Aondê: Um Serviço Web de Ontologias para Interoperabilidade em Sistemas de Biodiversidade

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Jaudete Daltio e aprovada pela Banca Examinadora.

Campinas, 31 de agosto de 2007.

Profª. Drª. Claudia M. Bauzer Medeiros
Instituto de Computação – UNICAMP
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues – CRB8a / 2116

Daltio, Jaudete
D17a Aondê: um serviço Web de ontologias para interoperabilidade em sistemas de biodiversidade / Jaudete Daltio -- Campinas, [S.P. :s.n.], 2007.

Orientadora: Claudia M. Bauzer Medeiros
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Ontologia. 2. Serviços Web. 3. Biodiversidade. 4. Interoperabilidade. I. Medeiros, Claudia Maria Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Aondê: an ontology Web service for interoperability across biodiversity information systems.

Palavras-chave em inglês (Keywords): 1. Ontology. 2. Web services. 3. Biodiversity. 4. Interoperability.

Área de concentração: Banco de Dados

Titulação: Mestre em Ciência da Computação

Banca examinadora: Profa. Dra. Claudia M. Bauzer Medeiros (IC-UNICAMP)
Profa. Dra. Karin Koogan Breitman (PUC-RIO)
Prof. Dr. Ricardo da Silva Torres (IC-UNICAMP)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)

Data da defesa: 31-08-2007

Programa de Pós-Graduação: Mestrado em Ciência da Computação

Instituto de Computação
Universidade Estadual de Campinas

Aondê: Um Serviço Web de Ontologias para Interoperabilidade em Sistemas de Biodiversidade

Jaudete Daltio[1]

Agosto de 2007

Banca Examinadora:

- Profª. Drª. Claudia M. Bauzer Medeiros
  Instituto de Computação – UNICAMP (Orientadora)
- Profª. Drª. Karin Koogan Breitman
  Departamento de Informática – PUC/RIO
- Prof. Dr. Ricardo da Silva Torres
  Instituto de Computação – UNICAMP
- Prof. Dr. Edmundo Roberto Mauro Madeira (Suplente)
  Instituto de Computação – UNICAMP

[1]Apoio financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (processo 05/57424-0) 2006–2007, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES 2005–2006 e *Microsoft Research* financiadora do projeto WeBios.

Resumo

A pesquisa em biodiversidade requer associar dados sobre seres vivos e seus habitats, construindo modelos sofisticados e correlacionando vários tipos de informações. Os dados manipulados são por natureza heterogêneos, sendo providos por grupos de pesquisa distintos (e distribuídos), que coletam seus dados usando diferentes vocabulários, suposições, metodologias, objetivos e uma grande variedade de restrições espaciais e temporais. Este cenário apresenta muitos tipos de desafio de pesquisa em Ciência da Computação, tanto no nível físico (por exemplo, diversidade de estruturas de armazenamento) quanto conceitual (por exemplo, diversidade de perspectivas e de domínios de conhecimento). O uso de ontologias está sendo proposto como uma forma de se resolver problemas de heterogeneidade. Este tipo de solução, entretanto, gera novos desafios de pesquisa, pois requer considerar problemas de projeto, gerenciamento e compartilhamento de ontologias.

Esta dissertação apresenta um novo tipo de Serviço Web, cujo objetivo é auxiliar a resolver tais desafios. Aondê ("coruja", em Tupi, uma referência à linguagem de representação de ontologias OWL) é um Serviço Web que provê um amplo espectro de operações para armazenamento, gerenciamento, busca, *ranking*, análise e integração de ontologias. O texto abrange a especificação e a implementação do Aondê, que foram validadas por um protótipo testado com ontologias volumosas e estudos de caso reais em biodiversidade.

Abstract

Biodiversity research requires associating data about living beings and their habitats, constructing sophisticated models and correlating all kinds of information. Data handled are inherently heterogeneous, being provided by distinct (and distributed) research groups, which collect these data using different vocabularies, assumptions, methodologies and goals, and under varying spatio-temporal frames. This poses many kinds of challenges in Computer Science research, from the physical (e.g., diversity of storage structures) to the conceptual level (e.g., diversity of perspectives and of knowledge domains). The adoption of ontologies has been proposed as a means to help solve heterogeneity issues. However, this kind of solution gives birth to new research issues, since it implies handling problems in ontology design, management and sharing.

This dissertation presents a new kind of Web Service whose goal is to help in solving such issues. Aondê (which means "owl" in Tupi, the main branch of native Brazilian languages) is a Web Service that provides a wide range of operations for storage, management, search, ranking, analysis and integration of ontologies. The text covers the specification and implementation of Aondê, which have been validated by a prototype tested with large ontologies and real biodiversity case studies.

Agradecimentos

A professora Claudia Bauzer Medeiros, que sempre acreditou em meu potencial e me guiou durante o mestrado. Pela sua atenção, paciência, ensinamentos e críticas que tanto contribuíram para o meu crescimento pessoal. Foi um grande previlégio tê-la como orientadora.

Ao professor Ricardo Torres, um dos idealizadores do projeto WeBios, pela disponibilidade para discussões, pelas sugestões e contribuições tão significativas para o trabalho.

Aos professores Thomas Lewinsohn e Paulo Inácio Prado, biólogos parceiros do projeto WeBios, que contribuíram para a validação do trabalho.

A minha família, pelo incentivo, apoio e confiança durante a realização deste mestrado. Em especial a minha mãe, um exemplo de força e perseverança em minha vida.

Ao Cristiano, pelo apoio incondicional, pela compreensão e companheirismo que tanto me ajudaram a superar os desafios e persistir em meus objetivos.

Aos amigos e companheiros do Laboratório de Sistemas de Informação – LIS, que contribuíram direta ou indiretamente na realização deste trabalho.

Aos meus primeiros orientadores ainda na graduação, professores Jugurta Lisboa Filho e Marcus Vinícius Alvim Andrade, que despertaram em mim o interesse por pesquisa e sempre me incentivaram.

Aos membros da banca, pelas sugestões e contribuições no trabalho.

As agências de fomento FAPESP (processo 05/57424-0), CAPES e *Microsoft Research*, pelo apoio financeiro na realização deste trabalho.

Sumário

Lista de Figuras

Capítulo 1

Introdução e Motivação

A pesquisa em biodiversidade é um campo multidisciplinar que requer a cooperação de vários tipos de pesquisadores. Os biólogos realizam diferentes tipos de atividades, incluindo coletas em campo, análises de dados sobre os espécimes coletados, seus habitats e correlações com outros seres vivos, construindo modelos capazes de descrever essas interações. Os dados disponíveis vêm sendo coletados em vários lugares do mundo por muitos grupos de pesquisadores, sendo publicados em formatos distintos e especificados em inúmeros padrões. Este cenário é caracterizado por sua heterogeneidade intrínseca – não apenas de dados e modelos conceituais utilizados, como também de necessidades e perfis dos especialistas que coletam e analisam os dados.

O grande volume de dados e a diversidade das espécies atuam como fatores complicadores deste cenário. As estimativas sobre o número de espécies vegetais e animais existentes no mundo variam entre 10 e 50 milhões, das quais apenas 1,5 milhões de espécies são atualmente classificadas pelos cientistas (estimativa realizada pela *WWF – Fundo Mundial para a Natureza*[1]). Entre os especialistas, o Brasil é considerado o país da "megadiversidade" – abrange cerca de 20% das espécies conhecidas em todo o mundo. Essa diversidade motiva diversos esforços na coleta de dados, dando origem, por conseqüência, a uma grande volume de informações. Isto gera uma demanda por mecanismos sofisticados de armazenamento, gerenciamento, compartilhamento, processamento e mineração, que permitam uma análise integrada e correlacionada desses dados.

Sistemas de Informação de Biodiversidade [69] representam soluções parciais para alguns desses problemas, permitindo a análise de espécies e suas interações. Seu propósito é auxiliar pesquisadores a aprimorarem ou complementarem seu conhecimento e entendimento sobre os seres vivos. Consultas típicas nesses sistemas combinam informações textuais sobre espécimes (*quando* e *onde* eles foram observados, por *quem* e *como*) e informações geográficas, caracterizando os ecossistemas onde os espécimes foram observados,

[1]World Wide Fund For Nature, fonte http://www.wwf.org.br/

além da distribuição espacial das ocorrências.

O *WeBios* [73] é um sistema de informação de biodiversidade que está sendo desenvolvido em uma parceria de pesquisadores dos Institutos de Computação e Biologia da UNICAMP. Seu objetivo é prover aos biólogos um sistema que permita consultas exploratórias multimodais sobre fontes de dados de biodiversidade heterogêneas e distribuídas na Web. Essas fontes incluem dados textuais, imagens (fotos de seres vivos e seus habitats), dados geográficos e metadados específicos do domínio. O sistema WeBios possui uma arquitetura orientada a serviços e emprega tecnologias da Web Semântica.

A manipulação de dados heterogêneos, assim como na maioria dos sistemas de biodiversidade, representa um dos maiores desafios do sistema WeBios. Uma dificuldade adicional é a necessidade de manipular dados coletados ao longo de vários anos. Tanto os ecossistemas considerados como os modelos ecológicos e a classificação taxonômica das espécies sofreram modificações, refletindo a evolução do conhecimento científico no mundo real. Conseqüentemente, os dados biológicos possuem muitos tipos de heterogeneidade, não só sintáticas ou semânticas, mas também temporais, tornando o problema de interoperabilidade ainda mais complexo. O uso de ontologias tem sido apontado como solução para alguns desses problemas.

Ontologias são descrições de um modelo abstrato de termos, relacionados entre si [34]. Modelam uma parte da realidade, suas entidades, relações taxonômicas, propriedades e restrições, visando definir um entendimento comum sobre um domínio. Ontologias têm adquirido importância tanto em aplicações industriais como acadêmicas. Muitos ramos da biologia possuem seus domínios descritos por ontologias consensuais, como a *Gene Ontology* [8] (em genética) e *TAMBIS Ontology* (*Transparent Access to Multiple Bioinformatics Information Sources*) [9] (em biologia molecular). Em biodiversidade, entretanto, ainda não existe uma ontologia consensual, embora existam vários especialistas envolvidos em iniciativas multinacionais, como o GBIF *(Global Biodiversity Information Facility)* [32]. Ainda há muito trabalho a ser realizado na especificação de uma ontologia para esse domínio – resultado da variedade de características dos dados e perfis de especialistas.

Esta é uma das razões que contribui para que a integração de dados de biodiversidade e de estudos ecológicos seja extremamente difícil. Soluções para interoperabilidade são necessidades reais para pesquisa nesse domínio. Até mesmo as classificações taxonômicas de espécies são alvo de discussão entre pesquisadores. Algumas árvores taxonômicas são claramente definidas – como em zoologia, para mamíferos. Entretanto, ainda existem divergências de autores em vários domínios além de muitas espécies para serem classificadas ou reclassificadas – como os insetos, por exemplo. Problemas similares afetam também descrições de habitats de espécies, como ocorre com as várias classificações de solo existentes no mundo. Somente no Brasil, existem dois sistemas de classificação de solos considerados oficiais. Com isso, uma mesma amostra de solo pode não apenas ser

representada por diferentes identificadores e nomes, como também pertencer a classes de ontologias distintas, de acordo com o sistema de classificação adotado.

Embora ontologias possam ser aplicadas para resolver problemas de heterogeneidade, seria necessário que os especialistas do domínio utilizassem os mesmos conceitos e significados para prover semântica a seus dados. Isto representa uma limitação em sistemas como o WeBios, que visam atender uma vasta comunidade de pesquisadores de domínios distintos e de diferentes visões do mundo, que desejam compartilhar seus dados. Com esse intuito, torna-se necessário um mecanismo capaz de manipular, de forma integrada, conjuntos de ontologias que representem a semântica de fontes distintas de dados.

O objetivo desta dissertação é especificar e implementar um Serviço Web de Ontologias – Aondê ("coruja", em Tupi, uma referência à linguagem de representação de ontologias OWL), dotado de funcionalidades para manipular ontologias tanto isoladamente quanto conjuntamente com outras que compartilhem domínios similares. As funcionalidades do Aondê provêem acesso, gerenciamento, análise, comparação e integração de ontologias. Embora a implementação realizada esteja direcionada para o domínio de biodiversidade, o serviço foi especificado de forma genérica, podendo ser estendido e adotado por outras aplicações que possuam necessidades de interoperabilidade similares.

Desta forma, esta dissertação apresenta contribuições direcionadas a solução de problemas de heterogeneidade semântica na Web, apresentando um Serviço Web de Ontologias caracterizado por: (i) utilizar repositórios distribuídos para armazenar e gerenciar ontologias e seus metadados; (ii) acessar esses repositórios por meio de Serviços Web; e (iii) especificar e implementar operações que permitem a manipulação integrada de conjuntos de ontologias. Com isto, aplicações podem enriquecer sua semântica e interoperabilidade tornando-se clientes desse serviço, trocando, reusando, integrando e adotando conceitos das ontologias publicadas na Web.

As principais contribuições desta dissertação são:

- Levantamento e comparação de técnicas e ferramentas para manipulação de ontologias propostas na literatura;
- Levantamento das necessidades de um Serviço de Ontologias para permitir a integração de dados heterogêneos e distribuídos;
- Especificação de um Serviço de Ontologias, com um conjunto de operações e uma estrutura de Repositórios Semânticos que disponibiliza ontologias por meio de protocolos padrões de Serviços Web;
- Implementação de um Serviço Web de Ontologias – Aondê – com considerações específicas para dados de biodiversidade e validado com estudos de caso reais em biodiversidade.

Os avanços alcançados no decorrer deste mestrado foram apresentados e publicados em dois congressos nacionais [21, 22].

1.1 Organização da Dissertação

Esta dissertação está organizada como segue. O Capítulo 2 discorre sobre conceitos importantes e trabalhos relacionados. O Capítulo 3 apresenta a especificação do Aondê, o Serviço Web de Ontologias proposto. Os aspectos de implementação relacionados ao protótipo do serviço são descritos no Capítulo 4. O Capítulo 5 descreve um estudo de caso realizado com dados reais coletados por biólogos parceiros do projeto WeBios. Finalmente, as conclusões e extensões desta dissertação são apresentadas no Capítulo 6.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta uma visão geral sobre Sistemas de Biodiversidade e, em particular, o sistema WeBios, projeto no qual esta dissertação está inserida (Seção 2.1). Em seguida, são descritos alguns conceitos básicos utilizados ao longo desta dissertação (Seção 2.2). Os trabalhos relacionados, apresentados na Seção 2.3, descrevem mecanismos para o gerenciamento de ontologias, incluindo técnicas e ferramentas propostas na literatura.

2.1 Sistemas de Biodiversidade e o WeBios

Em linhas gerais, *Sistemas de Biodiversidade* realizam correlações entre dados geográficos, informações sobre espécies e registros de coletas. Grande parte desses sistemas estão relacionados à determinação da distribuição espacial de uma ou mais espécies, suas correlações espaço-temporais e suas tendências de distribuição. Isto requer uma combinação de dados dos espécimes (*quando* e *onde* eles foram observados, *por quem* e *como*) com dados geográficos que caracterizam os ecossistemas onde foram observados. A integração de dados está baseada, na maioria das vezes, em propriedades espaciais, envolvendo bancos de dados geográficos.

Os principais desafios considerados nesses sistemas são: (i) a necessidade de mecanismos de interação com o usuário para facilitar a especificação de consultas, (ii) a dificuldade em combinar mecanismos de consulta de vários tipos de dados e (iii) a complexidade do gerenciamento diferenciado de dados de natureza tão distinta. Atualmente, existem várias iniciativas para o desenvolvimento de sistemas de biodiversidade [69]. Alguns desses sistemas possuem propósito específico – são centrados em características de uma categoria específica de seres vivos [37, 66] – enquanto outros possuem propósito geral, abrangendo uma ampla variedade de espécies [64]. A utilização de análise de imagens como predicados também pode contribuir para consultas mais complexas, como mostrado em [68].

O *WeBios* [73] pertence a esta segunda categoria de sistemas. Trata-se de um sistema

de biodiversidade proposto conjuntamente por pesquisadores dos Institutos de Computação e Biologia da UNICAMP. Seu objetivo é oferecer um mecanismo de consultas para pesquisadores que trabalham com questões ambientais e de biodiversidade. A idéia é que as diversas fontes de dados sejam acessadas por meio de Serviços Web, enquanto a formulação das consultas, o pré-processamento e a visualização dos resultados sejam executados como uma aplicação cliente remota. Desta forma, os pesquisadores poderão realizar um trabalho exploratório, que considere seus conhecimentos sobre espécies, suas interações, seus habitats e correlações entre eles. O principal diferencial do WeBios em relação aos demais sistemas de biodiversidade existentes é permitir a combinação de predicados baseados em conteúdo de imagens, predicados espaciais e textuais em uma mesma consulta. Os sistemas de biodiversidade disponíveis atualmente não atacam estas questões simultaneamente.

A Figura 2.1 ilustra a arquitetura do WeBios, organizada em quatro camadas principais: *Camada de Armazenamento*, *Serviços de Suporte*, *Serviços Avançados* e *Aplicação Cliente*. Esta dissertação refere-se aos dois componentes destacados na figura, o Serviço de Ontologias (Serviços de Suporte) e os Repositórios Semânticos (Camada de Armazenamento).

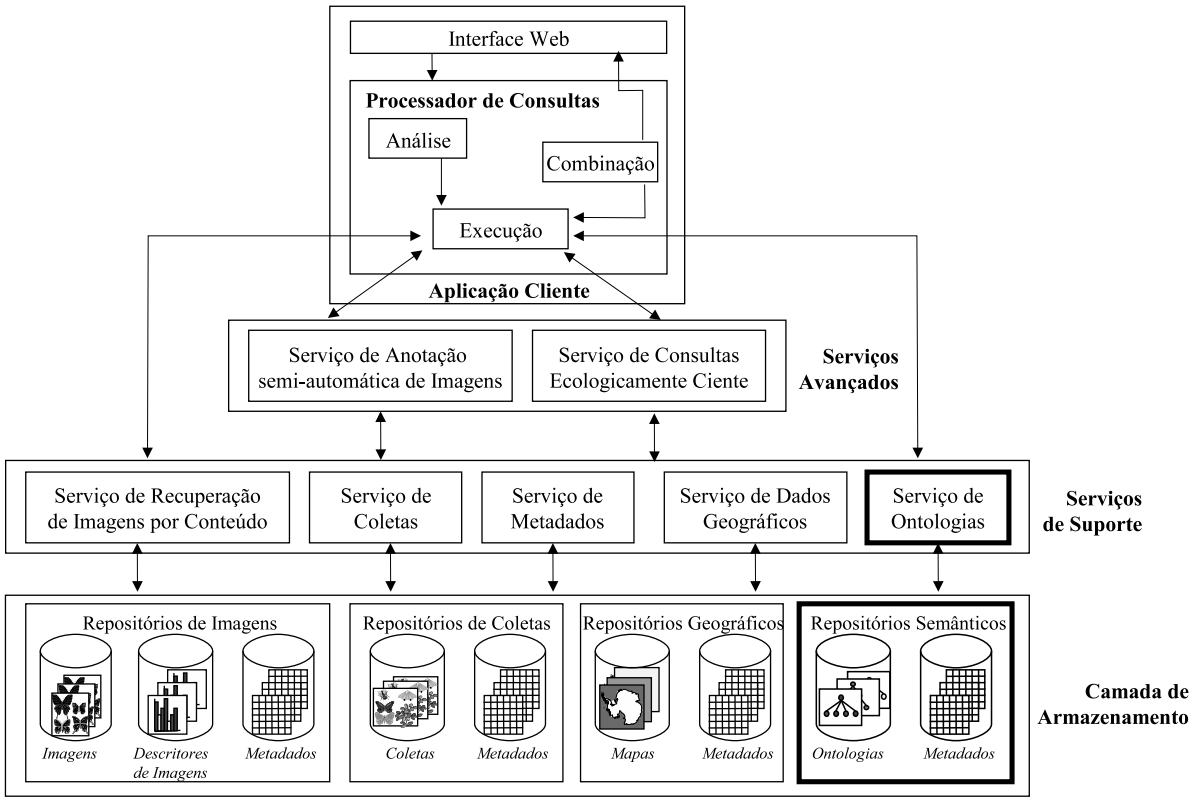

Figura 2.1: A Arquitetura do Sistema WeBios

Como pode ser observado na figura, as operações de acesso a dados são realizadas por meio de requisições às camadas de Serviço. Uma consulta submetida à *Interface Web* é traduzida, pelo *Mediador de Consultas*, para um conjunto de requisições remetidas aos serviços apropriados das camadas de Serviço *Avançados* ou de *Suporte*. Esses serviços acessam os dados dos repositórios da camada de *Armazenamento*, processam as requisições e retornam os resultados para o *Mediador*, que os combina e retorna o resultado final a ser exibido pela interface.

A Camada de Armazenamento é responsável pelo armazenamento e gerenciamento dos dados. Existem quatro tipos básicos de fontes de dados nos repositórios: imagens, registros de coletas de espécies, dados geográficos, ecológicos e ontologias. Metadados e ontologias são utilizados na anotação dos dados nos repositórios, auxiliando o processamento das consultas no sistema. Embora a Figura 2.1 ilustre um único repositório de cada tipo, a arquitetura do WeBios pressupõe a existência de vários repositórios distribuídos, gerenciados por pesquisadores, instituições ou projetos de pesquisa distintos.

Os repositórios de *Imagens* armazenam fotos e registros de seres vivos e seus habitats. Os repositórios de *Dados Geográficos* contêm informações a respeito das características de regiões georeferenciadas (ou seja, associadas a algum sistema de coordenadas que descreva sua localização na superfície da Terra). Os repositórios de *Coletas* possuem registros de ocorrências de espécies observadas em pesquisas de campo ou catalogadas em museus, com informações sobre observações de seres vivos. Ontologias são usadas pelos pesquisadores na descrição do contexto de seu trabalho, representado, dentre outros, árvores filogenéticas, descrições taxonômicas e morfológicas de espécies, relações ecológicas entre seres vivos ou ainda definições de habitats. Os repositórios *Semânticos*, que armazenam as ontologias, possuem um papel fundamental na desambiguação de consultas. A especificação dos repositórios semânticos faz parte desta dissertação, sendo descrita em detalhes nas Seções 3.2 e 4.2.

A camada de Serviços de Suporte abrange cinco Serviços Web, dedicados à recuperação específica de cada tipo de dados. O serviço de *Recuperação de Imagens por Conteúdo* processa requisições baseadas no conteúdo de imagens, utilizando descritores na codificação de vetores que expressam características como cor, forma e textura das imagens. Dada uma imagem de entrada I, esse serviço retorna o conjunto de imagens "mais similares" a I, considerando as funções de similaridade definidas pelo usuário. O serviço de *Coletas* provê acesso a registros de observações de seres vivos e dados coletados em visitas de campo. O serviço de *Metadados* é responsável por gerenciar e recuperar informações estruturadas em diferentes padrões de metadados adotados por grupos de pesquisa distintos. Desta forma, esse serviço permite recuperar não apenas o valor armazenado em um metadado, como também especificar qual padrão deve ser considerado na interpretação desse valor.

O serviço de *Dados Geográficos* é usado na recuperação de informações espaciais ne-

cessárias para a geração de mapas e dados sobre a distribuição de espécies em uma dada região. Requisições a esse serviço utilizam tanto predicados textuais (como o nome de habitats) quanto espaciais (por exemplo, encontrar regiões que estejam contidas ou que possuam sobreposição com outras regiões). O serviço de *Ontologias*, alvo desta dissertação, é descrito detalhadamente nas Seções 3.3 e 4.3.

A camada de Serviços Avançados é composta por Serviços Web que utilizam serviços de Suporte para atender a requisições que demandam a combinação de fontes de dados de tipos distintos. Até o momento, dois serviços desta camada foram projetados e parcialmente implementados: Serviço de Anotação de Imagens e Serviço de Consultas Ecologicamente Ciente. O serviço de *Anotação de Imagens* [31] auxilia usuários no processo de anotação de imagens utilizando metadados e termos ontológicos. O serviço de *Consultas Ecologicamente Ciente* [40] permite a realização de consultas que consideram os relacionamentos ecológicos entre as espécies (por exemplo, presa–predador). Relacionamentos ecológicos não são diretamente armazenados nos registros de espécies, porém podem ser inferidos por meio de ontologias desse domínio.

2.2 Conceitos Básicos

A pesquisa desenvolvida nesta dissertação trata de um Serviço Web para ontologias. Desta forma, Serviços Web e ontologias representam os conceitos básicos deste trabalho e são apresentados, respectivamente, nas Seções 2.2.1 e 2.2.2.

2.2.1 Serviços Web

Serviços Web (*Web Services*) podem ser definidos como aplicações auto-descritas e modulares, que expõem sua lógica de negócio como serviços na Web por meio de interfaces programáveis e de protocolos de Internet, com o propósito de proporcionar formas de registrar, encontrar e invocar tais serviços [6]. Por meio de padrões, Serviços Web encapsulam aplicações em diferentes linguagens de programação e hospedadas em diferentes plataformas, publicando-as como serviços que podem ser dinamicamente localizados e acessados.

Serviços Web são baseados no conceito de *Arquitetura Orientada a Serviços (SOA)* [26], que permite que componentes de *software* de diferentes sistemas sejam expostos como serviços. SOA não especifica apenas uma arquitetura de serviços, mas também o relacionamento entre participantes dessa arquitetura (*Provedor de Serviços*, *Cliente de Serviços* e o *Registro de Serviços*), como mostra a Figura 2.2. As interações entre essas entidades envolvem as operações *publicação*, *busca* e *invocação de serviços*. Um provedor define a descrição de um serviço e a publica em um registro de serviços, a partir do qual o serviço

pode ser encontrado. Um cliente utiliza uma operação de busca para encontrar a descrição do serviço e, então, a associa com o provedor para realizar requisições ao serviço.

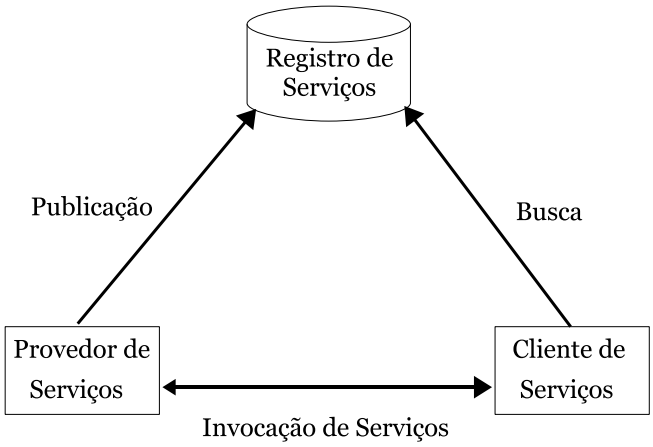

Figura 2.2: Arquitetura Orientada a Serviços: participantes e seus relacionamentos

As principais características dos Serviços Web são:

- **Independência de plataforma:** os serviços devem ser acessáveis utilizando tecnologias existentes em ambiente distintos e, com isso, protocolos e mecanismos de descrição e recuperação de serviços devem seguir padrões formalmente definidos e aceitos;

- **Fraco acoplamento:** os serviços não devem conhecer a estrutura e o funcionamento interno de outros serviços, o acesso é sempre intermediado pela interface do serviço. Isto aumenta a flexibilidade da arquitetura, uma vez que a alteração interna de um serviço não afeta seus clientes;

- **Transparência de localização:** serviços devem se encontrados pelos clientes independentemente de sua localização. Suas definições e informações de localização devem ser armazenadas em um diretório de serviços que é utilizado pelo cliente na busca.

A implementação de Serviços Web é baseada em padrões e tecnologias específicos, que incluem SOAP *(Simple Object Access Protocol)*, WSDL *(Web Services Description Language)* e UDDI *(Universal Description, Discovery and Integration)*, baseados em XML *(Extensible Markup Language)*.

O protocolo de transporte SOAP [12] rege a troca de mensagens entre aplicações em ambientes distribuídos e descentralizados. Sua especificação define o formato das mensagens para invocação e comunicação entre serviços, realizando o tráfego de documentos

XML por meio de protocolos de Internet como HTTP (*Hypertext Transfer Protocol*). As mensagens encapsulam informações que são transmitidas entre Serviços Web e especificam a operação que será invocada. Seu conteúdo é composto por informações e estruturas de dados, contendo três partes principais: (1) um envelope, que define o conteúdo da mensagem SOAP; (2) um conjunto de regras de codificação, que define os tipos de dados utilizados na aplicação; e (3) uma convenção, que define as chamadas e respostas usando procedimentos remotos.

A linguagem WSDL [16] descreve de forma padronizada e independente como e onde os serviços podem ser conectados e utilizados na Web. Um documento WSDL representa um contrato entre o provedor de serviços e seus clientes, especificando as funcionalidades do serviço, sua localização na Web e as instruções para acessá-lo. Além disso, o documento também define a estrutura das mensagens que um serviço envia e recebe.

A especificação UDDI [17] define mecanismos para a publicação, descoberta e integração de serviços. Tais mecanismos definem tanto as estruturas de dados necessárias para sua descrição e classificação, como uma interface baseada no protocolo SOAP que permite o acesso a essas informações. Provedores de serviços podem utilizar UDDI para publicar os serviços que oferecem e os clientes de serviços podem usar UDDI para descobrir serviços que lhes interessem e obter os metadados necessários para a utilização esses serviços.

2.2.2 Ontologias

Uma frase amplamente citada na literatura define que: *"uma ontologia é uma especificação explícita de uma conceitualização compartilhada"* [34]. A consideração mais importante nessa afirmação é a noção de *conceitualização*, que corresponde a uma definição consensual a respeito da representação de um domínio. Uma ontologia define um vocabulário para essa conceitualização e especifica restrições ao uso desse vocabulário. A Figura 2.3 ilustra parcialmente uma ontologia biológica relacionada ao domínio de biologia que representa interações entre insetos e plantas. Essa ontologia foi desenvolvida com o auxílio dos biólogos parceiros do projeto WeBios.

Numa perspectiva computacional, uma ontologia pode ser vista como um modelo de dados que representa um conjunto de conceitos de um domínio e seus relacionamentos. O conhecimento do domínio é formalizado em uma ontologia utilizando quatro tipos de componentes:

- **Classes:** representam conjuntos ou tipos de objetos (conceitos ou categorias de conceitos do domínio), comumente organizados em taxonomias – por exemplo, as classes *"Plant"* e *"Insect"* são subclasses de *"LivingBeing"* na Figura 2.3;

- **Instâncias:** materializam os objetos do domínio e são representados por instâncias

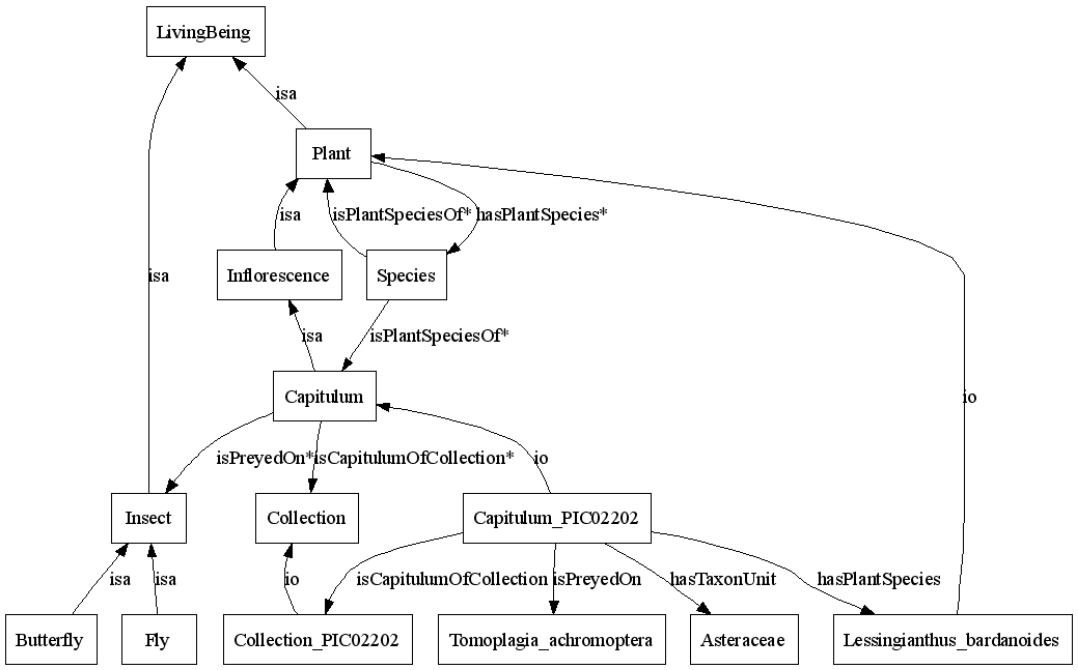

Figura 2.3: Exemplo de Ontologia Biológica

das classes – por exemplo, *"Lessingianthus bardanoides"* é uma instância da classe *Plant*;

- **Propriedades:** modelam as características das classes e instâncias. Propriedades podem expressar atributos (características ou parâmetros que uma classe deve possuir) ou expressar como as classes e instâncias se relacionam entre si – por exemplo, a propriedade *"isPreyedOn"* representa um relacionamento entre as classes *Capitulum* e *Insect*;

- **Restrições:** são definições abstratas que utilizam propriedades para descrever os conceitos do domínio usando condições – um exemplo de restrição para a ontologia ilustrada na Figura 2.3 seria: cada conceito *Collection* está relacionado a pelo menos um conceito *Capitulum* (*"Collection* $\forall$ *hasCapitulum only Capitulum"* e *"Collection hasCapitulum min 1"*).

Ontologias são comumente ilustradas como grafos. Na Figura 2.3, os vértices representam classes e instâncias (vértices terminais). As arestas representam os relacionamentos entre esses vértices, dos tipos: hierarquias de classes (rotuladas com *isa*), relações entre classes e instâncias (rotuladas com *io*) e propriedades (rotuladas com o nome das proprie-

dades). As restrições são expressas em axiomas (utilizando lógica descritiva, por exemplo) e não são representadas graficamente. Várias linguagens podem ser utilizadas na representação textual de uma ontologia, como RDF (*Resource Description Framework*) [46], RDF(S) (*RDF Schema*) [13] e OWL (*Web Ontology Language*) [7]. A maioria das linguagens textuais para a especificação de ontologias são baseadas em XML.

A representação em RDF se baseia na associação entre descrições e recursos por meio de declarações que estabelecem um valor para uma propriedade associada a um recurso. O bloco fundamental para qualquer construção nessa linguagem é definido por uma tripla da forma: *Sujeito–Predicado–Objeto*, onde *Sujeito* refere-se a um recurso (denotado por uma URI – *Unified Resource Identifier*), *Predicado* representa uma propriedade do recurso e *Objeto* representa o valor dessa propriedade, que pode ser um literal (representado por um inteiro ou uma *string*, por exemplo) ou um outro recurso. Para uma padronização de uso de RDF, foram criados os esquemas RDF (*RDF-Schemes* ou *RDFS*) [13], que fornecem tipos básicos para a criação de esquemas voltados a aplicações específicas. As primitivas usadas para modelar esquemas RDF incluem construtores para classes, subclasses, propriedades, subpropriedades e instâncias.

A linguagem OWL [7], recomendada pelo consórcio W3C, foi projetada com o intuito de melhor atender as necessidades de representação de termos e seus relacionamentos de forma ontológica. OWL é especificada em RDF e possui, adicionalmente, construtores específicos para os componentes básicos de uma ontologia (classes, instâncias e propriedades), além de construtores para a formalização de relacionamentos mais complexos (equivalência, interseção e união de classes, por exemplo). A linguagem OWL possui três classes de sub-linguagens, de acordo com sua expressividade:

- *OWL Lite:* permite a criação de hierarquias de classificação simplificadas com algumas restrições, não possuindo axiomas e estruturas para a representação de relacionamentos sofisticados. *OWL Lite* considera apenas restrições mais simples, como cardinalidade. A intenção dessa sub-linguagem é oferecer suporte a migração de taxonomias para o formato de ontologias;

- *OWL DL:* abrange os construtores oferecidos em *OWL Lite* adicionando a completude, a decidibilidade e a expressividade da lógica de descrições (*DL* é um acrônimo para *Description Logic*). Essa sub-linguagem possui construtores mais complexos que *OWL Lite*, permitindo a modelagem de classes por meio das operações união, interseção e complemento, além de representar enumerações de instâncias, disjunções entre classes e restrições de separação de tipos;

- *OWL Full:* fornece a máxima expressividade de OWL e a liberdade de usar RDF, abrangendo os construtores disponibilizados em *OWL DL* acrescidos de outros elementos lógicos sofisticados e eliminando as restrições de separação de tipos. O uso

desta sub-linguagem garante uma vasta gama de representações semânticas, porém não há garantias de computabilidade.

Ontologias construídas para representar domínios restritos têm sido amplamente reutilizadas e podem representar um papel importante de fornecedoras de conhecimento para a inferência dinâmica em agentes inteligentes. Atualmente, existem vários ambientes para manipulação de ontologias, incluindo editores, servidores e repositórios [20, 28, 33], com o objetivo de facilitar não só a construção de ontologias, como também sua disponibilização para reuso e extensão. Além desses ambientes, várias ferramentas foram propostas para a realização de operações em ontologias, como a integração, a análise de versões, a criação de anotações e a realização de consultas [57].

Embora esses ambientes e ferramentas possuam algumas similaridades, nenhum deles provê interoperabilidade com outras aplicações nem cobre todas as atividades previstas no ciclo de vida de uma ontologia (especificação, aquisição de conhecimento, conceitualização, integração, implementação, avaliação e documentação [29]). Essa falta de interoperabilidade acarreta problemas significativos quando duas ontologias, construídas em diferentes ferramentas ou linguagens, são combinadas usando ferramentas de integração. Isto representa um desafio real na integração de dados de diferentes grupos de pesquisa que, ainda que distantes geograficamente, realizam pesquisa sobre as mesmas áreas ou áreas afins e desejam compartilhar suas informações.

2.3 Trabalhos Correlatos

Os trabalhos correlatos relacionados a esta dissertação abrangem diferentes técnicas e ferramentas para ontologias existentes na literatura. A Seção 2.3.1 apresenta as abordagens para manipulação de ontologias comumente utilizadas em aplicações. A Seção 2.3.2 apresenta funcionalidades e ferramentas propostas para ontologias e suas principais características.

2.3.1 *Frameworks* e Servidores de Ontologias

É cada vez mais comum o uso de ontologias na associação de semântica a dados e operações. Apesar de auxiliarem na interoperabilidade entre aplicações, as ontologias impõem um novo tipo de sobrecarga aos sistemas (e usuários) – como definir as ontologias de interesse e, principalmente, como manipulá-las. Em muitos domínios de aplicação, especialistas constroem suas próprias ontologias e, em outros casos, ontologias existentes podem ser reusadas. Freqüentemente, ambas abordagens são combinadas – ou seja, especialistas desenvolvem ontologias a partir do reuso parcial ou total de outras existentes. Vários

repositórios foram recentemente criados para armazenar e compartilhar ontologias, como DOME, OntoServer, Ontaria e Swoogle [23].

Frameworks são comumente utilizados para auxiliar o desenvolvimento de aplicações que necessitem manipular ontologias. Entre os mais citados na literatura estão Jena [15], SnoBase [44] e SOFA[1]. Em geral, esses *frameworks* permitem acessar ontologias armazenadas em diferentes formatos (como RDF, RDF(S) e OWL) e acessar as informações descritas nas ontologias por meio de consultas.

A Figura 2.4 ilustra um trecho de código Java utilizando o *framework* Jena para percorrer uma ontologia e retornar todas as triplas RDF existentes. Nesse exemplo, a linha 1 inicializa um iterador para as triplas da ontologia, e seus valores *Sujeito–Predicado–Objeto* são obtidos nas linhas 4, 5 e 6, respectivamente. A linha 9 realiza um teste para verificar se o *Objeto* da tripla é um literal ou um outro recurso da ontologia.

```
1  StmtIterator iter = model.listStatements();
2  while (iter.hasNext()) {
3     Statement stmt = iter.next();
4     Resource subject = stmt.getSubject();
5     Property predicate = stmt.getPredicate();
6     RDFNode object = stmt.getObject();
7     System.out.print("(" + predicate.toString() + ",");
8     System.out.print(" " + subject.toString() + ",");
9     if (object instanceof Resource) {
10       System.out.print(" " + object.toString()); }
11    else {
12       System.out.print(" \"" + object.toString() + "\""); }
13    System.out.println(")"); }
```

Figura 2.4: Trecho de código Java utilizando o *framework* Jena para navegação em uma ontologia

Na maioria dos casos, *frameworks* não permitem o desenvolvimento de aplicações que necessitem considerar a evolução de ontologias, pois consideram as ontologias manipuladas como prontas (estáticas). No entanto, ontologias que descrevem domínios complexos podem sofrer constantes atualizações, refletindo a aquisição de conhecimento do mundo real. Em biodiversidade, por exemplo, isso ocorre quando novas espécies são encontradas ou reclassificadas taxonomicamente. Como a implementação das aplicações depende dos *frameworks*, a evolução das ontologias pode implicar alterações significativas nas próprias aplicações, contrariando o propósito de flexibilidade semântica proposto pelo uso de ontologias.

Servidores de ontologias foram propostos para suprir a necessidade de gerenciamento dinâmico de ontologias [25, 45, 67]. Alguns provêem acesso às ontologias por meio de suas URI's, outros armazenam as ontologias em repositórios locais facilitando o gerenciamento

[1] http://sofa.dev.java.net

e controle de versões. As funcionalidades desses servidores se restringem a realização de consulta nas ontologias, de forma similar aos *frameworks* e, em alguns casos, fornecem também mecanismos de inferência.

Entretanto, essas funcionalidades ainda não são suficientes em cenários distribuídos com múltiplas ontologias. Aplicações de um mesmo domínio podem ter focos e interesses distintos, utilizar diferentes terminologias ou ainda manipular o conhecimento em diferentes níveis de detalhe. Razões como essas podem levar a várias formas de heterogeneidade, tanto terminológica quanto conceitual.

A heterogeneidade terminológica entre ontologias relaciona-se com o processo de atribuição de nomes a seus elementos (classes, instâncias e propriedades). Dentre os tipos de heterogeneidade mais encontrados estão a utilização de sinônimos (palavras diferentes com o mesmo significado), homônimos (uma mesma palavra expressando conceitos diferentes) e variações sintáticas, como abreviações, sufixos e prefixos.

A heterogeneidade conceitual diz respeito ao conteúdo da ontologia. Esse tipo de heterogeneidade explica porque ontologias que descrevem um mesmo domínio podem diferir em suas classes primitivas ou em definições, às vezes contraditórias, de uma mesma entidade. A abordagem [11] sugere agrupar este tipo de heterogeneidade em três dimensões, ilustradas na Figura 2.5: (a) cobertura: ontologias podem diferir por cobrirem diferentes regiões – possivelmente sobrepostas – do domínio; (b) granularidade: ontologias podem diferir pelo nível de detalhamento de descrições das entidades; e (c) perspectiva: ontologias podem prover diferentes pontos de vista de um mesmo domínio, com variações temporais ou espaciais.

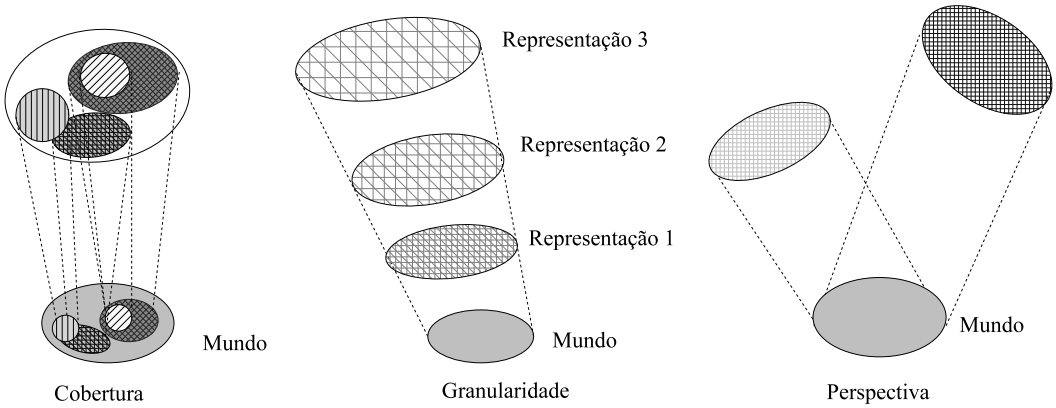

Figura 2.5: As três dimensões da heterogeneidade conceitual, sugerida por [11]

Aplicações que necessitem manipular várias ontologias de forma conjunta devem considerar esses tipos de heterogeneidade, seja na comparação da representação de um conceito

ou na descoberta de mapeamentos entre conceitos de ontologias diferentes. Além disso, muitas aplicações não utilizam de forma completa as ontologias de domínio, apenas parte do contexto descrito. Aplicações podem ainda desejar comparar versões distintas de uma dada ontologia para confrontar a evolução temporal dos dados. Atualmente, não existe uma solução única que abranja essas necessidades. O objetivo do Serviço de Ontologias proposto nesta dissertação é preencher esta lacuna, provendo acesso a essas funcionalidades via Serviços Web.

2.3.2 Técnicas e Ferramentas para Ontologias

As próximas seções descrevem brevemente algumas técnicas e ferramentas para ontologias propostas na literatura, destacando suas principais características. As técnicas incluem: *ranking* de ontologias (Seção 2.3.2.1), detecção de diferenças (Seção 2.3.2.2), extração de visões (Seção 2.3.2.3) e integração de ontologias (Seção 2.3.2.4).

2.3.2.1 *Ranking* de Ontologias

O objetivo dos mecanismos de *ranking* é auxiliar a determinação das ontologias potencialmente relevantes para um dado domínio de conhecimento, baseando-se em diferentes critérios. Alguns mecanismos adotam a análise de referências entre ontologias, favorecendo as ontologias mais referenciadas – abordagem similar à noção de popularidade de *Page–Ranking* da Web. Essa abordagem é adotada pelas ferramentas Swoogle [23] e OntoKhoj [56]. Entretanto, a baixa conectividade de grande parte das ontologias disponíveis na Web restringe a abrangência do método.

Outras abordagens baseiam o *ranking* na análise das estruturas internas de uma ontologia. Essa abordagem, utilizada na ferramenta AkTiveRank [2], utiliza métricas que avaliam como uma dada ontologia representa os conceitos de interesse de uma busca, considerando sua hierarquia de classes, suas propriedades e instâncias. A ferramenta AkTiveRank combina quatro métricas para determinar o *ranking* de um conjunto de ontologias O em relação a um conjunto de termos de entrada T, que são:

- **Casamento exato e parcial:** compara cada termo em T para cada ontologia $o \in O$. Computa o número de classes e instâncias em o cujos rótulos casam exata ou parcialmente com termos de T;
- **Densidade:** aplicado a cada classe $C \in o$ cujo rótulo "casa" com um termo de T. Computa o número de subclasses, superclasses, propriedades e vizinhos (subclasses da mesma superclasse);

- **Centralidade:** aplicado a cada classe $C \in o$ cujo rótulo "casa" com um termo de T. Computa os caminhos mínimos entre todos os pares de classes em o (por exemplo, utilizando o algoritmo Floyd-Warshall [19]) e conta quantos deles possuem a classe C. Quanto maior a centralidade de uma classe na ontologia, maior a probabilidade de que a ontologia represente adequadamente conceitos relacionados a essa classe e mais rica é a ontologia na descrição desse conceito;
- **Similaridade semântica:** aplicada a pares de classes $C_1, C_2 \in o$, cujos rótulos "casam" com termos de entrada T. Computa o tamanho do caminho mínimo entre C_1 e C_2. Quanto menor o tamanho desse caminho, maior a similaridade entre os conceitos.

A Figura 2.3.2.1 resume as principais características das ferramentas de *ranking* citadas. A figura mostra, por exemplo, que as métricas utilizadas pela ferramenta AkTiveRank tornam a análise da ontologia mais precisa, enquanto o *ranking* realizado pela ferramenta Swoogle possui um baixo tempo de execução. Os elementos sombreados indicam os aspectos adotados na dissertação.

Ferramenta	**Swoogle**	**AkTiveRank**	**OntoKhoj**
Abordagem	Referências entre ontologias	Análise das estruturas internas	Referências entre ontologias
Técnica Aplicada	Contagem das referências	Métricas de análise de estrutura	Contagem das referências
Formato das Ontologias	OWL, RDF, RDF(S), DAM+OIL	OWL	XML, RDF, DAML+OIL, OWL
Elementos Buscados	Todos os elementos	Classes	Todos os elementos
Vantagens	Baixo tempo de execução	Análise mais precisa	Baixo tempo de execução
Desvantagens	Uma mesma ontologia pode aparecer duplicada em uma busca, e em posições diferentes de *ranking*	Alto tempo de execução	Uma mesma ontologia pode aparecer duplicada em uma busca, e em posições diferentes de *ranking*

Figura 2.6: Tabela comparativa das ferramentas de *ranking* de ontologias

2.3.2.2 Detecção de Diferenças entre Ontologias

A detecção de diferenças, na maioria dos casos, compara duas versões de uma mesma ontologia, identificando as alterações existentes entre elas. Diferenças simples são aquelas

que não afetam a estrutura da ontologia – concentram-se apenas na alteração de nomes de classes e propriedades, tipos de dados ou restrições. Diferenças complexas incluem modificações na hierarquia das classes ou na semântica dos conceitos no domínio da aplicação.

Existem várias ferramentas que tratam da detecção de modificações em ontologias de forma automática ou semi-automática (ou seja, com ou sem a intervenção do usuário). As abordagens são comumente centradas na evolução de ontologias e na propagação de modificações. O trabalho descrito em [49] propõe o ambiente PromptDiff para o controle de versões de ontologias, auxiliando o processo de desenvolvimento colaborativo por especialistas do domínio. Este ambiente identifica alterações estruturais na ontologia e permite que usuários aceitem ou rejeitem cada alteração. A ferramenta OntoDiff [71] detecta automaticamente modificações entre a estrutura e o conteúdo de duas versões de uma ontologia, identificando termos adicionados, removidos e modificados.

A Figura 2.3.2.2 compara alguns aspectos desses enfoques. A figura mostra, por exemplo, que o desempenho da ferramenta OntoDiff está relacionado às heurísticas utilizadas e que a ferramenta PromptDiff não considera o conteúdo das ontologias no processo de comparação. Os elementos sombreados indicam os aspectos considerados na dissertação.

Ferramenta	**PromptDiff**	**OntoDiff**
Abordagem	Comparação da estrutura das ontologias	Comparação da estrutura e do conteúdo das ontologias
Técnica aplicada	Análise estrutural e hierárquica	Heurísticas para descoberta das modificações
Ambiente de Apoio	Protégé	Nenhum
Formato das Ontologias	RDF(S), Protégé, OWL	RDF(S)
Processamento	Semi-automático	Automático
Vantagens	Auxilia no processo de desenvolvimento colaborativo de ontologias	Comparação automática
Desvantagens	O usuário analisa todas as diferenças encontradas	Desempenho associado às heurísticas

Figura 2.7: Tabela comparativa das ferramentas de detecção de diferenças de ontologias

2.3.2.3 Visões de Ontologias

O reuso de ontologias é uma prática altamente recomendada no processo de desenvolvimento de novas ontologias. O reuso possui inúmeras vantagens – por exemplo, evitar o difícil trabalho de construir uma ontologia "do zero". Além disso, ontologias já disponíveis

podem ser consideradas consolidadas, uma vez que foram submetidas a especialistas do domínio e testadas por outras aplicações. Finalmente, o reuso estimula a integração de dados e a interoperabilidade entre aplicações que utilizam as mesmas ontologias.

Freqüentemente, o tamanho e a complexidade das ontologias disponíveis excedem as necessidades das aplicações – em geral, as aplicações necessitam apenas de parte do domínio descrito por uma ontologia. Apesar disso, por falta de alternativas disponíveis, desenvolvedores de aplicações utilizam essas ontologias completas, criando penalidades computacionais como problemas de desempenho, espaço e tempo de processamento, principalmente no uso de mecanismos de inferência.

Uma solução para este problema é a extração de visões, que permitem limitar um subconjunto do domínio descrito na ontologia. O termo *visão* tem sua origem na pesquisa em bancos de dados – uma visão é uma porção de um banco de dados relevante para um dado usuário ou aplicação, extraída pela execução de uma consulta no banco de dados. De forma similar, uma visão de uma ontologia é um subconjunto de uma ontologia, construído pela extração de partes relevantes da ontologia original para a criação de uma nova ontologia.

A abordagem [72] é baseada em linguagens de consulta para ontologias para a extração de visões. Outras abordagens consideram que visões de ontologias são conceitualmente diferentes e não podem ser comparadas com visões de bancos de dados. Em bancos de dados, visões extraem apenas dados que satisfazem uma determinada consulta. Visões em ontologias, sob esta perspectiva, podem incluir todos os tipos de elementos da ontologia – classes, instâncias, relacionamentos e axiomas – não sendo possível defini-las em uma consulta. Neste sentido, a abordagem proposta em [38] define uma linguagem para a definição de visões centradas em um conceito inicial, contendo operadores para a seleção de propriedades e instâncias. De forma similar, [52] apresenta o conceito de *View Traversal* que define, a partir de um conjunto de diretivas, quais elementos da ontologia original devem ser representados na visão. Essa proposta foi implementada como *plug-in* na ferramenta Protégé [33].

Uma outra abordagem utiliza visões com um tipo de *cache* para acelerar o processamento de consultas em ontologias e se baseia na extração automática de visões [3]. A idéia é realizar uma análise das consultas solicitadas pela aplicação. A visão é definida como uma sub-ontologia que contém os elementos mais freqüentemente requisitados nessas consultas.

A Figura 2.3.2.3 apresenta os principais aspectos das ferramentas apresentadas. A figura mostra, por exemplo, que as visões criadas com a abordagem de conceitos centrais são, necessariamente, ontologias consistentes, e que nas visões extraídas automaticamente não é possível especificar explicitamente quais elementos devem fazer parte da visão. Os elementos sombreados indicam os aspectos adotados na dissertação.

Ferramenta	OntoPathView	View Language	View Traversal	Ontology Winnowing
Abordagem	Baseada em conceito central	Baseada em linguagens de consulta	Baseada em conceito central	Extração automática baseada em consultas
Técnica Aplicada	Especificação do conceito central e dos elementos relacionados	Extensão de RQL	Especificação do conceito central e dos elementos relacionados	Análise dos elementos mais requisitados em consultas
Formato das Ontologias	RDF(S)	RDF(S)	RDF, RDF(S), Protégé, OWL	RDF(S), OWL
Vantagens	Consistência da visão	Restrição de condições na visão	Consistência da visão	Criação automática da visão
Desvantagens	Combinação de vários conceitos centrais	Explicitação de todos os tipos de elemento e combinação de consultas	Combinação de vários conceitos centrais	Não é possível determinar quais elementos serão incluídos

Figura 2.8: Tabela comparativa das ferramentas para extração de visões de ontologias

2.3.2.4 Integração de Ontologias

Em sistemas distribuídos e abertos, não é possível evitar a heterogeneidade dos dados apenas com o uso de ontologias. A integração de ontologias é uma solução cada vez mais comum para definir as relações existentes entre as representações heterogêneas. O processo de integração permite, por exemplo, criar expressões compatíveis entre ontologias diferentes. As abordagens para a integração de ontologias incluem [14, 41]:

- **Mapeamento:** etapa de pré-processamento da integração, que identifica entidades idênticas entre todos os conceitos de duas ontologias o_1 e o_2;
- **União:** constrói uma nova ontologia baseada nos mapeamentos entre o_1 e o_2, unindo os conceitos equivalentes. Esse novo conceito recebe o nome utilizado originalmente pelo conceito em o_1 ou o_2;
- **Alinhamento:** constrói uma nova ontologia que preserva as ontologias originais e materializa os mapeamentos como relacionamentos de equivalência entre os conceitos.

A identificação das similaridades entre ontologias é, em geral, baseada na análise e no reconhecimento de partes que "casam" umas com as outras. Este casamento pode ser a identificação de partes idênticas (equivalência) ou de elementos de relacionamentos (ex. parte-de, é-um). Segundo [63], existem duas principais classificações dessas técnicas:

Técnicas de combinação em nível de elemento: computam o mapeamento dos elementos analisando as entidades isoladamente, ignorando seus relacionamentos com outras entidades. Nessa categoria estão:

- Técnicas baseadas em *string*: utilizam a similaridade dos nomes e descrições dos elementos da ontologia – quanto mais similares as *strings*, mais provável que denotem os mesmos conceitos;

- Técnicas baseadas em linguagem: consideram *strings* como palavras em alguma linguagem natural e exploram as propriedades morfológicas dos termos, como a detecção da forma básica das palavras e a eliminação de artigos, preposições e conjunções;

- Técnicas baseadas em restrições: analisam as restrições aplicadas às definições das entidades, como seus tipos de dados e cardinalidades e instâncias (duas classes que possuem as mesmas instâncias possuem grande chance de denotarem conceitos similares);

- Técnicas baseadas em recursos lingüísticos: usam dicionários de domínio na tentativa de combinar palavras baseados nos relacionamentos lingüísticos entre elas (sinônimos, hiponímios, hiperonímios).

Técnicas em nível de estrutura: computam o mapeamento dos elementos analisando como as entidades aparecem na estrutura. Nessa categoria estão:

- Técnicas baseadas em grafos: consideram uma ontologia como uma estrutura de grafo e são usadas para identificar estruturas similares em duas ontologias analisando suas partes idênticas (baseando-se nas posições em seus respectivos grafos [59]);

- Técnicas baseadas em taxonomia: algoritmos de grafos que consideram apenas as relações de especialização, identificando classes idênticas a partir de seus atributos e das classes com as quais se relacionam. Considera-se que se relacionamentos do tipo *é-um* conectam termos que já são similares e, desta forma, seus vizinhos também podem ser de alguma forma similares [1].

Várias ferramentas são propostas na literatura para encontrar similaridade entre ontologias de forma semi-automática, segundo as diferentes abordagens de integração. A ferramenta GLUE [24] cria mapeamentos semânticos de forma semi-automática entre ontologias de mesmo domínio, considerando as taxonomias como componentes centrais. As ferramentas Chimaera [48], ODEMerge [60] e Prompt [50] realizam a união de ontologias. O processamento realizado pela ODEMerge é automático, baseado em informações de sinônimos e hiperonímios. Chimaera e Prompt são ferramentas iterativas, que direcionam o

usuário identificando potenciais candidatos à integração, baseado-se na similaridade entre os termos e em suas localizações nas hierarquias correspondentes. O protótipo CATO [27] realiza o alinhamento automático de ontologias, focando a taxonomia dos conceitos e utilizando dicionários de sinônimos para determinar similaridades entre classes.

A Figura 2.9 resume as características mais relevantes das ferramentas de integração citadas. A maioria delas são integradas a ambientes de manipulação de ontologias, como o Protégé (Prompt) e o Ontolingua (Chimaera). As ferramentas de processamento automático geram mapeamentos incorretos em alguns casos, enquanto as de processamento interativo muitas vezes sobrecarregam o usuário na verificação de todos os mapeamentos encontrados. Em geral, as ferramentas que combinam técnicas de elemento com técnicas de estrutura apresentam melhores resultados de integração. A última linha da tabela se refere ao domínio de conhecimento tratado pelas ontologias a serem integradas. Os elementos sombreados indicam os aspectos adotados na dissertação.

Ferramenta	**GLUE**	**Chimaera**	**ODEMerge**	**PROMPT**	**CATO**
Tipo de Integração	Mapeamento	União	União	Alinhamento e União	Alinhamento
Técnicas Aplicadas	Análise taxonômica, técnicas em string (similaridade) e restrições	Análise taxonômica, técnicas em string (similaridade)	Recursos lingüísticos (sinônimos, hiperônimos)	Análise taxonômica, técnicas em string, restrições de tipo, técnicas de grafos	Análise taxonômica, recursos lingüísticos (sinônimos)
Ambiente de Apoio	Nenhum	Ontolingua	WebODE	Protégé	Nenhum
Formato das Ontologias	XML (taxonomia)	Ontolingua, Protégé, XOL	RDF(S), DAML+OIL	Protégé, RDF(S)	OWL
Processamento	Semi-automático	Semi-automático	Automático	Semi-automático	Automático
Tipo de casamento entre elementos	classe-classe	classe-classe, propriedade-propriedade	classe-classe, propriedade-propriedade	classe-classe, propriedade-propriedade, instância-instância	classe-classe
Dependência de Domínio	Mesmo domínio	Domínios similares ou sobrepostos	Domínios similares ou sobrepostos	Domínios complementares	Domínio complementares similares e sobreposição

Figura 2.9: Tabela comparativa das ferramentas de integração de ontologias

2.4 Conclusões

Este capítulo apresentou uma visão geral sobre sistemas de biodiversidade, o sistema WeBios e os principais conceitos que embasam o trabalho proposto, ontologias e Serviços Web. Além disso, este capítulo descreveu os trabalhos correlatos utilizados nesta disser-

tação. O capítulo seguinte irá apresentar a especificação do Serviço de Ontologias, alvo desta dissertação.

Capítulo 3

Especificação do Serviço de Ontologias – Aondê

Este capítulo apresenta a especificação do Serviço Web de Ontologias Aondê. A Seção 3.1 apresenta uma visão geral da arquitetura proposta. As Seções 3.2 e 3.3 descrevem, em mais detalhes, as camadas que compõem a arquitetura.

3.1 Visão Geral

O objetivo do Serviço de Ontologias é prover acesso, manipulação, análise e integração de ontologias. Com isto, espera-se não apenas auxiliar a solução de problemas de heterogeneidade, como também associar mais semântica às operações das aplicações. A especificação do serviço é voltada às necessidades do sistema WeBios. Entretanto, essa especificação é suficientemente abrangente para ser utilizada em outros tipos de aplicações que necessitem prover semântica por meio da manipulação de ontologias.

A Figura 3.1 ilustra a arquitetura proposta para o Serviço de Ontologias Aondê, composta por duas camadas principais: *Repositórios* e *Operações*. A *Camada de Repositórios* é composta por vários repositórios distribuídos de ontologias, acessados via Serviços Web. Esses repositórios podem ser de dois tipos: *Repositórios Semânticos*, construídos e gerenciados pelo Serviço de Ontologias (que armazenam ontologias e metadados) e *Repositórios Externos de Ontologias*, não relacionados diretamente ao serviço e que publicam dados ontológicos via Serviços Web. A *Camada de Operações* provê as operações avançadas do serviço Aondê. Os módulos dessa camada acessam os repositórios por meio do módulo *Gerência dos Repositórios*.

Aplicações cliente podem interagir com Aondê tanto via a Camada de Operações quanto acessando diretamente os Repositórios Semânticos (permitindo, por exemplo, que especialistas do domínio possam consultar, validar ou atualizar ontologias nesses repo-

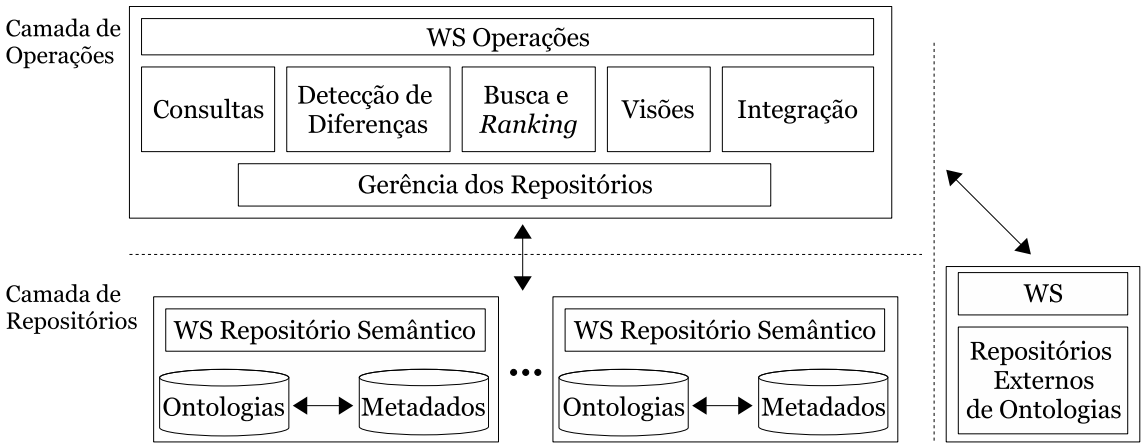

Figura 3.1: Arquitetura do Serviço de Ontologias

sitórios). A utilização do serviço pressupõe a existência de pelo menos um Repositório Semântico, que comporte tanto as ontologias requisitadas quanto aquelas produzidas pelas operações do serviço e seus metadados. É possível ter um cenário de uso do Aondê não atrelado a Repositórios Semânticos; considerações a esse respeito (e suas limitações) são discutidas na Seção 3.4.

As seções subseqüentes descrevem as camadas do serviço em mais detalhes. A Seção 3.2 apresenta a especificação da Camada de Repositórios e a Seção 3.3 descreve a Camada de Operações.

3.2 Camada de Repositórios

A Camada de Repositórios é composta por dois tipos de repositórios distribuídos:

- **Repositórios Externos de Ontologias:** repositórios de propósito geral, que são independentes do Serviço de Ontologias. As ontologias armazenadas nesses repositórios são gerenciadas por terceiros;

- **Repositórios Semânticos:** repositórios de propósito específico, que contêm ontologias e os metadados associados. Esses repositórios possuem uma estrutura pré-definida e são gerenciados pelo Serviço de Ontologias.

Ontologias extraídas de Repositórios Externos (por meio da operação de Busca, descrita na Seção 3.3.3) são armazenadas em Repositórios Semânticos, associadas a uma estrutura de metadados construída pelo serviço. O pressuposto é que as ontologias buscadas estão diretamente relacionadas às necessidades dos usuários e, portanto, devem estar disponíveis para futuras solicitações.

A necessidade do armazenamento de metadados nos Repositórios Semânticos está relacionada a questões de compartilhamento e reuso das ontologias por diferentes aplicações. Embora haja um grande número de ontologias disponíveis na Web, a maioria delas é difundida sem informações descritivas adicionais. Isso dificulta consideravelmente a identificação de ontologias de interesse por parte dos usuários potenciais. A construção de metadados e seu armazenamento junto às ontologias permite uma descrição de seu conteúdo, propósito e estrutura, por exemplo.

A Figura 3.2 ilustra a estrutura de um Repositório Semântico, composta por ontologias e estruturas de metadados. A figura mostra que uma ontologia sobre interações inseto-planta possui uma estrutura de metadados associada, indicando, por exemplo, sua URI, data de criação e palavras-chave contidas na ontologia.

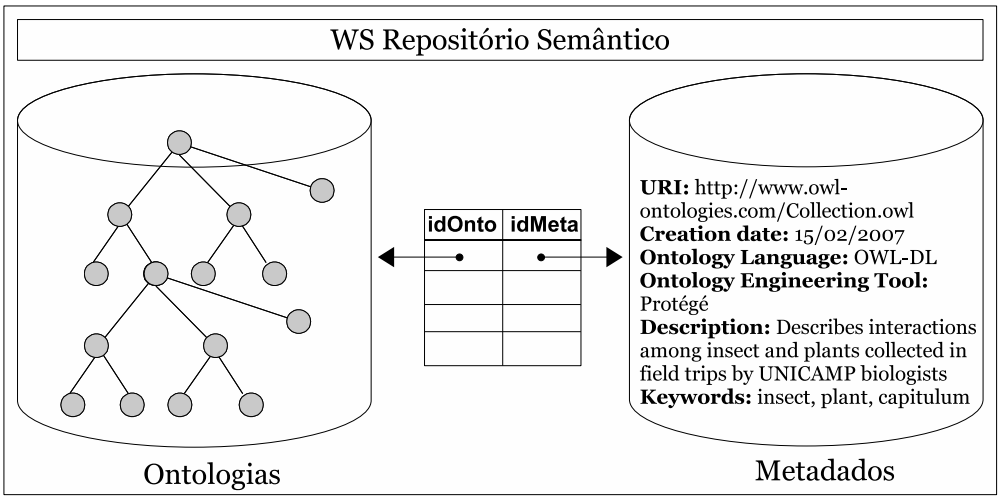

Figura 3.2: Estrutura de um Repositório Semântico

OWL (*Web Ontology Language*) [7], recomendada pelo consórcio W3C[1], é a linguagem utilizada na representação das ontologias. Como será visto no Capítulo 4, o padrão OMV (*Ontology Metadata Vocabulary*) é adotado para a representação das informações de metadados. Os metadados ilustrados na Figura 3.2 utilizam a terminologia OMV.

Considerando o contexto de biodiversidade, no qual está inserido o serviço Aondê, prevê-se que as ontologias dos Repositórios Semânticos armazenam conceitos sobre: (1) características geográficas, (2) informações biológicas e (3) outros tipos de informações gerenciadas pelo WeBios e relevantes para a pesquisa em biodiversidade (como metodologias de coletas e relacionamentos entre metadados de imagens, por exemplo). Ontologias geográficas são associadas a termos que descrevem características geográficas, climatológicas e ambientais de uma região, além de outros elementos naturais e artificiais que

[1]http://www.w3.org/

podem impactar o ecossistema. Ontologias biológicas possuem descrições de taxonomias e filogenia, evolução e morfologia de espécies, relacionamentos ecológicos e tróficos (posição ocupada por uma espécie na cadeia alimentar), dentre outros.

3.3 Camada de Operações

A Camada de Operações é responsável pelas operações avançadas de Aondê, organizadas nos seguintes módulos: Gerência de Repositórios (Seção 3.3.1), Consultas (Seção 3.3.2), Busca e *Ranking* (Seção 3.3.3), Visões (Seção 3.3.4), Integração (Seção 3.3.5) e Detecção de Diferenças (Seção 3.3.6). A escolha dessas funções foi baseada no estudo das necessidades observadas em alguns sistemas de biodiversidade (SinBiota [64], Spire [55] e GBIF [32], por exemplo) e complementada pelo processo de levantamento de requisitos conduzido conjuntamente com os biólogos parceiros do projeto WeBios.

3.3.1 Gerência dos Repositórios

Este módulo é responsável pelo gerenciamento (em memória) das ontologias manipuladas pelo serviço, realizando a mediação entre as requisições dos outros módulos de operações e os Repositórios Semânticos. Este módulo suporta a inserção e a eliminação de ontologias e metadados, a atualização de ontologias (novas versões) e a substituição de metadados em um repositório.

Um repositório pode conter várias ontologias; cada uma delas possui um identificador único em seu repositório. Cada estrutura de metadados inserida é associada a uma ontologia por meio de seus identificadores, e ambos devem ser armazenados no mesmo repositório. No caso de atualizações de ontologias, as diferentes versões são também associadas entre si por meio de seus identificadores.

Uma dada ontologia pode ser armazenada em mais de um Repositório Semântico – por exemplo, no caso de grupos de pesquisa distintos. Por esta razão, a recuperação de ontologias não é possível utilizando apenas seus identificadores "locais". A identificação única de uma ontologia requer o par: $<idOnto,\ URLRep>$, em que *idOnto* representa o identificador da ontologia no repositório endereçado por *URLRep*. Uma estrutura de metadados é sempre recuperada a partir do identificador de sua ontologia correspondente.

Este módulo também é responsável por gerenciar um catálogo de Repositórios Semânticos. Cada vez que um Repositório Semântico é utilizado pela primeira vez em uma invocação a algum módulo da Camada de Operações, este módulo armazena a URL desse repositório em seu catálogo. O objetivo desse catálogo é permitir que os Repositórios Semânticos disponíveis possam ser utilizados em outras invocações ao serviço (à semelhança de catálogos em bancos de dados). Isto facilita operações de busca por repositórios

distribuídos, por exemplo [40].

3.3.2 Consultas

Este módulo permite a extração de informações armazenadas em uma ontologia. Dada uma ontologia e uma consulta (expressa em uma linguagem de consulta para ontologias), esse módulo retorna o resultado da consulta no formato de saída requisitado. Uma invocação para este método possui a forma:

$$Consulta(idOnto,\ linguagem,\ strConsulta,\ inferencia,\ formato),$$

onde *idOnto* representa o identificador da ontologia a ser consultada, *linguagem* representa a linguagem de consulta utilizada na especificação da consulta descrita pela *string* *strConsulta*.

O campo *formato* define o formato de saída do resultado que pode ser textual (ou seja, triplas RDF ou partes dessas triplas) ou estruturado em arquivos XML [10]. O formato textual só pode representar apenas resultados cujos elementos possam ser designados por URI's. O formato XML pode ser utilizado na representação de qualquer tipo de resultado, contendo inclusive elementos que não possam ser designados por URI's. Considere o seguinte exemplo de consulta, na linguagem SPARQL [58], aplicado a uma ontologia *ontoA* previamente selecionada:

SELECT ?superclasse
WHERE classA rdfs:subClassOf ?superclasse

Essa consulta retorna todos os elementos da ontologia definidos como superclasses de *classA*. Em ontologias descritas em OWL-DL, por exemplo, os axiomas especificados na formalização das classes são representados como superclasses. Nesse caso, a consulta retornaria, além de classes, os axiomas de *classA*, que possivelmente envolvem outros elementos da ontologia (propriedades, classes e instâncias). Esses axiomas são representados como classes "anônimas" na ontologia e não podem ser designados por uma URI. Nessas situações, apenas o formato XML permite a representação do resultado da consulta.

O campo *inferencia* corresponde a um booleano que determina o uso ou não de mecanismos de inferência antes da execução da consulta requisitada (*true* = construir modelo inferido). A idéia é que ao invés de realizar a consulta diretamente sobre a ontologia armazenada, haja uma etapa prévia de inferência que gera uma ontologia estendida sobre a qual a consulta é realizada. Ao submeter a consulta ao modelo inferido, é possível obter resultados mais expressivos e completos, que não seriam obtidos pela aplicação direta à ontologia. Mecanismos de inferência utilizam a transitividade e a simetria de funções e as equivalências entre classes ou instâncias, por exemplo, na descoberta de novos fatos

ontológicos. Embora o uso de inferência permita obter informações ontológicas adicionais, isto requer um alto custo computacional (em tempo de execução) e, por esta razão, o uso de inferência é opcional, definido em um parâmetro da invocação.

A Figura 3.3 ilustra algumas implicações do uso de inferência. Considere uma consulta que deseja retornar as subclasses da classe A, aplicada uma ontologia que possui a hierarquia mostrada na Figura 3.3. Se essa consulta for realizada sobre as relações especificadas (Figura 3.3(a)), seu resultado retornará apenas a classe B. Aplicando essa consulta ao modelo inferido (Figura 3.3(b)), as classes C e D serão também retornadas, dada a transitividade da relação *subclasse*.

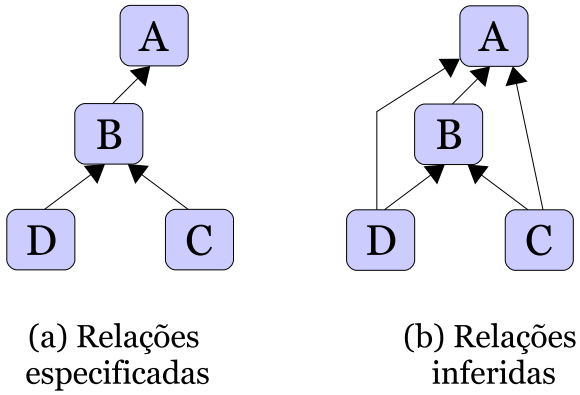

Figura 3.3: Exemplo do uso de mecanismos de inferência

Como se verá na Seção 4.3.2, o conteúdo do campo *strConsulta* é armazenado no Repositório Semântico associado à ontologia consultada. Isto permite disponibilizar aos usuários do Aondê as consultas já especificadas para uma dada ontologia, auxiliando especialistas do domínio na descoberta de informações adicionais sobre as ontologias, além das informações descritas em seus metadados.

3.3.3 Busca e *Ranking*

Este módulo realiza a busca, em um conjunto de repositórios, por ontologias que possuam certos termos de interesse. O retorno deste módulo é um conjunto de identificadores de ontologias que contém classes ou instâncias cujos nomes "casam" (exata ou parcialmente) com tais termos. Um Repositório Semântico é designado para armazenar o resultado da busca – ontologias encontradas em repositórios externos pesquisados são armazenadas nesse repositório.

O módulo permite dois tipos de operações: a busca por um conjunto de ontologias (com *ranking*) e a busca por uma ontologia específica (sem *ranking*). O *ranking* é baseado em

um conjunto de métricas que analisam as estruturas internas de cada ontologia retornada pela busca, visando mensurar como essas ontologias representam os termos de interesse da busca. Uma invocação de uma busca com *ranking* possui a forma:

$$BuscaRank(\{termo\}, \{peso\}, \{URLRep\},\ URLRepDestino),$$

em que $\{termo\}$ representa o conjunto de termos buscados nas ontologias, $\{URLRep\}$ representa o conjunto de repositórios (semânticos ou externos) designados para a realização da busca, cujo resultado é armazenado no Repositório Semântico *URLRepDestino* pelo próprio módulo. O conjunto de valores representado em $\{peso\}$ é utilizado na combinação das métricas de *ranking* utilizadas pelo módulo (as métricas adotadas foram adaptadas de [2]). Cada *peso* corresponde a um número não-negativo entre 0 e 1, sendo que a soma do conjunto $\{peso\}$ deve totalizar 1.

Buscas sem *ranking* são utilizadas quando o objetivo é retornar uma única ontologia, armazenada em repositórios de domínio biológico específico – contendo informações sobre classificações taxonômicas ou dados filogenéticos. Uma invocação para este tipo de busca possui a forma:

$$Busca(termo,\ \{diretiva\}, \{URLRep\},\ URLRepDestino),$$

em que, novamente, o conjunto de repositórios pesquisados é dado por $\{URLRep\}$ e a ontologia retornada como resultado tem *URLRepDestino* como Repositório Semântico de destino. O campo *termo* representa o nome do táxon científico buscado. Um táxon é uma unidade taxonômica associada a um sistema de classificação. Táxons podem estar em qualquer nível de um sistema de classificação (reinos, gêneros e espécie, por exemplo) e visam designar univocamente um tipo ou um grupo de seres vivos. Exemplos de táxons são: *Animalia* (reino), *Arthropoda* (filo) e *Lepidoptera* (ordem).

O campo $\{diretiva\}$ indica o conjunto de diretivas que deve ser utilizado na especificação de quais elementos da hierarquia taxonômica devem ser incluídos no resultado. Opções de diretivas são *antecessores*, *descendentes* e *irmãos* que denotam, respectivamente, que a busca irá retornar o táxon e seus antecessores, descendentes e táxons do mesmo nível na árvore taxonômica. A operação seleciona a primeira ontologia encontrada que representa o táxon. Este tipo de busca auxilia o sistema WeBios na descoberta de relacionamentos taxonômicos entre espécies.

3.3.4 Visões

Este módulo extrai uma visão de uma ontologia, baseada em um conceito central representado por uma classe dessa ontologia. O resultado produzido é uma sub-ontologia da ontologia original, contendo apenas essa classe e os elementos relacionados a ela por meio de diretivas especificadas. Uma invocação deste módulo possui a forma:

$$Visao(idOnto,\ conceito, \{diretivas\}, URLRepDestino),$$

em que *idOnto* representa o identificador da ontologia fonte e *conceito* refere-se ao nome da classe que representa o conceito central da visão. Depois de construída, a visão é armazenada no Repositório Semântico *URLRepDestino*. O conjunto {*diretivas*} é utilizado para especificar quais elementos relacionados ao conceito central devem ser incluídos na visão. As seguintes diretivas são permitidas:

- *subclasse:* subclasses do conceito central;
- *superclasse:* superclasses do conceito central;
- *irmao:* subclasses das mesmas superclasses do conceito central;
- *instancia:* instâncias das classes incluídas na visão;
- *axioma:* os axiomas que definem as classes incluídas na visão;
- *propriedade:* atributos ou relacionamentos que o conceito central possui com outras classes ou relacionamentos entre as instâncias da visão.

As diretivas *subclasse, superclasse* e *propriedade* (que representam relacionamentos com outras classes) são associadas a um inteiro não-negativo que representa sua profundidade – ou seja, quantos níveis de indireção da propriedade devem ser inseridos na visão. A diretiva {*superclasse:2*}, por exemplo, indica que todas as superclasses de distâncias 1 e 2 do conceito central devem ser inseridas na visão. As diretivas que envolvem relacionamentos do conceito central com outras classes da ontologia fazem com que essas classes relacionadas sejam incluídas na visão. Este é o caso das diretivas *subclasse, superclasse, irmao, axioma* e *propriedade*.

Uma invocação deste módulo com um conjunto vazio de diretivas produz uma visão contendo todos os elementos que a ontologia original possui e que estão, de alguma forma, relacionados à classe que representa o conceito central. Neste caso, considera-se que a profundidade das diretivas é ilimitada.

3.3.5 Integração

Este módulo é responsável por realizar a integração de duas ontologias. As atuais ferramentas de integração procuram por mapeamentos nas ontologias apenas entre elementos do mesmo tipo. Entretanto, é comum que ontologias difiram em nível de detalhe na descrição das entidades do domínio – uma classe em uma dada ontologia pode representar semanticamente o mesmo conceito que uma instância em outra ontologia. Por esta razão,

além de mapeamentos entre classes e entre instâncias, este módulo busca por mapeamentos entre classes e instâncias.

A abordagem de integração adotada é o alinhamento, descrito na Seção 2.3.2.4. Esse processo produz uma nova ontologia que preserva integralmente as ontologias originais, materializando os mapeamentos existentes entre seus elementos. A escolha dessa abordagem deve-se à expectativa de que a maioria das ontologias integradas terão domínios complementares e com sobreposições. Nesses casos, o processo de alinhamento permite mostrar como duas ontologias com diferentes coberturas podem ser utilizadas juntas na descrição de parte de um domínio do mundo real. Possibilita também mapear fatos entre ontologias com diferentes granularidades e mostrar um mesmo fato sob perspectivas distintas. O processo de união, uma abordagem alternativa, não preservaria integralmente as ontologias originais em seu resultado, além de privilegiar a nomenclatura utilizada por uma das ontologias – o que não é desejável pelos pesquisadores de biodiversidade. Uma invocação a este módulo possui a forma:

$$Integracao(idOntoA,\ idOntoB,\ \alpha,\ \beta,\ confiabilidadeMin,\ URLRepDestino),$$

em que *idOntoA* e *idOntoB* representam os identificadores das ontologias que serão alinhadas e o repositório de destino da ontologia produzida é endereçado por *URLRepDestino*. O campo *confiabilidadeMin* corresponde à confiabilidade mínima que um mapeamento identificado deve ter para ser materializado na ontologia alinhada.

A *confiabilidade* de um mapeamento é representada por um número entre 0 e 1, que corresponde à similaridade total entre os elementos que participam do mapeamento. A confiabilidade é calculada da seguinte forma:

$$confiabilidade = \alpha * max\{similarElemento\} + \beta * (similarEstrutura) \qquad (3.1)$$

Na Equação 3.1, o campo $\{similarElemento\}$ representa o conjunto de valores obtidos utilizando-se técnicas de similaridade de elemento, enquanto o campo *similarEstrutura* representa o valor da similaridade entre as estruturas desses elementos (vide Seção 2.3.2.4). A similaridade entre os elementos é inicialmente calculada e identifica possíveis candidatos ao mapeamento, que são posteriormente submetidos às técnicas de estrutura. Essa estratégia visa evitar casos em que classes ou instâncias com o mesmo nome, mas diferentes contextos e significados, sejam identificados como similares.

São aplicadas duas técnicas de similaridade de elemento: similaridade de *strings*, utilizando a métrica Jaro [18]), e dicionários de sinônimos. A métrica de comparação Jaro é baseada no número e na ordem dos caracteres comuns que existem entre duas *strings* e retorna valores entre 0 e 1. Considere as *strings* $s = a_1 \dots a_K$ e $t = b_1 \dots b_L$; um caractere a_i em s é comum à *string* t quando existe um caractere $b_j = a_i$ em t sendo que $i - H \leq j \leq i + H$, onde $H = \frac{min(|s|,|t|)}{2}$. Seja $s' = a'_1 \dots a'_k$ os caracteres em s que possuem

correspondentes em t (na mesma ordem em que aparecem em s) e seja $t' = b'_1 \dots b'_l$ análogo. Considere que uma transposição em s', t' é dada pela posição i em que $a'_i \neq b'_i$. Seja $T_{s',t'}$ metade do número de transposições existentes entre s' e t'. A métrica de similaridade Jaro para as *strings* s e t é dada por:

$$Jaro(s,t) = \frac{1}{3} * \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right) \tag{3.2}$$

A similaridade por dicionário de sinônimos retorna os valores 0 ou 1, indicando se as *strings* comparadas são sinônimos ou não. Como mostrado na equação 3.1, o valor máximo obtido entre essas técnicas (Jaro e dicionário de sinônimos) é utilizado no cálculo da confiabilidade e recebe peso α.

A similaridade de estrutura analisa as propriedades, os axiomas, as superclasses e as subclasses dos elementos comparados. O resultado de cada uma dessas quatro análises possui valores entre 0 e 1 e recebe peso de 0.25 no total da similaridade de estrutura, que resulta portanto em um valor entre 0 e 1. No caso de propriedades, comparam-se os rótulos e os elementos relacionados pela propriedade (classes ou tipos primitivos). Para axiomas, são comparadas as propriedades e as classes envolvidas. Na análise da hierarquia, procura-se por superclasses e subclasses comuns entre os elementos comparados. O valor de similaridade de cada análise é proporcional à quantidade de elementos similares em relação ao total de elementos existentes (número de superclasses similares/número total de superclasses, por exemplo). O total da similaridade de estrutura recebe peso β no cálculo da confiabilidade. É comum que as estruturas dos elementos difiram consideravelmente entre ontologias de diferentes granularidades. Além disso, alinhamentos entre elementos de diferentes tipos (classes e instâncias) não conseguem considerar a similaridade de estruturas.

Para exemplificar o cálculo da similaridade entre dois conceitos em ontologias, considere as classes *Pato* (Figura 3.4 (a)) e *Pata* (Figura 3.4 (b)). A classe *Pato* corresponde à ave pato, enquanto a classe *Pata* corresponde à parte do corpo de alguns animais relacionada a locomoção. Inicialmente, é calculada a similaridade de *string* entre esses termos, utilizando-se a métrica Jaro, cujo valor resultante é 0.83. Posteriormente, consulta-se o dicionário de sinônimos para ambos os termos e o valor resultante é 0 (pois os dicionários não relacionam esses termos como sinônimos).

Na análise de estrutura, ambas classes não possuem axiomas (valor 0 para essa métrica). O rótulo da propriedade *temHabitat* possui similaridade de *string* 0.71 com *temTipoParte* e os elementos relacionados nessas propriedades, *Habitat* e *Locomocao*, possuem similaridade 0.4. A propriedade *eMigratoria* da classe *Pato* não possui corresponde de comparação – trata-se de um atributo de *Pato*, e a classe *Pata* não possui atributos. Portanto, apenas uma das duas propriedades que a classe *Pato* possui pode ter um cor-

respondente em *Pata* e portanto essa métrica retorna o valor 0.5. Na análise da hierarquia, ambas classes não possuem subclasses e as superclasses *Ave* e *PartesCorpo* possuem similaridade 0. Desta forma, a similaridade total entre essas classes, considerando os pesos $\alpha = 0.8$ e $\beta = 0.2$, é dada por: $0.8{*}max(0.83, 0){+}0.2{*}(0.25{*}0{+}0.25{*}0.5{+}0.25{*}0{+}0.25{*}0)$, resultando em uma confiabilidade de $0,689$ para esse mapeamento.

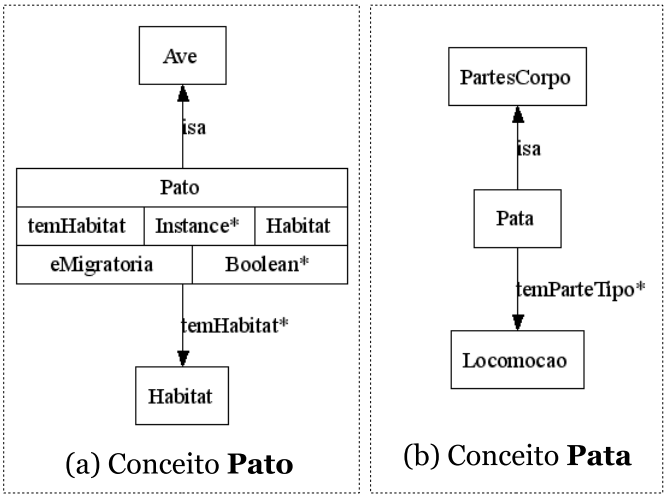

Figura 3.4: Exemplo do cálculo da similaridade entre as classes *Pato* e *Pata*

Valores para o parâmetro *confiabilidadeMin* muito próximos de 1 produzirão alinhamentos confiáveis, porém em quantidade significativamente inferior que uma confiabilidade 0.75, por exemplo.

3.3.6 Detecção de Diferenças

Este módulo realiza a comparação entre duas ontologias, que podem ser ou não duas versões de uma mesma ontologia, detectando as diferenças existentes em suas estruturas (classes e propriedades) e conteúdos (instâncias). O resultado retornado por este módulo consiste em um arquivo XML contendo as diferenças detectadas, categorizadas como modificações, inserções e exclusões. Uma invocação a este módulo possui a forma:

$$Diferenca(idOntoA,\ idOntoB,\ \alpha,\ \beta,\ confiabilidadeMin),$$

em que *idOntoA* e *idOntoB* representam os identificadores das ontologias a serem comparadas. O campo *confiabilidadeMin* corresponde à confiabilidade mínima que um mapeamento identificado deve ter para ser considerado uma modificação entre as ontologias comparadas, e seu cálculo combina a similaridade de elemento e de estrutura utilizando os pesos α e β fornecidos como parâmetros, de forma similar a realizada pelo módulo de integração.

Para exemplificar o funcionamento deste módulo, considere o exemplo ilustrado na Figura 3.5. A Figura 3.5 (a) ilustra uma ontologia criada para descrever algumas classificações de mamíferos, chamada *mammal.owl*. Essa ontologia foi reavaliada por especialistas e sofreu algumas alterações, ilustradas na Figura 3.5(b) (*mammal-V1.1.owl*).

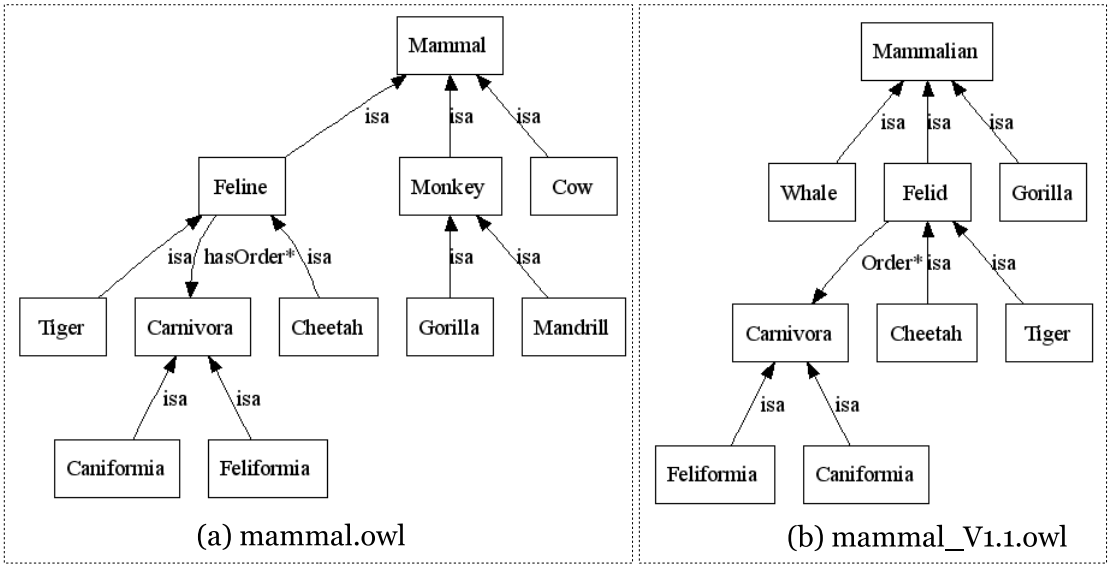

Figura 3.5: Duas versões da ontologia *mammal.owl*

O resultado gerado pela detecção de diferenças entre as ontologias *mammal.owl* (a) e *mammal-V1.1.owl* (b), utilizando como parâmetros $\alpha = 0.8$, $\beta = 0.2$ e *confiabilidadeMin* $= 0.7$, é ilustrado na Figura 3.6. Esse resultado é composto basicamente por três listas de diferenças: *ListModification*(modificações), *ListInsertion*(inserções) e *ListDelection*(exclusões). As listas de inserção e exclusão possuem as URI's dos elementos detectados. A lista de modificações possui as URI's dos elementos mapeados em suas respectivas ontologias e uma descrição textual do tipo de modificação identificada.

Como pode ser visto na Figura 3.5, as classes *Mammal, Feline, Gorilla* e a propriedade *hasOrder* foram mapeados como modificações entre as ontologias (a) e (b). As classes *Mammal* e *Feline* sofreram modificações em seus rótulos em relação às classes *Mammalia* e *Felid*(identificado como sinônimo na ontologia (b)). A classe *Gorilla* sofreu modificações em sua hierarquia em relação à classe *Gorilla* da ontologia (b), que não possui a superclasse *Mokey*. A propriedade *hasOrder* sofreu uma modificação em seu rótulo para *Order* na ontologia (b). Este último mapeamento só foi possível por que as classes relacionadas por essa propriedade foram mapeadas (*Feline* e *Felid*). Além disso, a Figura 3.5 mostra também que a classe *Whale* foi inserida na ontologia (b) e que as classes *Cow, Monkey* e *Mandrill* foram excluídas da ontologia (a).

Este módulo possui importantes aplicações no sistema WeBios, principalmente quando as ontologias A e B representam duas árvores distintas de uma mesma descrição taxonômica. Em biodiversidade, a classificação de espécies pode sofrer modificações ao longo do tempo. Além disso, classificações propostas por diferentes autores podem conter discordâncias pontuais na hierarquia das espécies. A comparação realizada por este módulo permite identificar as divergências em diferentes modelos taxonômicos utilizados por grupos de pesquisa distintos.

3.4 Cenário sem Repositórios Semânticos

É possível um cenário de utilização do Serviço de Ontologias não associado a Repositórios Semânticos. Neste caso, as próprias ontologias seriam enviadas como parâmetro às invocações dos módulos do serviço e as ontologias produzidas pelos módulos (arquivos OWL) seriam diretamente retornadas às aplicações cliente. Uma operação de busca com *ranking* que retornasse um conjunto de 10 ontologias poderia ser inviável, dado o tamanho das ontologias envolvidas. Além disso, as aplicações cliente teriam códigos mais complexos, para serem capazes de processar os resultados de invocações e recuperar as ontologias nesses resultados.

Na invocação dos demais módulos, embora viável, essa abordagem não se mostra adequada às necessidades do serviço Aondê. Uma mesma ontologia pode ser utilizada várias vezes, em diferentes invocações. Além disso, o resultado de uma invocação pode ser freqüentemente associado à invocação de outros módulos e, com isso, uma mesma ontologia seria submetida ou retornada seguidas vezes pelo serviço. Finalmente, sem a utilização de Repositórios Semânticos não seria possível prover o compartilhamento e reuso das ontologias. A recuperação de informações de relacionamentos entre ontologias não seria viável, como tampouco seria possível relacionar versões da mesma ontologia ou alinhamentos e visões baseados em uma mesma ontologia.

3.5 Conclusões

Este capítulo apresentou a especificação do Serviço Web de Ontologias – Aondê, sua arquitetura e funcionalidades. Os aspectos de implementação do serviço serão apresentados no Capítulo 4 e exemplos reais da utilização de cada funcionalidade desta especificação são mostrados no Capítulo 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Difference>
   <ListModification>
      <Modification>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Mammal</ElementInOntA>
         <ElementInOntB>http://www.owl-ontologies.com/Publications.owl#Mammalia</ElementInOntB>
         <Type>classe labels</Type>
      </Modification>
       <Modification>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Feline</ElementInOntA>
         <ElementInOntB>http://www.owl-ontologies.com/Publications.owl#Felid</ElementInOntB>
         <Type>class labels</Type>
      </Modification>
      <Modification>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Gorilla</ElementInOntA>
         <ElementInOntB>http://www.owl-ontologies.com/Publications.owl#Gorilla</ElementInOntB>
         <Type>class hierarchy</Type>
      </Modification>
       <Modification>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#hasOrder</ElementInOntA>
         <ElementInOntB>http://www.owl-ontologies.com/Publications.owl#Order</ElementInOntB>
         <Type>property labels</Type>
      </Modification>
   </ListModification>
   <ListInsertion>
      <Insertion>
         <ElementInOntB>http://www.owl-ontologies.com/Publications.owl#Whale</ElementInOntologyB>
      </Insertion>
       </ListInsertion>
   <ListDelection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Monkey</ElementInOntologyA>
      </Delection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Mandrill</ElementInOntologyA>
      </Delection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Publications.owl#Cow</ElementInOntologyA>
      </Delection>
    </ListDelection>
</Difference>
```

Figura 3.6: Resultado XML da operação de detecção de diferenças

Capítulo 4

Aspectos de Implementação

Um protótipo do Serviço Web de Ontologias Aondê foi implementado na linguagem Java. O acesso e a navegação ao conteúdo das ontologias é provido pelo *framework* Jena [15] versão 2.4. O *framework* é composto por uma API RDF, uma API OWL, mecanismos de armazenamento persistente em bancos de dados relacionais, suporte às linguagens de consulta RDQL e SPARQL e mecanismos de inferência baseada em regras.

A implementação de serviços Web utiliza o *framework* de código aberto Apache Axis. Esse *framework* consiste em uma implementação Java de um servidor SOAP e vários utilitários e APIs para criação e publicação de serviços Web. A publicação customizada de serviços Web requer a especificação de um descritor WSDD (*Web Service Deployment Descriptor*), utilizado para especificar quais recursos da aplicação devem ser expostos como serviços Web.

As ontologias para os teste do serviço Aondê foram construídas utilizando a ferramenta Protégé [33]. Essa ferramenta possui um *plug-in* que permite a manipulação de ontologias na linguagem OWL, com uma interface gráfica para a criação de classes, propriedades, instâncias e axiomas em lógica descritiva.

As próximas seções descrevem os aspectos de implementação de cada camada do serviço Aondê. A Seção 4.1 apresenta os aspectos relacionados às ontologias e aos metadados. A Seção 4.2 descreve a implementação dos Repositórios Semânticos e a Seção 4.3 descreve a implementação da Camada de Operações.

4.1 Construção de Ontologias e Estruturas de Metadados

Os Repositórios Semânticos são compostos, basicamente, por pares de arquivos <*ontologia, metadado*>, em que as ontologias são expressas em OWL (*Web Ontology Language*) [7] e

os metadados em OMV (*Ontology Metadata Vocabulary*) [36]. A linguagem OWL suporta diferentes níveis de expressividade (*Lite*, *DL* e *Full*, como mostrado na Seção 2.2.2). Essa é uma das razões pela qual essa linguagem foi adotada, pois o nível de detalhamento das ontologias manipuladas pelo serviço não é previamente conhecido. Essas ontologias podem ser extraídas de repositórios Web ou criadas pelos próprios usuários do serviço. A Figura 4.1 ilustra parte do arquivo OWL que representa a ontologia sobre interações entre insetos e plantas mostrada na Figura 2.3.

```
<owl:Class rdf:ID="Capitulum">
  <rdfs:subClassOf> <owl:Restriction> <owl:allValuesFrom>
      <owl:Class rdf:about="#Insect"/>
    </owl:allValuesFrom> <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isPreyedOn"/>
    </owl:onProperty> </owl:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf> <owl:Restriction> <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasPlantSpecies"/>
    </owl:onProperty> <owl:allValuesFrom>
      <owl:Class> <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Plant"/> <owl:Class rdf:ID="Species"/>
       </owl:intersectionOf> </owl:Class>
    </owl:allValuesFrom> </owl:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf> <owl:Restriction> <owl:onProperty>
     <owl:ObjectProperty rdf:ID="isCapitulumOfCollection"/>
    </owl:onProperty> <owl:allValuesFrom>
     <owl:Class rdf:ID="Collection"/> </owl:allValuesFrom> </owl:Restriction>
  </rdfs:subClassOf> </owl:Class>
<Capitulum rdf:ID="Capitulum_PIC02070_1">
  <hasNumInd rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1
  </hasNumInd>
  <isCapitulumOfCollection> <Collection rdf:ID="Collection_PIC02070"/>
  </isCapitulumOfCollection>
  <hasPlantSpecies> <Identified rdf:ID="Chromolaena_odorata">
    <hasAuthor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    (L.) R.M. King; H. Rob.</hasAuthor>
    <rdf:type rdf:resource="#Plant"/> </Identified> </hasPlantSpecies>
  <isPreyedOn> <Insect rdf:ID="Adaina_bipunctata">
    <isEndemic rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
    false</isEndemic>
    <hasAuthor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Tutt, 1905</hasAuthor>
    <rdf:type rdf:resource="#Identified"/> </Insect> </isPreyedOn>
 </Capitulum>
```

Figura 4.1: Representação OWL da ontologia ilustrada na Figura 2.3

O padrão OMV, adotado na representação dos metadados, é formalizado na linguagem RDF usando o *namespace* **omv** *http://omv.ontoware.org/omv-core-1.0#*. Os elementos especificados pela OMV são categorizados de acordo com seu tipo e propósito da seguinte forma:

- **Geral:** elementos que fornecem informações gerais sobre a ontologia: *URI, name,*

acronym, description, creationDate (obrigatório) e *documentation, keywords, status, modificationDate* (opcional);

- **Disponibilidade:** informações a respeito da localização da ontologia (publicação na Web): *resourceLocator, version* (obrigatório) e *hasLicense* (opcional);
- **Aplicabilidade:** elementos que descrevem a intenção de uso ou escopo da ontologia: *isOfType* (obrigatório) e *hasDomain, naturalLanguage, designedForOntologyTask, hasFormalityLevel* (opcional);
- **Formato**(obrigatório)**:** informações sobre a representação física da ontologia, incluindo a linguagem em que a ontologia está formalizada: *hasOntologyLanguage, hasOntologySyntax*;
- **Procedência:** elementos sobre organizações e colaboradores que contribuíram na criação da ontologia: *hasCreator* (obrigatório) e *hasContributor, usedOntologyEngineeringTool, usedOntologyEngineeringMethodology, usedKnowledgeRepresentationParadigm* (opcional);
- **Relacionamentos** (opcional)**:** informações sobre relacionamentos com outras ontologias, incluindo versões, extensões, generalizações/especializações e importações de outras ontologias: *useImports, hasPriorVersion, isBackwardCompatibleWith, isIncompatibleWith*;
- **Estatísticas** (obrigatório)**:** métricas fundamentadas na topologia do grafo que representa a ontologia: *numClasses, numProperties, numIndividuals, numAxioms*.

A Figura 4.2 apresenta uma estrutura de metadados OMV relacionada à ontologia ilustrada na Figura 4.1. Esses metadados mostram, por exemplo, que a ferramenta utilizada na construção da ontologia foi o Protégé e que a lógica descritiva foi utilizada na representação do conhecimento. Além disso, essa estrutura de metadados informa que a ontologia possui 1259 instâncias, e foi criada em 2007-02-15.

4.2 Implementação dos Repositórios Semânticos

Para a realização de testes do serviço Aondê, foi implementada uma infra-estrutura para a construção de Repositórios Semânticos. A infra-estrutura abrange um *script* (em SQL) para a construção do esquema físico do banco de dados, funções de manipulação e o encapsulamento desse repositório como serviço Web e um arquivo para a configuração dos parâmetros de conexão com o banco de dados. Desta forma, é possível criar facilmente

```
<rdf:RDF
   xmlns:xsd="http://www.w3c.org/2001/XMLSchema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:omv="http://omv.ontoware.org/omv-core-1.0#">
 <rdf:Description
       rdf:about="http://www.owl-ontologies.com/Collection.owl#">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
 <rdf:type rdf:resource="http://omv.ontoware.org/omv-core-1.0#Ontology"/>
       <omv:hasOntologySyntax>OWL-XML</omv:hasOntologySyntax>
  <omv:hasCreator>Computer science and biology researchers from
  UNICAMP</omv:hasCreator>
  <omv:resourceLocator>unknown</omv:resourceLocator>
  <omv:description>Describes interactions among insects and plants
       collected in capitula Zoology department, biology institute.
       </omv:description>
  <omv:URI>http://www.owl-ontologies.com/Collection.owl#</omv:URI>
  <omv:hasOntologyLanguage>OWL</omv:hasOntologyLanguage>
  <omv:version>1.0</omv:version>
  <omv:numClasses>16</omv:numClasses>
  <omv:numProperties>12</omv:numProperties>
  <omv:numAxioms>7</omv:numAxioms>
  <omv:numIndividuals>1259</omv:numIndividuals>
  <omv:acronym>colUN</omv:acronym>
  <omv:creationDate>2007-02-15</omv:creationDate>
  <omv:isOfType>Domain Ontology</omv:isOfType>
  <omv:name>collection.owl</omv:name>
  <omv:keywords>insect, plant, herbivore, capitulum</omv:name>
  <omv:usedKnowledgeRepresentationParadigm> Description Logics
       </omv:usedKnowledgeRepresentationParadigm>
  <omv:usedOntologyEngineeringTool>Protégé
       </omv:usedOntologyEngineeringTool>
  <omv:naturalLanguage>English</omv:naturalLanguage>
 </rdf:Description>
</rdf:RDF>
```

Figura 4.2: Metadado OMV para a ontologia ilustrada na Figura 4.1

tantos Repositórios Semânticos quantos desejáveis. Essa infra-estrutura deve estar disponível para que os usuários do serviço Aondê possam criar seus próprios Repositórios Semânticos e utilizá-los nas invocação aos módulos do serviço.

O *framework* Jena, adotado na implementação do serviço, manipula cada ontologia como um modelo orientado a objetos abstrato, com métodos que permitem a recuperação de elementos a partir de seus tipos definidos. Desta forma, uma ontologia OWL pode ser manipulada não só como um conjunto de triplas RDF (Sujeito–Predicado–Objeto), mas também como um conjunto de classes, instâncias e propriedades.

O mecanismo de persistência do Jena permite que ontologias sejam armazenadas em bancos de dados relacionais. O SGBD PostgreSQL foi adotado na implementação dos repositórios. A Figura 4.3 apresenta o diagrama entidade-relacionamento dos Repositórios Semânticos. Tanto as ontologias como os seus metadados são armazenados como triplas

de grafos RDF. Jena armazena essas triplas em entidades chamadas de *jena-stmt*, que são associadas a cada grafo RDF armazenado na entidade *jena-graph* (ontologia ou metadado). O identificador de um grafo RDF em Jena (campo *id* em *jena-graph*) é único, sendo utilizado como chave estrangeira para relacionar um grafo a seu conjunto de triplas (*jena-stmt*). O atributo *name* em *jena-graph* também é único, e é utilizado como chave pelo *framework* na recuperação de um grafo RDF. A Figura 4.3 mostra também que cada estrutura de metadados é associada a uma ontologia (entidades *MetadadoEstr* e *Ontologia*) por meio de seus identificadores (*id*). A caixa pontilhada no diagrama indica as entidades fornecidas pelo *framework* Jena.

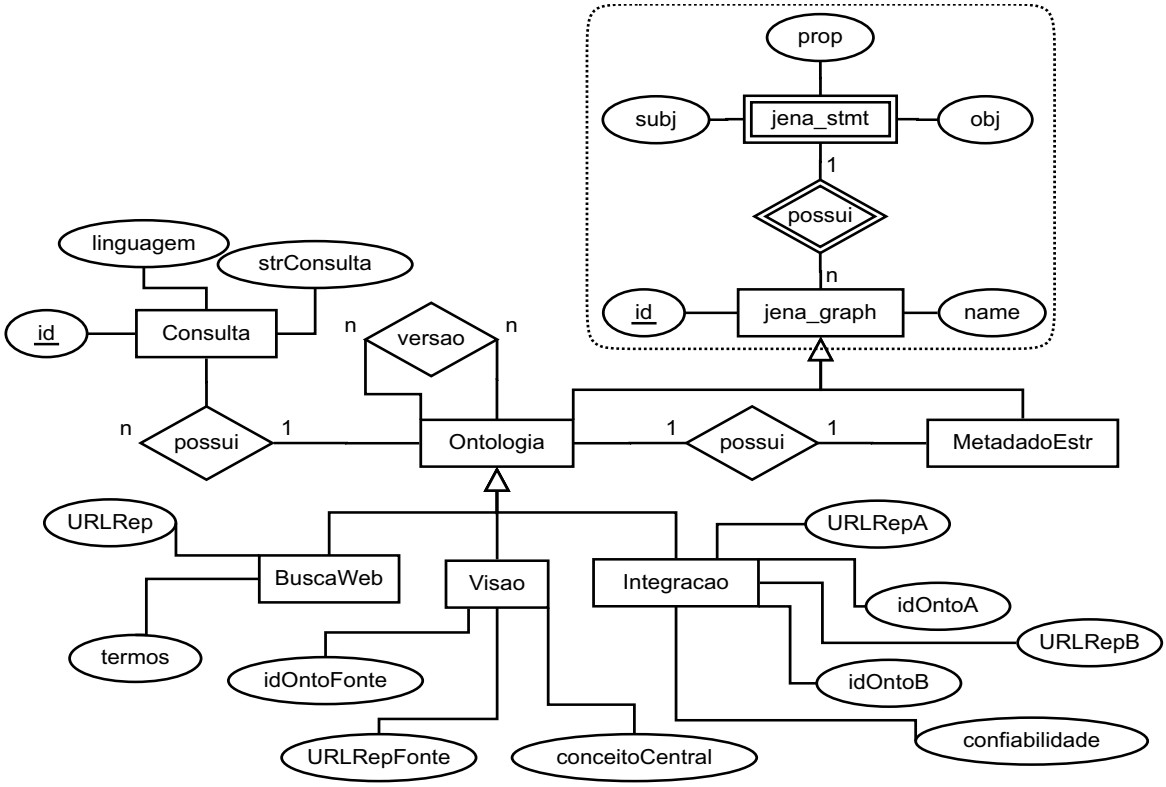

Figura 4.3: Diagrama Entidade-Relacionamento do Repositório Semântico

As ontologias geradas pelo serviço podem ser resultado de invocações aos módulos Busca, Visão e Integração da camada de operações, e essas categorias são representadas pelas entidades *BuscaWeb*, *Visao* e *Integracao*, especializações da entidade *Ontologia*. Ontologias das duas últimas categorias armazenam o identificador de suas ontologias fonte, composto pelo par: *<idOnto, URLRep>*. A entidade *Consulta* armazena as especificações de consultas requisitadas ao serviço e são associadas às ontologias correspondentes.

No encapsulamento dos repositórios como serviços Web, as invocações das operações e

os seus resultados são mensagens SOAP com arquivos OWL/RDF anexados. A inserção de ontologias e metadados via Jena requer que as aplicações cliente forneçam um "nome" (campo *name* em *jena-graph*) que deverá identificar univocamente um grafo RDF no banco de dados. Dada a dificuldade de se assegurar a validade de um identificador textual, as operações de inserção do serviço geram, a partir do nome fornecido pelo usuário (como sua preferência) um identificador único no repositório. Esse "nome" é retornado pelas operações em caso de sucesso.

Os métodos disponíveis para os Repositórios Semânticos são apresentados na Tabela 4.1.

Tabela 4.1: Métodos de Acesso aos Repositórios Semânticos

Método	**Parâmetros**	**Retorno**	**Descrição**
InsertOntologyFile	String ontName, DataHandler Ontology	String idOnto	Insere uma ontologia no repositório a partir de um arquivo.
InsertOntologyURL	String ontName, String urlOnto	String idOnto	Insere uma ontologia remota no repositório a partir de sua URL.
DeleteOntology	String ontName	String result	Elimina uma ontologia do repositório.
UpdateOntologyFile	String ontName, DataHandler Ontology	String result	Atualiza uma ontologia no repositório a partir de um arquivo.
UpdateOntologyURL	String ontName, String urlOnto	String result	Atualiza uma ontologia no repositório a partir de sua URL.
GetOntology	String ontName	DataHandler Ontology	Recupera uma ontologia do repositório.
ListOntologies		String [ ] Ontologies	Lista todas as ontologias armazenadas no repositório.
InsertQuery	String language, String query, String ontName	String result	Armazena a especificação de uma consulta realizada no repositório.

continua

Tabela 4.1 – continuação

Método	Parâmetros	Retorno	Descrição
RegisterView	String ontName, String ontSource, String URLRepSource, String centralConcept	String result	Registra no repositório uma ontologia gerada pelo módulo Visão.
RegisterSearchWeb	String ontName, String repository, String terms	String result	Registra no repositório uma ontologia gerada pelo módulo Busca e *Ranking*.
RegisterAlignment	String ontName, String ontSourceA, String URLRepA, String ontSourceB, String URLRepB, double α, double β, double trustworthiness	String result	Registra no repositório uma ontologia gerada pelo módulo Integração.
InsertMetadataFile	String ontName, DataHandler Metadata	String result	Insere a estrutura de metadados de uma ontologia no repositório a partir de um arquivo.
InsertMetadataURL	String ontName, String urlMeta	String result	Insere a estrutura de metadados de uma ontologia no repositório a partir de sua URL.
DeleteMetadataFile	String ontName	String result	Elimina a estrutura de metadados de uma ontologia do repositório.
ReplaceMetadataFile	String ontName, DataHandler Metadata	String result	Substitui a estrutura de metadados de uma ontologia no repositório a partir de um arquivo.

continua

Tabela 4.1 – continuação

Método	Parâmetros	Retorno	Descrição
ReplaceMetadataURL	String ontName, String urlMeta	String result	Substitui a estrutura de metadados de uma ontologia no repositório a partir de sua URL.
GetMetadataFile	String ontName	DataHandler Metadata	Recupera a estrutura de metadados de uma ontologia no repositório.
GetQuery	String ontName, String language	String[ ] Queries	Recupera as consultas em uma dada linguagem realizadas para uma ontologia do repositório.
GetRelatedOntology	String ontName, String type	String[ ] Ontologies	Recupera as ontologias relacionadas a uma ontologia no repositório. Essas ontologias podem ser novas versões, alinhamentos ou visões dessa ontologia.
GetInfoBuscaWeb	String ontName	String[ ] Info	Recupera os atributos relacionados a uma ontologia proveniente de uma busca – *URLRep* e *termos*.
GetInfoView	String ontName	String[ ] Info	Recupera os atributos relacionados a uma ontologia proveniente de uma visão – *URLRepFonte, idOntoFonte* e *conceitoCentral*.

continua

Tabela 4.1 – continuação

Método	Parâmetros	Retorno	Descrição
GetInfoIntegration	String ontName	String[ ] Info	Recupera os atributos relacionados a uma ontologia proveniente de uma integração – *idOntoA, URLRepA, idOntoB, URLRepB*, α, β e *confiabilidade*.

A Figura 4.4 apresenta um trecho da especificação WSDL do método *InsertOntologyFile*, que realiza a inserção de uma ontologia no repositório a partir de um arquivo OWL. As duas seções <*wsdl:message*> indicam os parâmetros enviados e recebidos por mensagens SOAP na invocação desta operação. Como ilustrado na figura, uma requisição a este método (*InsertOntologyFileRequest*) possui como parâmetros o identificador da ontologia (*idOnto*) e o arquivo OWL (*OWLFile*). A *string* retornada indica se a operação foi realizada com sucesso ou não. Em caso de sucesso, o identificador *idOnto* (ou uma adaptação dele, caso necessário) é retornado. Em caso de falha (arquivo OWL inválido, por exemplo), a *string* retornada indica o motivo do insucesso. Os demais métodos dos Repositórios Semânticos possuem especificações WSDL similares.

4.3 Implementação da Camada de Operações

Esta camada abrange a implementação de todas as operações previstas na arquitetura proposta: Gerência dos Repositórios (Seção 4.3.1), Consultas (Seção 4.3.2), Busca e *Ranking* (Seção 4.3.3), Visões (Seção 4.3.4), Integração (Seção 4.3.5) e Detecção de Diferenças (Seção 4.3.6). A Camada de Operações foi implementada de forma que, como previsto na arquitetura, todas as invocações a módulos realizam acessos aos Repositórios Semânticos. Apenas o módulo de Busca e *Ranking* acessa também Repositórios Externos.

A Tabela 4.2 apresenta os métodos disponíveis para esta camada no serviço Aondê. O funcionamento de cada um desses métodos é detalhado nas seções subseqüentes.

```
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/WSSemanticRepository
<wsdl:message name="InsertOntologyFileRequest">
   <wsdl:part name="idOnto" type="soapenc:string" />
   <wsdl:part name="OWLFile" type="apachesoap:DataHandler" />
</wsdl:message>
<wsdl:message name="InsertOntologyFileResponse">
   <wsdl:part name="InsertOntologyFileReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:portType name="WSSemanticRepository">
   <wsdl:operation name="InsertOntologyFile"
      parameterOrder="idOnto OWLFile">
      <wsdl:input message="impl:InsertOntologyFileRequest"
      name="InsertOntologyFileRequest" />
      <wsdl:output message="impl:InsertOntologyFileResponse"
      name="InsertOntologyFileResponse" />
   </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="WSSemanticRepositorySoapBinding"
   type="impl:WSSemanticRepository">
   <wsdlsoap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="InsertOntologyFile">
   <wsdlsoap:operation soapAction="" />
   <wsdl:input name="InsertOntologyFileRequest">
   <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://webService_pckg.OntologyService" use="encoded" />
</wsdl:input>
<wsdl:output name="InsertOntologyFileResponse">
   <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/WSSemanticRepository"
use="encoded" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

Figura 4.4: Exemplo WSDL do Repositório

Tabela 4.2: Métodos de Acesso à Camada de Operações

Método	Parâmetros	Retorno	Descrição
getCatalogue		String[ ] URLReps	Recupera o conjunto de Repositórios Semânticos cadastrados no catálogo do serviço.
			continua

Tabela 4.2 – continuação

Método	Parâmetros	Retorno	Descrição
Query	String ontName, String URLRep, String language, String strQuery, boolean inference, String outFormat	DataHandler result	Executa uma consulta em uma dada ontologia.
Difference	String ontNameA, String URLRepA, String ontNameB, String URLRepB, double alpha, double beta, double trustworthness	DataHandler result	Detecta as diferenças de estrutura e conteúdo entre duas ontologias dadas.
Search	String term, String[ ] directives, String[ ] URLReps, String targetRep	String idOnto	Realiza a busca por uma ontologia específica em repositórios externos distribuídos na Web.
SearchRank	String[ ] terms, double[ ] weights, String[ ] URLReps, String targetRep	String[ ] idOnto	Realiza a busca por ontologias em repositórios externos distribuídos na Web, e as retorna ordenadas de acordo com seus valores de *ranking* para os termos buscados.
View	String ontName, String URLRep, String concept, String[ ] directives, String targetRep	String idOnto	Extrai uma visão de uma ontologia dada, baseada em um conceito central.

continua

Tabela 4.2 – continuação

Método	Parâmetros	Retorno	Descrição
Integration	String ontNameA, String URLRepA, String ontNameB, String URLRepB, double alpha, double beta, double trustworthness, String targetRep	String idOnto	Realiza a integração de duas ontologias, alinhando seus elementos equivalentes.

4.3.1 Gerência dos Repositórios Semânticos

A implementação deste módulo consiste, basicamente, em disponibilizar métodos clientes para todas as invocações previstas para um Repositório Semântico (apresentados na tabela 4.1). Esses métodos são utilizados pelos demais módulos na comunicação com Repositórios Semânticos, seja na recuperação de ontologias – parâmetros de entrada na execução dos módulos – ou no armazenamento de ontologias construídas pelos módulos como resultado de operações.

O acesso a ontologias nos Repositórios Semânticos é realizado por meio de seus identificadores: <*idOnto, URLRep*>. Como ilustra a Figura 4.5, o parâmetro *URLRep* identifica a URL em que o Repositório Semântico está disponível. O *idOnto* é utilizado para recuperar a ontologia desse repositório (por meio da invocação do método *GetOntology* nesta URL). Os demais métodos deste módulo funcionam de forma análoga. Uma invocação particular deste módulo é por meio do método *getCatalogue*, que retorna o conteúdo do catálogo de Repositórios Semânticos, contendo todas as URL's de repositório previamente utilizadas em invocações ao Aondê.

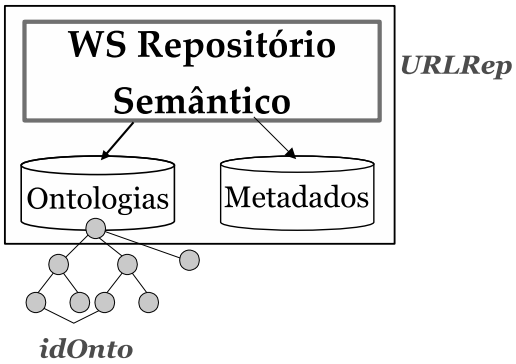

Figura 4.5: Acesso a ontologia *idOnto* no Repositório Semântico endereçado por *URLRep*

4.3.2 Consultas

A implementação do módulo de consultas suporta as linguagens RDQL [62] e SPARQL [58]. A versão mais atual do *framework* Jena, 2.5.3, não suporta consultas RDQL e por esta razão não foi adotada na implementação. Ambas linguagens realizam as consultas na representação RDF das ontologias, ou seja, no conjunto de triplas *Sujeito–Predicado–Objeto*. A execução de uma consulta consiste em selecionar quais triplas da ontologia satisfazem os predicados da cláusula *where*. A partir das triplas selecionadas, as variáveis da cláusula *select* são retornadas como resultado da consulta. O funcionamento deste módulo é descrito no Algoritmo 1.

Algoritmo 1 Consulta (idOnto, URLRep, linguagem, strConsulta, inferencia, formato)

```
1: ontoModel ← URLRep.GetOntologyFile(idOnto)
2: Se inferencia então
3:    infModel ← ontoModel.criarModeloIinferido()
4: Senão
5:    infModel ← ontoModel
6: Fim se
7: Se linguagem = RDQL então
8:    resultado ← infModel.executarConsultaRDQL(strConsulta)
9: Senão
10:   resultado ← infModel.executarConsultaSPARQL(strConsulta)
11: Fim se
12: URLRep.InsertQuery(linguagem, strConsulta, nomeOnto)
13: Se formato = XML então
14:   Retorne resultado.converterArquivoXML()
15: Senão
16:   Retorne resultado.converterArquivoTexto()
17: Fim se
```

Considerando a especificação de uma invocação (vide seção 3.3.2), um exemplo de invocação para esse módulo é:

Consulta(<ontoA, http://143.0106.23.89:8080/OntRepUNICAMP>, SPARQL, "SELECT ?subject ?object WHERE {?subject rdf:subClassOf ?object}", true, XML)

Nessa invocação, o identificador da ontologia consultada é dado por: *<ontoA, http://143.0106.23.89:8080/OntRepUNICAMP>*; a linguagem de consulta é *SPARQL*; a *string* que especifica a consulta é *"SELECT ?subject ?object WHERE {?subject rdf:subClassOf ?object}"*; o parâmetro *true* indica que a consulta deve ser realizada no modelo inferido da ontologia; e o formato de saída requisitado é *XML*. Para a construção do modelo inferido

(linha 3), utiliza-se a máquina de inferência *default* para OWL provida pelo Jena. A *string* que representa a consulta – o campo *strConsulta*– é armazenada no mesmo Repositório Semântico da ontologia *ontoA*.

O módulo de Gerência de Repositórios é utilizado nas operações das linhas 1 e 12, por meio dos métodos *GetOntologyFile* e *InsertQuery*, respectivamente.

4.3.3 Busca e *Ranking*

A implementação deste módulo consiste em buscar, em um conjunto de repositórios, ontologias que possuam classes ou instâncias cujos nomes casam (exata ou parcialmente) com os termos buscados. Na busca por vários termos, uma ontologia é retornada caso ela possua pelo menos um dos termos buscados.

Não foram encontrados repositórios para ser utilizados como Repositórios Externos de Ontologias – encapsulados em serviços Web que seguissem os padrões especificados na Seção 2.2.1 (interface WSDL e comunicação por mensagens SOAP). Por esta razão, os Repositórios Externos utilizados na implementação deste módulo são acessados via requisições HTTP. Essas requisições possuem a forma: *URLservico?parametros*, sendo que vários *parametros* podem ser combinados pelo conector "&". O funcionamento de uma busca com *ranking* é descrito no Algoritmo 2.

De acordo com a especificação dessa invocação (vide seção 3.3.3), um exemplo de invocação para essa operação é:

$$BuscaRank(\{insect,\ plant\}, \{0.3, 0.25, 0.2, 0.25\}, \{Swoogle\},\\ http://143.0106.23.89:8080/OntRepUNICAMP)$$

O Repositório Externo utilizado nesta invocação é o Swoogle[1] [23], acessado pela URL *"http://logos.cs.umbc.edu:8080/swoogle31/q"*. O retorno desta requisição é um arquivo RDF, que contém as URL's das ontologias que contêm os termos buscados e alguns metadados definidos pelo Swoogle, que são utilizados na construção automática da estrutura de metadados no padrão OMV (linha 6). A construção automática da estrutura de metadados utiliza apenas os campos obrigatórios do padrão OMV. Nesta invocação, os termos buscados são {*insect, plant*}. Como o repositório destino é diferente do repositório buscado, as ontologias retornadas na busca são armazenadas, criando-se uma estrutura de metadados associada a cada ontologia.

As métricas de *ranking* utilizadas na implementação (linhas 19 a 22) são as mesmas descritas na Seção 2.3.2.1: *Casamento exato e parcial*, *Densidade*, *Centralidade*, *Similaridade Semântica*. Algumas adaptações foram realizadas nessas métricas, permitindo que o termo buscado pudesse ser encontrado também em instâncias, e não apenas em classes nas

[1] http://swoogle.umbc.edu

Algoritmo 2 BuscaRank (termo[ ], peso[ ], URLRep[ ], URLRepDestino)

```
1: Para todo repositorio em URLRep[ ] faça
2:   retornoBusca ← repositorio.Busca(termo[ ])
3:   Se retornoBusca ≠ null então
4:     Se repositorio ≠ URLRepDestino então
5:       Para todo ontoURL em resultado faça
6:         metadado ← retornoBusca.criarEstruturaMetadado(ontoURL)
7:         identificador ← URLRepDestino.InsertOntologyURL(termo, ontoURL)
8:         URLRepDestino.InsertMetadataFile(metadado)
9:         URLRepDestino.RegisterSearchWeb(identificador, repositorio, termo)
10:        resultado.adicionar(identificador)
11:      Fim para todo
12:    Fim se
13:  Senão
14:    identificador ← repositorio.Identificador(ontoArquivo)
15:    resultado.adicionar(identificador)
16:  Fim se
17: Fim para todo
18: Para todo ontologia em resultado faça
19:   ontologia.metricaCasamento()
20:   ontologia.metricaDensidade()
21:   ontologia.metricaCentralidade()
22:   ontologia.metricaSimilaridadeSemantica()
23: Fim para todo
24: resultado.normalizarMetricas()
25: resultado.obterRank(peso[ ])
26: resultado.ordenar()
27: Retorne resultado.Identificadores()
```

ontologias. Os pesos $\{0.3, 0.25, 0.2, 0.25\}$ são atribuídos da seguinte forma: *Casamento exato e parcial*: 0.3, *Densidade*: 0.25, *Centralidade*: 0.2, *Similaridade semântica*: 0.25. Note que os valores do conjunto de pesos possuem soma igual a 1.

O módulo de Gerência de Repositórios é responsável pelas operações *InsertOntologyURL*, *InsertMetadataFile* e *RegisterSearchWeb* (linhas 7, 8 e 9, respectivamente). As ontologias são diretamente acessadas por suas URL's no armazenamento no Repositório Semântico *http://143.0106.23.89:8080/OntRepUNICAMP*.

O funcionamento de uma busca simples (sem *ranking*) é descrito no Algoritmo 3.

Algoritmo 3 Busca (termo, diretiva[ ], URLRep[ ], URLRepDestino)

```
1: Para todo repositorio em URLRep[ ] faça
2:   ontoArquivo ← repositorio.Busca(termo, diretiva[ ])
3:   Se ontoArquivo ≠ null então
4:     Se repositorio ≠ URLRepDestino então
5:       metadado ← ontoArquivo.criarEstruturaMetadado()
6:       identificador ← URLRepDestino.InsertOntologyFile(termo, ontoArquivo)
7:       URLRepDestino.InsertMetadataFile(metadado)
8:       URLRepDestino.RegisterSearchWeb(identificador, repositorio, termo)
9:       Retorne identificador
10:    Fim se
11:  Senão
12:    identificador ← repositorio.Identificador(ontoArquivo)
13:    Retorne identificador
14:  Fim se
15: Fim para todo
16: Retorne null
```

De acordo com a especificação de Busca (vide seção 3.3.3), um exemplo de invocação para essa operação é:

$$Busca(asteraceae, \{siblings,\ descendants\}, \{Spire\},\ http://143.0106.23.89:8080/OntRepUNICAMP)$$

O Repositório Externo utilizado neste caso é o portal Spire[2] [55], acessado pela URL *"http://spire.umbc.edu/ont/ethan-part.php"*. O retorno dessa requisição é um arquivo OWL que contém uma ontologia com a hierarquia de classes que representa a árvore taxonômica do termo buscado (no caso, um táxon). O táxon buscado nessa invocação é *asteraceae* (uma família de inflorescências), e as diretivas {*siblings, descendants*} indicam que a ontologia resultado deve conter os táxons irmãos e os de nível hierárquico inferior a *asteraceae*.

[2] http://spire.umbc.edu/ont/

Nessa invocação, a ontologia retornada é armazenada no repositório designado pela URL *http://143.0106.23.89:8080/ OntRepUNICAMP* e é criada uma estrutura de metadados para essa ontologia. Além disso, essa ontologia é registrada no repositório como proveniente de uma busca. O módulo de Gerência de Repositórios é utilizado nas operações das linhas 6, 7 e 8, por meio dos métodos *InsertOntologyFile*, *InsertMetadataFile* e *RegisterSearchWeb*, respectivamente. Caso a ontologia não seja encontrada, o módulo retorna *null* como resultado (linha 16).

Apesar dos exemplos de invocação terem mostrado apenas requisições a Repositórios Externos, os Repositórios Semânticos também podem ser utilizados em buscas, com ou sem *ranking*.

4.3.4 Visões

A implementação do módulo de visões utiliza a abordagem proposta em [52], descrita na Seção 2.3.2.3. Segundo essa abordagem, a extração de uma visão é baseada em um conceito central e seus elementos relacionados. Essa abordagem foi estendida para preservar os axiomas na visão. O funcionamento deste módulo é descrito no Algoritmo 4.

De acordo com a especificação de Visões (vide seção 3.3.4), um exemplo de invocação para essa operação é:

$$Visao(<ontoA,\ http://143.0106.23.89:8080/OntRepUNICAMP>,\ Plant,\{superclass{:}2,\ subclass{:}3\},http://150.0471.42.75:8080/OntRepUNICAMP)$$

Nessa invocação, a visão extraída da ontologia *ontoA* tem como conceito central *Plant*, e as diretivas indicam que as superclasses com distâncias 1 e 2 e as subclasses de *Plant* com distâncias 1, 2 e 3 devem fazer parte da visão (linhas 2, 5 e 7 do algoritmo). A Figura 4.6 ilustra, em alto nível, o processo de construção da visão. A Figura 4.6(a) mostra a visão dentro da ontologia original e a Figura 4.6 (b) apresenta o resultado gerado pelo módulo.

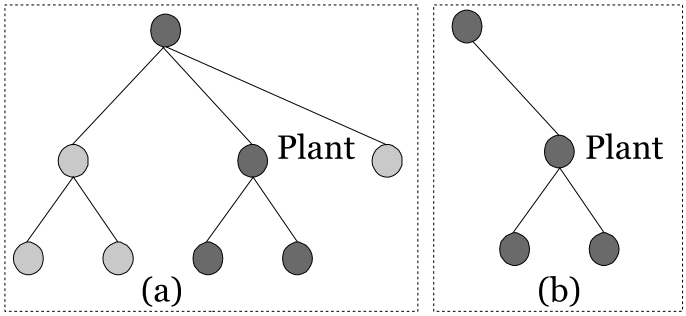

Figura 4.6: Seleção dos conceitos e extração da visão

Algoritmo 4 Visao (idOnto, URLRep, conceito, diretiva[ ], URLRepDestino)

```
1: ontoModel ← URLRep.GetOntologyFile(idOnto)
2: visao ← criarOntologia(conceito)
3: Para todo dirt em diretiva[ ] faça
4:    Se irt = superclasse então
5:       visao.adicionarSuperclasse(dirt.profundidade)
6:    Senão Se dirt = subclasse então
7:       visao.adicionarSubclasse(dirt.profundidade)
8:    Senão Se dirt = axioma então
9:       visao.adicionarAxioma()
10:   Senão Se dirt = instancia então
11:      visao.adicionarInstancia()
12:   Senão Se dirt = irmao então
13:      visao.adicionarIrmao()
14:   Senão
15:      visao.adicionarPropriedade(dirt)
16:   Fim se
17: Fim para todo
18: identificador ← URLRepDestino.InsertOntologyFile(visao, conceito)
19: metadado ← ontoArquivo.criarEstruturaMetadado()
20: URLRepDestino.InsertMetadataFile(metadado)
21: URLRepDestino.RegisterView(identificador, idOnto, URLRep, conceito)
22: Retorne identificador
```

O módulo de Gerência de Repositórios é utilizado nas linhas 1, 18, 20 e 21 do algoritmo, por meio dos métodos *GetOntology*, *InsertOntologyFile*, *InsertMetadataFile* e *RegisterView*, respectivamente. A ontologia resultado do módulo é registrada no repositório como proveniente da extração de uma visão.

A dificuldade na implementação deste módulo está em percorrer a ontologia fonte. De fato, a construção da visão requer navegar na ontologia fonte para recuperar os elementos definidos pelas diretivas e reconstruir tais elementos na visão. Essa navegação torna-se ainda mais difícil quando o domínio (*domain*) ou o contra-domínio (*range*) de propriedades ou axiomas são compostos pela união, interseção ou complemento de outros elementos (classes e instâncias).

Embora não implementado diretamente, é possível criar visões centradas em mais de um conceito por meio de invocações sucessivas ao serviço Aondê. Para isto, basta extrair visões de cada conceito separadamente e, em seguida, realizar a integração das ontologias obtidas.

4.3.5 Integração

A implementação do módulo de integração realiza o alinhamento entre duas ontologias fontes. Este alinhamento é baseado na combinação de técnicas de elemento e técnicas de estrutura, conforme descrito na Seção 3.3.5. Os elementos analisados no processo de mapeamento são: classes-classes, instâncias-instâncias e classes-instâncias. Esta última abordagem, com elementos de diferentes tipos, ainda não é explorada na literatura, mas possui importantes aplicações no mapeamento de ontologias com diferentes granularidades. O funcionamento deste módulo é descrito no Algoritmo 5. O cálculo da confiabilidade dos mapeamentos utiliza os pesos α e β para combinar as similaridades de elemento e de estrutura.

De acordo com a especificação deste módulo (vide seção 3.3.5), um exemplo de invocação desta operação é:

$$Integracao(<ontoA,\ http://143.0106.23.89:8080/OntRepUNICAMP>, <ontoB,\ http://150.0471.42.75:8080/OntRepBIO>, 0.5, 0.5, 0.75,\ http://150.0471.42.75:8080/OntRepBIO),$$

Nessa invocação, as ontologias *ontoA* e *ontoB*, de dois repositórios diferentes, serão alinhadas e apenas os mapeamentos com confiabilidade superior a 0.75 serão materializados na ontologia construída como resultado, utilizando-se os pesos $\alpha = \beta = 0.5$ para combinar as técnicas de elemento e de estrutura. A Figura 4.7 ilustra, em alto nível, o que ocorre nas linhas 14 e 23 desse algoritmo: a descoberta de mapeamentos candidatos (Figura 4.7 (a)) e a materialização na ontologia apenas daqueles que atendem a confiabilidade mínima (Figura 4.7 (b)).

Algoritmo 5 Integracao (idOntoA, URLRepA, idOntoB, URLRepB, α, β, confiabilidadeMin, URLRepDestino)

```
1: ontoModelA ← URLRepA.GetOntologyFile(idOntoA)
2: ontoModelB ← URLRepB.GetOntologyFile(idOntoB)
3: Para todo elemA em ontoModelA.listar(classes,instancias) faça
4:   Para todo elemB em ontoModelB.listar(classes,instancias) faça
5:     simString ← Jaro(elemA,elemB)
6:     simSinon ← consultarDicionario(elemA,elemB)
7:     simElemento ← Max(simString, simSinon)
8:     Se simElemento ≥ α * confiabilidade então
9:       propr ← 0.25∗ simProp(elemA, elemB)
10:      axioma ← 0.25∗ simAxiom(elemA, elemB)
11:      supercl ← 0.25∗ simSuper(elemA, elemB)
12:      subcl ← 0.25∗ simSub(elemA, elemB)
13:      simEstrutura ← propr + axioma + supercl + subcl
14:      Se α∗simElemento +β∗ simEstrutura ≥ confiabilidadeMin então
15:        mapeamento.adicionar(elemA, elemB)
16:      Fim se
17:    Fim se
18:  Fim para todo
19: Fim para todo
20: alinhamento.adicionarOntologia(ontoModelA)
21: alinhamento.adicionarOntologia(ontoModelB)
22: Para todo map em mapeamento faça
23:   alinhamento.adicionar(map)
24: Fim para todo
25: identificador ← URLRepDestino.InsertOntologyFile(alinhamento, idOntoA.idOntoB)
26: metadado ← ontoArquivo.criarEstruturaMetadado()
27: URLRepDestino.InsertMetadataFile(metadado)
28: URLRepDestino.RegisterAlignment(identificador, idOntoA, URLRepA, idOntoB,
    URLRepB, confiabilidadeMin)
29: Retorne identificador
```

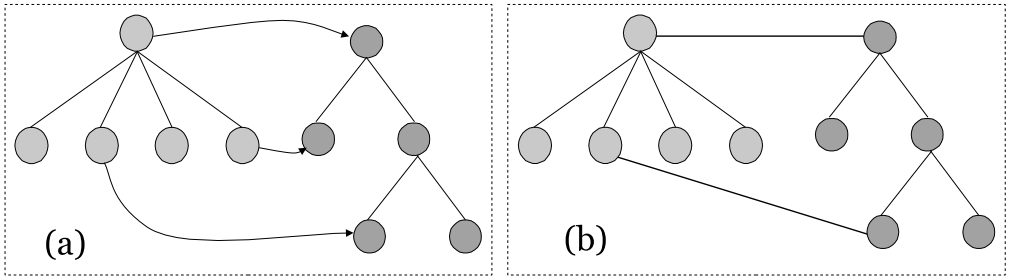

Figura 4.7: Seleção de candidatos e materialização dos mapeamentos no processo de alinhamento

O módulo de Gerência de Repositórios é utilizado nas linhas 1 e 2 para recuperar as ontologias que serão alinhadas (método *GetOntologyFile*). Nas linhas 25, 27 e 28, a ontologia alinhada e sua estrutura de metadados são armazenadas no repositório destino, sendo que a ontologia é registrada como proveniente de uma integração – métodos *InsertOntologyFile*, *InsertMetadataFile* e *RegisterAlignment*.

Na descoberta dos candidatos a mapeamento, são utilizadas as técnicas de elemento similaridade de *string* (implementação de Jaro extraída de [18]) e dicionários de sinônimos. O dicionário de propósito geral WordNet [43] é utilizado na busca por sinônimos dos termos. A versão utilizada desse dicionário é a disponível na URL *"http://jws-champo.actoulouse.fr: 8080/wordnet-xml"*, acessada via requisições HTTP. O resultado dessas requisições são arquivos XML, que são analisados para a extração dos sinônimos (campo *synset*).

Essas técnicas de elemento foram adaptadas para o caso dos nomes tratados serem táxons. Em muitos casos, os nomes científicos dos táxons diferem apenas em seus sufixos, representando diferentes categorias taxonômicas. Por exemplo, considere as *strings Asterales* e *Asteraceae*. Essas *strings*, embora muito similares, se referem a táxons distintos: o sufixo *-ales* indica "ordem" enquanto o sufixo *-aceae* indica "família". Além disso, táxons necessitam de um dicionário de sinônimos específico que possua táxons sinônimos (usados em diferentes épocas ou por diferentes autores) e nomes populares. Por esta razão, a base de dados ITIS (*Integrated Taxonomic Information System*)[3] [47] é utilizada na verificação de sinônimos desse tipo. Esta base de dados está disponível na Web em formato textual. Para facilitar seu processamento, essa base foi importada para o serviço e estruturada em um banco de dados relacional que é acessado por este módulo.

Os candidatos a mapeamento selecionados pelas técnicas de elemento são submetidos às técnicas de estrutura (comparação de propriedades, axiomas e hierarquias). A confiabilidade do mapeamento será representada pela soma ponderada desses valores obtidos, representando um número entre 0 e 1 (vide Seção 3.3.5). A linha 23 do algoritmo 5

[3]http://www.itis.gov

seleciona quais dos mapeamentos obtidos possuem confiabilidade superior à desejada.

Na ontologia alinhada, os mapeamentos encontrados são materializados usando *tags* OWL. As classes alinhadas são relacionadas pela *tag* <*owl:equivalentClass*> e as instâncias pela *tag* <*owl:sameAs*>. O alinhamento entre classes e instâncias é também representado pela *tag* <*owl:sameAs*>.

4.3.6 Detecção de Diferenças

A implementação deste módulo detecta diferenças tanto na estrutura quanto no conteúdo das ontologias. Sua estratégia é baseada em, inicialmente, encontrar os mapeamentos diretos que existem entre os elementos das ontologias (elementos idênticos). Posteriormente, são procurados mapeamentos "indiretos", ou seja, mapeamentos identificados por *strings* similares ou sinônimas com estruturas similares. A descoberta de mapeamentos indiretos utiliza as mesmas técnicas do módulo de integração e representa modificações nas ontologias. Os elementos não mapeados (indireta ou indiretamente) da primeira ontologia são considerados exclusões e os elementos não mapeados na segunda ontologia são considerados inserções no resultado do módulo. O funcionamento deste módulo é descrito no Algoritmo 6.

De acordo com a especificação da Detecção de Diferenças (vide Seção 3.3.6), um exemplo de invocação para essa operação é:

$$Diferenca(<ontoA,\ http://143.0106.23.89:8080/OntRepUNICAMP,\ 0.5,\ 0.5,\ 0.8>, <ontoB,\ http://150.0471.42.75:8080/OntRepBIO>)$$

Nesta invocação, as ontologias *ontoA* e *ontoB* são comparadas, considerando-se o mesmo peso para as similaridades de elemento e de estrutura ($\alpha = \beta = 0.5$) e a confiabilidade mínima das modificações é 0.8. O resultado obtido é um arquivo XML que categoriza as diferenças encontradas de 3 formas: modificações, exclusões e inserções, correspondentes as linhas 24, 30 e 33, respectivamente (conforme descrito anteriormente na Seção 3.3.6). A Figura 4.8 ilustra, em alto nível, o resultado final obtido na comparação – a Figura 4.8 (a) apresenta as ontologias originais a serem comparadas e, na Figura 4.8 (b), os elementos marcados com um "X" foram eliminados e os de cor mais escura (acima da ontologia) foram inseridos. Os demais elementos permaneceram inalterados (foram diretamente mapeados).

O pressuposto para o funcionamento correto deste módulo é que o número de mapeamentos entre as ontologias seja muito superior ao número de elementos não-mapeados entre as ontologias. Caso isto não ocorra, as modificações dificilmente são identificadas e, com isso, este módulo detecta apenas uma seqüência de exclusões e inclusões. A detecção das modificações depende intimamente dos mapeamentos previamente encontrados. O

Algoritmo 6 Diferenca (idOntoA, URLRepA, idOntoB, URLRepB, α, β, confiabilidadeMin)

```
1: ontoModelA ← URLRepA.GetOntologyFile(idOntoA)
2: ontoModelB ← URLRepB.GetOntologyFile(idOntoB)
3: Para todo tipo em {classe, instancia, propriedade} faça
4:   Para todo elemA em ontoModelA.listar(tipo) faça
5:     Para todo elemB em ontoModelB.listar(tipo) faça
6:       Se elemA = elemB então
7:         mapeado.adicionar(elemA, elemB)
8:       Fim se
9:     Fim para todo
10:  Fim para todo
11: Fim para todo
12: Para todo elemA em ontoModelA.listarNaoMapeado(mapeado) faça
13:   Para todo elemB em ontoModelB.listarNaoMapeado(mapeado) faça
14:     simString ← Jaro(elemA,elemB)
15:     simSinon ← consultarDicionario(elemA,elemB)
16:     simElemento ← Max(simString, simSinon)
17:     Se simElemento ≥ α∗ confiabilidadeMin então
18:       propr ← 0.25∗ simProp(elemA, elemB)
19:       axioma ← 0.25∗ simAxiom(elemA, elemB)
20:       supercl ← 0.25∗ simSuper(elemA, elemB)
21:       subcl ← 0.25∗ simSub(elemA, elemB)
22:       simEstrutura ← propr + axioma + supercl + subcl
23:       Se (α∗simElemento +β∗ simEstrutura) ≥ confiabilidadeMin então
24:         alterado.adicionar(elemA, elemB)
25:       Fim se
26:     Fim se
27:   Fim para todo
28: Fim para todo
29: Para todo elemA em ontoModelA.listarNaoMapeado(mapeado,alterado) faça
30:   excluido.adicionar(elemA)
31: Fim para todo
32: Para todo elemB em ontoModelB.listarNaoMapeado(mapeado,alterado) faça
33:   adicionado.adicionar(elemB)
34: Fim para todo
35: resultado.adicionar(alterado)
36: resultado.adicionar(excluido)
37: resultado.adicionar(adicionado)
38: Retorne resultado
```

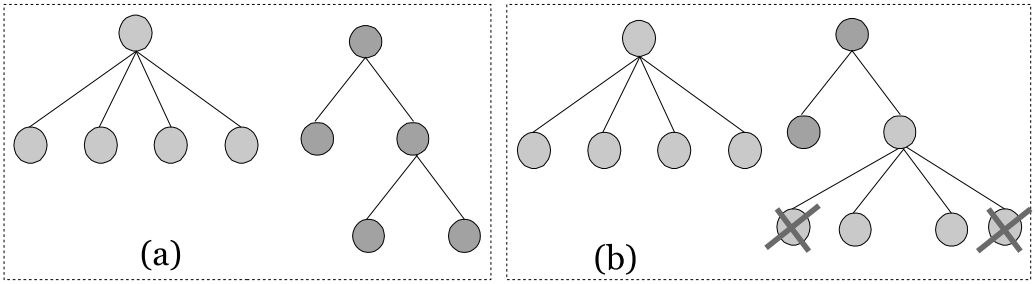

Figura 4.8: Resultado da detecção de diferenças entre duas ontologias

módulo de Gerência de Repositórios é responsável pelas linhas 1 e 2, na recuperação das ontologias comparadas, utilizando o método *GetOntologyFile*.

4.4 Conclusões

Este capítulo apresentou os aspectos de implementação relativos ao protótipo desenvolvido do Aondê, o Serviço Web de Ontologias proposto nesta dissertação. O próximo capítulo apresenta cenários reais de utilização do serviço, em biodiversidade, ressaltando o impacto das decisões de especificação e implementação abordadas neste capítulo.

Capítulo 5

Estudo de Caso

O estudo de caso é baseado em dados reais sobre insetos e plantas, que foram coletados para pesquisas dos biólogos parceiros do projeto WeBios ao longo de 3 anos. O estudo está relacionado com interações entre insetos e inflorescências (conjuntos de flores). Embora aparentem ser apenas uma flor (como margaridas e girassóis), tratam-se na verdade de um conjunto de flores, em que cada flor irá dar origem a uma nova semente da planta. Os biólogos concentram suas pesquisas em plantas da família *Asteraceae* (subfamílias *Asteroideae*, *Barnadesioideae* e *Cichorioideae*), cujas inflorescências são comumente chamadas de *capítulo*.

Os insetos de interesse nas interações estudadas são classificados como "endófagos predadores de sementes" – as larvas dos insetos estão dentro dos capítulos (endófagos) e alimentam-se de suas sementes. A Figura 5.1 ilustra um capítulo dissecado em laboratório, que demonstra esse tipo de interação. Os insetos analisados pertencem às ordens *Diptera* (moscas) e *Lepidoptera* (borboletas). Alguns estudos desenvolvidos pelos biólogos a respeito desses dados podem ser encontrados em [4, 5, 30].

No total, a ontologia criada para o estudo de caso representa 290 registros de coletas compostas por 777 capítulos, que representam interações entre 77 espécies diferentes de insetos e 108 espécies de plantas. A ontologia possui cerca de 2000 elementos, incluindo classes, propriedades, axiomas e instâncias.

5.1 Construção dos Repositórios Semânticos

A primeira etapa do estudo de caso consistiu na construção de uma ontologia que descrevesse o cenário semântico da pesquisa realizada pelos biólogos. Esse cenário é caracterizado da seguinte forma: são realizadas várias coletas em visitas a campo, que ocorrem em diferentes localidades e cada coleta é composta por um conjunto de capítulos (inflorescências de *Asteraceae*) de uma mesma espécie de planta. A intenção é que a coleta

Figura 5.1: Capítulo de uma *Asteraceae* predado por duas larvas de moscas da família *Tephritidae*

represente a população da espécie de planta naquela determinada região. As coletas são levadas aos laboratórios para análise – os capítulos podem conter ovos ou larvas de insetos que, após passarem por várias metamorfoses e atingirem a fase adulta, são submetidos a um processo de identificação.

Em geral, grande parte das espécies dos insetos são identificadas. Essas espécies são associadas a classificações taxonômicas determinadas pelos autores responsáveis pela catalogação da espécie na literatura. Entretanto, considerando a grande dificuldade encontrada pelos biólogos nesse processo – tanto pelo pequeno tamanho dos indivíduos, como pela alta similaridade entre as espécies – parte dos insetos não é identificada. Um espécime não identificado pode ser classificado em três categorias:

- *Quando se trata de uma espécie não descrita:* os biólogos reconhecem que é uma espécie diferente, mas não são capazes de achar o nome adequado, muito provavelmente por que a espécie não foi descrita. Em alguns casos, os biólogos conseguem identificar somente até a tribo ou o gênero de uma espécie. A nomenclatura dada a esses indivíduos contém o táxon mais restritivo identificado adicionado da *string sp.X* ou *indet*. Exemplos: *Cecidochares sp.e*, *Achyrocline indet.*;

- *Quando não se tem certeza da identificação:* em alguns casos, os indivíduos se assemelham muito com uma dada espécie, mas possuem alguma estrutura com formato que não é comum aos indivíduos utilizados como referência. A nomenclatura dada a esses indivíduos é a mesma da espécie semelhante, contendo adicionalmente a *string cf.* após o nome do gênero (representando o termo latim *conferatum*). Exemplo: *Dioxyna cf. thomae*;

- *Quando se acredita ser uma espécie diferente, porém muito parecida com uma já descrita:* embora muito similares a uma espécie já descrita, alguns indivíduos possuem alguma estrutura muito diferente, dando quase a certeza de se tratar de outra espécie. Em geral, não são encontradas na literatura informações a respeito das características mais ambíguas. A nomenclatura dada a esses indivíduos é a mesma da espécie similar, contendo a *string aff.* após o nome do gênero (representando o termo latim *affinis*). Exemplo: *Tomoplagia aff. fiebrigi*.

Em todos os casos de não-identificação, é necessário recorrer ao auxílio de um especialista em taxonomia para se ter certeza de uma classificação. Casos de não identificação podem ocorrer também em relação aos capítulos coletados.

A Figura 5.2 ilustra parte da ontologia desenvolvida para descrever esse cenário, criada na linguagem OWL DL, utilizando a ferramenta Protégé [33]. A figura mostra, por exemplo, que em uma dada coleta (na parte inferior à esquerda) um inseto de uma espécie não identificada (*conferatum*) representada por *"Dioxyna cf. thomae"* é predador de um capítulo da espécie identificada *"Bidens segetum"*. Esta interação foi observada na coleta *"Collection G0105"*, realizada no habitat *Cerradao*. As demais instâncias da ontologia foram omitidas para simplificar a figura.

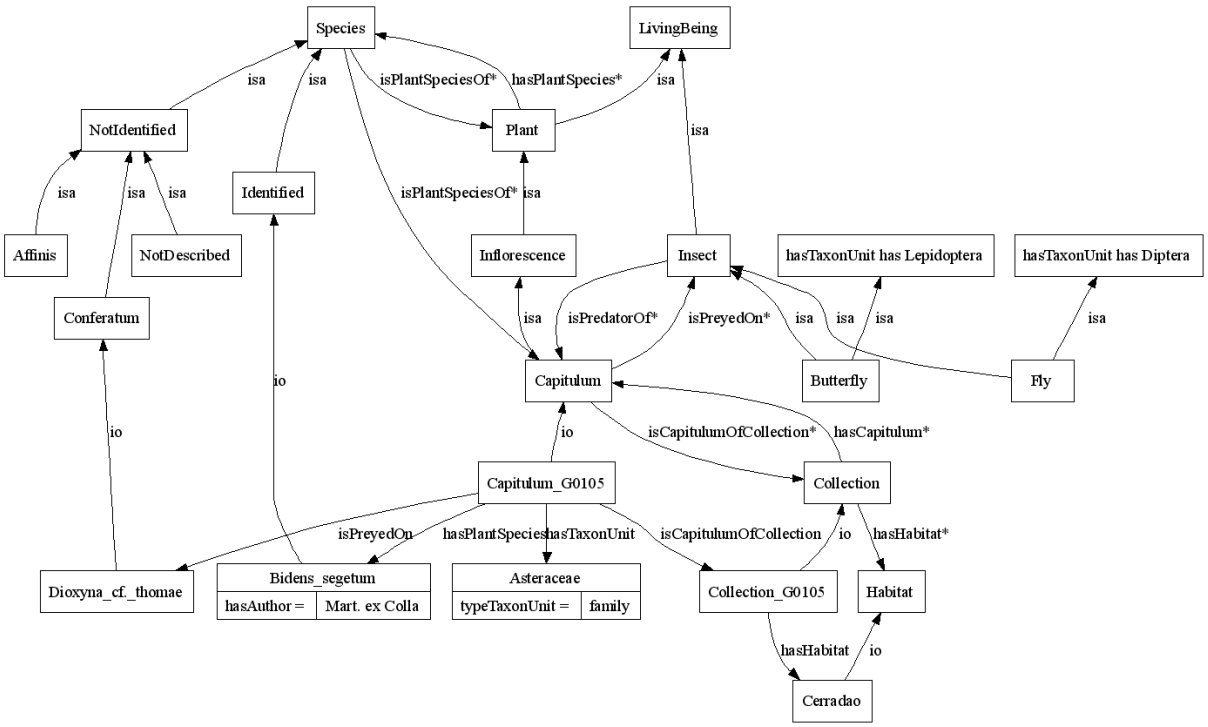

Figura 5.2: Parte da ontologia de coletas, construída com o auxílio dos biólogos parceiros do WeBios

Um Repositório Semântico foi criado para armazenar essa ontologia e seus metadados. A URL deste repositório é *http://143.0106.23.89:8080/OntRepUNICAMP*. A ontologia foi inserida no repositório usando o módulo de Gerência de Repositórios, pela invocação do método *InsertOntologyFile*. O identificador local da ontologia nesse repositório é *colUN*. Em invocações futuras ao Serviço de Ontologias, esta ontologia será, portanto, identificada pelo par: *<colUN, http:// 143.0106.23.89:8080/OntRepUNICAMP>*.

5.2 Uso dos Módulos do Serviço Aondê

Considere uma consulta típica no sistema WeBios em pesquisas sobre interações inseto-planta: *"Retorne as espécies de insetos que se alimentam de Asteraceae e que foram coletadas no ano 2000 no Cerrado brasileiro"*. Esta consulta requer vários tipos de desambiguação semântica – o que é um *inseto*, o que significa o relacionamento *"alimentar-se de"*, o que é um *Asteraceae*, quais espécies de insetos foram coletadas no ano 2000, o que é *Cerrado brasileiro* e qual sua extensão geográfica, dentre outros. Cada uma dessas considerações pode necessitar de invocações ao serviço Aondê, e o resultado dessas invocações é posteriormente utilizado na invocação dos demais serviços do sistema WeBios (vide arquitetura do sistema na Seção 2.1).

Os exemplos desse estudo de caso resolvem problemas relativamente simples, porém suficientes para ilustrar as possibilidades distintas de invocação ao serviço Aondê e sua aplicação em consultas reais. Este estudo de caso se restringe a mostrar a solução de algumas das questões semânticas que envolvem diretamente o serviço. Analogamente, outras variáveis necessárias à solução também podem ser desambiguadas, seja pelo uso do próprio Aondê, pela combinação de invocações a outros serviços e repositórios de dados, ou ainda por intervenção do usuário (por exemplo, fornecendo as coordenadas geográficas dos polígonos que representam o cerrado brasileiro). Dados referentes a coletas de campo, por exemplo, como data, pesquisador responsável e metodologia de coleta são tipicamente armazenados em Repositórios de Coletas (gerenciados pelo Serviço de Coletas do WeBios). Desta forma, o predicado *"ano = 2000"* é, em geral, resolvido por meio de invocações a esse serviço. Discussões sobre o processamento e a combinação de consultas aos vários serviços fogem ao escopo desta dissertação. Aqui é suficiente saber que uma consulta ao sistema WeBios é processada combinando invocações ao serviço Aondê para desambiguações semânticas com invocações aos demais módulos do sistema.

As seções subseqüentes ilustram exemplos de invocações aos módulos do Aondê: Consultas (Seção 5.2.1), Visões (Seção 5.2.2), Detecção de Diferenças (Seção 5.2.3), Busca (Seção 5.2.4), Integração (Seção 5.2.5) e Busca com *Ranking* (Seção 5.2.6). Esses exemplos mostram a utilização das operações propostas no serviço para a solução de consultas reais, típicas do cenário descrito. Alguns desses exemplos combinam dados fictícios simulando

outro grupo de pesquisa, gerados para ilustrar algumas particularidades do funcionamento desses módulos.

5.2.1 Consultas

A associação entre os insetos coletados e suas respectivas localidades provê informações importantes para o estudo de endemismos. Uma espécie é considerada endêmica quando sua ocorrência é registrada apenas em uma região muito restrita, não sendo naturalmente encontrada em outras localidades. A informação de endemismo é uma característica já conhecida para determinadas espécies e pode impactar na análise de dados de ocorrência. Considere que os biólogos desejam realizar a seguinte consulta: *"Retorne as espécies endêmicas de insetos coletados"*. A informação de endemismo é representada na ontologia *colUN* como uma propriedade (*datatype*) da classe *Insect*, com valor booleano (*true* = endêmico). Esta consulta pode ser resolvida por meio de uma invocação direta ao *Módulo de Consultas*, da seguinte forma:

Consulta(<colUN,http://143.0106.23.89:8080/OntRepUNICAMP>, SPARQL, strConsulta, false, XML)

Esta invocação mostra que a consulta contida em *strConsulta* está especificada na linguagem *SPARQL* e será realizada na ontologia *colUN* acessada em seu repositório *http://143.0106.23.89:8080/OntRepUNICAMP*. O parâmetro *false* indica que não será necessário o uso de mecanismos de inferência na ontologia antes da execução da consulta. O conteúdo da consulta (parâmetro *strConsulta*) é mostrado na Figura 5.3. A consulta consiste basicamente em encontrar, dentre os elementos da ontologia, os que são simultaneamente: instâncias de insetos, predadores de capítulos e com valor verdadeiro para a propriedade de endemismo.

```
PREFIX col:<http://www.owl-ontologies.com/Collection.owl#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
SELECT distinct ?insect
WHERE {
?insect rdf:type col:Insect .
?insect col:isEndemic ?endemism .
FILTER (?endemism = "true"^^xsd:boolean) .
?capitulum rdf:type col:Capitulum .
?capitulum col:isPreyedOn ?insect }
```

Figura 5.3: Consulta *"Retorne as espécies endêmicas de insetos coletados"* em SPARQL

O resultado da consulta é retornado em um arquivo XML (padrão para SPARQL [10]) – parâmetro *XML* da invocação. A consulta retorna 8 espécies de insetos (entre as 77 da

ontologia), e seu resultado é mostrado na Figura 5.4.

```
<?xml version="1.0"?>
<sparql
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema#"
   xmlns="http://www.w3.org/2005/sparql-results#" >
 <head> <variable name="insect"/>  </head>
 <results ordered="false" distinct="true">
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Dyseuaresta brasiliensis</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Euarestoides sp.05</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Dictyotrypeta sp.24</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Xanthaciura "pelotensis"
          aberrante</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Tetreuaresta sp.b</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Dyseuaresta sp.b</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Trupanea sp.11</uri>
   </binding> </result>
  <result> <binding name="insect">
     <uri>http://www.owl-ontologies.com/Collection.owl#Trupanea sp.10</uri>
   </binding> </result>
 </results>
</sparql>
```

Figura 5.4: Resultado XML da invocação do Módulo de Consultas

Outro tipo de estudo importante para os biólogos consiste em analisar o impacto de determinadas espécies de inseto na produção de sementes de uma dada planta. Como os insetos são predadores de sementes, é possível realizar análises temporais a respeito da diminuição da população de uma planta em uma dada região, relacionando-a com a intensidade dos ataques de insetos predadores. Uma consulta típica neste contexto é representada por: *"Retorne a espécie de inseto mais abundante em capítulos da espécie Trixis verbasciformis"*. Estima-se que quanto mais abundante a espécie de inseto em um capítulo (isto é, maior número de ocorrências do inseto em um mesmo capítulo) maior a intensidade do ataque. Essa informação é representada com uma propriedade dos capítulos coletados, que representa o número de indivíduos (insetos). Esta consulta pode ser resolvida de forma direta por meio de uma invocação ao *Módulo de Consultas*, similar

a realizada anteriormente, porém considerando a consulta SPARQL mostrada na Figura 5.5.

```
PREFIX col:<http://www.owl-ontologies.com/Collection.owl#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?insect ?numInd
WHERE {
?capitulum rdf:type col:Capitulum .
?capitulum col:hasPlantSpecies col:Trixis_verbasciformis .
?capitulum col:isPreyedOn ?insect .
?capitulum col:hasNumInd ?numInd }
ORDER BY DESC[?numInd]
```

Figura 5.5: Consulta *"Retorne a espécie de inseto mais abundante em capítulos da espécie Trixis verbasciformis"* em SPARQL

O resultado da consulta, ilustrado na Figura 5.6, contém 5 espécies de insetos ordenados decrescentemente pelo número de indivíduos (cláusula *"ORDER BY DESC[?numInd]"* na consulta SPARQL). Desta forma, a espécie mais abundante está na primeira posição desse resultado – a espécie *Unadilla erronella*, que apresentou 38 indivíduos em um mesmo capítulo.

Espécie de Inseto	NumInd
Unadilla erronella	38
Tomoplagia costalimai	04
Melanagromyza indet.	02
Adaina bipunctata	01
Blastobasidae sp.01	01

Figura 5.6: Resultado da consulta SPARQL especificada na Figura 5.5

5.2.2 Visões

A ontologia ilustrada na Figura 5.2 classifica as espécies de insetos em duas categorias principais: espécies identificadas (classe *Identified*) e espécies não identificadas (classe *NotIdentified*). Outra análise importante, relacionada à segunda consulta descrita na Seção 5.2.1, é analisar o impacto de espécies de inseto não identificadas (nas coletas) na produção de sementes de uma dada planta. Observe que a espécie retornada pela consulta mais abrangente, *Unadilla erronella*, é uma espécie identificada (isto pode ser verificado pela sua nomenclatura, como descrito na Seção 5.1). Considere, agora, que os biólogos desejam realizar a seguinte consulta: *"Retorne a espécie de inseto não-identificada mais abundante em capítulos da espécie Trixis verbasciformis"*.

Uma possibilidade para se resolver essa consulta é similar à descrita na Seção 5.2.1, com uma simples invocação ao *Módulo de Consultas*, porém inserindo mais restrições na cláusula *where* da consulta SPARQL mostrada na Figura 5.5. Outra possibilidade é restringir a consulta às partes da ontologia *colUN* que representam as espécies de interesse e, posteriormente, executar a consulta. A restrição da ontologia para a representação apenas de insetos não identificados consiste na extração de uma visão dessa ontologia, por meio de uma invocação do *Módulo de Visões*, da seguinte forma:

Visao(<colUN,http://143.0106.23.89:8080/OntRepUNICAMP>,
NotIdentified,{subclasse:1, isPlantSpeciesOf:2, instance, isPreyedOn:1,
hasPlantSpecies:1, hasNumInd:1 }, http://143.0106.23.89:8080/OntRepUNICAMP)

O parâmetro *NotIdentified* representa o conceito central da visão, correspondendo às espécies não identificadas. A visão é extraída da ontologia *colUN* acessada em seu repositório *http://143.0106.23.89:8080/OntRepUNICAMP*. O conjunto de diretivas é interpretado da seguinte forma:

- {*subclasse:1*}: inclui todas as subclasses imediatas de *NotIdentified*, ou seja, as subclasses de distância 1 (neste caso, as classes *Affinis*, *Conferatum* e *NotDescribed* – vide Figura 5.2);

- {*isPlantSpeciesOf:2*}: inclui todas as classes relacionadas a *NotIdentified*, de distâncias 1 e 2, por meio da propriedade *isPlantSpeciesOf* (neste caso, as classes *Capitulum* e *Plant*);

- {*instance*}: indica que as instâncias de todas as classes da visão devem ser incluídas;

- {*isPreyedOn:1*}: como esta propriedade não está diretamente relacionada a *NotIdentified*, esta diretiva indica que esta propriedade entre instâncias (caso ocorra) deve participar da visão (neste caso, entre as instâncias de *Capitulum* e *NotIdentified*);

- {*hasPlantSpecies:1*}: mesma situação anterior (neste caso, entre as instâncias de *Capitulum* e *Plant*);

- {*hasNumInd:1*}: mesma situação anterior (neste caso, preserva a propriedade – atributo – para as instâncias de *Capitulum*).

A ontologia gerada a partir da visão é parcialmente ilustrada na Figura 5.7. Note que a visão preserva os relacionamentos entre as instâncias da ontologia original especificados nas diretivas. Como o número de espécies de insetos não identificadas é consideravelmente

pequeno em relação ao total de espécies coletadas, o número de triplas da visão também é significativamente inferior em relação à ontologia original (a visão possui menos da metade das triplas da ontologia original).

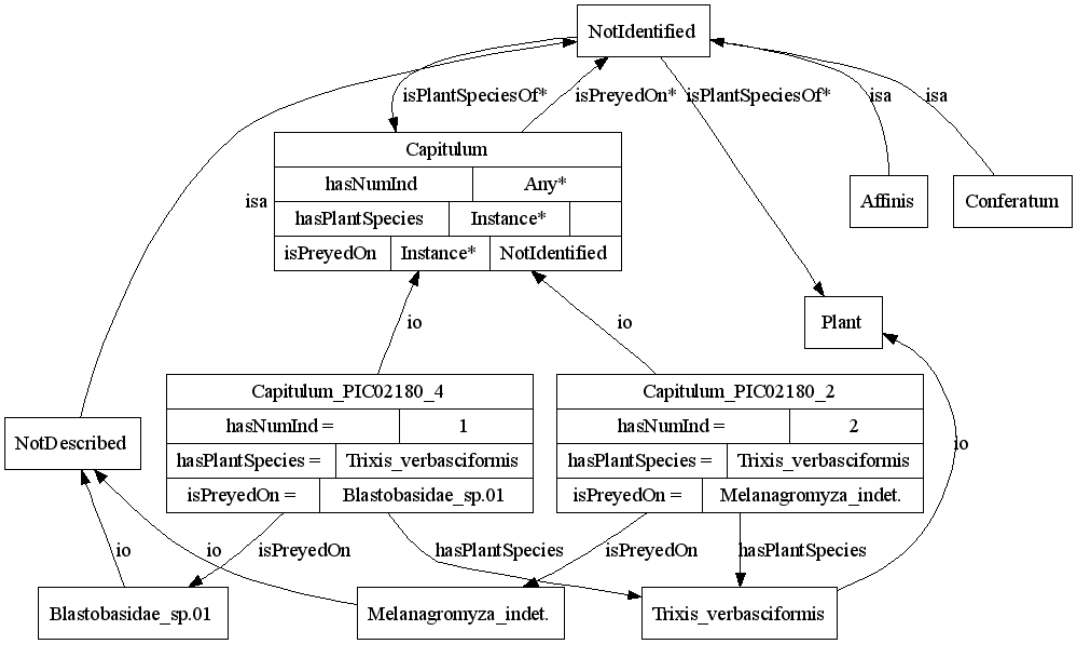

Figura 5.7: Visão com conceito central *"NotIdentified"*

A visão é armazenada no Repositório Semântico *http://143.0106.23.89:8080/ OntRepUNICAMP* (registrada como uma visão da ontologia *colUN*), e a invocação retorna o identificador criado para essa visão, *colUN-Visao1*. Esse identificador é a seguir utilizado para uma invocação ao *Módulo de Consultas*, executando uma consulta que retorne as espécies de inseto não identificadas em capítulos da espécie *Trixis verbasciformis*, de forma similar a realizada anteriormente. As duas espécies que resultam desta consulta são as ilustradas na Figura 5.7, sendo que *Melanagromyza indet.* é a mais abundante ($hasNumInd = 2$).

5.2.3 Detecção de Diferenças

Considere o seguinte cenário: um segundo grupo de pesquisadores (G2) deseja utilizar os dados da ontologia *colUN* em suas análises. Para isto, os biólogos de G2 criam seu próprio Repositório Semântico com a URL *http://158.2547.12.50:8080/OntRepBIO*. Posteriormente, acessam o repositório dos pesquisadores da UNICAMP por meio do serviço Aondê, recuperam a ontologia de coletas *colUN* (utilizando o *Módulo de Gerência de Repositórios*, com a operação *GetOntology*) e a armazenam em seu próprio repositório

(invocação do *Módulo de Gerência de Repositórios*, com a operação *InsertOntologyFile*).

Como dito na Seção 5.1, as espécies não identificadas são, em geral, enviadas para especialistas em taxonomia. A partir da análise dos taxonomistas, muitas dessas espécies passam a ser corretamente identificadas, enquanto outras tornam-se alvo de estudos mais aprofundados. Algumas espécies não identificadas pelos biólogos da UNICAMP foram identificadas pelos especialistas, sendo assim atualizadas na ontologia de coletas *colUN*. Com isso, *colUN* foi atualizada em seu repositório original (por meio de uma invocação ao *Módulo de Gerência de Repositórios*, operação *UpdateOntologyFile*), dando origem a uma nova versão da ontologia original com o identificador *colUN-v1*.

Os biólogos do grupo G2, sabendo da existência de alterações, gostariam de atualizar suas análises sem que fosse necessário refazer todo o trabalho novamente. Isto pode ser realizado por meio de uma invocação ao *Módulo de Detecção de Diferenças* que compara a ontologia do repositório próprio (de G2) e a ontologia atualizada, da seguinte forma:

$$Diferenca(colUN\text{-}v1,http://143.0106.23.89:8080/OntRepUNICAMP>,\\ colUN,http://158.2547.12.50:8080/OntRepBIO,\ 0.8, 0.2, 0.7 >)$$

A Figura 5.8 ilustra as modificações detectadas por essa invocação. A Figura 5.8 (a) corresponde à ontologia original *colUN*, mostrando as espécies da classe *Conferatum* que foram analisadas e reclassificadas pelos taxonomistas. A Figura 5.8 (b) apresenta as modificações que essa reclassificação causou à ontologia *colUN-v1*. O indivíduo *Dioxyna cf. thomae* foi identificado como sendo, de fato, da espécie *Dioxyna thomae*. Os demais indivíduos, *Cecidochares cf. connexa* e *Cochylis cf. sagittigera*, não foram identificados, apenas reclassificados como muito similares (porém de espécies diferentes) e, portanto, passaram a pertencer à classe *Affinis*.

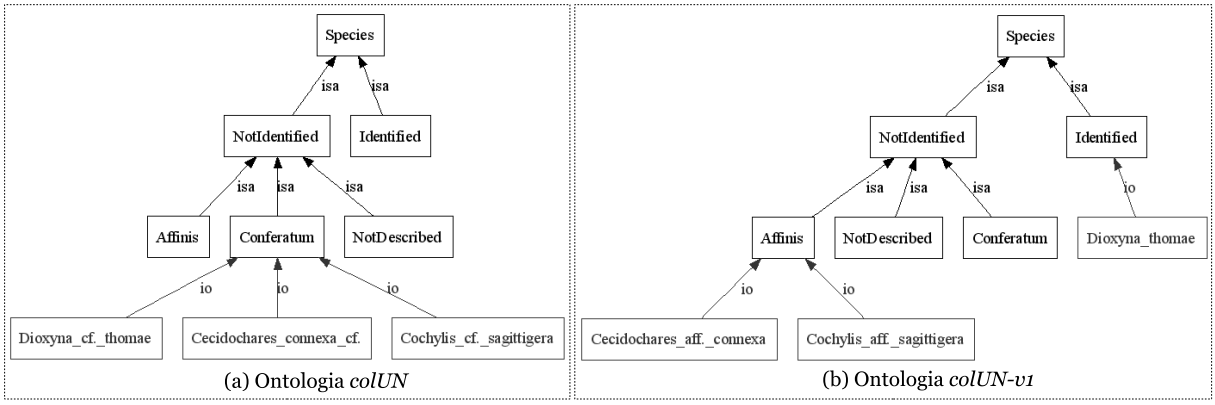

Figura 5.8: Instâncias da classe *Conferatum* em *colUN* reclassificadas em *colUN-v1*

As modificações detectadas são ilustradas na Figura 5.9 e mostram que as instâncias *Dioxyna cf. thomae*, *Cecidochares cf. connexa* e *Cochylis cf. sagittigera* foram excluídas

da classe *Conferatum* e que as instâncias *Dioxyna thomae*, *Cecidochares aff. connexa* e *Cochylis aff. sagittigera* foram inseridas nas classes *Identified* e *Affinis*, respectivamente. Como as diferenças estão no conteúdo das ontologias, e tratam-se de instâncias de classes distintas que não possuem propriedades que possam ser utilizadas em sua comparação, apesar do nome das instâncias continuar muito similar, o módulo não identifica essas diferenças como modificações, mas sim como uma seqüência de exclusões e inclusões.

```
<?xml version="1.0" encoding="UTF-8"?>
<Difference>
   <ListInsertion>
      <Insertion>
         <ElementInOntB>http://www.owl-ontologies.com/Collection.owl#Cecidochares_aff._connexa
         </ElementInOntologyB>
      </Insertion>
      <Insertion>
         <ElementInOntB>http://www.owl-ontologies.com/Collection.owl#Cochylis_aff._sagittigera
        </ElementInOntologyB>
      </Insertion>
      <Insertion>
         <ElementInOntB>http://www.owl-ontologies.com/Collection.owl#Dioxyna_thomae
         </ElementInOntologyB>
      </Insertion>
   </ListInsertion>
   <ListDelection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Collection.owl#Cecidochares_connexa_cf.
         </ElementInOntologyA>
      </Delection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Collection.owl#Cochylis_cf._sagittigera
         </ElementInOntologyA>
      </Delection>
      <Delection>
         <ElementInOntA>http://www.owl-ontologies.com/Collection.owl#Dioxyna_cf._thomae
         </ElementInOntologyA>
      </Delection>
    </ListDelection>
</Difference>
```

Figura 5.9: Resultado XML da detecção de diferenças entre as ontologias *colUN* e *colUN-v1*

5.2.4 Busca

A análise das espécies não identificadas consiste, muitas vezes, em comparações com outras espécies descritas na literatura. De certa forma, cria-se um subconjunto de possíveis espécies e, de acordo com determinadas características, é determinada a classificação taxonômica mais restrita de uma espécie, mesmo que isso não represente a identificação de fato da espécie.

Considere que o grupo de pesquisadores G2 realiza análises deste tipo nos dados de borboletas representados na ontologia *colUN*. As borboletas nessa ontologia (classe *Butterfly*) são descritas como insetos cuja unidade taxonômica ordem é *Lepidoptera* (propriedade *hasTaxonUnit has Lepidoptera*, cujo tipo é *Order*), como ilustrado na Figura 5.2. Entretanto, a ontologia *colUN* não fornece informações adicionais a respeito da ordem *Lepidoptera*.

É preciso prover aos biólogos informações sobre quais classificações taxonômicas e espécies são abrangidas pela ordem *Lepidoptera*. Isto pode ser feito realizando uma invocação ao *Módulo de Busca* utilizando Repositórios Externos de Ontologias. Neste caso, como o termo buscado representa um táxon, o repositório adequado para recuperar esse tipo de ontologia é o Spire[1] [55]. A invocação é realizada da seguinte forma:

$$Busca(\{Lepidoptera\}, \{descendants\}, \{Spire\}, \textit{http://158.2547.12.50:8080/OntRepBIO})$$

O parâmetro $\{Spire\}$ indica que a busca será realizada no Repositório Externo Spire, e $\{Lepidoptera\}$ é o táxon buscado. A diretiva $\{descendants\}$ indica que a ontologia retornada pela busca deve conter níveis taxonômicos inferiores (descendentes) ao táxon buscado (tribos, gêneros e espécies, por exemplo). A ontologia retornada como resultado desta busca é parcialmente ilustrada na Figura 5.10.

A figura mostra que a ontologia retornada consiste basicamente em uma hierarquia de classes, que possui diferentes níveis de descendentes da ordem *Lepidoptera*(hierarquia não-homogênea). Os desníveis representam, principalmente, subclassificações taxonômicas que não são classificações taxonômicas comuns a todas as espécies, como subfamílias, subclasses e tribos. Algumas espécies de insetos são representadas nos nós terminais dessa ontologia. Essa ontologia é armazenada no repositório indicado na invocação da busca (*http://158.2547.12.50:8080/OntRepBIO*). Ela é registrada como uma ontologia proveniente de uma busca e seu identificador retornado pelo módulo é *lepdDesc*.

5.2.5 Integração

Para agilizar as análises das espécies de inseto coletadas, além de ter informações sobre as classificações taxonômicas abrangidas pela ordem *Lepidoptera*, os biólogos desejam correlacionar essas classificações com os nomes preliminares dados a cada espécie. A partir dessa correlação, além da categorização em identificados ou não, será possível realizar consultas que recuperem grupos de borboletas (*Lepidoptera*) que pertençam a um mesmo gênero ou tribo, por exemplo.

[1] http://spire.umbc.edu/ont/

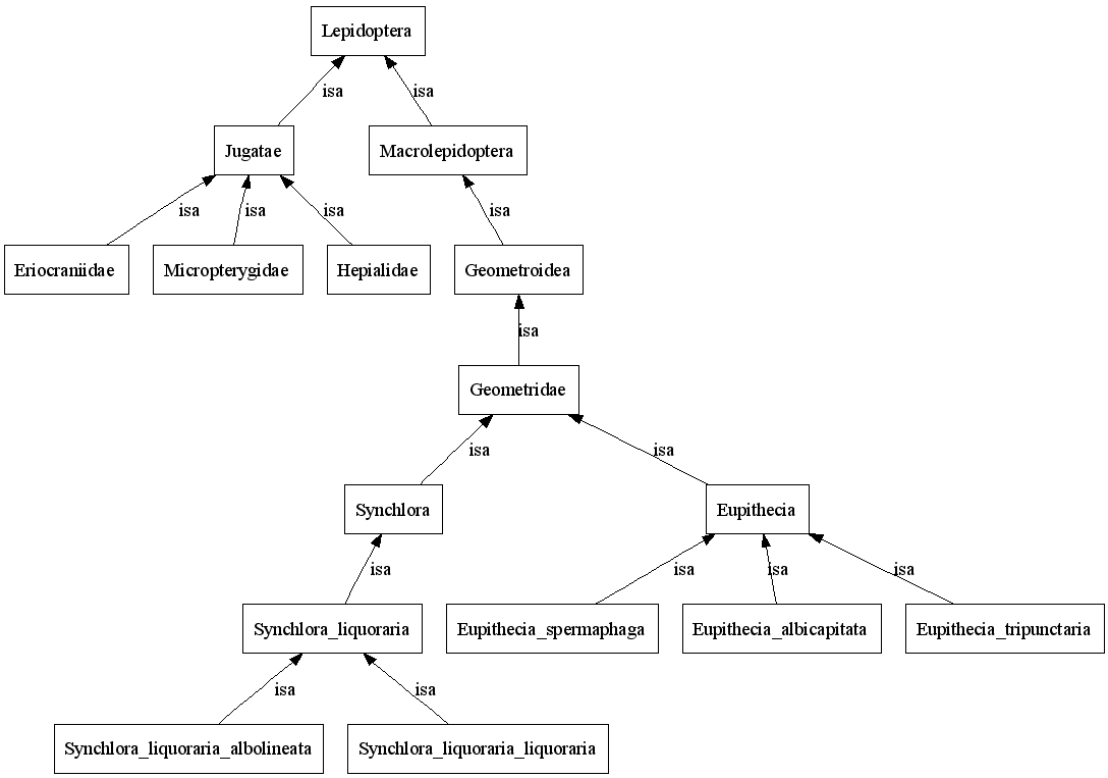

Figura 5.10: Ontologia de descendentes da ordem *Lepidoptera*, recuperada do Repositório Externo Spire

Essa correlação pode ser realizada por meio de uma invocação ao *Módulo de Integração*, que alinha as ontologias de coleta (*colUN*) e de descendentes de *Lepidoptera* (*lepdDesc*). A invocação é realizada da seguinte forma:

$$Integracao(<colUN,http://143.0106.23.89:8080/OntRepUNICAMP>,$$
$$<lepdDesc,http://158.2547.12.50:8080/OntRepBIO>,\ 0.8,\ 0.2,\ 0.7,$$
$$http://143.0106.23.89:8080/OntRepUNICAMP)$$

Os dois primeiros parâmetros da invocação designam as ontologias que serão alinhadas, *colUN* e *lepdDesc*. O parâmetro 0.7 representa a confiabilidade mínima que um mapeamento deve ter para ser materializado no alinhamento e os parâmetros 0.8 e 0.2 representam, respectivamente, os valores de α e β utilizados na combinação das similaridades de elemento e de estrutura, conforme descrito na Seção 3.3.5.

Para exemplificar o cálculo da confiabilidade, considere os candidatos: *Eupithecia sp.01* (da ontologia *colUN*) e *Eupithecia* (da ontologia *lepdDesc*) ilustrados na Figura 5.11. A similaridade de elemento entre eles é calculada de duas formas: similaridade de *strings*, cujo valor retornado é 0.875; e similaridade por sinônimo, cujo valor retornado é 0 pois

ambos dicionários consultados, ITIS[2] [47] e WordNet[3] [43], não possuem sinônimos para os termos. Como a similaridade de elemento é dada pelo valor máximo entre esses dois valores, no caso 0.875 e 0, o valor considerado é 0.875. A similaridade de estrutura é dada pela comparação entre as propriedades, os axiomas e as hierarquias desses elementos. Como se tratam de elementos de diferentes tipos (classe e instância), a similaridade de estrutura é 0. A confiabilidade do mapeamento será portanto: $0.8 * 0.875 + 0.2 * 0 = 0.7$. Como esse mapeamento possui a confiabilidade mínima determinada na invocação, ele será inserido na ontologia alinhada. Em outras palavras, a ontologia alinhada possuirá uma *tag* $<sameAs>$ relacionando a classe *Eupithecia* (originalmente da ontologia *colUN*) e a instância *Eupithecia sp.01* (originalmente da ontologia *lepdDesc*).

O último parâmetro representa o Repositório Semântico onde a ontologia resultado desse alinhamento é armazenada (e registrada como proveniente de um alinhamento) – neste caso, o mesmo repositório da ontologia *colUN*. A ontologia resultado desse alinhamento é parcialmente ilustrada na Figura 5.11 – a árvore à esquerda é proveniente da *lepdDesc*; à direita de *colUN*. O identificador gerado automaticamente pelo módulo, *colLep*, é retornado pela invocação.

O alinhamento realizado entre as ontologias originais, embora aparentemente simples (uma junção "natural" das ontologias), apresenta algumas particularidades. Os mapeamentos, que dão origem ao alinhamento, são encontrados entre instâncias da ontologia *colUN* e classes da ontologia *lepdDesc* – ou seja, elementos de diferentes tipos. Esses mapeamentos correlacionam os nomes das espécies coletadas a suas classificações taxonômicas, e são representados na Figura 5.11 pelos vértices rotulados por *sameAs*. A descoberta dos candidatos a mapeamentos, neste caso, precisa considerar as regras de nomenclatura científica em biologia – por exemplo, não comparando elementos de níveis taxonômicos diferentes (uma família com um gênero, por exemplo).

A partir da ontologia alinhada, é possível realizar sobre os dados de coleta análises do tipo: *"Retorne as espécie de borboletas predadoras de capítulos da espécie Mikania cordifolia"*. A especificação dessa consulta na linguagem SPARQL é ilustrada na Figura 5.12. A consulta consiste basicamente em encontrar, dentre os elementos da ontologia, os que são simultaneamente: instâncias de insetos, predadores de capítulo da espécie *Mikania cordifolia* e que são borboletas – ou seja, foram alinhados (*tag* $<owl:sameAs>$) com subclasses de *Lepidoptera*.

Para obter todas as espécies de interesse, é necessário executar essa consulta a partir do modelo inferido da ontologia alinhada (ou seja, na invocação do *Módulo de Consultas*, o parâmetro que determina o uso de inferência deve ser *"true"*). O uso de inferência está relacionado à descoberta de quais classes dessa ontologia são subclasses de *Lepidoptera*.

[2] http://www.itis.gov

[3] "http://jws-champo.ac-toulouse.fr: 8080/wordnet-xml"

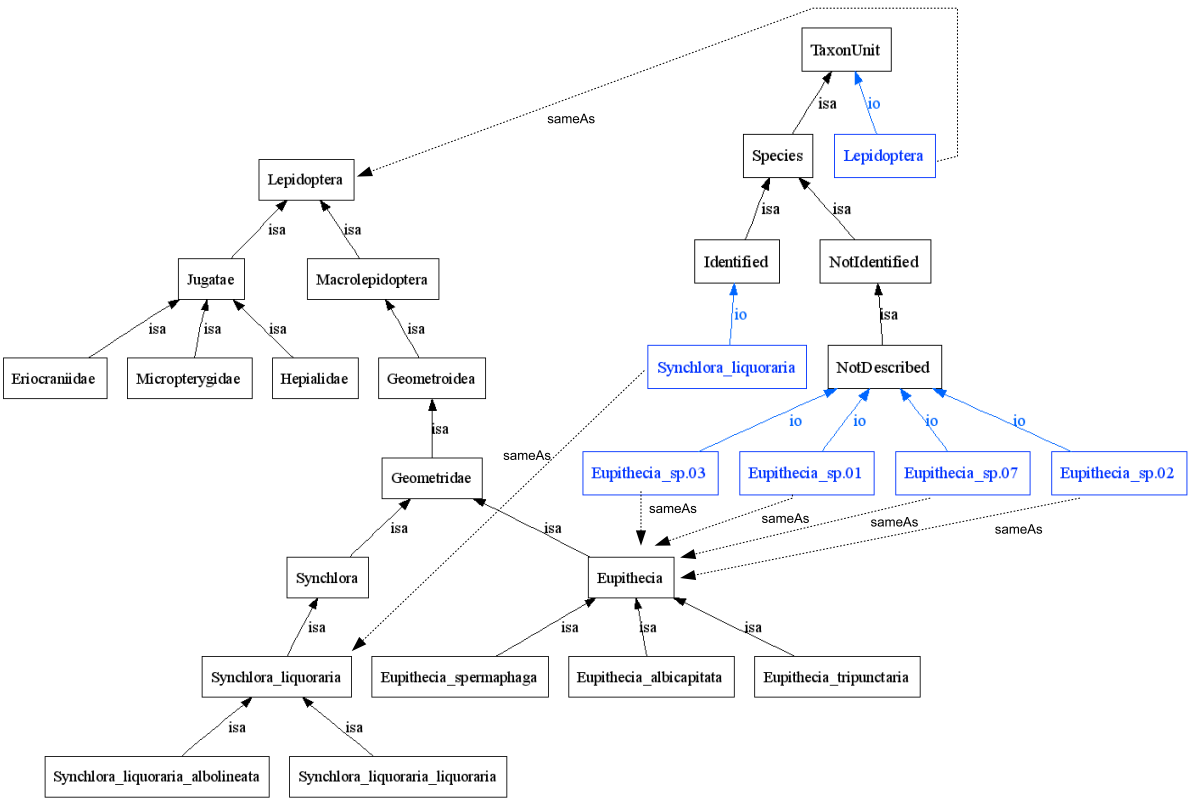

Figura 5.11: Parte da ontologia *colLep*, produzida pelo Módulo de Integração

Conforme descrito na seção 3.3.2, sem a utilização de inferência apenas as subclasses diretas são retornadas. Considerando a transitividade da propriedade *subClassOf*, as subclasses indiretas também podem ser retornadas, recuperando os alinhamentos gerados em diferentes níveis da classificação hierárquica do táxon *Lepidoptera*. O resultado dessa consulta retorna 12 espécies de inseto, dentre elas as instâncias *Eupithecia sp.03*, *Eupithecia sp.07* e *Synchlora liquoraria* ilustradas na figura 5.11.

Outra análise importante para os biólogos é identificar o número de espécies distintas de plantas que são atacadas por uma dada espécie de insetos. Um inseto pode atacar poucas espécies de plantas em uma localidade, mas quando se considera um conjunto de localidades, a gama de plantas hospedeiras pode ser significativamente maior. Ao contrário desse comportamento generalista, alguns insetos podem apresentar um comportamento *"especialista"* – atacam apenas uma espécie de planta, ou um pequeno conjunto de espécies. Por exemplo, um inseto que ataca 10 espécies de plantas de um mesmo gênero é considerado mais especialista que um inseto que ataca apenas duas espécies de plantas de tribos diferentes (já que "tribo" está em um nível taxonômico superior ao "gênero"). Uma análise sobre o comportamento especialista dos insetos coletados pode ser expressa pela

```
PREFIX col:<http://www.owl-ontologies.com/Collection.owl#>
PREFIX lep:<http://spire.umbc.edu/ontologies/EthanAnimals.owl#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
SELECT ?insect
WHERE {
?capitulum rdf:type col:Capitulum .
?capitulum col:hasPlantSpecies col:Mikania_cordifolia .
?insect owl:sameAs ?butterfly .
?butterfly rdfs:subClassOf lep:Lepidoptera .
?capitulum col:isPreyedOn ?insect }
```

Figura 5.12: Consulta *"Retorne as espécie de borboletas predadoras de capítulos da espécie Mikania cordifolia"* em SPARQL

seguinte consulta: *"Retorne os insetos coletados que predam capítulos pertencentes a uma única espécie de planta"*.

Para obter resultados mais significativos nessa consulta, ela deve ser submetida a dados relativos a coletas distintas. Considere, portanto, que os biólogos do grupo G2 responsáveis pelo Repositório Semântico *http://158.2547.12.50:8080/OntRepBIO* criaram sua própria ontologia de coletas, cujo identificador é *colBIO*, ilustrada na Figura 5.13. A ontologia foi armazenada nesse repositório por meio de uma invocação ao *Módulo de Gerência de Repositórios*, pela operação *InsertOntologyFile*.

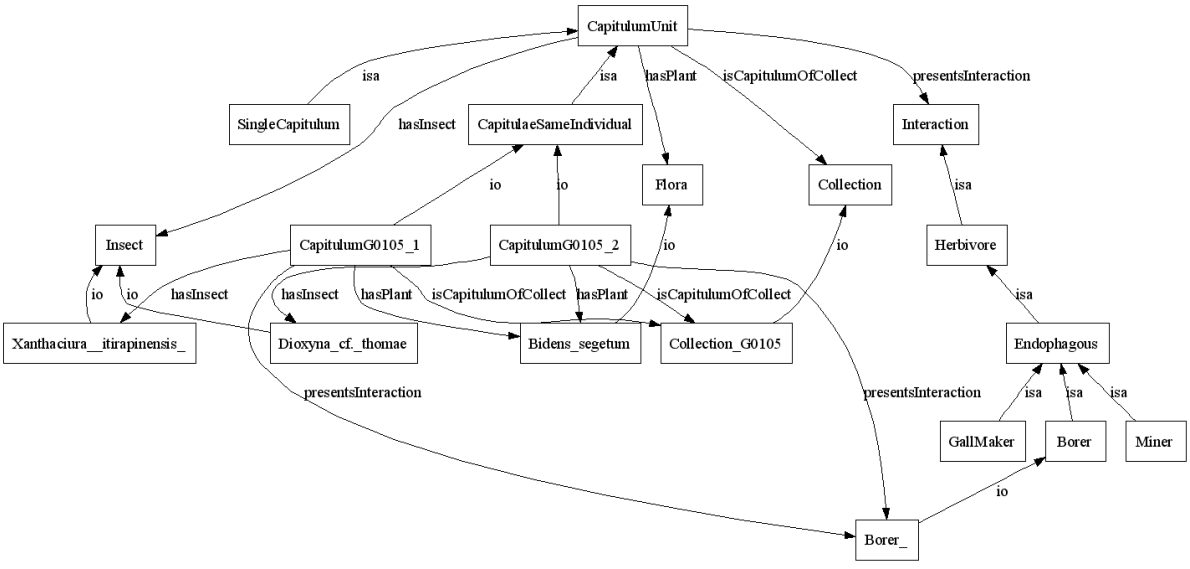

Figura 5.13: Ontologia de coletas *colBIO* do grupo de pesquisa G2

Embora essa ontologia possua algumas particularidades, como a unidade de capítulo utilizada nas coletas (*CapitulumUnit*) e categorizações para os tipos de predação dos

insetos (brocador (*Borer*), minerador (*Miner*) e galhador (*GallMaker*)), as instâncias dessa ontologia são também representadas por insetos (classe *Insect*) e plantas (classe *Flora*). A correlação entre os dados das duas ontologias de coletas, *colUN* e *colBIO*, pode ser realizada por meio de uma invocação ao *Módulo de Integração*, que alinha as espécies de plantas e insetos de ambas ontologias. Essa invocação é realizada da seguinte forma:

Integracao(<colUN,http://143.0106.23.89:8080/OntRepUNICAMP>,
<colBIO,http://158.2547.12.50:8080/OntRepBIO>, 0.7, 0.3, 0.65,
http://158.2547.12.50:8080/OntRepBIO)

Os dois primeiros parâmetros dessa invocação representam os identificadores das ontologias que serão alinhadas, *colUN* e *colBIO*. O parâmetro 0.65 representa a confiabilidade mínima que um mapeamento deve ter para ser materializado no alinhamento e o último parâmetro representa o Repositório Semântico onde a ontologia resultado desse alinhamento é armazenada (e registrada como proveniente de um alinhamento), que neste caso, o mesmo repositório da ontologia *colBIO*. O identificador *colUNcolBIO* é gerado automaticamente pelo módulo e retornado pela invocação.

A particularidade do alinhamento produzido neste caso está na descoberta de mapeamentos utilizando os dicionários de sinônimos. Usando o dicionário *WordNet*, é possível descobrir que *Flora* e *Plant*, são sinônimos, e portanto, sua similaridade de elemento tem valor 1. Com isso, a confiabilidade do mapeamento, utilizando apenas esse resultado, possui valor 0.7 – superior à confiabilidade mínima especificada no parâmetro da invocação 0.65. Esse mapeamento entre classes é representado na ontologia alinhada por meio da *tag <owl:equivalenceClass>*.

Além disso, as ontologias de coleta utilizam nomenclaturas distintas para algumas espécies de plantas – os pesquisadores responsáveis adotam classificações propostas por autores diferentes. Neste caso, por meio de consultas à base de sinônimos ITIS[4] [47] é possível identificar mapeamentos entre as instâncias das ontologias, representadas na ontologia alinhada por meio da *tag <owl:sameAs>*.

Os mapeamentos encontrados na ontologia alinhada são usados na consulta *"Retorne os insetos coletados que predam capítulos pertencentes a uma única espécie de planta"*, para detectar comportamento especialista. A consulta deve ser executada no modelo inferido da ontologia alinhada, permitindo o acesso a todas as instâncias de plantas por meio de uma das classes, *Flora* ou *Plant* (por inferência, instâncias individuais de classes equivalentes são instâncias comuns a ambas as classes). A execução dessa consulta retorna resultados que dificilmente seriam observados em um processamento manual: um inseto coletado da espécie *Platphalonidia fusifera* é predador das espécies de plantas *Chromolaena odorata*

[4]http://www.itis.gov

(nas coletas de *colUN*) e *Eupatorium odoratum* (nas coletas de *colBIO*). Entretanto, esse inseto possui de fato um comportamento especialista, pois tratam-se de duas classificações taxonômicas distintas (identificadas como sinônimos na base ITIS) dadas a uma mesma espécie de planta. Este caso é ilustrado na Figura 5.14, que representa parte da ontologia produzida por esse alinhamento.

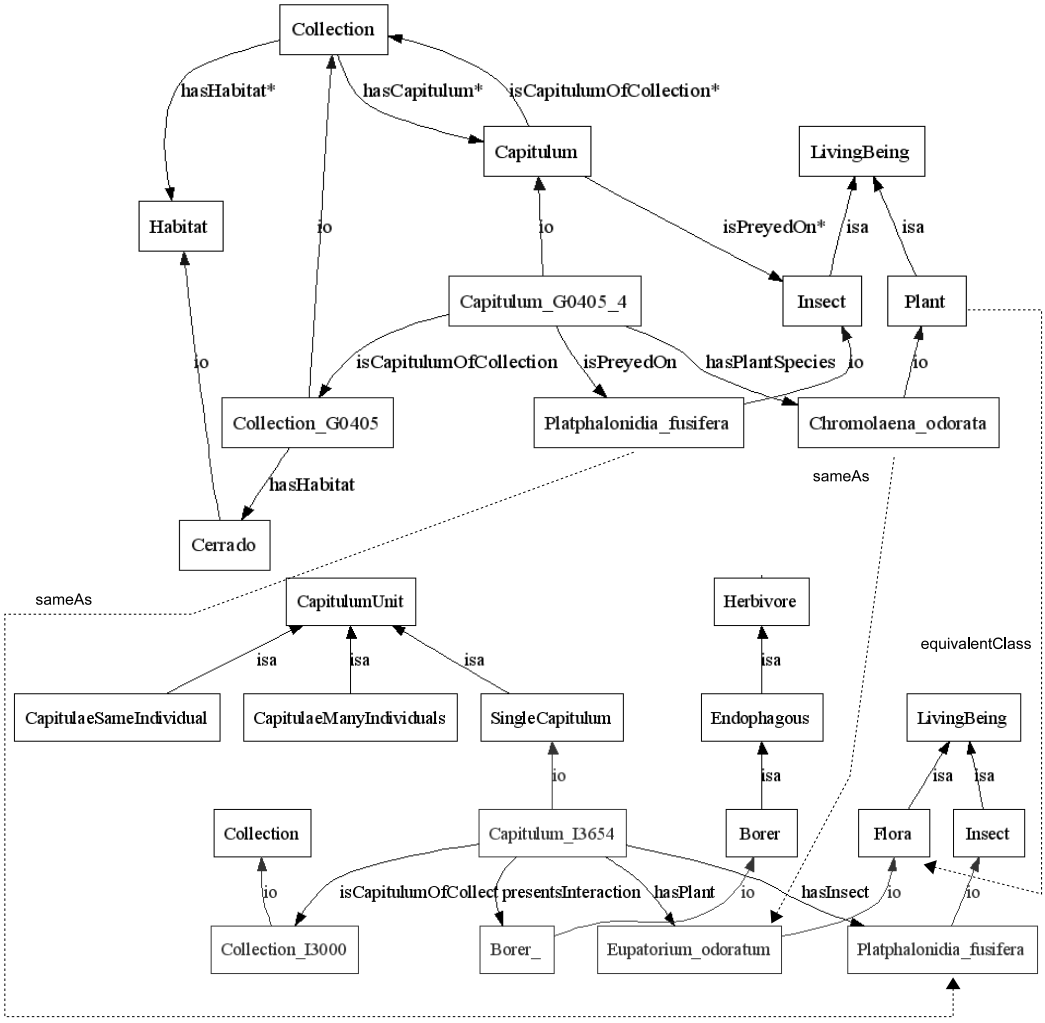

Figura 5.14: Parte da ontologia *colBIO*, produzida pelo *Módulo de Integração*

5.2.6 Busca com Ranking

Considere agora que os biólogos da UNICAMP desejam obter informações adicionais sobre o conceito *planta* – não detalhado na ontologia *colUN* – como informações sobre sua estrutura básica ou suas interações com outras entidades bióticas. Uma forma de pro-

ver esse tipo de informação é realizando uma invocação ao *Módulo de Busca* utilizando Repositórios Externos de Ontologias. Neste caso, como é buscado um termo (*plant*) não específico do domínio, o repositório Swoogle[5] [23] é adequado para recuperar esse tipo de ontologia. A invocação é realizada da seguinte forma:

$$BuscaRank(\{plant\}, \{0.4, 0.2, 0.4, 0\}, \{\text{Swoogle}\},\\ http://143.0106.23.89:8080/OntRepUNICAMP)$$

O campo $\{plant\}$ indica o termo buscado no repositório $\{Swoogle\}$. O conjunto de pesos determinado nessa invocação é atribuído da seguinte forma: *Casamento Exato e Parcial:* 0.4; *Densidade:* 0.2; *Centralidade:* 0.4 e *Similaridade Semântica:* 0. A invocação busca por um único termo; por esse motivo, a métrica de *Similaridade Semântica* não é aplicável e recebe peso 0 (zero). A distribuição dos demais pesos deve, portanto, somar o valor 1. Neste exemplo, os pesos dão mais importância às métricas de *Centralidade* e *Casamento Exato e Parcial* e menos importância à métrica de *Densidade*. Os pesos $0.4, 0.4$ e 0.2 indicam essa decisão. A invocação retorna uma lista com 10 identificadores de ontologias, que serão todas armazenadas em *http://143.0106.23.89:8080/OntRepUNICAMP*, registradas como provenientes de uma busca. Os identificadores são ordenados pelos seus valores de *ranking*.

A Figura 5.15 ilustra a ontologia com mais alto valor de *ranking*, disponível na URL *http://wow.sfsu.edu/ontology/rich/EcologicalConcepts.owl*. Analisando essa ontologia, é possível perceber que o termo buscado, $\{plant\}$, aparece recorrentemente como parte dos nomes das classes – *Plant, PlantDescription, PlantPartDescriptiveTrait, PlantParts, PlantSpecies, AboveGroundPlantParts* e *PlantTrait* destacados na figura – justificando o alto valor dessa ontologia pela métrica de *Casamento Exato e Parcial*. A quantidade de classes nesse caso é, de fato, superior à das outras nove ontologias retornadas. Além disso, esta ontologia contém muitos outros conceitos relacionados a essas classes (critério de centralidade), como as classes *Organism, BioticItem,Leaves, Seeds* e *Stems*.

Essa ontologia é registrada no repositório *http://143.0106.23.89:8080/ OntRepUNICAMP* como proveniente de uma operação de busca no repositório *Swoogle* com o termo $\{plant\}$, e seu identificador é *ecoConcept*. Essa ontologia pode ser posteriormente utilizada em invocações a outros módulos, como o *Módulo de Consultas*, de forma similar aos exemplos anteriores.

5.3 Conclusões

Este capítulo apresentou exemplos reais de uso de cada um dos módulos do serviço Aondê na solução de consultas relevantes a dados de biodiversidade. Estes exemplos demons-

[5] http://swoogle.umbc.edu

tram a importância do levantamento de requisitos realizado com os biólogos parceiros do WeBios, que teve impacto direto na determinação dos módulos e em considerações sobre suas especificações e implementações.

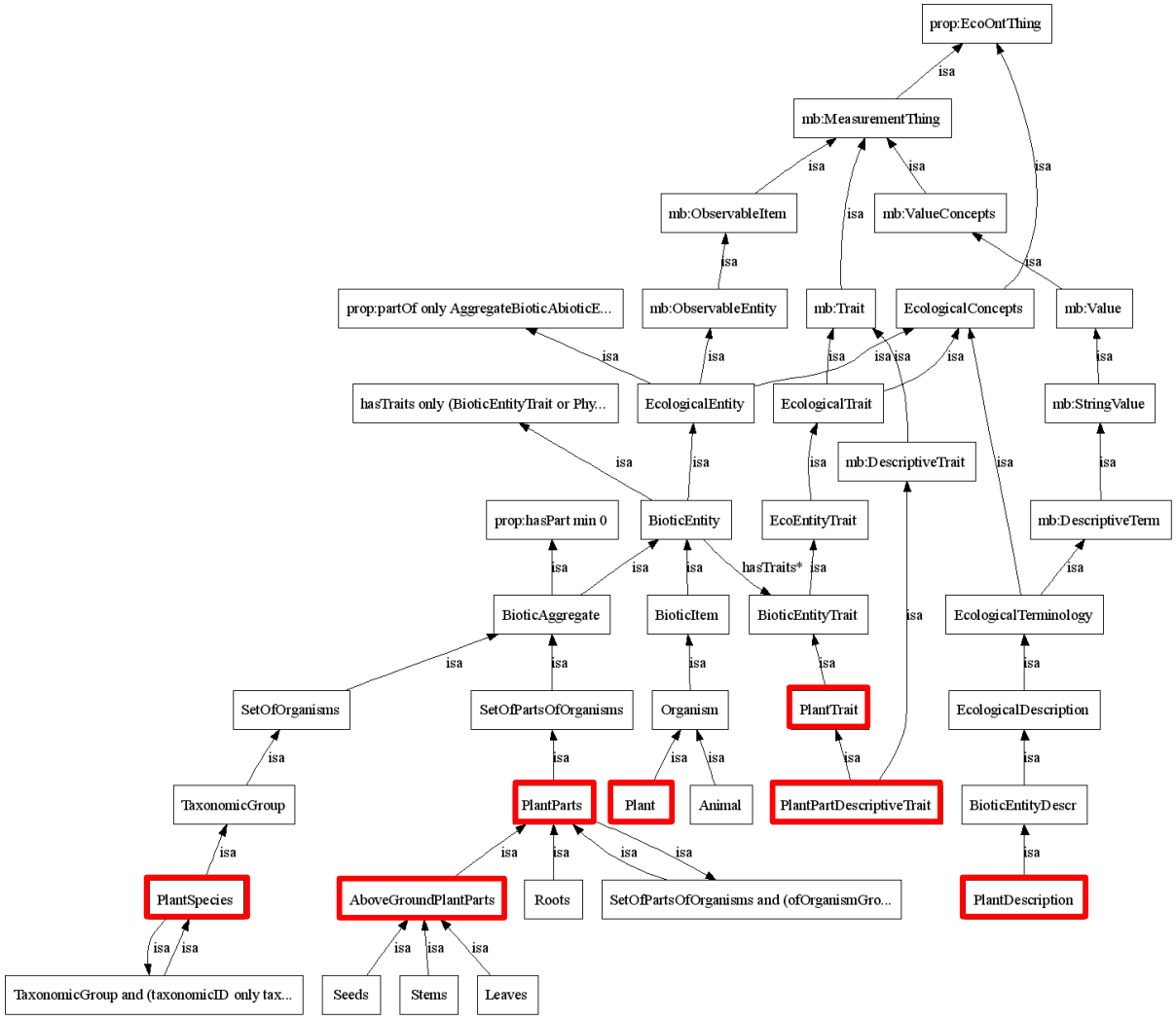

Figura 5.15: Ontologia *ecoConcept*, retornada na busca pelo termo *plant* no repositório Swoogle

Capítulo 6

Conclusões e Extensões

6.1 Conclusões

O trabalho apresentado nesta dissertação consiste na especificação e implementação de um Serviço Web para Ontologias – Aondê. O serviço proposto suporta acesso a ontologias distribuídas e comporta uma gama de operações que combinam algumas das abordagens existentes. Este serviço foi especificado e desenvolvido para apoiar consultas requisitadas ao sistema WeBios, um Sistema de Informação de Biodiversidade que está sendo desenvolvido no IC-UNICAMP. A especificação do serviço foi centrada no levantamento de aspectos ontológicos de sistemas de biodiversidade e nos requisitos dos pesquisadores em biologia, parceiros do projeto WeBios, que necessitam de acesso transparente a fontes de dados heterogêneas e distribuídas. A solução proposta pelo serviço Aondê é capaz de prover interoperabilidade e integração de dados por meio de uma combinação de serviços Web e gerenciamento de ontologias.

Os Repositórios Semânticos, propostos como parte da arquitetura do serviço, são compostos por ontologias e metadados e podem ser acessados por outras aplicações (pois também são encapsulados por serviços Web). O serviço pode acessar também ontologias disponíveis em Repositórios Externos gerenciados por terceiros. Além disso, o serviço suporta a atualização de ontologias, sendo portanto capaz de atender as necessidades de aplicações que refletem as evoluções do mundo real. Os módulos disponíveis no serviço são: consultas, busca (com e sem *ranking*), detecção de diferenças, extração de visões e integração, além de funcionalidades de gerência para Repositórios Semânticos.

O serviço Aondê foi testado com dados reais, fornecidos pelos biólogos parceiros do projeto WeBios, em consultas relevantes no cenário de biodiversidade. A implementação do serviço desenvolvida nessa dissertação será incorporada ao sistema WeBios e utilizada pelos demais módulos do sistema. Essa implementação possui algumas características específicas para dados de biodiversidade, que auxiliam o tratamento de termos que repre-

sentam classificações taxonômicas – consideração de sufixo na comparação entre *strings* e incorporação de dicionários específicos para táxons sinônimos e vernáculos (nomes populares). Apesar disso, o serviço pode ser utilizado em outros domínios que apresentem as mesmas necessidades de manipulação de dados heterogêneos e distribuídos, cenário típico em aplicações *e-Science*.

O desenvolvimento deste trabalho apresentou várias dificuldades. Uma das mais marcantes está relacionada à falta de consolidação dos padrões utilizados para manipulação de ontologias. Por tratar-se de uma área relativamente recente em computação, as propostas de linguagens para ontologias ainda estão convergindo para uma padronização. As linguagens adotadas neste trabalho foram escolhidas de acordo com as recomendações (ou candidatas a recomendação) da W3C[1]. Ainda não existe uma recomendação para linguagens de consulta em ontologias – a linguagem RDQL estava sendo analisada como candidata quando, em novembro de 2005, a linguagem SPARQL foi proposta a esse consórcio.

Outra dificuldade encontrada está relacionada com as operações propostas para o serviço. Existem vários trabalhos na literatura para cada uma dessas operações, e muitos desses trabalhos apresentam abordagens divergentes em suas soluções. Este é o caso, por exemplo, da operação de extração de visões – existem abordagens que consideram que uma visão de ontologia, assim como em banco de dados, deve ser especificada por uma consulta enquanto outras abordagens consideram que visões são baseadas em conceitos centrais. Há também divergências na operação de *ranking*, que pode ser baseada em referências (similar à técnica de *page-ranking*) ou em suas próprias estruturas internas. Não é possível determinar se existe uma abordagem melhor que todas as demais – cada uma possui suas vantagens e desvantagens. A escolha das abordagens utilizadas no serviço Aondê foi baseada na adequabilidade aos casos analisados.

Houve, além disso, dificuldades relacionadas a implementação do serviço, como o aprendizado do *framework* Jena e dificuldades de uso do padrão OMV, principalmente na tentativa de uso da ferramenta Oyster [54] para criação de estruturas de metadados para ontologias – essa ferramenta gera arquivos RDF's inconsistentes (isto pode ser observado nos próprios *snapshots* reportados pelos desenvolvedores) e por este motivo, não pôde ser adotada.

As principais contribuições desta dissertação são:

- Levantamento e comparação de técnicas e ferramentas para manipulação de ontologias propostas na literatura;
- Levantamento das necessidades de um Serviço de Ontologias para permitir a integração de dados heterogêneos e distribuídos;

[1] http://www.w3.org/

- Especificação de um Serviço de Ontologias, com um conjunto de operações e uma estrutura de Repositórios Semânticos que disponibiliza ontologias por meio de protocolos padrões de Serviços Web;
- Implementação de um Serviço Web de Ontologias Aondê, com considerações específicas para dados de biodiversidade e validado com estudos de caso reais em biodiversidade.

6.2 Extensões

Esta dissertação possui várias extensões possíveis, relacionadas tanto a aspectos de pesquisa quanto de implementação, algumas das quais listadas a seguir.

6.2.1 Refinamento dos Módulos do Serviço de Ontologias

– Módulo de Gerência de Repositórios

A estrutura proposta para os Repositórios Semânticos permite recuperar várias informações sobre as ontologias armazenadas. Essas informações poderiam ser melhor exploradas a partir de operações mais específicas, como *"retorne todas as visões obtidas da ontologia ontA"* ou *"retorne todas as ontologias alinhadas que tiveram a participação da ontologia ontA"*. Além disso, seria interessante criar mecanismos internos que gerenciassem a procedência de uma ontologia. Para isto, seria necessário definir operações específicas que permitam tais manipulações.

Outra melhoria prevista para este módulo relaciona-se à gerência de versões de ontologias. O serviço permite a atualização de ontologias nos repositórios; entretanto, essa atualização consiste no armazenamento completo da nova versão da ontologia. Algumas abordagens propostas na literatura especificam métodos sofisticados para a manipulação de versões [42, 53]. Algumas são associadas a ferramentas de desenvolvimento e edição de ontologias em processos colaborativos ou ainda a ferramentas de detecção de diferenças, focadas em interfaces intuitivas para os usuários finais [49].

– Módulo de Consultas

A implementação do módulo de consultas utiliza a máquina de inferência (*reasoner*) OWL do *framework* Jena. Esse raciocinador não é completo para a linguagem OWL DL e possui várias limitações tanto em desempenho quanto na compreensão de axiomas[2]. Existem outros raciocinadores externos – como Pellet [65], Racer [35] e FaCT [70] – que podem ser

[2] http://jena.sourceforge.net/inference/index.html#owl

conectados ao *framework* Jena e, desta forma, acoplados a este módulo. Esses raciocinadores são mais completos e possuem regras mais sofisticadas de inferência.

– Módulo de Detecção de Diferenças
O módulo de detecção de diferenças usa um algoritmo simples para detectar modificações em ontologias. Existem várias abordagens que usam heurísticas na identificação dessas alterações [49, 51]. Essas heurísticas poderiam ser acopladas ao módulo para auxiliar, principalmente, casos de modificações que muitas vezes são identificados como seqüências de exclusões e inserções. Além disso, grande parte das ferramentas de detecção de diferenças apresentam seus resultados visualmente, em interfaces para usuários finais. A apresentação dos resultados deste módulo é estruturada em um arquivo XML, que não segue nenhuma padronização. Seria importante investigar outras possibilidades de representação de diferenças de forma padronizada.

– Módulo de Busca
O módulo de busca se concentra em analisar classes e instâncias da ontologia que sejam rotuladas com os termos buscados. Entretanto, muitas informações a respeito das ontologias estão descritas em sua estrutura de metadados, armazenadas junto às ontologias nos Repositórios Semânticos. Seria interessante prover um novo tipo de busca, que considerasse as estruturas de metadados na busca por ontologias de interesse. Desta forma, seria possível recuperar ontologias de acordo com elementos de seus metadados, por exemplo, ontologias com uma determinada palavra-chave, autor ou data de criação, ou ainda com instâncias (ou seja, número de instâncias diferente de zero).

– Módulo de *Ranking*
O módulo de *ranking* usa apenas quatro métricas na análise de estruturas internas das ontologias (descritas na Seção 2.3.2.1). Outras métricas poderiam ser implementadas e adaptadas ao módulo, para permitir a ordenação de ontologias segundo outros critérios que melhor representassem as necessidades de um usuário. Uma possível métrica é a proposta em [61], que analisa a similaridade entre conceitos de acordo com a proximidade com um conceito comum hierarquicamente superior (análise do "pai" mais próximo).

– Módulo de Visões
A implementação deste módulo suporta apenas visões com um conceito central. Visões centradas em mais de um conceito precisam ser construídas individualmente e, posteriormente, alinhadas pelo módulo de integração. Essa abordagem possui a desvantagem de duplicar os elementos que aparecem em ambas visões. Mesmo que esses elementos sejam mapeados, a duplicação faz com que a ontologia alinhada possua um número de

triplas superior ao que seria necessário. A extensão desse módulo para comportar vários conceitos centrais poderia torná-lo mais robusto.

– Módulo de Integração
A abordagem utilizada pelo módulo de integração representa os alinhamentos entre classes e instâncias por meio da *tag* $<sameAs>$. Isto faz com que as classes sejam interpretadas como instâncias da classe base $<owl:class>$ e, com isso, a linguagem da ontologia alinhada é promovida a OWL Full. Nesta dissertação, não foram analisados os impactos que isso pode causar na utilização da ontologia alinhada, seja em outros módulos ou em aplicações finais. Seria importante analisar outras possibilidades de representação desses mapeamentos.

Além disso, o módulo de integração suporta apenas o alinhamento de ontologias. Esse módulo poderia ser estendido para outras técnicas, como a união e o mapeamento. Na união, seria necessário especificar qual ontologia daria origem à nomenclatura final e como proceder em caso de mapeamentos entre elementos de diferentes tipos (classes e instâncias). Para o mapeamento, seria necessário especificar um formato padronizado de saída para a representação dos mapeamentos.

Outra consideração importante neste módulo é aprimorar as técnicas adotadas no cálculo da similaridade entre os elementos mapeados. As abordagens utilizadas poderiam ser estendidas e melhoradas visando obter um maior número de mapeamentos. Na implementação, a descoberta desses mapeamentos é totalmente automática, ou seja, sem a intervenção do usuário. É possível que uma abordagem híbrida (semi-automática, de acordo com as necessidades do processo) traga ganhos significativos no aumento de mapeamento corretos na integração das ontologias.

6.2.2 Proposta de Outros Módulos

Vários operadores poderiam ser incorporados como novos módulos do Serviço de Ontologias. Uma possibilidade seria um módulo de tradução que compatibilizasse ontologias representadas em diferentes linguagens (como OWL, RDF(S), DAML+OIL, OBO). Isto traria flexibilidade para que os demais módulos do serviço pudessem ser utilizados com ontologias em diferentes linguagens. Um complicador para este novo módulo é lidar com os diferentes níveis de expressividade de cada linguagem.

Outro operador interessante para o serviço seria um módulo de anotação, que auxiliasse a construção de anotações baseadas em conceitos ontológicos. Este módulo seria integrado a outros tipos de serviços que gerenciam tipos de dados específicos (imagens, ou registros de coletas por exemplo), com metadados associados que comportem anotações baseadas em conceitos das ontologias gerenciadas por este serviço.

6.2.3 Mecanismos para Operações Complexas

É possível realizar a combinação dos módulos propostos no Serviço de Ontologias para a execução de operações complexas. Neste cenário, o resultado de um módulo poderia ser utilizado como entrada para outros módulos. O processamento em sequência desses módulos certamente traria ganho em tempo de execução, pois diminuiria o tempo gasto nas trocas de mensagens SOAP entre o serviço e as aplicações cliente. Seria interessante estender o Serviço de Ontologias para permitir esse tipo de execução. Esta execução poderia ser estática, ou seja, pela construção de módulos "avançados" no serviço, que realizassem a execução de determinadas operações seqüencialmente. Outra opção seria a execução dinâmica, que permitisse determinar diferentes seqüências de execução combinando as entradas e saídas de cada módulo (por meio de *workflows* [39], por exemplo).

6.2.4 Análise de Desempenho e Escalabilidade

Não foram realizadas análises a respeito do desempenho do Serviço de Ontologias. Seria interessante analisar o desempenho do serviço no acesso a vários repositórios distribuídos em uma mesma requisição, por exemplo. Além disso, vários módulos do serviço empregam algoritmos que utilizam a estrutura de grafos das ontologias. A análise de tempo de execução desses algoritmos poderia trazer resultados interessantes sobre a viabilidade do uso de Serviço de Ontologias junto a outras aplicações, ou no pré-processamento das consultas do sistema WeBios, por exemplo.

6.2.5 Aplicação do Serviço de Ontologias em outros Cenários

Apesar do estudo de caso e a motivação da dissertação terem sido voltados a problemas em biodiversidade, o serviço não possui nenhuma particularidade que restrinja sua utilização em outros domínios. Apenas algumas decisões de funcionalidades e a escolha do método de integração utilizado, o alinhamento, foram direcionadas considerando o domínio abrangido pelas ontologias. Seria interessante, como extensão deste projeto, realizar testes com ontologias de outros domínios e avaliar os resultados obtidos.

Referências Bibliográficas

[1] S. Abels, L. Haak, and A. Hahn. Identification of common methods used for ontology integration tasks. In *IHIS '05: First International Workshop on Interoperability of Heterogeneous Information Systems*, pages 75–78. ACM Press, 2005.

[2] H. Alani, C. Brewster, and N. Shadbolt. Ranking Ontologies with AKTiveRank. In *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[3] H. Alani, S. Harris, and B. O'Neill. OntologyWinnowing: A Case Study on the AKT Reference Ontology. In *CIMCA/IAWTIC*, pages 710–715. IEEE Computer Society, 2005.

[4] A. M. Almeida, C. R. Fonseca, P. I. Prado, M. Almeida-Neto, S. Diniz, U. Kubota, M. R. Braun, R. L. G. Raimundo, L. A. Anjos, T. G. Mendonça, S. M. Futada, and T. M. Lewinsohn. Diversidade e ocorrência de Asteraceae em cerrados de São Paulo. *Biota Neotropica*, 5:27 – 43, 2005.

[5] A. M. Almeida, C. R. Fonseca, P. I. Prado, M. Almeida-Neto, S. Diniz, U. Kubota, M. R. Braun, R. L. G. Raimundo, L.A. Anjos, T. G. Mendonça, S. M. Futada, and Thomas M. T. M. Lewinsohn. Assemblages of endophagous insects on Asteraceae in São Paulo Cerrados. *Neotropical Entomology*, 35:458 – 468, August 2006.

[6] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer Verlag, Berlin Heidelberg New York, November 2004.

[7] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*, pages 76–92, 2003.

[8] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver,

A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29, May 2000.

[9] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS–Transparent Access to Multiple Bioinformatics Information Sources. In *Int Conf Intelligent Systems for Molecular Biology*, volume 6, pages 25–34, Montreal, Canada, June 1998.

[10] D. Beckett and J. Broekstra. SPARQL Query Results XML Format. Technical report, W3C, April 2006.

[11] P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krötzsch, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris. Specification of a common framework for characterizing alignment. Knowledge Web Deliverable 2.2.1v2, University of Karlsruhe, DEC 2004.

[12] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1. W3C Note Note-SOAP-20000508, World Wide Web Consortium, May 2000.

[13] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, RDF Core Working Group, World Wide Web Consortium – W3C, 2004.

[14] J. Bruijn, F. Martin-Recuerda, D. Manov, and M. Ehrig. State-of-the-art survey on Ontology Merging and Aligning. Technical report, SEKT project D4.2.1, 2004.

[15] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *WWW Alt. '04: Proc. of the 13th international World Wide Web*, pages 74–83. ACM Press, 2004.

[16] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. W3c note, World Wide Web Consortium, March 2001.

[17] L. Clement, A. Hately, C. Riegen, and T. Rogers. UDDI Spec Technical Committee Draft 3.0.2. OASIS Committee Draft, 2004.

[18] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 73–78, Acapulco, Mexico, August 2003.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge MA, 2nd edition, 2001.

[20] Z. Cui and P. O'Brien. Domain Ontology Management Environment. In *HICSS '00: Proc. of the 33rd Hawaii International Conference on System Sciences-Volume 8*, Washington, DC, USA, 2000. IEEE Computer Society.

[21] J. Daltio and C. B. Medeiros. Um servidor de ontologias para apoio a sistemas de biodiversidade (an ontology server to support biodiversity information systems). In *V Workshop Thesis and Dissertations on Databases, Proc. XXI Brazilian Symposium on Databases (SBBD 2006)*, pages 71–76, Florianópolis, Brazil, October 2006.

[22] J. Daltio and C. B. Medeiros. Um serviço de ontologias para sistemas de biodiversidade (an ontology service for biodiversity information systems). In *XXXIV SEMISH: Seminário Integrado de Software e Hardware*, pages 2143–2157, Rio de Janeiro, Brazil, July 2007.

[23] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a Search and Metadata Engine for the Semantic Web. In *CIKM '04: Proc. of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[24] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to Map between Ontologies on the Semantic Web. In *WWW '02: Proc. of the 11th international conference on World Wide Web*, pages 662–673. ACM Press, 2002.

[25] M. Duke and M. Patel. An Ontology Server for Agentcities.NET. Agentcities Task Force Technical Note, September 2003.

[26] T. Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, Hardcover, August 2005.

[27] C. H. Felicíssimo. Semantic Web Interoperability: One strategy for the Taxonomic Ontology Alignment (in Portuguese). Master's thesis, Pontifical Catholic University of Rio de Janeiro (PUC–FRJ, August 2004.

[28] D. Fensel, F. van Harmelen, I. Horrocks, D. L. Mcguinness, and P. F. Patel-Schneider. OIL: an ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems and Their Applications*, 16(2):38–45, 2001.

[29] M. Fernandez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *Proc. of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.

[30] C. R. Fonseca, P.I. Prado, M. Almeida-Neto, U. Kubota, and T. M. Lewinsohn. Flower-heads, herbivores, and their parasitoids: food web structure along a fertility gradient. *Ecological Entomology*, 30:36–46, February 2005.

[31] R. B. Freitas and R. S. Torres. Ontosaia: An ontology-based tool for image retrieval and semi-automatic annotation (in portuguese). In *I Workshop in Digital Libraries, Proc. XX Brazilian Symposium on Databases - SBBD 2005*, pages 60–79, October 2005.

[32] Global Biodiversity Information Facility (GBIF). GBIF website. http://www.gbif.org (accessed February 26, 2007).

[33] J. H. Gennari, M. A. Musen, R. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The Evolution of Protege: An Environment for Knowledge-Based Systems Development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003.

[34] T. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[35] V. Haarslev and R. Möller. Racer: A Core Inference Engine for the Semantic Web. In York Sure and Óscar Corcho, editors, *Proc. of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON 2003) located at the 2nd International Semantic Web Conference (ISWC 2003)*, volume 87 of *CEUR Workshop Proceedings*, pages 27–36. CEUR-WS.org, October 2003.

[36] J. Hartmann, Y. Sure, P. Haase, R. Palma, and M. C. Suárez-Figueroa. OMV – Ontology Metadata Vocabulary. In *ISWC 2005 - In Ontology Patterns for the Semantic Web*, Galway, Ireland, November 2005.

[37] J. S. Hong, H.Y. Chen, and J. Hsiang. A digital museum of taiwanese butterflies. In *ACM Digital Library*, pages 260–261, 2000.

[38] E. Jiménez, R. Berlanga, I. Sanz, M. J. Aramburu, and R. Danger. OntoPathView: A Simple View Definition Language for the Collaborative Development of Ontologies. *In B. López et al. (Eds.): Artificial Intelligence Research and Development*, pages 429–436, 2005.

[39] G. Z. Pastorello Jr. Publication and Integration of Scientific Workflows on the Web. Master's thesis, Instituto de Computação - Unicamp, April 2005.

[40] L. C. Gomes Jr. An architecture to query biodiversity data on the Web (in portuguese). Master's thesis, State University of Campinas - UNICAMP, May 2007.

[41] Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: the State of the Art. *Knowledge Engineering Review*, 18(1):1–31, 2003.

[42] M. C. A. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology Versioning and Change Detection on the Web. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 02). Ontologies and the Semantic Web*, pages 197–212, London, UK, 2002. Springer-Verlag.

[43] H. Kong, M. Hwang, and P. Kim. A New Methodology for Merging the Heterogeneous Domain Ontologies Based on the WordNet. In *NWESP '05: Proc. of the International Conference on Next Generation Web Services Practices*, page 235, Washington, DC, USA, 2005. IEEE Computer Society.

[44] J. Lee. An Application Programming Interface for Ontology. IBM T. J. Watson Research Center. Document from SNOBASE v.1.0 release documentation, November 2003.

[45] Y. Li, S. G. Thompson, Z. Tan, N. Giles, and H. Gharib. Beyond Ontology Construction; Ontology Services as Online Knowledge Sharing Communities. In *International Semantic Web Conference - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 2003.

[46] F. Manola and E. Miller. Resource Description Framework (RDF) Model and Syntax Specification, February 2004. http://www.w3.org/TR/rdf-primer/.

[47] R. McDiarmid. The Integrated Taxonomic Information System. In *Proc. of the Taxonomic Authority Files Workshop*, June 1998.

[48] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera Ontology Environment. In *Proc. of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 1123–1124, 2000.

[49] N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen. Tracking changes during ontology evolution. In Sheila A. Mcilraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Third International Semantic Web Conference*, pages 259–273, Hiroshima, Japan, November 2004. Springer Berlin.

[50] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Seventeenth International Joint Conference on Artificial IntelligenceAAAI/IAAI*, pages 450–455, 2000.

[51] N. F. Noy and M. A. Musen. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems*, 19(4):6–13, August 2004.

[52] N. F. Noy and M. A. Musen. Specifying Ontology Views by Traversal. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 713–725, 2004.

[53] D. Ognyanov and A. Kiryakov. Tracking Changes in RDF(S) Repositories. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 02). Ontologies and the Semantic Web*, pages 373–378, London, UK, 2002. Springer-Verlag.

[54] R. Palma, P. Haase, and A. Gómez-Pérez. Oyster: sharing and re-using ontologies in a peer-to-peer community. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 1009–1010. ACM, 2006.

[55] C. S. Parr, A. Parafiynyk, J. Sachs, L. Ding, S. Dornbush, T. W. Finin, D. Wang, and A. Hollander. Integrating Ecoinformatics Resources on the Semantic Web. In *Proc. in 15th International Conference on World Wide Web*, pages 1073–1074. ACM, 2006.

[56] C. Patel, K. Supekar, Y. Lee, and E. K. Park. OntoKhoj: a Semantic Web Portal for Ontology Searching, Ranking and Classification. In *WIDM '03: Proc. of the 5th ACM international workshop on Web information and data management*, pages 58–61, New York, NY, USA, 2003. ACM Press.

[57] A. G. Perez, J. Angele, M. F. Lopez, V. Christophides, A. Stutt, and Y. Sure. A survey on ontology tools. Deliverable 1.3, EU IST Project IST-2000-29243 OntoWeb, 2002.

[58] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, World Wide Web Consortium - W3C, 2006.

[59] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[60] J. A. Ramos. Mezcla automática de ontologías y catálogos electrónicos. Final Year Project. Facultad de Informática de la Universidad Politécnica de Madrid. Spain, 2001.

[61] P. Resnik. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[62] A. Seaborne. RDQL: A Query Language for RDF. Technical report, World Wide Web Consortium - W3C, 2003.

[63] P. Shvaiko and J. Euzenat. A Survey of Schema-based Matching Approaches. Technical Report DIT-04-087, University of Trento, 2004.

[64] SINBIOTA. São Paulo Biodiversity System site. http://sinbiota.cria.org.br/ (accessed May 10, 2007).

[65] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[66] J. A. Sánchez, C. A. Flores, and J. L. Schnase. Mutant: Agents as guides for multiple taxonomies in the floristic digital library. In *ACM Digital Library*, pages 244–245, 1999.

[67] H. Suguri, E. Kodama, M. Miyazaki, H. Nunokawa, and S. Noguchi. Implementation of FIPA ontology service. In *Proc. of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, pages 61–68, May 2001.

[68] R. S. Torres. *Ambiente de Gerenciamento de Imagens e Dados Espaciais para Desenvolvimento de Aplicações em Biodiversidade*. PhD thesis, IC-UNICAMP, 2004.

[69] R. S. Torres, C. B. Medeiros, M. A. Gonçalves, and E. A. Fox. A Digital Library Framework for Biodiversity Information Systems. *International Journal on Digital Libraries*, 6(1):3 – 17, February 2006.

[70] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[71] M. Tury and M. Bieliková. An Approach to Detection Ontology Changes. In *ICWE '06: Workshop proceedings of the sixth international conference on Web engineering*, page 14, New York, NY, USA, 2006. ACM Press.

[72] R. Volz, D. Oberle, and R. Studer. Implementing Views for Light-Weight Web Ontologies. In *Proc. of Int. Database Engineering and Application Symposium - IDEAS*, pages 160–169, Hong Kong, China, July 2003. IEEE Computer Society.

[73] WeBios. WeBios – Web Service Multimodal Tools for Strategic Biodiversity Research, Assessment and Monitoring. Home page http://www.lis.ic.unicamp.br/projects/webios, 2007.