadas como necessárias para a construção de efetivo compartilhamento de conhecimento. Todas as considerações assumidas devem ser documentadas, tanto para os conceitos definidos na ontologia como as primitivas utilizadas para expressar as definições.

Uschold e Gruninger (1996) propõem então um conjunto de critérios para o desenvolvimento de ontologias, ou seja, características que devem ser seguidas para que se consiga efetuar a construção obtendo os objetivos a que as ontologias se propõem: clareza de informações, coerência e capacidade de ser extensível.

Assim como a mudança do conhecimento, as ontologias construídas devem possibilitar a evolução. Dessa forma, durante a construção da ontologia, deve-se assumir o mínimo de compromisso possível do domínio que está sendo modelado, possibilitando que o resultado seja extensível.

A representação em código deve ser minimizada, de modo que possa ser compartilhada entre agentes de software (SILVA, 2000).

A criação de ontologias, embora possa ser parcialmente automatizada, ainda exige a intervenção humana e uma metodologia respectiva para isso. A criação de ontologias parte do principio de metodologias para gerenciamento do conhecimento e a introdução das tecnologias de conhecimento em organizações tende a assumir uma abordagem centralizada, indo de encontro aos métodos flexíveis com que as organizações operam. A necessidade atual é por uma evolução distribuída de ontologias (DAVIES; STUDER; WARREN, 2006).

3.2 O papel da ontologia na Web

O uso de ontologia na Web visa formular o conhecimento compartilhado do domínio, do inglês *shared understanding of a domain*. Ao lidar com diferenças na terminologia de usuários, comunidades, disciplinas e línguas como aparecem nos textos, torna-se necessário efetuar o mapeamento das palavras, de acordo com o seu significado, ou seja, semântica. As aplicações ainda podem usar o mesmo termo com diferentes significados entre eles. Um exemplo dessa situação citado por Cardoso (2007) refere-se à palavra escola. Em um contexto, pode referir-se a um local em uma cidade, em outro contexto a uma instituição social e em um terceiro contexto, a um estilo artístico.

Essas diferenças sutis podem ser superadas a partir do mapeamento de terminologias particulares para uma ontologia compartilhada ou definindo mapeamentos diretos entre ontologias. Essas aplicações demonstram que as ontologias suportam interoperabilidade semântica. Os mapeamentos a partir de ontologias das palavras chaves ampliam o retorno das buscas na Web. Dessa forma, as buscas na Web passam a explorar a possibilidade de generalizar ou especializar as informações – se uma busca falha na tentativa de encontrar documentos a partir de determinado termo, uma nova tentativa de encontrar resultados a partir da especialização do termo pode ser realizada. Por outro lado, o mecanismo de busca pode sugerir ao usuário uma busca mais genérica. Consegue-se então estabelecer maior amplitude ou precisão em buscas baseadas na Web.

3.3 O papel da ontologia na Web Semântica

A iniciativa da Web Semântica propõe o uso de anotações semânticas para descrever o significado de certas partes das informações na Web e adicionalmente, o significado das mensagens trocadas entre os serviços web.

Anotações adequadas são úteis para aprimorar a acurácia das buscas na Web. Os mecanismos de busca podem efetuar pesquisas por páginas que contenham os mesmos conceitos representados em ontologias ao invés de uma coleção de páginas onde os termos envolvidos possam estar representados de forma ambígua. Além disso, e tão importante quanto, é a possibilidade de integrar ou transformar elementos de estrutura de dados.

Ontologias possibilitam a especificação formal de um domínio de aplicação que pode ser compartilhado entre diferentes sistemas. Tendo em vista que cada sistema tem a sua forma de organizar as informações, as ontologias representam um mediador entre essas formas de organização. Essa habilidade constitui o maior pré-requisito para o acesso global aos serviços web (CARDOSO, 2007). Uma aplicação particularmente interessante de ontologias é a integração transparente de serviços, sistemas de informações e base de dados que contenham conhecimento geral.

3.4 *Ontology Web Language* – OWL

As linguagens para ontologia permitem que os usuários possam escrever explicitamente a *conceitualização* formal dos modelos de domínio. Para isso, entre os principais requisitos para escrever uma ontologia em forma de linguagem, destacados por Cardoso (2007), podemos citar sintaxe e semântica bem definida, suporte eficiente para inferência e poder de expressão suficiente.

A sintaxe bem definida é condição necessária para que a informação seja processada por uma máquina. As linguagens para ontologias baseadas na Web utilizam o XML como padrão fundamental, apesar de haver outros tipos de sintaxes. Atualmente, o desenvolvimento de ontologias é realizado a partir de ferramentas idealizadas para isso, como a *Protégé* (HORRIDGE *et al.*, 2004), eliminando a necessidade de escrita direta em linguagem representativa.

A OWL é um padrão W3C desenvolvido a partir de seus predecessores OIL (FENSEL *et al.*, 2001) e DAML+OIL (PATEL-SCHNEIDER; HORROCKS; HARMELEN, 2002) e construído a partir do RDF. Representa o padrão atual para linguagem de ontologia na Web e possui a sintaxe baseada em XML, como dito anteriormente. A OWL foi proposta com a intenção de prover uma linguagem que possa ser usada para descrever as classes e relacionamentos entre classes, e que

sejam inerentes aos documentos web e aplicações de software (SMITH; WELTY; MCGUINNESS, 2004).

A especificação da OWL disponibiliza mecanismos para criação de todos os componentes de uma ontologia: conceitos, instâncias e relacionamentos (ou propriedades). Podem ser definidos dois tipos de propriedades: de objetos e de tipo de dados. As propriedades de objetos relacionam as instâncias entre si. As propriedades dos tipos de dados relacionam instâncias com definições que representam tipos de dados, como strings e números. Os conceitos podem ter super e sub-conceitos, possibilitando dessa forma classificação e herança das propriedades através de um esquema hierárquico (DAVIES; STUDER; WARREN, 2006).

Assim como as outras linguagens de representação de ontologias, a OWL é utilizada para representar de forma explícita o conjunto de termos de um vocabulário e como esses termos se relacionam. Como vantagem, destaca-se que a linguagem possibilita à máquina uma maior legibilidade do conteúdo da Web e permite o seu processamento ao invés de apenas mostrá-las ao usuário. A OWL fornece um vocabulário adicional juntamente com uma semântica formal. Além disso, a OWL possibilita maior facilidade para expressar a semântica, superando as linguagens XML, RDF e RDF *Schema*, por possuir habilidade para representar de forma legível o conteúdo da Web (LUSTOSA; FAGUNDES; BRITO, 2003).

A OWL possui três *sub-linguagens* incrementais que foram criadas para uso em comunidades especificas de desenvolvedores e usuários: OWL-Lite, OWL-DL e OWL-Full. O que define essa subdivisão da OWL é sua expressividade. OWL-Lite é a *sub-linguagem* menos expressiva da OWL enquanto que a OWL-Full é a *sub-linguagem* mais expressiva. Logo, a expressividade da OWL-DL fica entre as duas anteriores. A escolha de qual das OWL os desenvolvedores devem usar dependerá das necessidades da ontologia a ser construída.

A OWL Lite suporta os usuários primários, com necessidade de classificação hierárquica e simples restrições de relacionamento. Por ser mais simples, é a subdivisão da OWL que possui maior aproveitamento em sistemas computacionais. A OWL Lite suporta restrições de cardinalidade, permitindo cardinalidade 0 ou 1. A sua implementação pode tornar-se ainda mais simples através da utilização de ferramentas que dão respectivo suporte e também rápido caminho de migração entre taxonomias.

A OWL DL proporciona o máximo de expressividade sem perder completude computacional e decidibilidade dos sistemas de software. Dessa forma, garante-se ao usuário que todas as implicações são garantidas e todas as computações são finalizadas em tempo finito quando utilizar essa subdivisão. A OWL DL inclui todas as construções da linguagem OWL com restrições como separação de tipo, ou seja, uma classe não pode ser uma propriedade e

uma propriedade não pode ser uma classe. OWL DL é nomeada dessa forma devido a sua correspondência com as descrições lógicas (DL), um campo de pesquisa que se refere à lógica de primeira ordem.

A OWL Full foi projetada para garantir aos usuários uma máxima expressividade e liberdade de sintaxe do RDF, mas sem garantias computacionais. A OWL Full e a OWL DL suportam o mesmo conjunto de construções da linguagem OWL, embora com restrições um pouco diferentes. Enquanto a OWL DL impõe restrições sobre o uso de RDF e requer disjunção de classes, propriedades, indivíduos e valores de dados, a OWL Full permite misturar OWL com RDF *Schema* e não requer a disjunção de classes, propriedades, indivíduos e valores de dados. Isto é, uma classe pode ser ao mesmo tempo uma classe e um indivíduo.

3.5 Implementação de ontologia

Baseado na rede semântica demonstrada na figura 4, o trabalho propõe a construção de uma ontologia simplificada. Para isso, a construção será iniciada a partir do básico: a criação de um modelo OWL. Segue-se então a inclusão de classes OWL e a criação de propriedades para esse modelo. Para a construção dessa ontologia, será utilizada a aplicação *Protégé* (HORRIDGE *et al.*, 2004). A aplicação foi escolhida por ser bem citada na bibliografia relacionada a construção de ontologias e por possuir ampla documentação para sua utilização, incluindo tutoriais e exemplos. Será utilizada a linguagem OWL DL, por ser utilizada como referência na Web Semântica.

3.5.1 A ferramenta *Protégé*

O *Protégé* é uma aplicação desenvolvida pela Universidade de Stanford e é uma ferramenta que permite a construção de ontologias de domínio, personalizar formulários de entrada de dados, inserir e editar dados possibilitando assim a criação de uma base de dados guiada por uma ontologia (SEMPREBOM; CAMADA; MENDONÇA, 2007).

A principal interface gráfica do *Protégé* está disponível na figura 6 – aba *OWL Classes*. Essa aba apresenta as áreas de visualização que funcionam como módulos de navegação e edição de classes incluindo sua descrição e propriedades. Além dessa aba, o *Protégé* oferece ainda as abas *Metadata*, *Properties*, *Individuals* e *Forms*. A aba de metadados é responsável pelos metadados da ontologia, como *namespaces*, prefixos e URI. A aba *Properties* possibilita a criação de propriedades de instâncias de classes – *object properties*, propriedades de tipo de dados – *datatype properties* e *annotations*. Na aba *Individuals* é possível instanciar as classes

definidas, a partir do painel *Instance Browser*. Na figura 7 é possível conferir a interface do *Protégé* para a aba *Individuals*. Finalmente, a aba *Forms* possibilita a personalização de formulários para entrada de dados.

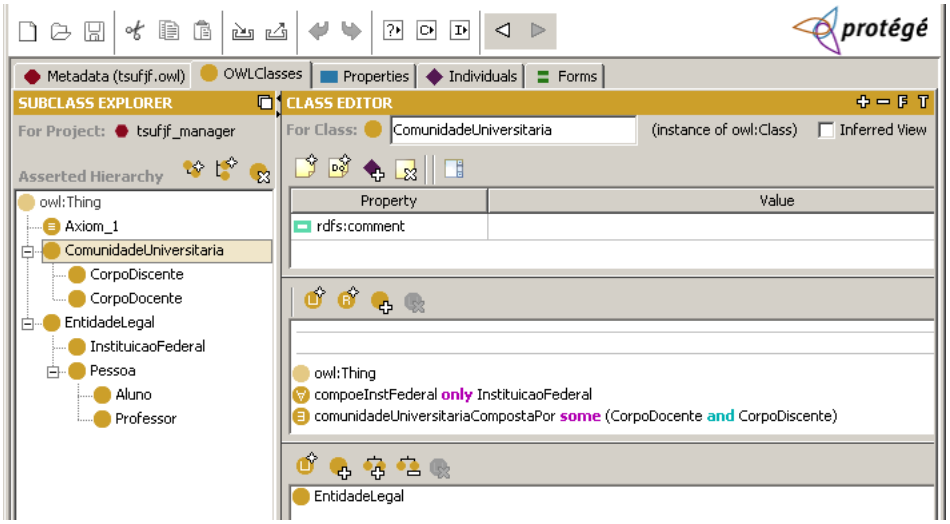

Figura 6: Aba *OWL Classes*, interface principal do *Protégé*

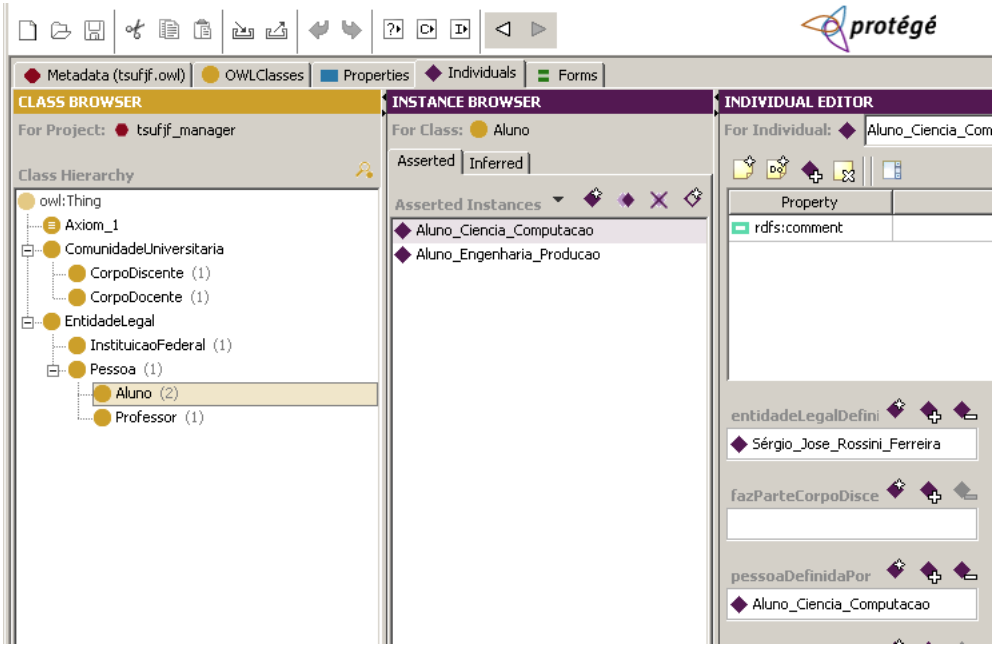

Figura 7: Aba *Individuals* do *Protégé*, com classe Aluno instanciada

3.5.2 Criação das classes

Para o início da construção da ontologia proposta neste trabalho, foi criado o modelo de classes para o domínio, identificando quais classes devem ser inseridas. O modelo seguiu a especificação UML (OMG, 2008b) para diagramas de classes, sendo utilizado para isso a ferramenta

Omondo EclipseUML Free (OMONDO, 2008). De acordo com a figura 4, essas classes são: Entidade Legal, Instituição Federal, Pessoa, Professor, Aluno, Corpo Discente, Corpo Docente e Comunidade Universitária. No modelo de classes, os relacionamentos também foram devidamente identificados. Esse modelo pode ser conferido na figura 8.

Com o modelo de classes criado e a identificação dos relacionamentos devidamente realizada, o tutorial da ferramenta *Protégé* (HORRIDGE *et al.*, 2004) foi utilizado para início da construção da ontologia. Desta forma, espera-se ao final dessa construção obter um documento OWL que represente a ontologia proposta para o domínio desse trabalho.

A etapa inicial para a construção da ontologia refere-se a criação das classes dentro de um novo projeto OWL Lite no *Protégé* – para isso foi utilizada a aba *OWL Classes*, já identificada na figura 6. A figura 9 detalha o painel Subclass Explorer da aba citada com as classes devidamente criadas, onde ainda é possível verificar a hierarquia proposta.

A característica de duas classes serem disjuntas significa que um objeto individual não pode ser uma instância de mais do que uma dessas duas classes, ou seja, um objeto individual não pode representar Aluno e Professor ao mesmo tempo. Dessa forma foram classificadas como disjuntas as classes que estão no mesmo nível hierárquico disponível na figura 9:

- ComunidadeAcademica e EntidadeLegal;
- CorpoDiscente e CorpoDocente;
- InstituiçãoFederal e Pessoa;
- Aluno e Professor.

O *Protégé* oferece a funcionalidade *"Add all siblings..."* no painel *Disjoints* da aba *OWLClasses* para que essa classificação seja feita automaticamente para uma classe selecionada. A figura 10 destaca o painel *Disjoints* e sua barra de ferramentas.

3.5.3 Propriedades de objeto

A etapa seguinte definida pelo tutorial refere-se à criação das propriedades da ontologia. As propriedades em OWL representam os relacionamentos entre dois indivíduos, sendo que existem dois tipos de propriedades: dos objetos e de tipo de dados. As propriedades de objetos realizam uma ligação entre indivíduos. Propriedades de tipo de dados realizam a ligação de um individuo a um valor de tipo de dado em um *XML Schema* ou um literal RDF. As propriedades do tipo *Annotations* não foram utilizadas nessa construção e por isso não serão detalhadas.

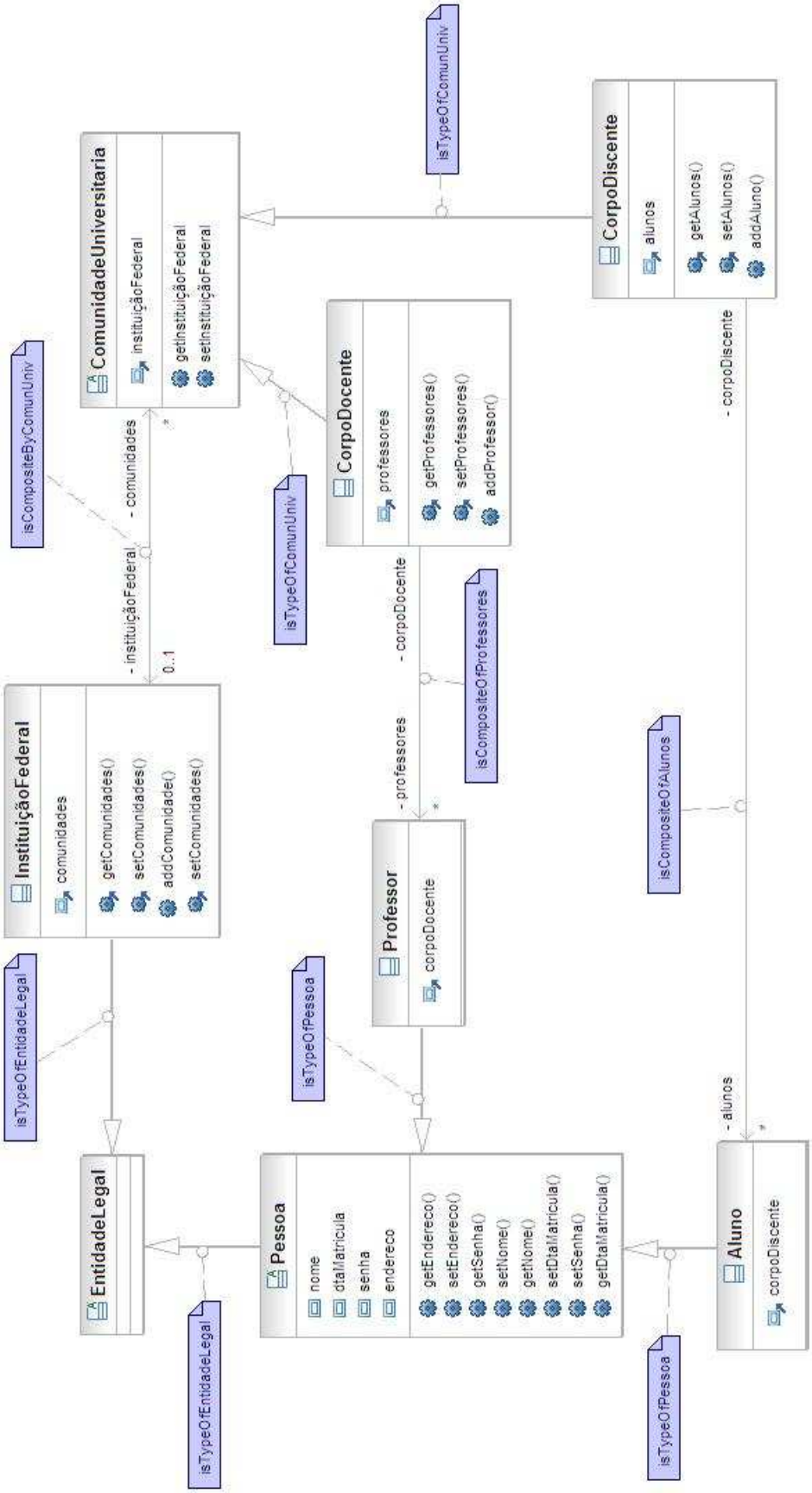

Figura 8: O modelo de classes para o domínio instituição federal, com seus relacionamentos e atributos devidamente identificados

Figura 9: Detalhe da aba *OWL Classes* com as classes criadas e sua hierarquia

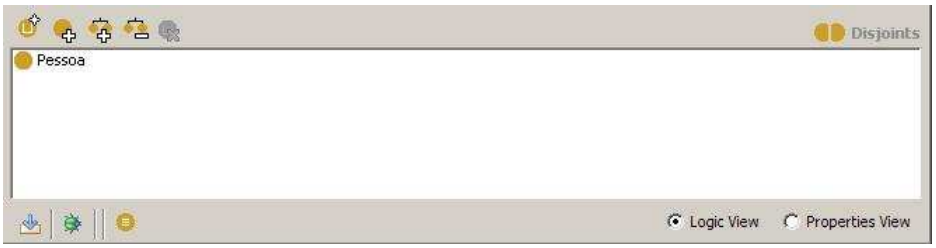

Figura 10: O painel *Disjoints* da aba *OWLClasses* e sua barra de ferramentas

Para o domínio proposto, inicialmente foram inseridas as propriedades de objetos identificadas no modelo de classes. Para que não se introduzisse ambigüidade na definição das propriedades de objetos, os relacionamentos isTypeOfPessoa, isTypeOfEntidadeLegal, isTypeOfComunUniv disponíveis no modelo de classes foram divididos de acordo com as classes a que cada um se refere, separadamente. Ou seja, isTypeOfPessoa foi dividido em alunoIsTypeOfPessoa e professorIsTypeOfPessoa. O mesmo conceito segue para as outras propriedades. As propriedades foram identificadas da seguinte maneira:

- instituicaoFederalIsTypeOfEntidadeLegal – propriedade que representa o relacionamento de herança entre as classes InsituicaoFederal e Pessoa com EntidadeLegal.
- pessoaIsTypeOfEntidadeLegal – propriedade que representa o relacionamento de herança entre as classes Pessoa e EntidadeLegal.
- alunoIsTypeOfPessoa – propriedade que representa o relacionamento de herança entre as classes Aluno e Pessoa.

- professorIsTypeOfPessoa – propriedade que representa o relacionamento de herança entre as classes Professor e Pessoa.
- corpoDocenteIsCompositeByProfessor – essa propriedade representa o relacionamento que indica CorpoDocente ser composto por Professor.
- corpoDocenteIsTypeOfComunidadeUniversitaria – propriedade que representa o relacionamento de herança entre as classes CorpoDocente e ComunidadeUniversitaria.
- corpoDiscenteIsCompositeByAluno – essa propriedade representa o relacionamento que indica CorpoDiscente ser composto por Aluno.
- corpoDiscenteIsTypeOfComunidadeUniversitaria – propriedade que representa o relacionamento de herança entre as classes CorpoDiscente e ComunidadeUniversitaria.
- comunidadeUniversitariaAddsToInstituicaoFederal – propriedade que representa o relacionamento entre ComunidadeUniversitaria e InsituicaoFederal. Um conjunto de comunidades universitárias compõe uma instituição federal.

Durante a criação de propriedades de objetos, a OWL e o *Protégé* possibilitam a criação de *subPropriedades*, ou seja, realizar a criação de uma hierarquia de propriedades introduzindo heranças entre elas. Para esse trabalho não foi necessário a criação de *subPropriedades*.

Para cada propriedade de objeto, deve existir uma propriedade inversa correspondente (HORRIDGE *et al.*, 2004). Se uma propriedade realiza a ligação entre o individuo Aluno com o individuo ComunidadeUniversitaria, então sua propriedade inversa realiza a ligação entre o individuo ComunidadeUniversitaria com Aluno necessariamente. Desta forma, baseando-se nas propriedades de objetos criadas até o momento, as propriedades inversas da ontologia foram identificadas e criadas. A figura 11 destaca o painel *Property Browser* da aba *Properties* do *Protégé*, com todas as propriedades criadas. No painel ainda é possível identificar a propriedade inversa de cada uma delas.

Os indivíduos das classes EntidadeLegal e InstituicaoFederal devem ser especificadas por ao menos um indivíduo das classes a quem se relacionam. Para isso, a OWL dispõe do atributo booleano *Functional* de uma propriedade de objeto. Se uma propriedade de objeto *Functional* é configurada como verdadeira, então para um dado individuo, deve existir no mínimo outro indivíduo relacionado através dessa propriedade. Dessa forma, o atributo *Functional* das propriedades de objeto instituicaoIsCompositeByComunidade, corpoDiscenteIsCompositeByAluno e corpoDocenteIsCompositeByProfessor tiveram seu valor modificado para verdadeiro.

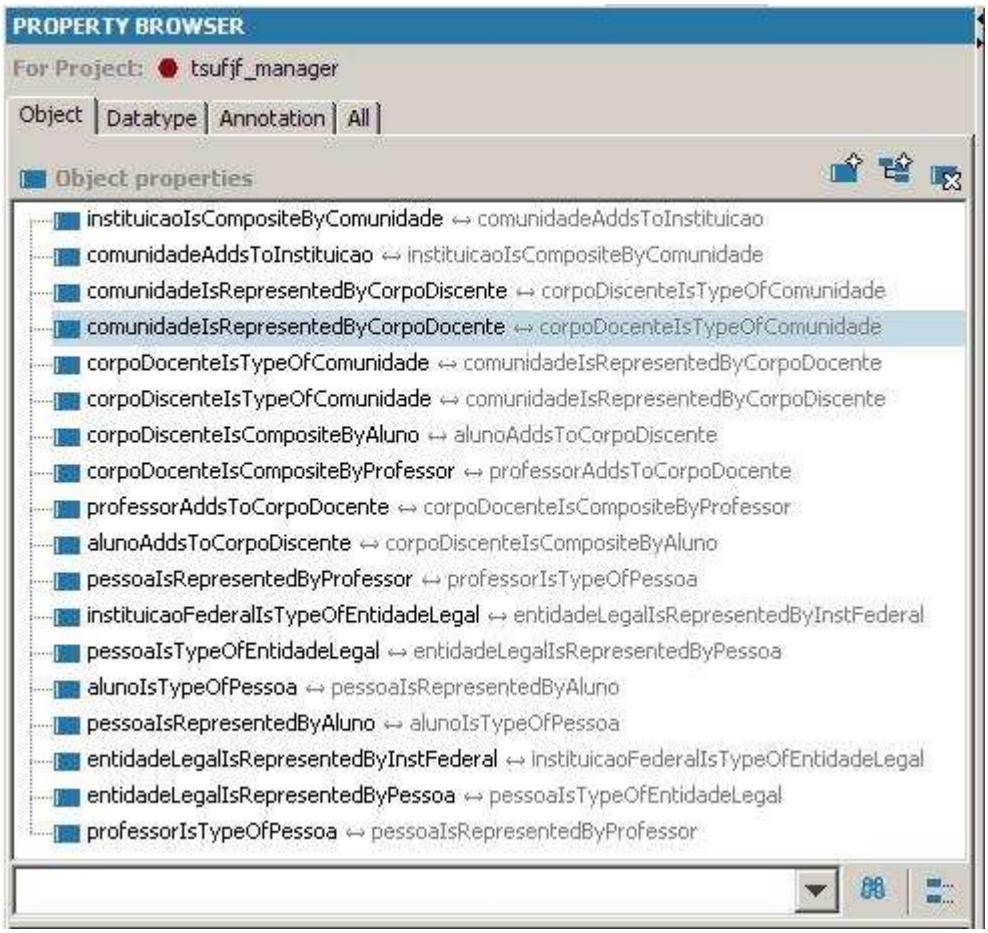

Figura 11: Detalhe da aba *Properties* com as propriedades de objetos criados com respectiva propriedade inversa

As propriedades de objeto possuem ainda os atributos de domínio (*domain*) e de alcance (*range*). As propriedades de objeto devem efetuar uma ligação entre um objeto individual de domínio a um objeto individual de alcance. Definir o domínio e o alcance de uma propriedade de objeto significa eliminar inferências inconsistentes que podem acontecer durante o processamento da ontologia, pois são utilizados como “axiomas” durante esse processo. Assim, esses atributos devem ser especificados. Como exemplo, devemos definir para a propriedade alunoIsTypeOfPessoa o domínio Aluno e o alcance Pessoa. Com o mesmo raciocínio, o domínio e o alcance de todas as outras propriedades foram definidos.

Seguindo o mesmo padrão para definir propriedades de objeto, as propriedades de dados (*dataType properties*) foram criadas:

- A classe Pessoa encapsula as propriedades *hasNome*, *hasDataMatricula*, *hasSenha* e *hasEndereco*;
- A classe CorpoDocente possui a propriedade *hasProfessores*;

- A classe CorpoDiscente encapsula a propriedade *hasAlunos* e;
- A classe InstituicaoFederal possui a propriedade *hasComunidades*.

O *Protégé* ainda permite especificar outros atributos e adicionar informações que possibilitem atingir uma ontologia com maior eficiência, o que vai além da proposta desse trabalho.

3.5.4 A descrição da ontologia

Um projeto da aplicação *Protégé* é armazenado em três tipos de arquivos. O arquivo do tipo pprj armazena as informações referentes ao projeto. O arquivo repository armazena informações quanto ao repositório de ontologias importadas no projeto, podendo ser locais ou remotas. Finalmente, o arquivo owl armazena a ontologia desenvolvida no formato XML utilizando a linguagem OWL, foco desse trabalho.

A listagem 3.1 disponibiliza o resultado obtido com a construção da ontologia através da ferramenta *Protégé*. Nessa listagem é possível identificar as classes citadas, a hierarquia definida entre essas classes e as propriedades criadas.

Listagem 3.1: A ontologia criada representada em XML utilizando a linguagem OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http:://localhost:8080/TSUFJFManager/ontology/2008/6/5/
Ontology1212691584.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://localhost:8080/TSUFJFManager/ontology/2008/6/5/
       Ontology1212691584.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="EntidadeLegal">
    <owl:disjointWith>
      <owl:Class rdf:ID="ComunidadeAcademica"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="CorpoDocente">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#ComunidadeAcademica"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="CorpoDiscente"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Professor">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Pessoa"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
```

```
    <owl:Class rdf:ID="Aluno"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#CorpoDiscente">
  <owl:disjointWith rdf:resource="#CorpoDocente"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ComunidadeAcademica"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="InstituicaoFederal">
  <owl:disjointWith>
    <owl:Class rdf:about="#Pessoa"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#EntidadeLegal"/>
</owl:Class>
<owl:Class rdf:about="#Pessoa">
  <rdfs:subClassOf rdf:resource="#EntidadeLegal"/>
  <owl:disjointWith rdf:resource="#InstituicaoFederal"/>
</owl:Class>
<owl:Class rdf:about="#ComunidadeAcademica">
  <owl:disjointWith rdf:resource="#EntidadeLegal"/>
</owl:Class>
<owl:Class rdf:about="#Aluno">
  <owl:disjointWith rdf:resource="#Professor"/>
  <rdfs:subClassOf rdf:resource="#Pessoa"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="comunidadeIsRepresentedByCorpoDocente">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="corpoDocenteIsTypeOfComunidade"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#ComunidadeAcademica"/>
  <rdfs:range rdf:resource="#CorpoDocente"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#corpoDocenteIsTypeOfComunidade">
  <owl:inverseOf rdf:resource="#comunidadeIsRepresentedByCorpoDocente"/>
  <rdfs:range rdf:resource="#ComunidadeAcademica"/>
  <rdfs:domain rdf:resource="#CorpoDocente"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="professorIsTypeOfPessoa">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="pessoaIsRepresentedByProfessor"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Pessoa"/>
  <rdfs:domain rdf:resource="#Professor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="pessoaIsRepresentedByAluno">
  <rdfs:range rdf:resource="#Aluno"/>
  <rdfs:domain rdf:resource="#Pessoa"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="alunoIsTypeOfPessoa"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="instituicaoFederalIsTypeOfEntidadeLegal">
  <rdfs:range rdf:resource="#EntidadeLegal"/>
  <rdfs:domain rdf:resource="#InstituicaoFederal"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="entidadeLegalIsRepresentedByInstFederal"/
      >
```

```
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="pessoaIsTypeOfEntidadeLegal">
  <rdfs:range rdf:resource="#EntidadeLegal"/>
  <rdfs:domain rdf:resource="#Pessoa"/>
  <owl:inverseOf>
    <owl:ObjectProperty
         rdf:ID="entidadeLegalIsRepresentedByPessoa"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#entidadeLegalIsRepresentedByInstFederal">
  <rdfs:domain rdf:resource="#EntidadeLegal"/>
  <rdfs:range rdf:resource="#InstituicaoFederal"/>
  <owl:inverseOf rdf:resource="#instituicaoFederalIsTypeOfEntidadeLegal"/
     >
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#entidadeLegalIsRepresentedByPessoa">
  <rdfs:domain rdf:resource="#EntidadeLegal"/>
  <rdfs:range rdf:resource="#Pessoa"/>
  <owl:inverseOf rdf:resource="#pessoaIsTypeOfEntidadeLegal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#pessoaIsRepresentedByProfessor">
  <rdfs:range rdf:resource="#Professor"/>
  <rdfs:domain rdf:resource="#Pessoa"/>
  <owl:inverseOf rdf:resource="#professorIsTypeOfPessoa"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="corpoDiscenteIsTypeOfComunidade">
  <rdfs:domain rdf:resource="#CorpoDiscente"/>
  <rdfs:range rdf:resource="#ComunidadeAcademica"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="comunidadeIsRepresentedByCorpoDiscente"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#comunidadeIsRepresentedByCorpoDiscente">
  <rdfs:range rdf:resource="#CorpoDiscente"/>
  <rdfs:domain rdf:resource="#ComunidadeAcademica"/>
  <owl:inverseOf rdf:resource="#corpoDiscenteIsTypeOfComunidade"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:ID="corpoDocenteIsCompositeByProfessor">
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="professorAddsToCorpoDocente"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Professor"/>
  <rdfs:domain rdf:resource="#CorpoDocente"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="corpoDiscenteIsCompositeByAluno">
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="alunoAddsToCorpoDiscente"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#CorpoDiscente"/>
  <rdfs:range rdf:resource="#Aluno"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="instituicaoIsCompositeByComunidade">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
     >
  <rdfs:range rdf:resource="#ComunidadeAcademica"/>
```

```
    <rdfs:domain rdf:resource="#InstituicaoFederal"/>
    <owl:inverseOf>
      <owl:InverseFunctionalProperty rdf:ID="comunidadeAddsToInstituicao"/>
    </owl:inverseOf>
  </owl:FunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#comunidadeAddsToInstituicao">
    <rdfs:domain rdf:resource="#ComunidadeAcademica"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#InstituicaoFederal"/>
    <owl:inverseOf rdf:resource="#instituicaoIsCompositeByComunidade"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#professorAddsToCorpoDocente">
    <rdfs:domain rdf:resource="#Professor"/>
    <rdfs:range rdf:resource="#CorpoDocente"/>
    <owl:inverseOf rdf:resource="#corpoDocenteIsCompositeByProfessor"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#alunoAddsToCorpoDiscente">
    <rdfs:domain rdf:resource="#Aluno"/>
    <rdfs:range rdf:resource="#CorpoDiscente"/>
    <owl:inverseOf rdf:resource="#corpoDiscenteIsCompositeByAluno"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#alunoIsTypeOfPessoa">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#pessoaIsRepresentedByAluno"/>
    <rdfs:domain rdf:resource="#Aluno"/>
    <rdfs:range rdf:resource="#Pessoa"/>
  </owl:InverseFunctionalProperty>
    <owl:DatatypeProperty rdf:ID="has_Alunos">
        <rdfs:domain rdf:resource="#CorpoDiscente"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_Comunidades">
        <rdfs:domain rdf:resource="#InstituicaoFederal"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_DataMatricula">
        <rdfs:domain rdf:resource="#Pessoa"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_Endereco">
        <rdfs:domain rdf:resource="#Pessoa"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/
            >
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_Nome">
        <rdfs:domain rdf:resource="#Pessoa"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/
            >
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_Professores">
        <rdfs:domain rdf:resource="#CorpoDocente"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="has_Senha">
        <rdfs:domain rdf:resource="#Pessoa"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/
            >
    </owl:DatatypeProperty>
</rdf:RDF>
```

```
<!-- Created with Protege (with OWL Plugin 3.4, Build 128)  http://protege.
    stanford.edu -->
```

A partir da ontologia criada e representada pela listagem é possível adicionar semântica a um sistema de software e ou site web, onde a mesma será inferida para se conseguir os mapeamentos e resultados necessários à busca das informações.

3.6 Considerações finais

A criação de ontologias para um domínio de interesse representa uma grande importância para um conhecimento unificado e uniforme. As ontologias possibilitam a realização de mapeamentos entre base de dados com diferentes representações, mas de conteúdo em comum, adicionando assim semântica às estruturas de dados disponíveis e ou utilizadas na aplicação de interesse.

Por possuir uma proposta bem abrangente, a construção de uma ontologia não é definida como tarefa trivial. É necessário o envolvimento de pessoas especializadas no domínio do qual se quer desenvolver a representação para o sucesso do processo.

Nenhuma ontologia bem definida é necessariamente descartada durante a construção de outra que possua domínio correlato. O conceito de ontologia envolve o aproveitamento de outras ontologias, o que enriquece ainda mais o trabalho desenvolvido. Necessário destacar que esse aproveitamento é um dos maiores desafios durante sua construção.

A Web Semântica propõe a utilização da grande quantidade de informações disponíveis na Web de uma forma muito mais proveitosa e correta, procurando eliminar ambigüidades e alcançar buscas precisas. Para isso, a Web Semântica conta com o auxílio de ontologias e suas linguagens representativas para conseguir estabelecer o seu objetivo.

As ontologias possuem mais de uma linguagem para sua representação. A OWL é a mais utilizada por possibilitar um maior entendimento humano, sendo descrita em XML – uma estrutura de dados textual e de fácil processamento por computadores.

4 Serviços Web Semânticos

4.1 Introdução

Os *serviços web* são definidos como unidades de funcionalidades que podem ser acessadas através da Web. Dessa forma, consegue-se adicionar à web um novo nível de funcionalidade, em direção a uma integraçãoo consistente entre os componentes de software distribuídos usando os padrões web (DAVIES; STUDER; WARREN, 2006). Atualmente os serviços web possibilitam a integração de aplicações através da troca dinâmica de informações. A indústria tem se esforçado para alcançar a padronização da descrição, descoberta e invocação de serviços web a partir de tecnologias como WSDL, UDDI e SOAP, respectivamente.

Na forma com que os padrões das tecnologias citadas são apresentados e manipulados, eles são desenvolvidos para representar a informação referente às interfaces de serviços, como eles são organizados e como podem ser invocados (CARDOSO, 2007) além de especificar a estrutura das mensagens que o serviço pode aceitar ou produzir. Essa especificação é chamada de descrição sintática (DAVIES; STUDER; WARREN, 2006) e, mesmo que operem à um nível sintático e suportem a interoperabilidade entre diversas aplicações em diferentes plataformas de desenvolvimento através de padrões comuns, essas tecnologias ainda requerem a interação humana para que se estabeleça um maior alcance.

Para combinar os serviços web apropriados de uma maneira útil é necessário que o desenvolvedor realize a busca de cada um daqueles que devem compor a aplicação. Essa limitação atinge a escalabilidade da aplicação e encurta bastante a economia vislumbrada para o desenvolvimento com a utilização de serviços web (DAVIES; STUDER; WARREN, 2006). Significa dizer então que os padrões para serviços web atualmente não permitem a sua representação semântica.

Essa falta de representação semântica atinge a promessa de integração automática de aplicações durante as tarefas de descoberta e invocação de serviços web, o que vai de encontro à proposta da Web Semântica. Se todos os provedores de serviços em todos os domínios de aplicação concordassem em representar seus serviços sob um formato padronizado único, não

haveria necessidade de semântica. Entretanto é presunçoso assumir que todas as aplicações e todos os seus serviços possíveis possam ser padronizados. Dessa forma ocorrem disparidades entre a especificação da busca e a publicação de serviços similares, lembrando que estas duas etapas são realizadas por indivíduos diferentes: a busca é feita pelo requisitante ou consumidor do serviço e a publicação é feita pelo provedor de serviço (CARDOSO, 2007).

Especificar como os serviços web podem ser integrados não é suficiente. Se a semântica não é agregada ao serviço, o requisitante pode não ser capaz de encontrar um provedor de serviços que seja capaz de realizar as combinações adequadas mesmo que existam diferenças sutis na especificação das interfaces disponíveis pelo provedor.

Com a visão de que os sites web não devem apenas disponibilizar informações estáticas, mas também possibilitar ações representativas para o mundo real, como a venda de um produto ou o controle de um dispositivo físico, os estudos recentes da Web Semântica têm sido direcionados também para o contexto dos serviços web, de forma que esses serviços possam ser acessíveis através do processamento por máquinas. O surgimento de linguagens como a OWL que possibilitam a representação de ontologias de qualquer domínio e que possam ser instanciadas na descrição de sites web fortalecem o estudo da adoção de marcação semântica para os serviços web com a intenção de permitir que as tarefas de descoberta, composição e invocação sejam automatizadas, minimizando a intervenção humana (MARTIN *et al.*, 2004).

Ao possibilitar essa automatização, espera-se um grande impacto nas áreas de *e-commerce* e de soluções EAI (*Enterprise Application Integration*), assim como também é esperada a cooperação dinâmica e escalável entre diferentes sistemas e organizações – um cenário onde os serviços web são compostos de forma a se alcançar uma maior complexidade, maior flexibilidade e onde são adicionados valores às suas funcionalidades, alcançando também maior custo-benefício para a integração (DAVIES; STUDER; WARREN, 2006).

4.2 Serviços web semânticos

A proposta do conceito de *serviços web semânticos* parte da visão de que as tecnologias proporcionadas pela Web Semântica sejam combinadas com os serviços web, permitindo habilitar a descoberta e interação automática entre sistemas de software (STUDER; GRIMM; ABECKER, 2007).

De uma forma geral, as tecnologias proporcionadas pela Web Semântica têm como objetivo estabelecer uma Web que possibilite uma interpretação via máquina, ou seja, onde os algoritmos sejam capazes de processar e inferir as informações disponibilizadas atualmente

somente para o entendimento humano. De outro lado, as tecnologias que estão sendo desenvolvidas para os serviços web buscam proporcionar uma infraestrutura em que as organizações possam disponibilizar funcionalidades de seus negócios através da Web.

As interfaces que definem os serviços passam a ser *anotadas* de maneira que possam ser entendidas por processamento de máquina, disponibilizando informações referentes às funções que executam e como essas funções são executadas, indicando seus atributos funcionais e não funcionais. Além disso, com ontologias capazes de descrever os domínios aos quais os serviços se referem, proporcionam-se descrições mais ricas, capazes de estabelecer a esperada interpretação por máquina. Assim, é possibilitada uma descoberta mais sofisticada dos serviços do que a disponível atualmente a partir de UDDI (STUDER; GRIMM; ABECKER, 2007).

Cardoso (2007) define os *serviços web semânticos* como serviços web onde suas propriedades, capacidades, interfaces e efeitos estão disponíveis em forma de código de uma maneira em que não exista ambigüidade e que possam ser interpretados por máquina.

Davies, Studer e Warren (2006) citam a necessidade de se propor um *framework* completo e consistente para o desenvolvimento de serviços web semânticos. Propõem a tarefa de criação do modelo conceitual como inicial, sendo o passo seguinte referente à utilização de uma linguagem que seja capaz de referenciar de maneira sintática formal e semântica o modelo conceitual criado na etapa anterior. Finalmente, todos os componentes que utilizem a linguagem escolhida e que possibilitem a automação do serviço devem ser envolvidos num ambiente de execução.

A literatura ainda define um ciclo de vida para os serviços web semânticos. O ciclo de vida documentado por Cardoso (2007) identifica o papel das semânticas utilizadas para sua construção.

4.2.1 Atividades no ciclo de vida dos serviços web semânticos

De maneira geral, as atividades no ciclo de vida dos serviços web podem ser categorizadas como atividades de modelagem, atividades de construção – *build-time* e atividades de disponibilização e execução – *run-time* (CARDOSO, 2007). As semânticas desempenham papel fundamental em todas as atividades do ciclo de vida. A figura 12 ilustra a interação entre as três etapas e as subseções seguintes destacam de maneira mais detalhada os papéis que as semânticas desempenham em cada uma dessas etapas.

Durante as atividades de modelagem, o provedor dos serviços pode explicar as semânticas planejadas através de anotações em partes apropriadas do serviço web com os conceitos

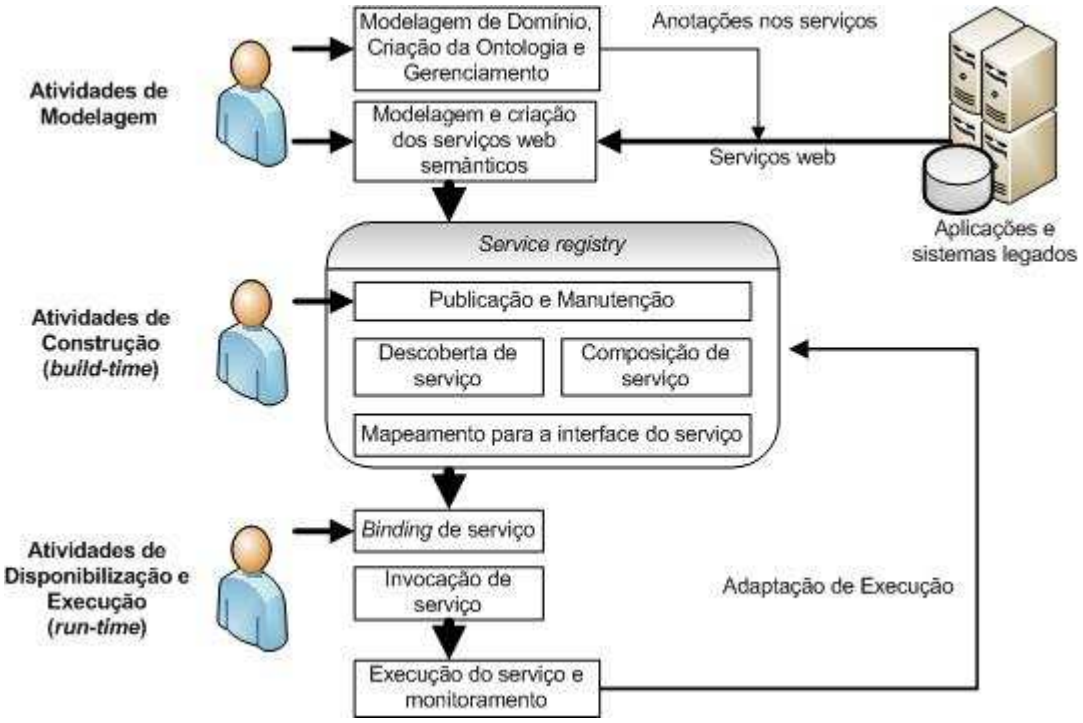

Figura 12: As atividades do ciclo de vida dos serviços web semânticos e sua interação (CARDOSO, 2007)

disponibilizados por um modelo semântico mais rico. Uma vez que os modelos semânticos possibilitam acordo comum no significado e intenção no uso de termos, as definições formais e informais para as entidades envolvidas no serviço são estabelecidas, alcançando assim o mínimo de ambigüidade por parte do provedor do serviço. O serviço web semântico pode então ser publicado num repositório, ou *service registry*.

Na etapa de descoberta, o requisitante do serviço pode descrever as condições usando termos a partir do modelo semântico. Técnicas de inferência podem ser utilizadas para encontrar a similaridade semântica entre a descrição do serviço e as condições do requisitante. Para os casos em que não ocorra uma compatibilidade direta, pode haver a composição de diversos serviços de uma forma que se estabeleça a funcionalidade requisitada. Para isso, os aspectos funcionais das anotações podem ser utilizados com a intenção de se estabelecer composições úteis. Para verificar se essas composições são legais e válidas, são utilizadas as semânticas de aspectos não funcionais dos serviços que as compõem.

Durante a etapa de disponibilização, as semânticas podem ser novamente utilizadas para localizar um serviço instanciado especifico que corresponda à interface definida. Durante a invocação, as semânticas podem ser utilizadas para representar as transformações dos dados. Nos casos de falha do serviço durante sua execução, as semânticas podem ser tomadas nova-

mente, desta vez para realizar a descoberta de serviços que possam substituir o que falhou. Essa tarefa é mostrada na figura 12, identificada como “Adaptação de execução”.

4.2.2 O papel das semânticas nos serviços web

Identificar similaridade entre os serviços web pode não ser uma tarefa trivial. Isso se deve ao fato de que a terminologia utilizada para descrevê-lo e a utilizada pelo requisitante podem ser diferentes. A estrutura e o tipo de informação presentes na descrição do serviço também devem ser levados em consideração e dessa forma semânticas explícitas podem assumir um papel importante na minimização do problema de ambigüidade. Cardoso (2007) identifica de forma específica o papel das anotações semânticas durante a descoberta, invocação, composição e monitoração de execução dos serviços.

Essa subseção tem como objetivo ilustrar a importância do papel que as semânticas desempenham na solução de ambigüidades relativas aos serviços web e nas etapas que compõem o seu ciclo de vida.

4.2.2.1 Descoberta de serviços

A descoberta automática de serviços envolve a tarefa de encontrar aquele que satisfaça um conjunto de requisitos tanto funcionais como não-funcionais. Essa descoberta pode ser feita a partir de um repositório de serviços central ou distribuído. A compatibilidade pode ocorrer de forma sintática quando é baseada em tipo e estrutura de dados ou de forma semântica, baseada no léxico ou por similaridades nominais e inferências a partir de ontologias associadas.

A descoberta de serviços pode ser utilizada para encontrar recursos satisfatórios e ou componentes de software que podem ser utilizados para implementar uma nova funcionalidade ou parte dela, sendo dessa forma necessária a utilização das anotações semânticas, onde desempenham papel fundamental. Alguns cenários possíveis onde podem ocorrer diferenças na descoberta de serviços devem ser considerados (CARDOSO, 2007):

- Serviços sintaticamente similares e semanticamente diferentes podem ocorrer quando dois serviços, por exemplo, possuem o parâmetro xsd:string de entrada e retorna um valor xsd:float, mas executam funções completamente diferentes - enquanto o primeiro verifica a quantidade em estoque o segundo verifica a disponibilidade em estoque. Além disso, um serviço pode esperar uma string simples como entrada enquanto outro pode esperar um documento XML.

- Serviços sintaticamente diferentes e semanticamente similares podem ocorrer quando dois serviços possuem sintaxe diferente mas executam a mesma função semântica. Como exemplo, dois serviços podem fornecer a quantidade disponível de um item de estoque sendo que o primeiro recebe os parâmetros necessários como código do produto, quantidade requisitada e data de entrega encapsulados num documento XML enquanto o segundo recebe todos esses parâmetros ou parte deles de forma separada.

- Serviços sintaticamente diferentes e com semântica aparentemente diferente ocorrem quando provedores de serviços optam por diferentes terminologias para expressar os mesmos conceitos.

- Serviços sintaticamente e semanticamente similares podem representar à primeira vista mesmas funcionalidades mas desempenham atividades totalmente diferentes - tanto os parâmetros quanto seus nomes podem ser similares.

Os cenários identificados possibilitam conclusões falsas entre os serviços – positivas falsas e negativas. A visão dos serviços web semânticos refere-se ao aperfeiçoamento das buscas e comparações entre os serviços baseada no contexto semântico e ontologias associadas, minimizando o quanto for possível a conclusão dessas positivas falsas e negativas.

4.2.2.2 Invocação de serviços

Para que as aplicações invoquem os serviços escolhidos de forma automática, um nível maior de combinação é necessário para identificar os mapeamentos atuais das interfaces.

Como exemplo, Cardoso (2007) cita que os códigos UPC – código de barras e SKU – número atribuído a um produto que serve para acompanhar o seu nível de estoque, podem ser combinados a nível semântico, já que ambos os conceitos identificam um produto de forma única. Entretanto essa combinação não pode ser executada devido às limitações da representação sintática, já que UPC é representado por 14 dígitos e SKU por 12. Um serviço intermediário de conversão ou mapeamento entre esses dois códigos pode ser necessário antes que a invocação do serviço inicialmente buscado possa ser feita.

A invocação de serviços web envolve em alguns casos, a criação de interfaces para efetuar o mapeamento entre o serviço do requisitante e o serviço escolhido. Muitas vezes, essa característica necessita a análise mais profunda da semântica do que aquela que acontece durante a etapa de descoberta. Os parâmetros de entrada *firstName* e *lastName* para alguns serviços devem ser concatenados de forma que possam representar o parâmetro de entrada *fullName*

em um outro serviço. Assim, um modelo semântico com o uso de mapeamentos podem ser aproveitados para possibilitar a derivação desses tipos de associação, facilitando dessa forma a invocação dos serviços.

A criação de novos serviços baseados em outros já disponíveis é feita atualmente pelos desenvolvedores a partir de vasta análise da documentação disponível, criando códigos para invocar serviços específicos. A visão da semântica para a invocação de serviços web possibilita diminuir essa sobrecarga (CARDOSO, 2007).

4.2.2.3 Composição de serviços

Os serviços web podem ser classificados de maneira geral como serviços simples ou complexos. A comunidade da Web Semântica define o serviço web simples ou atômico como um recurso invocado a partir de uma mensagem que, após executar suas tarefas, produz uma simples resposta para o requisitante, sendo que não existe interação entre o usuário e o serviço além da requisição e resposta. Dessa forma os serviços web simples são classificados como *stateless* (CARDOSO, 2007).

Por outro lado, um serviço complexo é composto por outros serviços complexos e simples. Essa classificação pode exigir uma interação entre o requisitante e o conjunto de serviços que está sendo utilizado, ou seja, um serviço web complexo possivelmente envolve fluxo de dados entre as etapas ou entre os serviços que o compõe. Espera-se que as linguagens utilizadas para a representação de serviços web semânticos suportem ambas as categorias de serviços identificadas (MARTIN *et al.*, 2004).

É importante ressaltar que, em vários domínios de negócio, possibilitar a composição de serviços web disponíveis para criação de uma nova funcionalidade é condição necessária para a representação de processos empresariais e fluxos de negócio.

Como a composição de serviços permite satisfazer as condições de uma determinada descrição de tarefa em alto nível, ela pode ser vista como um processo descrito em termos de um modelo de processo (CARDOSO, 2007). O modelo de processo e seus detalhes de estrutura para controle e fluxo de dados podem ser especificados por uma linguagem de modelagem de processos de negócio como BPEL (JORDAN; EVDEMON, 2007; RAO, 2004). A figura 13 ilustra como um processo pode ser composto do ponto de vista de serviços web, onde são necessários procedimentos de autenticação e criptografia.

Atualmente o desenvolvedor deve selecionar manualmente os serviços web necessários para realizar a composição de seu processo e assegurar que todas as informações trocadas entre

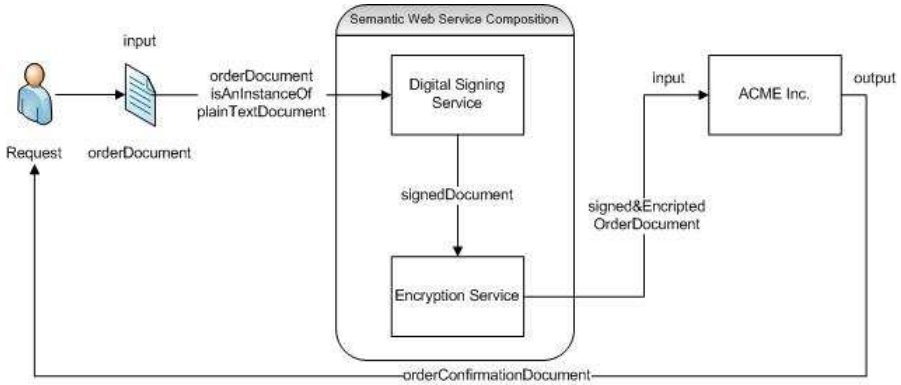

Figura 13: Um processo de autenticação e criptografia de um documento eletrônico a partir da composição de serviços web (CARDOSO, 2007)

eles estão sendo realizadas corretamente. Com a anotação semântica nos serviços web, as informações necessárias para selecionar e compor os serviços complexos ficam encapsualadas nos próprios serviços web. Dessa forma, permite-se escrever um código capaz de manipular essas representações de maneira conjunta com os objetivos do processo alcançando a composição de maneira automática (MARTIN *et al.*, 2004).

4.2.2.4 Monitoração da execução dos serviços

Mais uma vez, o papel das semânticas nos serviços web deve ser destacado, dessa vez no contexto da execução dos serviços.

A monitoração da execução dos serviços utiliza as anotações semânticas para o caso em que uma tarefa do processo de negócio seja interrompida por indisponibilidade de um ou mais serviços que ela representa, realizando as substituições necessárias por outros que sejam adequados e compatíveis à tarefa.

Essa dinâmica de execução e rearranjo caracteriza os sistemas como mais robustos e confiáveis.

4.3 O processo de anotações semânticas dos serviços web

O *processo de anotação semântica* permite efetuar a descrição dos serviços web em um nível maior de detalhamento. De maneira geral, as informações necessárias para a utilização correta de um serviço web envolvem detalhes como o que ele faz, o que assume, seu propósito e, em alguns casos, como alcança esse propósito. A proposta de um processo de anotação semântica

é realizada por Cardoso (2007), sendo que para tal, o mesmo introduz alguns aspectos gerais relevantes para que isso possa ser feito.

A descrição de um serviço web precisa incluir a exposição de informações técnicas numa ordem em que elas podem ser utilizadas. Isto inclui:

- O que ele faz;
- Como o serviço web é invocado, incluindo o protocolo de comunicação;
- Quais parâmetros a serem repassados para o serviço.

Se o serviço web faz parte de um processo organizacional, ou da composição de um serviço complexo, sua descrição ainda deve conter informações sobre:

- Qual protocolo deve ser seguido na sua utilização, ou seja, qual o padrão de troca de mensagens deve acontecer. Como exemplo um *login* de duas etapas: repassar o código inicial e assim que validado, repassar o código secundário.

Para atender às expectativas de um consumidor de serviço web, eles devem possuir então outras informações:

- Propriedades que dizem respeito a significados de segurança por exemplo.

Apesar de enumerar tais informações como sendo necessárias para a descrição de um serviço web deve-se atentar para o fato de que tal lista não está totalmente completa. Algumas outras informações relativas às descrições técnicas como transações, aspectos legais como assuntos contratuais para o caso de contratos eletrônicos para serviços web (FANTINATO, 2007), e descrições não-técnicas como requisitos não-funcionais, também se fazem necessárias.

A exploração da descrição técnica e não-técnica dos serviços web pode ser feita de várias maneiras: selecionar um serviço para que isso possa ser feito; compor essa funcionalidade com outro serviço ou; derivar propriedades relevantes para a composição de serviços web (como exemplo, combinar custo e validade para uma especificação dada) (CARDOSO, 2007).

O processo de anotação semântica para os serviços web requer em geral múltiplas fontes de informações para que seja realizado. As informações referentes a um serviço web podem ser reunidas a partir de código fonte de um serviço (desde que as anotações tenham sido feitas pelo provedor desse serviço), pela descrição e documentação de uma API, de toda documentação

textual de um serviço web disponível e a partir de outras fontes. Dependendo da estrutura dessas fontes, as anotações semânticas podem ser feitas de forma manual - quando texto é a forma de entrada; de maneira semi-automática - para algumas descrições de serviços web e; de maneira automática quando, como exemplo, interfaces Java constituem a forma de entrada essas anotações (CARDOSO, 2007).

4.3.1 Utilização de ontologias para anotações semânticas em serviços web

Com a utilização de ontologias é possível estabelecer semânticas formais para as anotações realizadas nos serviços web. De forma geral, a descrição semântica é realizada por metadados baseados em ontologias e conecta a propriedade de um serviço web com o conceito correspondente definido em uma ontologia.

Uma ontologia para serviços web estabelece conceitos gerais como serviço ou operação assim como relações existentes entre esses conceitos. Os metadados que descrevem os serviços web podem instanciar conceitos disponíveis na ontologia associada. A conexão entre o serviço web e os conceitos da ontologia possibilita aos desenvolvedores efetuar a comparação entre os metadados de diferentes serviços descritos pelas mesmas ontologias ou por aquelas que sejam similares e assim estabelecer maior possibilidade de uso, reuso e verificação do serviço.

4.3.2 Realizando anotações em serviços web

As anotações semânticas possibilitam às máquinas realizar um processamento de informações de gerenciamento, alcançada a automação esperada. Para isso, deve-se analisar diferentes aspectos a partir de quatro questões relevantes à esse contexto (CARDOSO, 2007):

1. Quais fontes de dados contém informações relevantes para a especificação dos serviços web?
2. Quais descrições necessárias para uso e reuso dos serviços web?
3. Como os metadados devem ser representados?
4. Como a descrição pode ser explorada?

As subseções seguintes procuram explorar de maneira geral as características das fontes de informações e as semânticas das descrições, buscando avaliar os itens enumerados. Possibilita assim expor o processo de anotação semântica para os serviços web proposto por Cardoso (2007) logo em seguida.

4.3.2.1 Fontes de informação

As fontes de informações para descrição dos serviços web representam ferramentas importantes para que essa tarefa seja realizada. Nessa subseção são indicadas as facilidades e dificuldades que cada uma delas fornece (CARDOSO, 2007).

A documentação textual de um serviço web é uma das fontes de dados onde o processamento de máquina é impossível de ser realizado por apresentar linguagem natural e não formalizada.

Os códigos fontes possuem todas as informações necessárias referentes às operações, incluindo todas as entradas e saídas, mas essas informações podem ser extraídas de diferentes maneiras entre linguagens de programação - enquanto a linguagem Java revela os tipos de dados das entradas e saídas dos métodos, outras linguagens podem não realizar isso. Além do mais, o código fonte define a programação lógica do serviço, revela pré-condições e efeitos alcançados a partir de uma operação.

Os comentários incluídos nos códigos fontes apresentam informações que podem ser utilizadas para determinar detalhes sobre operações assim como a funcionalidade de todo o serviço web. Como desvantagem, se um código fonte não estiver acessível, os comentários conseqüentemente não estarão acessíveis.

As descrições de API consistem normalmente de declaração de interfaces e explicações em linguagem natural, consistindo assim outra fonte de informação. Ao contrário dos códigos fontes, a descrição de API pode ser mais detalhada e focada em métodos mais relevantes. As explicações contidas nas APIs em linguagem natural apresentam as mesmas desvantagens que a documentação textual de um serviço.

As especificações de software e os diagramas UML (OMG, 2008b) são fontes de informações comparadas às descrições por API.

Outras fontes de informação envolvem contratos como o acordo entre a empresa e o usuário que permite a utilização de um programa - *user license agreements, EULA*. Como são escritos em linguagem natural, o processamento de máquina para essa fonte de informações também apresenta desvantagem, assim como qualquer outra fonte escrita em linguagem natural.

O conhecimento de domínio da aplicação ao qual o serviço web se destina é necessário do ponto de vista técnico, para realizar sua devida implementação. Esse conhecimento também representa informação que usualmente não é fornecida com um serviço web e pode ser útil na especificação de suas propriedades semânticas.

4.3.2.2 Semânticas de descrições

O estudo do processo de anotações semânticas em serviços web engloba a análise das informações disponibilizadas e como podem ser classificadas dentro desse processo. Cardoso (2007) distingue quatro categorias de semânticas: semânticas de dados, semânticas de protocolo, semânticas funcionais e semânticas não-funcionais.

As *semânticas de dados* incluem a descrição dos sentidos de todas as entradas e saídas de um serviço web. Por exemplo, um parâmetro válido do tipo data em que o formato específico seja mês e ano da forma "mm/aaaa". Para especificar as semânticas de dados das operações do serviço web, devem ser construídas conexões entre suas entradas e saídas e seus tipos de dados e os correspondentes conceitos de uma ontologia. Essas conexões são realizadas ao instanciar conceitos apropriados da ontologia para cada parâmetro e operação do serviço web.

Para a categoria das *semânticas de protocolo*, procura-se especificar semânticas num contexto em que o protocolo define as dependências entre serviços web e também dependências entre suas operações, como ordem de operações incluindo seqüências, laços, *joins* entre outros. De maneira geral, as semânticas de protocolo descrevem o fluxo de tarefas (*workflow*) e de dados (*dataflow*) relevantes para a execução do serviço web.

A categoria das *semânticas funcionais* tenta abranger todas as semânticas que se referem à funcionalidade de um serviço web. Pode-se afirmar que antes da execução de uma operação são exigidas que certas condições devam ser atendidas (pré-condições) e também que a execução de uma operação pode refletir "efeitos".

- As pré-condições e efeitos de uma operação podem ser utilizadas para expressar as *semânticas de protocolo* e as *semânticas funcionais*. Dentro do contexto de semânticas funcionais, as pré-condições podem ser internas - quando são verificadas pela lógica da aplicação; e externas - quando dependem de fatores externos ao serviço e referem-se à atributos que devem ser verificados antes que o serviço web seja invocado, possibilitando que a operação seja realizada com sucesso. As pré-condições internas ainda podem ser subdivididas em implícitas - conseqüência de padrões, tipos e definições, isto é, quando se assume que o tipo passado no parâmetro está de acordo com o especificado; e explícitas - que são especificadas para uma determinada operação, por exemplo, para a verificação da data de validade presume-se que a data passada como parâmetro a ser comparada com a data atual, seja no futuro.

- Os efeitos descrevem valores de saída e o impacto da execução dos serviços web. Pode-se afirmar como exemplo de efeito que numa operação de transferência monetária, uma conta têm o seu saldo aumentado enquanto outra tem o seu saldo diminuído.

A categoria das *semânticas não-funcionais* tenta abranger as semânticas das propriedades não funcionais dos serviços web. Como propriedades não-funcionais, Cardoso (2007) cita segurança e qualidade de serviço. Para alcançar uma interpretação correta de propriedades não-funcionais, descrições lógicas como padrões devem ser consideradas. Por exemplo, se um padrão define que determinado algoritmo para criptografia deve ser usado para que a chave tenha um tamanho específico então essa informação pode ser interpretada como propriedade de segurança.

O conjunto das categorias expostas representam as *especificações semânticas de serviço* e, reunidas, contêm toda a informação necessária para a publicação e descoberta de um serviço web. A combinação das *semânticas de dados* com a localização do serviço especificado por uma URI e as semânticas utilizadas nos protocolos representam um conjunto especial das *especificações semânticas de serviço* e descrevem como acessar um serviço web.

4.3.2.3 O processo de anotação

A coleta de informações para especificar semanticamente um serviço foi verificada nas subseções anteriores (4.3.2.1 e 4.3.2.2) sendo que dessa forma é possível introduzir um processo, definido em duas etapas, que suporte a anotação semântica em serviços web (CARDOSO, 2007).

O passo inicial definido pela literatura é a anotação não-semântica. Como artefatos de entrada para essa etapa, definem-se diferentes fontes de informações, como documentação e interface referente às operações que serão desempenhadas pelos serviços web. A anotação não-semântica pode ser realizada de forma semi-automática sendo suportada por diferentes ferramentas. A subseção 2.4 apresenta a anotação não-semântica de um serviço web com a utilização de uma ferramenta para geração semi-automática de sua descrição. O resultado dessa etapa são arquivos de descrição WSDL, como o da listagem 2.2, e outros tipos de documentos como descrições API.

A segunda etapa consiste na anotação semântica, onde são utilizados os mesmos documentos da primeira etapa como artefatos de entrada além da descrição WSDL gerada anteriormente que, combinados com ontologias relevantes - ontologias de serviços web e ontologias de domínio, resultam numa especificação semântica do serviço web - o *serviço web semântico*. Essa segunda etapa é dividida em outras sub etapas.

A primeira sub etapa classifica o serviço web, resultando em sua associação com algum domínio. Aqui, o desenvolvedor do serviço classifica-o a partir de decisões baseadas em seu conhecimento (*background knowledge*) e, quando possível, com a ajuda de ferramentas. Essas ferramentas podem aplicar algoritmos de classificação de textos (HEß; KUSHMERICK, 2004) nos documentos WSDL e dessa forma associá-los a um domínio específico, além de classificar a funcionalidade das operações e seus parâmetros de entrada e saída. A associação do serviço web com um domínio pode ser feita de forma manual ou semi-automática. Em seguida, é necessário selecionar uma ontologia de domínio ou desenvolver uma nova para que seus conceitos possam ser conectados às descrições do serviço web. Os metadados referentes aos tipos dos elementos das interfaces de operação devem ser mapeados para os conceitos da ontologia, resultando numa descrição de como o serviço web pode ser acessado. As especificações de protocolo e das semânticas funcionais fazem parte da sub etapa seguinte: se as semânticas de dados já foram especificadas na sub etapa anterior, então partes do protocolo e da funcionalidade foram especificadas implicitamente a partir das semânticas adicionadas aos parâmetros de entrada. Dessa forma, pode-se derivar quais informações devem ser reunidas antes que uma operação seja invocada (protocolo) e identificar partes das semânticas funcionais.

Todas as fontes de informações citadas na subseção 4.3.2.1 tornam-se artefatos de entrada para a etapa seguinte do processo: adicionar outras semânticas para o serviço web identificando dependências e definindo relacionamentos entre elementos. Como exemplo, se for identificado que a dependência entre uma informação de validade qualquer e a data atual é necessária então especifica-se o relacionamento éValido entre esses elementos. Já as especificações de pré-condições devem ser realizadas pelo desenvolvedor em um nível mais baixo de abstração, por se revelar como uma tarefa mais complexa.

A última etapa para o processo proposto é a anotação das semânticas não-funcionais do serviço web. Algumas das propriedades não-funcionais simples como o nome do serviço podem ser extraídas automaticamente do arquivo de descrição WSDL. Propriedades não-funcionais mais complexas, como as que envolvem características de semântica devem ser descritas pelo desenvolvedor.

Com o processo definido e suas etapas identificadas, Cardoso (2007) conclui que a descrição de um serviço contém todas as propriedades relevantes para alavancar o reuso e habilitar melhor suporte através de ferramentas apesar de ainda requerer um grande esforço do desenvolvedor. A figura 14 condensa de maneira geral o processo de anotações semânticas para os serviços web.

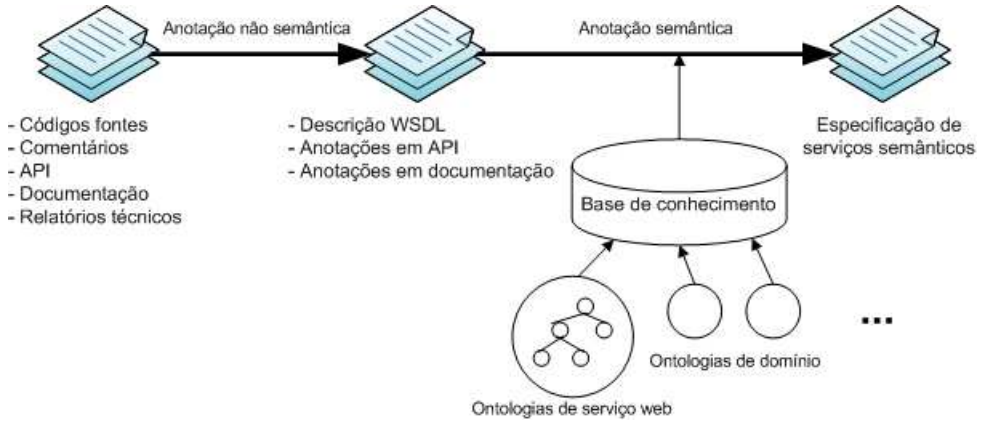

Figura 14: O processo de anotações semânticas para serviços web proposto por Cardoso (2007)

4.4 Considerações finais

A contextualização e a definição de serviços web semânticos possibilitam identificar as informações necessárias para sua descrição semântica e propor um processo para que essa atividade seja desempenhada de forma padronizada pelos desenvolvedores.

Devem ser identificadas as atividades que fazem parte de um ciclo de vida para os serviços web semânticos. Essas atividades devem possibilitar uma integração consistente e, tarefas automatizadas como descoberta, invocação, composição e monitoração da execução dos serviços web, devem ser associadas com possíveis anotações semânticas para que sejam realizadas.

As anotações semânticas devem ser realizadas de maneira formal e utilizando linguagem que possibilite processamento de máquina. Como suporte a um processo de anotações semânticas, toda a documentação disponível referente a um serviço, além do conhecimento de domínio, devem ser disponibilizados.

O processo de anotação semântica baseado em documentos, descrição WSDL, modelos de domínios além de ontologias para serviços web é proposto por Cardoso (2007).

5 *Implementação de Serviços Web Semânticos*

5.1 Introdução

No contexto do desenvolvimento de serviços web semânticos, existem algumas abordagens propostas pela comunidade para que as mesmas sejam utilizadas como padrão. Cada uma apresenta suas vantagens e desvantagens, mas todas as abordagens procuram aproximar as semânticas de seus papéis necessários (identificados na subseção 4.2.2) e explorar ao máximo as descrições em um serviço web - as semânticas de descrição, subseção 4.3.2.2.

É necessário destacar que todas as abordagens têm como objetivo possibilitar a descoberta, invocação, composição e monitoração da execução automáticas dos serviços web, através de linguagens formais que possam ser inferidas e processadas por máquina.

Esse capítulo apresenta de forma geral as principais abordagens contidas na literatura, identificando elementos de sua arquitetura, características e quando possível as vantagens e desvantagens.

5.2 OWL-S - *Semantic Markup for Web Services*

A abordagem OWL-S (MARTIN *et al.*, 2004) define uma ontologia para serviços web que consiste de um conjunto de classes e propriedades básicas para declaração e descrição de serviços, baseada na linguagem OWL (SMITH; WELTY; MCGUINNESS, 2004), já que essa linguagem fornece representação apropriada e compatível com a Web. O termo *upper ontology* é utilizado nessa abordagem OWL-S para identificar uma *super-ontologia* que seja padrão para definição e descrição dos serviços web.

O objetivo da proposta de OWL-S destacado em Martin *et al.* (2004) é suportar ambas as categorias de serviços web: tanto os serviços simples quanto os serviços complexos, sendo esse último tipo o que tem motivado o desenvolvimento dessa abordagem.

5.2.1 Uma ontologia para serviços web (*upper-ontology*)

A estrutura da ontologia é proposta a partir da necessidade de fornecer três tipos essenciais de conhecimento sobre um serviço: o que o serviço provê para seus possiveis clientes, como ele é utilizado e como alguém pode interagir com ele.

Assim, ela é montada a partir da classe *Service*. Associa-se então cada um dos conhecimentos buscados, citados no parágrafo anterior, às propriedades *presents*, *supports* e *describedBy* daquela classe. A propriedade *presents* está associada à classe *ServiceProfile*; a propriedade *supports* está associada à classe *ServiceGrounding* enquanto *describedBy* associa-se com a classe *ServiceModel*. Todas essas classes são atributos de alcance da classe *Service* a partir dessas propriedades. A figura 15 apresenta a estrutura da ontologia de serviços.

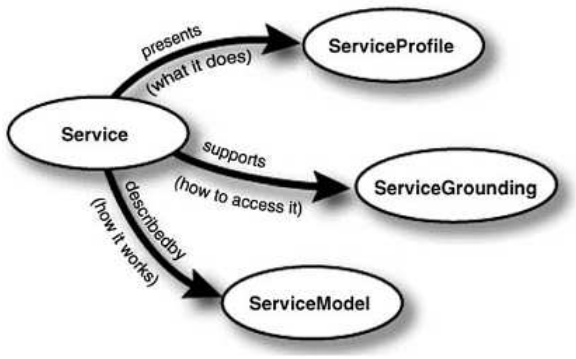

Figura 15: A ontologia de serviço em um nível mais alto de abstração (MARTIN *et al.*, 2004)

5.2.2 A classe *ServiceProfile*

A classe *ServiceProfile* possibilita especificar de maneira geral o que o serviço web faz. Ela representa as interfaces do serviço incluindo suas entradas, saídas, pré-condições e efeitos (IOPEs) (CARDOSO, 2007). As descrições disponibilizadas nessa classe desempenham um papel semântico fundamental para a descoberta de serviços.

Para desempenhar o papel proposto para essa classe, ela inclui informações como requisitos necessários para que ele seja acessado de maneira correta, a organização que provê o serviço, a função que o serviço realiza e um conjunto de atributos que especificam as carac-

terísticas do serviço (categoria e a avaliação de sua qualidade) (MARTIN *et al.*, 2004; DAVIES; STUDER; WARREN, 2006).

5.2.3 A classe *ServiceModel*

Os detalhes de funcionamento interno do serviço são disponbilizados por essa classe, já que se torna necessário especificar uma maneira detalhada como a interação com esse serviço deve acontecer, descrevendo de forma semântica o conteúdo necessário para sua solicitação (DAVIES; STUDER; WARREN, 2006).

A interação com um serviço pode ser vista como um processo. Um processo não é um programa a ser executado, mas uma especificação de como o cliente deve interagir com um serviço. Um processo atômico é a descrição de um serviço que ao receber uma mensagem, talvez complexa, retorna outra como resposta, complexa ou não. Um processo composto (*composite process*) é aquele que mantém algum estado, onde cada mensagem do cliente possibilita um avanço nas etapas que o compõem (MARTIN *et al.*, 2004).

De maneira geral, OWL-S considera que um processo pode ter vários parâmetros de entrada que, sob certas condições, são necessários para que possa ser iniciado. Podem ocorrer várias pré-condições que, devidamente atendidas na ordem, garantem que o processo seja iniciado e executado com sucesso. Para a abordagem OWL-S, se as pré-condições não são atendidas, os efeitos referentes a execução do processo são desconhecidas (DAVIES; STUDER; WARREN, 2006).

Para processos compostos, OWL-S permite a sua decomposição em outros processos menores ou atômicos utilizando controles de construção. Entre os controles de construção suportados pela OWL-S podem ser citados: *Sequence* - permite que um conjunto de processos seja executado sequencialmente; *Split* - permite que um conjunto de processos possa ser executado de forma paralela; *Split + Join* - consiste na execução paralela de um pacote de processos com sincronização; *Any-Order* - permite a execução dos processos em qualquer ordem; *If-Then-Else* - um controle de construção que possui as propriedades *if-Condition, then* e *else* possibilitando construção condicionada; *Iterate* - um controle de construção que permite repetidas vezes a execução de um processo e; *Repeat-While* e *Repeat-Until* - onde cada um dos controles de construção iteram a partir de uma condição sendo verdadeira ou falsa (DAVIES; STUDER; WARREN, 2006).

Dessa forma, possibilita-se efetuar a composição de serviços para uma tarefa específica e durante a execução do serviço, coordenar as atividades de diferentes participantes além de monitorar sua execução.

5.2.4 A classe *ServiceGrounding*

Essa classe define as conexões necessárias com a descrição WSDL, para usar o seu modelo de invocação (CARDOSO, 2007). Além disso, fornece os detalhes necessários sobre os protocolos de transporte, ou seja, como um cliente pode realizar o acesso ao serviço. Especifica ainda detalhes como formato das mensagens e número de portas utilizadas para conexão com o serviço.

5.2.5 Considerações sobre OWL-S

A abordagem OWL-S propõe uma nova maneira de descrever serviços web e assume que somente a linguagem OWL é utilizada para representar modelos de domínios. Além disso, as descrições contidas na classe *ServiceModel* que se referem aos dados de entrada e saída devem estar replicados no documento WSDL, levando assim ao inconveniente de se criar múltiplas definições de descrição para o mesmo serviço (AKKIRAJU *et al.*, 2005).

5.3 WSDL-S - *Web Service Semantics*

A abordagem WSDL-S provê uma especificação para anotações semânticas diretamente nos documentos WSDL, através da descrição de elementos semânticos para cada um dos já conhecidos dessa estrutura (subseção 2.3.3). Assim, esta proposta fornece uma evolução compatível com os padrões de serviços web existentes e define a versão 2.0 do padrão WSDL.

Nessa abordagem assume-se que os modelos formais semânticos relevantes para os serviços já existem, sendo que esses modelos podem ser mantidos fora do documento WSDL, possibilitando ainda que a utilização de padrão para sua representação seja independente, caracaterizando uma vantagem da proposta WSDL-S.

A conexão entre os elementos da descrição WSDL e os conceitos dos modelos de domínio são representados pela figura 16.

Além de possibilitar representação de modelos semânticos de maneira independente, como visto no parágrafo anterior, uma outra vantagem sobre a abordagem OWL-S é permitir

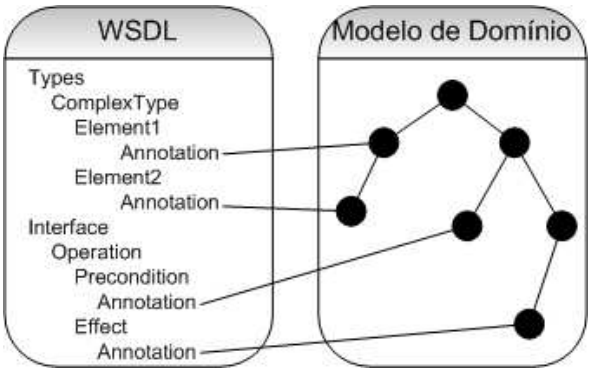

Figura 16: A associação entre os elementos da descrição WSDL e os conceitos disponíveis em modelos de domínio externos (AKKIRAJU *et al.*, 2005)

que tanto os detalhes para descrever as operações quanto as semânticas do serviço web podem ser feitas apenas no documento WSDL, de forma compatível, sendo esse padrão já conhecido pelos desenvolvedores. Isto possibilita fácil atualização das ferramentas de WSDL para suportar essa abordagem incremental.

Os tipos de informações semânticas aplicáveis na descrição dos serviços web considerados nessa proposta, abrangem os conceitos definidos pela comunidade da Web Semântica para OWL-S (MARTIN *et al.*, 2004) além de outras, como WSMO (BRUIJN *et al.*, 2005). Assim, como na abordagem anterior, as informações semânticas consideradas para WSDL-S incluem definições de pré-condições, entrada, saída e efeitos da operação do serviço web (AKKIRAJU *et al.*, 2005).

5.3.1 Requisitos para abordagem WSDL-S

Os seguintes princípios são definidos para que se possa definir um *framework* para os serviços web semânticos, sendo que eles são seguidos para o desenvolvimento do padrão WSDL-S (AKKIRAJU *et al.*, 2005):

- Construção baseada em padrões de serviços web já existentes de forma que as estruturas atuais (serviços web disponibilizados e utilizados) não sejam interferidas;
- O mecanismo de anotações semânticas deve possibilitar ao desenvolvedor escolher a linguagem para representação do domínio, oferencendo assim maior flexibilidade no desenvolvimento dos serviços web semânticos;
- As anotações semânticas devem possibilitar a associação de múltiplas descrições especificadas em diferentes linguagens de representação semântica, possibilitando assim que

a busca dos serviços web semânticos possa ser realizada por diferentes mecanismos de descoberta (*discovery engines*);

- Suporte a anotações semânticas em serviços web onde os tipos de dados possam ser descritos em *XML Schema*, já que a definição de dados utilizando esse esquema tem sido muito utilizada e;

- Possibilitar suporte para um mecanismo de mapeamento rico entre os tipos de dados dos serviços web descritos em *XML Schema* (*web service schema types*) e conceitos de ontologias.

5.3.2 As anotações semânticas na descrição WSDL

A descrição WSDL 2.0, possui os seguintes elementos para representar a descrição do serviço web: *interface, operation, message, binding, service* e *endpoint*. Enquanto os três primeiros elementos referem-se a definições abstratas do serviço os seguintes referem-se à sua implementação. A abordagem tratada foca nos três primeiros elementos uma vez que anotações semânticas para as definições abstratas de um serviço web permitem a sua descoberta, composição e invocação dinâmica. Dessa forma, são criados elementos de extensão para *interface, operation* e *message* sendo suas anotações semânticas realizadas a partir de URIs que fazem referências a modelos de domínio definidos para o serviço (AKKIRAJU *et al.*, 2005).

O atributo de extensão *modelReference* determina a associação entre uma entidade do WSDL e um conceito disponível em algum modelo semântico. Esse atributo pode ser adicionado para um tipo complexo (*complexType*), *element, operation* assim como os elementos de extensão *precondition* e *effect*

O atributo de extensão *schemaMapping* pode ser adicionado para elementos definidos em XSD e elementos *complexType* para que as diferenças estruturais existentes entre os tipos definidos por *XML Schema* e os conceitos disponíveis nos conceitos dos modelos de domínio possam ser manipulados e mapeados.

O elementos *precondition* e *effect* foram introduzidos como elementos-filho de *operation* e são utilizados durante a descoberta do serviço. Eles podem ser anotados semanticamente com o atributo *modelReference*.

O atributo de extensão *category* faz parte das propriedades do elemento *interface* e consiste numa informação de categorização que pode ser utilizada durante o processo de publicação do serviço web.

5.3.3 Anotações semânticas em tipos complexos

Os tipos complexos de um serviço web podem ser anotados de múltiplas formas. A abordagem WSDL-S propõe dois esquemas alternativos para que isso seja feito: anotação *bottom level* e anotação *top level*.

Na anotação *bottom level* todos os elementos folha de um tipo complexo, *complexType*, podem ser anotados. Por ser um esquema simples, oferece maior vantagem. Nos casos em que não há conceito semântico relacionado, a anotação não precisa ser realizada. Como desvantagem, esse esquema assume um relacionamento um-para-um entre seus elementos e os conceitos do domínio. Assim, quando associações são do tipo um-para-muitos e muitos-para-um as anotações semânticas em elementos folha do *complexType* não são possíveis.

Quando os próprios elementos *complexType* são associados a algum conceito do modelo de domínio através do atributo *modelReference* tem-se a anotação semântica *top-level*. Dessa forma, consegue-se estabelecer associações um-para-muitos e muitos-para-um, o que é apontado como vantagem desse esquema. Como desvantagem, essa forma de anotação tende a ser complexa.

A anotação semântica em um nível mais alto, para o elemento *complexType* pode ser utilizada durante a descoberta do serviço, sendo que passa a ser possível determinar de maneira preliminar em qual ponto duas estruturas que estão sendo avaliadas têm relação semântica. Como exemplo, um tipo complexo “chip” em um esquema XSD pode ter uma anotação semântica “*Microprocessor chip*” a partir de uma ontologia (AKKIRAJU *et al.*, 2005).

5.3.4 Considerações sobre WSDL-S

A proposta WSDL-S foi submetida ao W3C em 2005 (W3C, 2008). Por apresentar vantagens já citadas como amplo conhecimento de WSDL por desenvolvedores para descrição de serviços web minimizando necessidade de treinamentos; possibilidade de atualizar ferramentas existentes para descrição WSDL com pouco esforço; utilização de modelos de domínios externos e já definidos em linguagens independentes e; transformações XSLT entre conceitos, essa proposta define-se como a mais promissora entre as disponíveis para tornar-se padrão na descrição de serviços web semânticos.

5.4 WSMO - *Web Services Modeling Ontology*

Além dos objetivos expostos anteriormente para as abordagens de serviços web semânticos, a iniciativa WSMO (BRUIJN *et al.*, 2005) ainda tem como proposta a padronização unificada a partir de um *framework* para permitir suporte à modelagem conceitual e representação formal de serviços assim como sua execução automática (DAVIES; STUDER; WARREN, 2006).

A abordagem WSMO é composta pelos elementos WSMO (*Web Service Modeling Ontology*) - um modelo conceitual para serviços web semânticos; WSML (*Web Service Modeling Language*) - uma linguagem que possibilita uma sintaxe formal e semântica para WSMO e; WSMX (*Web Service Modeling Execution Environment*) - um ambiente de execução referência para a implementação de WSMO, que ainda oferece suporte para serviços web semânticos. A figura 17 apresenta esses elementos.

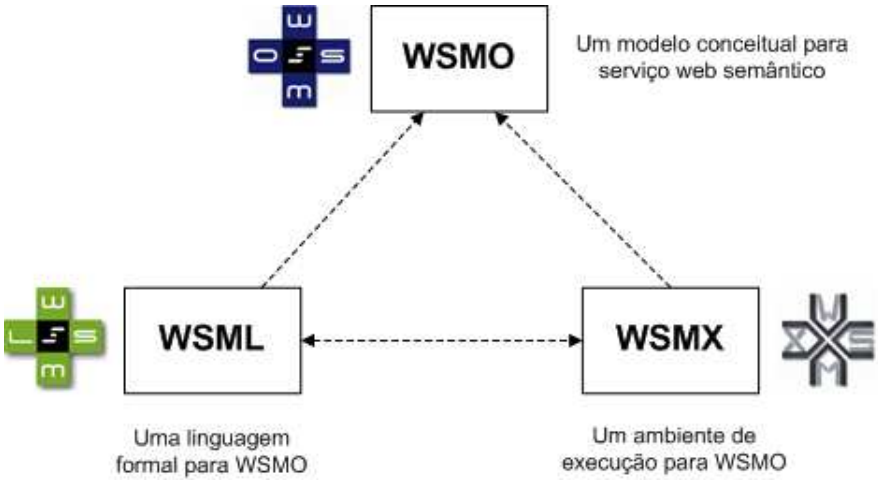

Figura 17: Os elementos da abordagem WSMO (BRUIJN *et al.*, 2005)

5.4.1 O modelo conceitual - *Web Services Modeling Ontology* (WSMO)

O projeto propõe quatro elementos principais para WSMO: *ontologies* - que permitem especificar a terminologia usada por outros elementos WSMO, definir uma ontologia e importar outras (no caso de ontologias importadas, exige a especificação de um objeto verificador a ser utilizado nos casos em que deva ocorrer alinhamento e união entre elas, chamado *OOMediators*); *web services* - possibilitam descrever aspectos de funcionalidade e de comportamento de um serviço web; *goals* - especificam objetivos que um cliente possa ter enquanto efetua a consulta a um serviço web, descrevendo aspectos relacionados aos pedidos dos usuários que dizem respeito a funcionalidade buscada e de comportamento e; *mediators* - que apontam para tratamento automático dos problemas de interoperabilidade entre diferentes elementos WSMO (DAVIES; STU-

DER; WARREN, 2006; CARDOSO, 2007). Esses elementos são definidos como classes e possuem respectivamente as assinaturas:

- **Class** ontology
- **Class** webService
- **Class** goal
- **Class** mediator

Cada uma dessas classes possuem atributos como *hasNonFunctionalProperties*, *importsOntology* e *hasInterface*, que devem ser definidos durante o processo de desenvolvimento do modelo.

Assim como o WSDL-S, o WSMO possui um conjunto que princípios que norteiam o seu desenvolvimento. Alguns desses princípios são listados em seguida:

- Conformidade à Web, herdando conceitos como URI (*Universal Resource Identifier*), adotando definições como *namespaces* e suporte ao XML e outras tecnologias recomendadas pelo W3C;
- Baseado em ontologia para modelagem de domínio e representação de conhecimento;
- Desacoplamento de recursos - cada recurso deve ser especificado de forma independente;
- Como complemento do desacoplamento de recursos, a intervenção em problemas que possam ocorrer entre tipos heterogêneos como dados, ontologias, protocolos e processos é necessária, criando assim um mecanismo de tratamento dessa situação;
- Papéis bem definidos para diferenciar solicitações de clientes e serviços disponíveis e;
- Diferenciação entre as descrições dos elementos dos serviços web semânticos e a tecnologias de execução.

5.4.2 A linguagem - *Web Service Modeling Language* (WSML)

WSML é uma linguagem formal para descrição dos elementos baseados em WSMO, *ontologies*, *goals*, *web services* e *mediators* e provê um *framework* coerente em que une as tecnologias web com diferentes paradigmas de linguagem lógica conhecidas (DAVIES; STUDER; WARREN, 2006).

Alguns estudos ainda estão sendo desenvolvidos para que se consiga relacionar WSML com os padrões existentes.

A linguagem WSML apresenta variantes baseadas em formalismos lógicos: WSML-Core, WSML-DL, WSML-Flight, WSML-Rule e WSML-Full.

5.4.3 O ambiente de execução - *Web Service Modeling Execution Environment* (WSMX)

O WSMX representa um ambiente de execução que possibilita a descoberta, seleção, intervenção (*mediation*) e invocação de serviços web semânticos. É baseado no modelo conceitual fornecido pelo WSMO, que também serve como referência para implementação desse ambiente.

Um dos papéis do WSMX é servir como um artefato de teste para WSMO, além de confirmar sua viabilidade ao mesmo tempo que possibilita alcançar interoperabilidade dinâmica de serviços web semânticos.

5.4.4 Considerações sobre WSMO

A abordagem WSMO propõe novos conceitos e um *framework* para a definição de serviços web e ontologias a partir de um conjunto de linguagens baseadas em formalismo lógico. Define uma arquitetura completa, desde a definição de domínios de conhecimento até a plataforma de execução e monitoração de serviços web semânticos.

Em comparação a OWL-S, que não faz distinção entre tipos de serviços web, WSMO também oferece suporte a todos os tipos de serviços, mas exige a especificação de programas de mapeamentos para resolver conflitos durante a operação entre eles, o que pode se tornar uma tarefa complexa.

5.5 A anotação semântica para o serviço *inclusaoAluno*

Como estudo complementar e com finalidade de ilustrar o trabalho, propõe-se efetuar a anotação semântica de um serviço web. Para isso, será utilizado o serviço implementado na seção 2.4 e a ontologia desenvolvida na seção 3.5.

A abordagem escolhida para a descrição semântica do serviço web foi a WSDL-S, pelos mesmos motivos citados como vantagens de sua aplicação:

- O modelo de domínio já é conhecido e especificado externamente ao serviço, a manutenção pode ser realizada sem maiores esforços de desenvolvimento;
- O serviço web já está especificado e não haverá a necessidade de modificar a estrutura na qual ele é disponibilizado;
- O padrão utilizado já é de domínio para desenvolvimento e;
- A especificação para WSDL-S oferece suporte para anotações semânticas na versão 1.1 do WSDL sem necessidade de modificação do documento.

Essas vantagens reiteram a escolha da proposta WSDL-S para descrição de serviços web semânticos.

5.5.1 Especificando semânticas para o serviço

Inicialmente são incluídos os prefixos *OntologyUFJF* e *wssem* para o documento WSDL. O *namespace OntologyUFJF* faz referência a um modelo de domínio, a ontologia especificada em OWL. O *namespace wssem* refere-se ao arquivo de defingção XSD que especifica os elementos de extensão para anotação semântica na descrição WSDL. A listagem 5.1 apresenta o código resultante dessa etapa, com destaque para as linhas 8 e 9.

Listagem 5.1: O elemento inicial do documento WSDL com *namespaces* de semântica adicionados

```
1  <wsdl:definitions targetNamespace="http://service.ufjf.org.br"
2          xmlns:apachesoap="http://xml.apache.org/xml-soap"
3          xmlns:impl="http://localhost:8080/TSUFJFWeb/service/"
4          xmlns:intf="http://localhost:8080/TSUFJFWeb/service/"
5          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
6          xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
7          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
8          xmlns:OntologyUFJF="http://localhost:8080/TSUFJFManager/ontology
               /2008/6/5/tsufjf_ontology.owl#"
9          xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics.xsd">
10 </wsdl:definitions>
```

5.5.1.1 Inserindo semântica de dados

Para efetuar a anotação semântica num tipo *complexType*, WSDL-S permite as anotações do tipo *bottom-level* e *top-level*. Os dois tipos serão abordados nessa implementação.

Modificando a idéia inicial do serviço web *inclusaoAluno* e, assumindo que o seu parâmetro de entrada passa a ser um conceito Aluno, a anotação semântica pode ser feita através do

atributo *modelReference* do elemento folha *xs:element* de *complexType*. A listagem 5.2 apresenta uma abordagem *bottom-level* em que o parâmetro do serviço está conectado a um conceito Aluno.

Listagem 5.2: O elemento *complexType* com anotação semântica do tipo *bottom-level*

```
<xs:complexType name="insereAlunoRequest">
        <xs:all>
                <xs:element name="aluno" wssem:modelReference="OntologyUFJF
                    #Aluno" />
        </xs:all>
</xs:complexType>
```

Outra alternativa de realizar anotação semântica do parâmetro de um serviço web é definir o atributo *wssem:modelReference* para o elemento *complexType*, conectando-o a um conceito disponível na ontologia. Esse tipo de anotação é definido como *top-level* por apresentar uma característica de anotação de mais alto nível. A listagem 5.3 apresenta a descrição semântica realizada.

Listagem 5.3: O elemento *complexType* com anotação semântica do tipo *top-level*

```
<xs:complexType
        name="insereAlunoRequest"
        wssem:schemaMapping =
                "http://localhost:8080/TSUFJFWeb/services/model/
                    tsufjf_OOAluno.xsl"
        wssem:modelReference="OntologyUFJF#Aluno">
        <xs:all>
                <xs:element name="nome" type="string" />
                <xs:element name="senha" type="string" />
                <xs:element name="endereco" type="string" />
                <xs:element name="codCurso" type="string" />
        </xs:all>
</xs:complexType>
```

Ao realizar uma anotação do tipo *top-level* para *complexType*, seus elementos filhos ficam sem correspondência semântica. Para resolver essa situação, eles podem ser mapeados através de um arquivo de transformação XSLT, indicado pelo atributo *wssem:schemaMapping*.

De forma geral:

- O *complexType* é conectado ao conceito Aluno da ontologia;
- Os elementos internos do *complexType* Aluno passam a ser mapeados pelo atributo *wssem:schemaMapping*, utilizando transformação XSLT, sendo esses elementos definidos em um documento XSD;
- Os elementos definidos no documento XSD são atributos de um tipo complexo Aluno que, por sua vez, está conectado ao conceito respectivo na ontologia.

Dessa forma, todos os elementos do parâmetro de entrada passam a estar anotados semanticamente. A figura 18 apresenta um diagrama de conexão entre os conceitos do tipo complexo Aluno e as definições do atributo nome. A listagem 5.4 apresenta o arquivo de mapeamento XSLT.

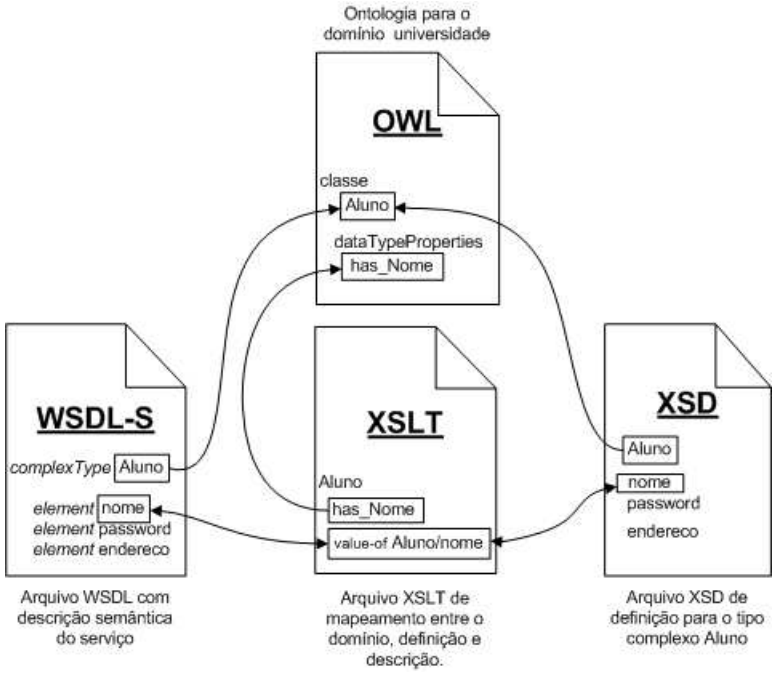

Figura 18: As conexões entre o serviço web e os modelos de domínio com suporte a mapeamento de atributos, num contexto *top-level*

Listagem 5.4: O documento XSLT de transformação do conceito Aluno

```
<?xml version='1.0' ?>
<xsl:transform version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <xsl:template match="/">
        <Aluno rdf:ID="Aluno1">
            <hasNome rdf:datatype="xs:string">
                <xsl:value-of select="Aluno/Pessoa/nome"/>
            </hasNome>
            <hasPassword rdf:datatype="xs:string">
                <xsl:value-of select="Aluno/Pessoa/password"/>
            </hasPassword>
            <hasEndereco rdf:datatype="xs:string">
                <xsl:value-of select="Aluno/Pessoa/endereco"/>
            </hasEndereco>
            <hasCodCurso rdf:datatype="xs:int">
                <xsl:value-of select="Aluno/codCurso"/>
            </hasCodCurso>
        </Aluno>
    </xsl:template>
</xsl:transform>
```

A listagem 5.5 apresenta o arquivo de definição XSD para o tipo complexo. Nesse arquivo, os tipos complexos dos quais Aluno é dependente também estão conectados ao conceitos disponíveis na ontologia.

Listagem 5.5: O documento XSD de definição do conceito Aluno, e seus atributos

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:br.org.ufjf.model="http:///br/org/ufjf/model.ecore"
        xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:wssem="http://localhost:8080/TSUFJFWeb/services/
           new_tsufjf_service.wsdl"
        xmlns:OntologyUFJF="http://localhost:8080/TSUFJFWeb/services/model/
           ontology/tsufjf_ontology.owl#"
        ecore:nsPrefix="br.org.ufjf.model"
        ecore:package="br.org.ufjf.model"
        targetNamespace="http:///br/org/ufjf/model.ecore">
  <!-- Arquivo de definição para o tipo complexo Aluno, respeitando a
        hierarquia proposta, 1/7/2008 -->
  <xsd:element
        name="Aluno" type="br.org.ufjf.model:Aluno" />
  <xsd:element
        name="EntidadeLegal" type="br.org.ufjf.model:EntidadeLegal" />
  <xsd:element
        name="Pessoa" type="br.org.ufjf.model:Pessoa"/>
  <xsd:complexType
        name="Aluno"
        wssem="OntologyUFJF#Aluno">
    <xsd:complexContent>
      <xsd:extension base="br.org.ufjf.model:Pessoa">
        <xsd:attribute name="codCurso" type="xsd:int"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType
        name="EntidadeLegal"/>
  <xsd:complexType
        ecore:implements="xsd:anyType"
        name="Pessoa"
        wssem="OntologyUFJF#Pessoa">
    <xsd:complexContent>
      <xsd:extension
        base="br.org.ufjf.model:EntidadeLegal"
        wssem="OntologyUFJF#EntidadeLegal">
        <xsd:attribute name="endereco" type="xsd:string"/>
        <xsd:attribute name="senha" type="xsd:string"/>
        <xsd:attribute name="nome" type="xsd:string"/>
        <xsd:attribute name="dtaMatricula" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Importante destacar que, apesar dessa implementação utilizar transformação XSLT, a proposta WSDL-S permite que a especificação para mapeamento seja feita por qualquer outra linguagem escolhida pelo desenvolvedor, como RDF (AKKIRAJU *et al.*, 2005).

5.5.1.2 Inserindo semântica funcional

Na versão 2.0 do WSDL, as semânticas funcionais são definidas pelos elementos *wssem:precondition* e *wssem:effect* e devem ser inseridas como filhos de *operation*, em *interface*.

Para WSDL versão 1.1, a definição de semântica realizada por esses elementos é feita a partir de sua inclusão em *portType*, como elementos filhos de *documentation*.

Como o serviço web *insereAluno* foi gerado com versão 1.1, a segunda abordagem será utilizada para anotação semântica.

A inclusão de um aluno exige no mínimo que o curso no qual ele está sendo matriculado exista. Essa é uma pré-condição identificada para a execução correta do serviço e que pode ser verificada através do atributo *codCurso*. Dessa forma, acompanhando o exemplo disponível em (AKKIRAJU *et al.*, 2005), associa-se *wssem:precondition* em *portType* ao conceito *CursoDisponivel* do modelo de domínio - considerando-se que o mesmo tenha sido incluído na ontologia. Outra pré-condição identificada é a existência de vagas no curso. Uma vez que a pré-condição anterior tenha sido verificada, o número de vagas disponível deve ser suficiente para a inclusão desse novo aluno. A proposta WSDL-S permite que mais de uma pré-condição possa ser descrita. A listagem 5.6 apresenta a alteração realizada em *portType*.

Listagem 5.6: As anotações semânticas para a operação do serviço web

```
<wsdl:portType name="TSUFJFService">
    <wsdl:operation name="insereAluno">
        <documentation>
                <wssem:precondition name="CursoExistentePreCondition"
                    modelReference="OntologyUFJF#CursoDisponivel"/>
                <wssem:effect name="AlunoMatriculadoCursoEffect"
                    wssem:modelReference="OntologyUFJF#AlunoMatriculadoCurso
                        "/>
                <wssem:effect name="CursoAtualizadoEffect"
                    wssem:modelReference="OntologyUFJF#CursoAtualizado"/>
                <wssem:effect name="NumeroMatriculaEffect"
                    wssem:modelReference="OntologyUFJF#NumeroMatricula"/>
        </documentation>
        <wsdl:input message="impl:insereAlunoRequest" name="
            insereAlunoRequest"/>
        <wsdl:output message="impl:insereAlunoResponse" name="
            insereAlunoResponse"/>
    </wsdl:operation>
</wsdl:portType>
```

Como efeitos da execução correta da operação, espera-se um aluno inserido no curso, diminuição do número de vagas nesse curso e um número de matrícula gerado para o aluno. Esses efeitos podem ser incluídos com o elemento *wssem:effect*, sendo um para cada efeito esperado. A listagem 5.6 também indica essa anotação semântica. Efeitos com os nomes *Alu-*

noMatriculadoCursoEffect, *CursoAtualizadoEffect* e *NumeroMatriculaEffect* foram incluídos e oferecem para o requisitante maior significado para o resultado esperado na invocação desse serviço web.

Na versão 2.0 do WSDL, essas anotações são realizadas no elemento *interface*, seguindo o mesmo raciocínio para *portType*.

5.5.1.3 Inserindo categoria de serviço

A categoria do serviço deve ser identificada e inserida na descrição do serviço web. Para isso, o *namespace* disponibiliza o elemento *category* e pode ser inserido também em *portType*, uma vez que ele é definido em *interface* na versão 2.0. A categoria é utilizada para publicar e catalogar o serviço web possibilitando descoberta dinâmica.

O elemento *category* exige os atributos *name* e *taxonomyURI*. Enquanto o primeiro refere-se ao nome da categoria do serviço em uma taxonomia, o segundo especifica o endereço do qual a taxonomia pode ser obtida.

Uma vez que o serviço refere-se à inclusão de um aluno numa instituição federal, e assumindo que exista a taxonomia que define categoria para esse tipo de serviço, o mesmo pode ser anotado na forma apresentada na listagem 5.7.

Listagem 5.7: As anotações semânticas para a categoria do serviço web

```
<wssem:category name="MatriculaAlunoInstituicaoFederal"
        taxonomyURI="http://www.ufjf.br/" />
```

5.6 Considerações finais

A implementação dos serviços web semânticos deve ser possibilitada a partir de um proposta que seja de fácil entendimento, baseada em padrões já utilizados na Web, que possibilite fácil manutenção e que não exija alteração da infraestrutura em utilização para serviços web.

As propostas para implementação devem ter como objetivo possibilitar a construção de processos de negócio baseados em serviços web e que as atividades de descoberta, invocação e execução possam ser feitas de forma semi-automática ou automática, com o mínimo de intervenção humana.

Dentre as propostas apresentadas, WSDL-S foi escolhida para a anotação do serviço web *inclusaoAluno*. As etapas apresentadas possibilitaram a criação de um serviço web semântico.

O processo a partir dessa abordagem torna-se simplificado uma vez que utiliza padrões já conhecidos e permite desacoplamento entre a descrição e o modelo de domínio. As tarefas de descrição de um serviço web e definição de conceitos com utilização de ontologias podem ser independentes, o que possibilita agilidade de desenvolvimento e fácil manutenção do sistema.

Com o serviço web semântico criado, é possível disponibilizá-lo em um registro UDDI ou adicioná-lo ao *Resource Manager* de um ambiente WSMX. Através de uma plataforma que possibilite inferência e processamento das informações disponibilizadas pela ontologia associada e pela descrição semântica do serviço, o mesmo pode ser incluído na composição de um processo de negócio; localizado a partir de informações de requisição bem definidas; e permitir segurança na sua invocação uma vez que suas pré-condições são conhecidas.

6 Conclusão

6.1 Considerações finais

Os serviços web têm sido destacados dentro de um contexto de utilização onde se possibilita a integração de diferentes tecnologias e sistemas legados, oferecendo suporte para definição de processos de negócios entre empresas. Esses serviços podem utilizar a plataforma Web para serem disponibilizados, facilitando ainda mais o seu acesso.

A Arquitetura Orientada a Serviço - SOA, relaciona-se com o conceito de processos de negócio e apresenta-se num contexto em que a crescente utilização de tecnologias diferentes passam a ser integradas através de componentes de software independentes. Essa arquitetura permite a criação de processos dinâmicos e dá destaque à utilização de serviços web possibilitando maior agilidade e rapidez às demandas de mercado nas quais as empresas operam.

As ontologias representam domínios de conhecimento de uma determinada área. O objetivo de sua utilização em sistemas de software é permitir maior integração e comunicação entre plataformas, possibilitando interoperabilidade uma vez que permite o mapeamento de conceitos utilizados em tecnologias diferentes.

A utilização de ontologias na Web é defendida pelos conceitos de Web Semântica, uma vez que se pretende facilitar a busca de informações, não se limitando à palavras-chaves mas também realizá-la através de conceitos associados aos sites web. Os conceitos similares em sites web diferentes podem ser mapeados entre si, enriquecendo a forma com que as informações estão dispostas.

A utilização de ontologias é possibilitada a partir de sua representação por linguagens formais, como OWL. Essas linguagens permitem que as ontologias sejam inferidas e processadas por máquinas.

A união dos conceitos de serviços web e Web Semântica possibilitou o surgimento do estudo de serviços web semânticos. Sua definição e implementação são baseadas na necessidade de realizar descoberta, invocação e execução automática, composição de processos de negócio

e monitoração da execução de maneira semi-automática ou automática, permitindo substituição de serviços com o mínimo de intervenção humana.

O estudo dos serviços web semânticos envolve a definição de anotações semânticas, ou seja, conceitos de domínio da aplicação que devem ser adicionados à descrição dos serviços possibilitando alcançar os objetivos aos quais seu estudo foi iniciado.

Algumas propostas para implementação de serviços web semânticos fazem parte do conjunto de especificações que aguardam estudo e aprovação do W3C. Entre as apresentadas, foi dado destaque para WSDL-S através da anotação semântica de um serviço web. Ela foi selecionada por permitir um desenvolvimento sem modificações na estrutura desse serviço.

6.2 Trabalhos futuros

O estudo de serviços web semânticos envolve ainda alguns conceitos não abordados nesse trabalho. Explorar questões como arquitetura dos serviços, algoritmos de descoberta e anotações semânticas para requisitos não funcionais possibilita o desenvolvimento de um trabalho mais rico para esse assunto.

A especificação de uma arquitetura para serviços web semânticos está disponível em (STUDER; GRIMM; ABECKER, 2007). Métodos, algoritmos e ferramentas para descoberta de serviços web semânticos podem ser encontrados em (CARDOSO, 2007), além de um estudo aprofundado do *framework* proposto com WSMO.

Existem ferramentas disponíveis para cada uma das propostas de descrição semântica de serviços web, algumas são apresentadas por Studer, Grimm e Abecker (2007). Um estudo envolvendo a comparação dessas ferramentas permite maior compreensão das propostas apresentadas.

A descrição de semânticas não-funcionais envolvem questões como segurança. Esse tipo de descrição torna-se necessário em contextos como transações entre empresas ou execução de processos de negócios, onde também são necessárias as informações quanto qualidade, disponibilização do serviço e permissão de acessos simultâneos.

O estudo da composição dos seviços web semânticos ainda pode ser realizado em conjunto com a disponibilização de um um ambiente para execução, descoberta e invocação apresentando assim um contexto completo de sua utilização.

Referências

AKKIRAJU, R. *et al.* ***Web Service Semantics - WSDL-S***. [S.l.], Novembro 2005. Disponível em: http://www.w3c.org/Submission/2005/SUBM-WSDL-S-20051107/. Acesso em junho/2008.

AMORIM, S. S. ***A Tecnologia Web Services e sua Aplicação num Sistema de Gerência de Telecomunicações***. Dissertação (Mestrado Profissional de Engenharia de Computação) — Universidade Estadual de Campinas, Instituto de Computação, Março 2004. Disponível em: http://libdigi.unicamp.br/document/?code=vtls000332721. Acesso em março/2008.

APACHE. ***Javadocs for the Apache Axis runtime API***. [S.l.], 2006. Disponível em: http://ws.apache.org/axis/java/apiDocs/index.html. Acesso em abril de 2008.

BELLWOOD, T. ***UDDI Version 2.04 API Specification***. [S.l.], 2002. Disponível em: http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm. Acesso em abril de 2008.

BRAGA, R. M. M.; WERNER, C. M. **Desenvolvimento Baseado em Componentes**. *XIV Simpósio Brasileiro de Engenharia de Software SBES2000 – Minicursos e Tutoriais,* p. 297–329, out. 2000.

BRAY, T. *et al.* ***Extensible Markup Language (XML) 1.1 (Second Edition)***. [S.l.], 2006. Disponível em: http://www.w3.org/TR/2006/REC-xml11-20060816/. Acesso em abril de 2008.

BRUIJN, J. *et al.* ***Web Service Modeling Ontology (WSMO)***. [S.l.], 2005. Disponível em: http://www.w3.org/Submission/WSMO/. Acesso em junho/2008.

CARDOSO, J. ***Semantic Web Services - Theory, Tools and Applications***. 1. ed. [S.l.]: Information Science Reference, 2007.

DAVIES, J.; STUDER, R.; WARREN, P. ***Semantic Web Technologies - trends and research in ontology-based systems***. 1. ed. [S.l.]: John Wiley and Sons, 2006.

FANTINATO, M. ***Uma Abordagem Baseada em Características para o Estabelecimento de Contratos Eletrônicos para Serviços Web***. Tese (Doutorado) — Universidade Estadual de Campinas, 2007.

FENSEL, D. *et al.* **OIL: An Ontology Infrastructure for the Semantic Web**. *IEEE Intelligent Systems*, p. 38–44, mar./abr. 2001. Disponível em: http://www.cs.vu.nl/ frankh/postscript/IEEE-IS01.pdf. Acesso em maio/2008.

GAMMA, E.; HELM, R.; JOHNSON, R. ***Padrões de Projeto***. 2. ed. [S.l.]: Bookman Companhia Editora, 2000.

GOMES, J. C. ***Utilização da Arquitetura de Web Services no Desenvolvimento de Sistemas de Informação em Micro e Pequenas Empresas***. Dissertação (Mestrado Profissionalizante em Administração) — Faculdades IBMEC, Programa de Pós-Graduação e Pesquisa em Administração e Economia, Novembro 2005.

HEß, A.; KUSHMERICK, N. **ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services**. *Proceedings of the International Semantic Web Conference 2004*, p. 320–334, 2004. Disponível em: http://iswc2004.semanticweb.org/demos/34/paper.pdf. Acesso em junho/2008.

HORRIDGE, M. *et al*. ***A Practical Guide to Building OWL Ontologies Using the Protégé – OWL Plugin and CO-ODE Tools Edition 1.0***. [S.l.], Agosto 2004.

HORSTMANN, M.; KIRTLAND, M. **DCOM Architecture**. *DCOM Technical Articles*, Julho 1997.

JORDAN, D.; EVDEMON, J. ***Web Services Business Process Execution Language Version 2.0***. [S.l.], 2007. Disponível em: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. Acesso em junho/2008.

KEAN, L. ***Domain Engineering and Domain Analysis***. Carnegie Mellon University, Fevereiro 1998. Disponível em: http://www.sei.cmu.edu/str/descriptions/deda.html. Acesso em julho/2008.

LUSTOSA, P. A.; FAGUNDES, F.; BRITO, P. F. **OWL e Protégé-2000 na definição de uma ontologia para o domínio Universidade**. *In: Anais do V Encontro de Estudantes de Informática do Tocantins*, p. 303–312, out. 2003. Centro Universitário Luterano de Palmas.

MACHADO, G. B. ***Uma Arquitetura baseada em Web Services com Diferenciação de Serviços para Integração de Sistemas Embutidos a outros Sistemas***. Dissertação (Mestrado em Ciência da Computação Área de Concentração Sistemas de Computação) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Ciência da Computação, 2006.

MARTIN, D. *et al*. ***OWL-S: Semantic Markup for Web Services***. [S.l.], 2004. Disponível em: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/. Acesso em abril de 2008.

OMG, I. ***Documents Associated with CORBA, 3.1***. [S.l.], 2008. Disponível em: http://www.omg.org/spec/CORBA/3.1. Acesso em abril de 2008.

OMG, I. ***Unified Modeling Language***. [S.l.], 2008. Disponível em: http://www.uml.org/. Acesso em junho de 2008.

OMONDO. ***Omondo EclipseUML Europa***. [S.l.], 2008. Disponível em: www.eclipsedownload.com/product.html. Acesso em maio de 2008.

PATEL-SCHNEIDER; HORROCKS; HARMELEN, V. **Reviewing the design of DAML+OIL: An ontology language for the semantic web**. *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002. AAAI Press.

RAO, J. ***Semantic Web Service Composition via Logic-based Program Synthesis***. Tese (Doutorado) — Norwegian University of Science and Technology, Department of Computer and Information Science, 2004.

SEMPREBOM, T.; CAMADA, M. Y.; MENDONÇA, I. ***Ontologias e Protégé***. [S.l.], 2007. Disponível em: http://www.das.ufsc.br/ gb/pg-ia/Protege07/ontologias.pdf. Acesso em junho/2008.

SILVA, A. R. **Os Agentes de Software no Futuro da Internet**. *Nova Economia e Tecnologias de Informação: Desafios para Portugal*, p. 279–298, 2000. Disponível em: http://isg.inesc-id.pt/alb/static/papers/2000/jn1-as-livro-sti-ucp-2000.pdf. Acesso em julho/2008.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. ***OWL Web Ontology Language Guide***. [S.l.], Fevereiro 2004. Disponível em: http://www.w3.org/TR/owl-guide/. Acesso em maio de 2008.

STUDER, R.; GRIMM, S.; ABECKER, A. ***Semantic Web Services - Concepts, Technologies and Applications***. 1. ed. [S.l.]: Springer, 2007.

SUN, M. ***Javadocs for the Java RMI runtime API***. [S.l.], 2003. Disponível em: http://java.sun.com/j2se/1.4.2/docs/api/java/rmi/package-frame.html. Acesso em abril de 2008.

SUN, M. ***Javadocs for the Java Swing runtime API***. [S.l.], 2003. Disponível em: http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-frame.html. Acesso em abril de 2008.

SUN, M. ***Java API for XML Web Services Annotations***. [S.l.], 2005. Disponível em: https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html. Acesso em abril de 2008.

TAURION, C. Aprendendo SOA: primeiros passos. ***Mundo Java***, n. 26, p. 74, nov. 2007.

USCHOLD, M.; GRUNINGER, M. **Ontologies: Principles, Methods and Applications**. *Knowledge Engineering Review*, v. 11, n. 2, 1996. University of Edinburgh.

W3C. ***W3C Semantic Web Activity***. [S.l.], 2001. Disponível em: http://www.w3.org/2001/sw/. Acesso em maio/2008.

W3C. ***Acknowledged Member Submissions to W3C***. [S.l.], 2008. Disponível em: http://www.w3.org/Submission/. Acesso em julho/2008.