Universidade Federal da Bahia

Universidade Estadual de Feira de Santana

Dissertação de Mestrado

SWoDS: Semantic Web (of Data) Service

Leandro José Silva Andrade

Mestrado Multi-institucional em
Ciência da Computação – MMCC

Salvador - BA
2015

Leandro José Silva Andrade

SWoDS: Semantic Web (of Data) Service

Dissertação apresentada ao Mestrado Multi-institucional em Ciência da Computação da Universidade Federal da Bahia e Universidade Estadual de Feira de Santana como requisito parcial à obtenção de título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Cássio Vinicius Serafim Prazeres

Salvador
2015

Andrade, Leandro J. S.
SWoDS: Semantic Web (of Data) Service
126 páginas
Dissertação (Mestrado) - Instituto de Matemática da Universidade Federal da Bahia. Departamento de Ciências da Computação.

1. Cássio Vinicius Serafim Prazeres (Orientador)
2. Paulo Caetano da Silva
3. Leandro Krug Wives

I. Universidade Federal da Bahia. Instituto de Matemática. Departamento de Ciências da Computação.

Banca Examinadora:

Prof. Dr.
Cássio Vinicius Serafim Prazeres
MCC-UFBA

Prof. Dr.
Paulo Caetano da Silva
PPGCOMP-UNIFACS

Prof. Dr.
Leandro Krug Wives
PPGC-UFRGS

Em memória a minha avó Hermínia, que com toda sua simplicidade e amor, deixou muita saudade.

Caçador de Mim

"Por tanto amor
Por tanta emoção
A vida me fez assim
Doce ou atroz
Manso ou feroz
Eu, caçador de mim

Preso a canções
Entregue a paixões
Que nunca tiveram fim
Vou me encontrar
Longe do meu lugar
Eu, caçador de mim

Nada a temer senão o correr da luta
Nada a fazer senão esquecer o medo
Abrir o peito a força, numa procura
Fugir às armadilhas da mata escura

Longe se vai
Sonhando demais
Mas onde se chega assim
Vou descobrir
O que me faz sentir
Eu, caçador de mim"

Milton Nascimento

Agradecimentos

Primeiramente, agradeço a Deus por nunca ter me abandonado e me deixado desistir. Por ter me dado forças e saúde para superar todos os desafios, que a vida me apresentou.

Agradeço a toda a minha família, que é um alicerce indispensável para toda minha trajetória de vida. Em especial agradeço aos pais José Luiz e Cristina, que são presenças fortes no meu dia-a-dia, que sempre me deram o exemplo de que com trabalho e dedicação somos capazes de alcançar nossos sonhos. A minha querida avó Maria Joana, que com suas orações, amor e atenção me faz sentir mais forte. A minha irmã Daiane, que além de ser uma grande amiga, sempre me incentivou e nunca mediu esforços para me ajudar. A minha terceira mãe Rita, que no seu jeito reservado me dedica muito carinho e atenção. Aos meus primos-irmãos Alex, Gerson, Pedro Henrique e Dom. As minhas afilhadas (Luísa, Lara e Maria Eduarda) que são flores na minha vida. A meu cunhado, aos meus tios, tias, padrinhos e madrinhas.

A Fabiane Barreto, que nunca faltou-me com incentivos e paciência em todos os momentos de percalços nessa trajetória. Que com muito amor e carinho soube me ouvir, comemorar, partilhar e me mostrar que sou capaz.

Ao meu orientador professor Cássio Prazeres, sempre me motivou e foi apoio fundamental para alcançar o objetivo final. Foram muitos ensinamentos, sempre acompanhados de serenidade e incentivos. Sou muito grato por todas as horas que ele abdicou (muito além da obrigação) em prol do meu crescimento profissional. Obrigado por ser um exemplo nessa carreira acadêmica.

A meus alunos, que foram fontes inspiradoras e decisivas para eu encarar com total firmeza minha trajetória acadêmica. Foi através deles, que tive a convicção que esse é o futuro profissional que desejo me dedicar com bastante entrega.

A meus amigos, que nos quais sempre encontrei apoio, seja através de conselhos e desabafos, seja através de momentos de descontração. Aos meus amigos de infância, aos eternos amigos do Colégio São José, aos amigos da Igreja da Penha, aos amigos da

UFBA e a todas novas amizades.

Por fim, agradeço a todas as pessoas que de algum modo cruzaram minha vida por algum período e deixaram bons aprendizados, boas recordações e boas energias. Pois, como disse Gonzaguinha: “Toda pessoa sempre é as marcas das lições diárias de outras tantas pessoas”

Resumo

Criada com a proposta inicial de conectar basicamente documentos HTML, a Web hoje expandiu suas capacidades, tornando-se um ambiente bastante heterogêneo de aplicações, recursos, dados e usuários que interagem entre si. A proposta da Web Semântica, associada aos Serviços Web, busca estabelecer padrões que viabilizem a comunicação entre aplicações heterogêneas na Web. A Web de Dados, outra linha de evolução da Web, fornece orientações (*Linked Data*) sobre como usar as tecnologias da Web Semântica para publicar e definir ligações semânticas entre dados de diferentes fontes. Contudo, existe uma lacuna na integração entre aplicações baseadas em Serviços Web e aplicações da Web de Dados. Essa lacuna ocorre porque os Serviços Web são "executados", enquanto que a Web de Dados é "consultada". Dessa forma, esta dissertação apresenta o *Semantic Web (of Data) Services* (SWoDS) com objetivo de prover Serviços Web a partir de bases *Linked Data*. O *Semantic Web (of Data) Services* pode preencher a lacuna entre Serviços Web e aplicações baseadas na Web de Dados, fazendo que a Web de Dados seja "executada" através de Serviços Web Semânticos. Assim, permitindo que dados *Linked Data*, através do SWoDS, integrem aos Serviços Web, por meio de operações de composição automática e descoberta de serviços.

Palavras-chave: SWoDS, Serviços Web Semânticos, Web de Dados, Linked Data, OWL-S, Descoberta, Composição.

Abstract

The Web was originally created to link HTML documents. Nowadays, the Web has improved its potential, and heterogeneous applications, resources, data and users can interact between each other. Two proposals for improvement of the current Web, Semantic Web and Web Services, have established standards that make the interoperability between heterogeneous Web applications possible. Another way to improve the current Web is the Web of Data, which provides guidelines (Linked Data) about how to use Semantic Web standards for publishing and definition of semantic links on diverse data sources. However, there is a gap in the integration between Web Service based applications and Web of Data applications. Such a gap occurs because Web Services are "executed" and Web of Data applications are "queried". Therefore, this dissertation introduces the Semantic Web (of Data) Services (SWoDS), in order to provide Web Services for data sources from the Web of Data. Therefore, the Semantic Web (of Data) Services can fulfill the current gap between Web Services and Web of Data applications by making the Web of Data "executed" through Semantic Web Services. Thus, allowing Linked Data bases, through SWoDS, integrate with Web Services, by means operations of automatic composition and service discovery.

Keywords: SWoDS, Semantic Web Services, Web of Data, Linked Data, OWL-S, Discovery, Composition.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CERN: *European Organization for Nuclear Research*

HTML: *Hypertext Markup Language*

HTTP: *Hypertext Transfer Protoco*

LOD: *Linking Open Data*

OWL-S: *Semantic Markup for Web Services*

OWL: *Web Ontology Language*

RDF: *Resource Description Framework*

REST: *Representational state transfer*

SAWSDL: *Semantic Annotations for WSDL and XML Schema*

SOA: *Service Oriented Architecture*

SOAP: *Simple Object Access Protoco*

SPARQL: *SPARQL Protocol and RDF Query Language*

SWoDS: *Semantic Web (of Data) Service*

SWSF: *Semantic Web Services Framework*

URI: *Uniform Resource Identifiers*

W3C: *World Wide Web Consortium*

WADL: *Web Application Description Language*

Web API: *Web Application Programming Interface*

WSDL-S: *Web Service Semantics*

WSDL: *Web ServicesDescription Language*

WSMO: *Web Service Modeling Ontology*

Sumário

Capítulo 1

Introdução

A *World Wide Web*, em sua proposta[1] inicial, foi construída sobre a ideia de hiperligações entre documentos HTML (*Hypertext Markup Language*) [6]. A Web inicial foi chamada de Web dos Documentos, que é um espaço global em que a informação possui como ponto de integração os *hyperlinks*, que por sua vez permitem a navegação entre os documentos [6]. A Web de Documentos também é conhecida como Web 1.0 e tem como foco principal prover conteúdo para leitura por pessoas através de navegadores Web, sem utilização de interfaces para interação do usuário [7].

A partir do projeto inicial, a Web expandiu suas capacidades, de modo a permitir a interatividade e colaboração entre usuários. Essa transformação foi denominada de Web 2.0 [8] e foi fortemente apoiada por aplicações como redes sociais, *wiki's* e *blogs* [9]. Além disso, a Web 2.0 agrega o desenvolvimento de outras tecnologias, as quais visam aprimorar as formas de intercâmbio de informações entre aplicações [7]. Assim, diversos provedores de dados elaboraram aplicações Web, que permitiam acesso e consulta a suas bases de informações por intermédio da *Web Application Programming Interface* (Web API) [6] e também por meio de Serviços Web [10], que são um conjunto de funcionalidades relacionadas que podem ser acessadas por meio de programação através

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

da Web [11].

Os Serviços Web são uma importante tecnologia para a implantação de interações automatizadas entre aplicações distribuídas e heterogêneas [12]. O atual uso dos Serviços Web estende-se a diversas aplicações da vida cotidiana, como a verificação de crédito em contas bancárias, negociação nas bolsas de valores, e informações climáticas de cidades. Tudo isso é possível através de protocolos e arquiteturas padronizadas para viabilizar a comunicação entre as partes. Entre as mais utilizadas, destaca-se o SOAP (*Simple Object Access Protocol*) [13], que é um protocolo para troca de informações estruturas executada acima do HTTP (*Hypertext Transfer Protocol*) e a arquitetura RESTful [14], que é a implementação da arquitetura REST (*Representational state transfer*) para Serviços Web. O protocolo (SOAP) e a arquitetura (RESTful) são respectivamente descritos sintaticamente pelas linguagens WSDL (*Web ServicesDescription Language*) [15] e WADL (*Web Application Description Language*)[16].

Berners Lee et al. [17] afirmam que, apesar da descrição sintática, as aplicações da Web ainda não podem ser interpretadas automaticamente por agentes de *softwares*, pois carecem de informações semânticas. Dessa forma, Berners-Lee et al. [17] propuseram uma arquitetura para extensão da Web atual, na qual os dados são escritos de forma estruturada e com semântica explícita, provendo significado bem definido, com a finalidade de permitir a sua interpretação por agentes de *software*. Essa proposta ficou conhecida como Web Semântica. As principais abordagens para a Web Semântica utilizam ontologias como base dessa padronização. As ontologias definem a representação do conhecimento através de classes, taxonomias e regras de inferência [17].

Seguindo outra linha de evolução da Web, Bizer et al. [1] propuseram um conjunto de boas práticas para publicação de dados na Web, que deu origem aos princípios *Linked Data*. Esses princípios fornecem orientações sobre como usar as tecnologias da Web Semântica para publicar e definir ligações semânticas entre dados de diferentes fontes [6]. Os dados *Linked Data* utilizam a linguagem RDF (Resource Description

Framework) [18], que apresenta recursos do tipo "sujeito-predicado-objeto", como uma das linguagens bases para descrever recursos semânticos. A adoção de tais princípios na publicação de dados deu origem à chamada Web de Dados [19], que foi impulsionada pelo desenvolvimento de um projeto da comunidade de pesquisadores de Web Semântica para publicação de dados abertos [1]: o projeto *Linking Open Data* (LOD) [19]. Esse projeto reuniu pesquisadores ligados a diversas organizações para publicar seus dados usando as práticas do *Linked Data*, dando origem a importantes fontes de informações como, por exemplo, a DBpedia[2] [1].

Como desdobramento das pesquisas sobre Web Semântica, sugiram trabalhos com o objetivo de associar descrições semânticas aos Serviços Web, para automatizar tarefas como a descoberta, composição e invocação de serviços [20]. Descoberta automática é o processo de encontrar serviços que atendam à uma requisição de um usuário ou mesmo de outra aplicação (por exemplo, um agente de *software*) sem a intervenção humana. A composição automática visa atender requisições a partir da integração automática de instâncias (subserviços) que correspondem a partes da requisição [11]. E a invocação automática refere-se à execução automática do Serviço Web. Nesse sentido, diversas abordagens foram propostas como padrões para descrição semântica de serviços, dentre os quais destacam-se *Semantic Markup for Web Services* (OWL-S) [21] e o *Web Service Modeling Ontology* (WSMO) [22].

1.1 Motivação

Apesar de ter mais de uma década de estudos e pesquisas ativas, os Serviços Web Semânticos (SWS) ainda apresentam pouco impacto prático, uso em aplicações reais, no contexto de uso de Serviços Web [23]. Isso é motivado pelo fato de que muitos trabalhos relacionados ao SWS têm como alvo serviços SOAP/WSDL, que por sua vez

[2]Projeto DBpedia - http://www.dbpedia.org

não são prevalentes na Web atual [24]. Segundo Pedrinaci et al. [23] os SWS e a Web de Dados podem de forma combinada eliminar parte das atuais limitações que dificultam o efetivo uso de ambos. A integração dessas duas tendências (SWS e Web de Dados), além disso, pode potencializar o uso dos SWS, pois agregará ao seu campo de alcance uma crescente base dados, servindo como elemento complementar na descoberta e na composição de Serviços Web.

Vários autores buscam desenvolver soluções para viabilizar a integração de Web de Serviços à Web de Dados. Dentre as diversas abordagens destaca-se a proposta do iServe [25], que é uma plataforma para publicação e descoberta de Serviços Web na Web de Dados, utilizando os princípios *Linked Data*. O iServe almeja integrar a Web de Serviços e a Web de Dados através de descrições semânticas simples (de fácil interpretação para desenvolvedores e máquinas) e associadas à Web de Dados. Todavia, tal abordagem não atende a serviços que são descritos por linguagens mais expressivas semanticamente, que aquelas tratadas no iServe.

Seguindo uma linha de pesquisa diferente, porém dentro do mesmo contexto, Taheriyan et al. [26] identificam a necessidade da composição de serviços de diferentes origens para aprimorar o desenvolvimento de aplicações. Assim, os autores propuseram uma abordagem para integrar API's da Web na nuvem *Linked Data* com o uso de uma descrição semântica utilizando RDF e SPARQL (*SPARQL Protocol and RDF Query Language*). Norton e Stadtmüller [27, 28] ressaltam a necessidade de compor serviços RESTful reduzindo o esforço de programação manual do desenvolvedor, com isso propõem a descrição dos serviços usando princípios *Linked Data* e descrevendo semanticamente suas entradas e saídas com SPARQL e RDF.

Em especial ao que diz respeito a Serviços Web e Web de Dados, existe uma lacuna na integração entre aplicações das duas abordagens. Essa lacuna ocorre porque os Serviços Web são "executados" por meio de requisições HTTP, enquanto que a Web de Dados é "consultada" por meio de consultas SPARQL. Assim, essa diferença de

abordagem dificulta o uso de dados da Web de Dados em composição com Serviços Web. Esse problema é bastante atual e relevante, dado que diversos autores têm proposto trabalhos [12, 25, 26, 28, 29, 30, 31] na direção de resolver os problemas envolvidos na integração de serviços e dados. Portanto, é justamente na integração dos Serviços Web e Web de Dados que se motiva este trabalho, sobe o qual foi desenvolvida uma solução que possibilite a descoberta composição e invocação de Serviços Web com dados oriundos da Web de Dados.

1.2 Objetivos

O objetivo geral deste trabalho consistiu em pesquisar e desenvolver uma solução para prover Serviços Web a partir da Web de Dados, especificamente a partir de bases *Linked Data*. A proposta utiliza Serviços Web Semânticos descritos em OWL-S, a fim de tornar a nuvem *Linked Data* um provedor de Serviços Web. Isso possibilitará a integração automática (descoberta e composição) de dados provenientes da Web de Dados com outros tipos de serviços (SOAP e RESTful) suportados pelo OWL-S. Isso através da definição Serviços Web denominados de *Semantic Web (of Data) Service* (SWoDS).

Como desdobramento para atingir o objetivo geral, observou-se algumas etapas intermediárias, que assim compõem os seguintes objetivos específicos:

- Estender a parte ontologia OWL-S, que é referente à descrição da execução do serviço, a fim de permitir o suporte aos Serviços Web provenientes da Web Dados com a descrição semântica OWL-S. Tal objetivo também viabilizará a composição dos SWoDS com serviços de outras fontes;

- A extensão da OWL-S API para incorporar as extensões feitas na ontologia OWL-S, de forma que seja possível executar os SWoDS de modo similar à execução de um Serviço Web tradicional (SOAP ou RESTful);

- O desenvolvimento de uma interface para descoberta automática de serviços SWoDS a partir de requisições semânticas de Serviços Web;
- Realizar um estudo de caso que avalie a integração dos Serviços Semânticos da Web de Dados, desde aspectos relacionados a invocação, descoberta e composição com outros serviços de diferentes fontes.

1.3 Contribuições

Diante dos objetivos relacionados na Seção 1.2, tem-se como consequência as seguintes contribuições:

- Desenvolvimento de uma extensão da ontologia de suporte e execução para serviços OWL-S (Grounding) para incluir o suporte a serviços oriundos da Web de Dados;
- Extensão da principal biblioteca de suporte a serviços OWL-S (OWL-S API) para dar sustentação ao novo Grounding, definido no item anterior;
- O desenvolvimento de um módulo que a partir de requisições de serviços descritas em OWL-S gera automaticamente consultas (na linguagem SPARQL) para buscar na Web de Dados informações equivalentes a requisição de serviço;
- Desenvolvimento de um estudo de caso para avaliar a consistência dos SWoDS, aplicando invocação de serviços, descoberta e composição, na qual é relacionado com outras fontes de serviços dentro de um cenário simulado de estudo.

1.4 Estrutura da dissertação

Os próximos capítulos da presente dissertação estão estruturados nos seguintes agrupamentos. No Capítulo 2 é apresentado os principais conceitos, que se relacionam com este trabalho. Os Conceitos estão agrupados dentro da denominação “Web Semântica”,

em que inicialmente é apresentado seus principais conceitos e em seguida subdivide-se em Serviços Web Semânticos e *Linked Data*. Cada subseção desse capítulo aborda importantes assuntos, os quais são pré-requisitos para a compreensão deste trabalho.

O Capítulo 3 apresenta a definição dos SWoDS juntamente com a ontologia proposta para suporte do SWoDS. Além disso, é apresentado o desenvolvimento da extensão aplicada na biblioteca OWL-S, para permitir o suporte e integração do SWoDS aos demais serviços OWL-S.

No Capítulo 4, explana-se sobre a Descoberta e Execução de SWoDS, ressaltando os passos necessários para descobrir novos serviços oriundos da Web de Dados a partir de requisições de serviços descritas em OWL-S. Posteriormente, são apresentados os aspectos relacionados a execução de tal serviço.

O Capítulo 5 é responsável por relatar a avaliação realizada a cerca do SWoDS. A avaliação consiste em um cenário que envolve todos os objetivos relacionados a este trabalho, executando-se a partir de uma requisição de serviço OWL-S, descoberta de serviço SWoDS e composição deste com serviços de outras fontes.

O Capítulo 6 apresenta as pesquisas que se relacionam com o problema abordado pelo presente trabalho, ressaltando os principais aspectos e suas diferenças em relação a esta abordagem.

Por fim, o Capítulo 7 relata as conclusões obtidas a partir da execução deste trabalho, além das oportunidades de trabalhos futuros, que dão continuidade a esta pesquisa.

Capítulo 2

Web Semântica

A World Wide Web, na sua origem, foi desenvolvida para viabilizar o compartilhamento remoto de informações entre pessoas, sem exigir alto conhecimento especializado pra publicação de conteúdo [32]. Com intuito inicial de atingir o público de físicos e engenheiros do laboratório CERN (*European Organization for Nuclear Research*), a Web hoje está presente na vida de bilhões de pessoas e tornou-se elemento indispensável para diversos serviços e situações em inúmeros contextos.

Desde o período da sua criação até os tempos atuais, a Web tem passado por transformações no seu uso e aplicabilidade. O'Reilly [8] relata a transformação ocorrida na Web, onde o conteúdo passa a ser publicado de forma colaborativa entre pessoas e aplicações. Aplicações como *wiki's*, rede sociais e *blogs* fortaleceram a massiva publicação de conteúdo, onde a partir de então essa fase ficou conhecida como Web 2.0. Contudo, essa transformação da Web não resultou em mudanças significativas estruturais na publicação de dados, isto é, apesar da maior interatividade na publicação de dados, as informações ainda continuam utilizando a mesma estruturação projetada na origem da Web.

A estrutura desenvolvida para configuração e apresentação dos dados publicados na Web, através dos documentos HTML, foi projetada de modo simples o suficiente para

viabilizar uma rápida expansão e popularização da Web [33]. Todavia, essa simplicidade tem limitado e aumentado o desafio de prover soluções que processem semanticamente a grande quantidade de informações disponíveis [34]. Isso ocorre porque os documentos HTML apresentam estruturação simples, com alguns recursos sintáticos, mas com quase ausência de representações semânticas. Assim, com o explosivo crescimento da Web, alguns problemas resultantes da falta de estruturação semântica dos dados resulta em problemas cada vez mais comuns, como a baixa precisão dos mecanismo de buscas de conteúdo, com entraves na correlação de vocabulários e relações semântica entre páginas do mesmo tema [35].

O conteúdo da Web tradicional foi desenvolvido para a leitura e compreensão por pessoas, o que consequentemente inviabiliza a interpretação automática do significado de tais dados por agentes de *software* (programas de computador). Assim, a Web Semântica propõe uma estrutura que torne capaz a compreensão semântica de conteúdos publicados na Web por agentes de *software*. De modo que estes sejam capazes de processar tais informações e realizar complexas associações relacionadas ao significado semântico dos dados [17].

Segundo Berners-Lee et al. [17], a proposta da Web Semântica não é de substituir a Web atual e sim de estendê-la a partir da adição de estrutura e de semântica explícita aos documentos da Web e à infraestrutura já existente. Essa estruturação é fundamental para dar suporte à representação do conhecimento, o qual apoia-se em informações logicamente estruturadas e regras de inferência para produzir raciocínios automatizados.

Os tópicos a seguir apresentam tecnologias para suporte da Web Semântica [17]: RDF, que é a principal linguagem usada para descrição dos documentos semânticos; Ontologias, que são modelos de dados para representação de conhecimentos; e por fim a arquitetura da Web Semântica com suas diferentes camadas de maturidade.

RDF

A linguagem *Resource Description Framework*, ou simplesmente RDF, é uma linguagem abstrata que reflete em um modelo de grafo simples baseado em dados e semântica formal com uma notação rigorosa [36]. A linguagem RDF é bastante adequada para o contexto da Web, pois agrega uma descrição semântica dos recursos, permitindo a interligação entre as entidades de modo semelhante aos *hiperlinks* da Web de Documentos.

O modelo RDF é descrito em triplas no formato sujeito, predicado e objeto [36]. O sujeito e o objeto podem ser identificados por URI, que representa um recurso, ou uma URI e um literal respectivamente. Já o predicado (propriedade) especifica a relação semântica entre o sujeito e o objeto e é representado por uma URI. A Figura 2.1 exibe a relação sujeito-predicado-objeto em uma visão de um grafo. É possível notar que o sujeito e objeto são vértices do grafo e o predicado é a aresta que os conecta.

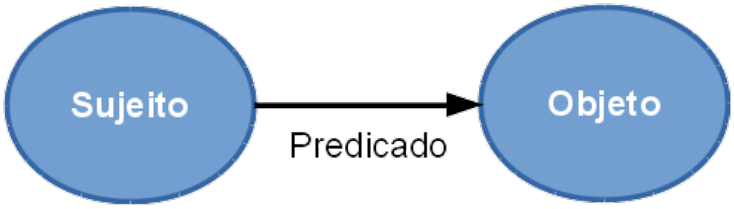

Figura 2.1: Grafo conceitual da tripla RDF

Em RDF, um documento faz afirmações de que as coisas particulares (pessoas, páginas da Web ou qualquer outro recurso) têm propriedades (por exemplo, "é uma irmã de", "é o autor de") com certos valores (outra pessoa, outra página Web)[17]. Por exemplo, na Figura 2.2 a primeira tripla RDF descreve a relação do grupo DIG com o membro Tim Berners-Lee. A tripla descreve que o grupo representado pela URI `http://dig.csail.mit.edu/data#DIG` (sujeito) possui como membro (predicado descrito por `http://xmlns.com/foaf/0.1/member`) Tim Berners-Lee (URI `http://www.w3.org/People/Berners-Lee/card#i`) (objeto). Já a segunda parte do exemplo mostra o recurso referente a um filme, a saber *Pulp Fiction*, o qual o predicado indica

uma relação de equivalência do sujeito com o objeto.

```
Sujeito: http://dig.csail.mit.edu/data#DIG
Predicado: http://xmlns.com/foaf/0.1/member
Objeto: http://www.w3.org/People/Berners-Lee/card#i

Sujeito: http://data.linkedmdb.org/resource/film/77
Predicado: http://www.w3.org/2002/07/owl#sameAs
Objeto: http://dbpedia.org/resource/Pulp_Fiction_%28film%29
```

Figura 2.2: Exemplos de triplas RDF [1]

O formato padrão de serialização do RDF é o XML, que foi introduzido pela W3C nas especificações do *framework* [37]. Entretanto existem outros formatos que são utilizados, como o *Notation3* [38], que foi desenvolvido pela W3C e apresenta um modelo alternativo ao XML para descrever documentos RDF.

Ontologia

O termo ontologia tem origem na filosofia, cujo o significado relaciona-se a um relato sistemático da existência. De acordo Tom Gruber [39], o termo ontologia para a ciência da computação, mais propriamente para sistemas baseados no conhecimento, está relacionada ao que pode ser representado. Assim, pode-se descrever a ontologia de determinado sistema ou domínio através de um formalismo declarativo composto de um conjunto de objetos, relações e axiomas formais que restringem a interpretação e o uso desses termos. Uma outra definição bastante utilizada pela comunidade de Inteligência Artificial é a definição de Fensel [40]: "uma especificação formal e explícita de uma conceitualização compartilhada".

Já com estudos voltados para Web Semântica, Hendler [41] define ontologia como um conjunto de termos de conhecimento, incluindo vocabulário, interconexões semânticas, regras de inferência e lógica. Em outros termos, tal definição argumenta que as ontologias são definidas para representar o conhecimento, de modo que tal descrição possa ser

compreensível para agentes de software. Nesse sentido, o grupo *Web Ontology* da W3C identificou a necessidade de desenvolver uma linguagem para descrição de ontologias que fosse poderosa e expressiva. Assim, surgindo a partir da linguagem DAML+OIL, esse grupo desenvolveu a linguagem OWL (*Web Ontology Language*), a qual foi destinada a ser a linguagem padronizada e amplamente aceita da Web Semântica [42].

A linguagem OWL faz uso das linguagens RDF e RDFS (RDF Schema), porém define novos elementos, o que acresce sua expressividade. Os principais elementos da OWL são:

- **Classes**: Definem conceitos, entidades ou conjunto de objetos do mesmo tipo;
- **Propriedades**: Definem propriedade sobre as classes. OWL suporta dois tipos de propriedades: *ObjectProperty*, que define uma relação entre dois objetos (exemplo professor *ensina a* aluno); *DataTypeProperty*, que relaciona objetos com valores (Exemplo a idade ou nome de uma pessoa);
- **Restrições**: São propriedade que definem o uso, relação ou valor de uma determinada classe;
- **Instâncias**: Definem implementações das classes como indivíduos do mundo real;
- **Regras**: São regras lógicas, que permitem inferência de conhecimento a partir das definições descritas na ontologia.

A OWL possui três sub-linguagens: OWL Lite, OWL DL e OWL Full, que possuem diferentes níveis de expressividade [42]. A **OWL Full** é a que possui máxima expressividade e a liberdade sintática do RDF(S), sem nenhuma garantia computacional. Por exemplo, em OWL Full é possível impor uma restrição de cardinalidade sobre a classe de todas as classes, essencialmente, limitando o número de classes que podem ser descritos em qualquer ontologia. OWL Full permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL. É improvável que algum software de inferência venha

a ser capaz de suportar completamente cada recurso da OWL Full, pois tal linguagem apresenta pontos que são computacionalmente indecidíveis. A **OWL DL** possui parte da expressividade da OWL Full, a qual mantém computabilidade e a decidibilidade. OWL DL inclui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições. A **OWL Lite** suporta principalmente classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. OWL Lite também tem uma menor complexidade formal que OWL DL.

Arquitetura da Web Semântica

A Web Semântica é constituída por uma série de padrões estalecidos e controlados pelo consórcio W3C (*World Wide Web Consortium*). Berners-Lee propôs uma arquitetura em camadas, a qual apresenta um modelo geral para Web Semântica [2]. A Figura 2.3 exibe essa arquitetura, em que é possível notar que nas camadas inferiores são utilizados padrões já estabelecidos da Web, pois o intuito da Web Semântica é estender a Web atual. Já nas camadas superiores são acrescentadas novos padrões, que visam dar maior expressividade semântica aos dados da Web.

Na Figura 2.3, as camadas mais inferiores (**Unicode e URI**) constituem a base de codificação de caracteres e endereçamento da Web. A *Unicode* é uma padronização de codificação de caracteres mundialmente utilizada, que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente [43]. Já o padrão URI (*Uniform Resource Identifiers*) é utilizado para identificação única de recursos da Web Semântica. A URI já é adotada na Web atual como identificação de páginas Web e passa ter seu uso estendido para identificação de recursos, que podem ser qualquer coisa do mundo real (abstrata ou física) que tenha uma identidade.

A camada **XML + NS + xmlschema** ilustrada na Figura 2.3 é responsável pela

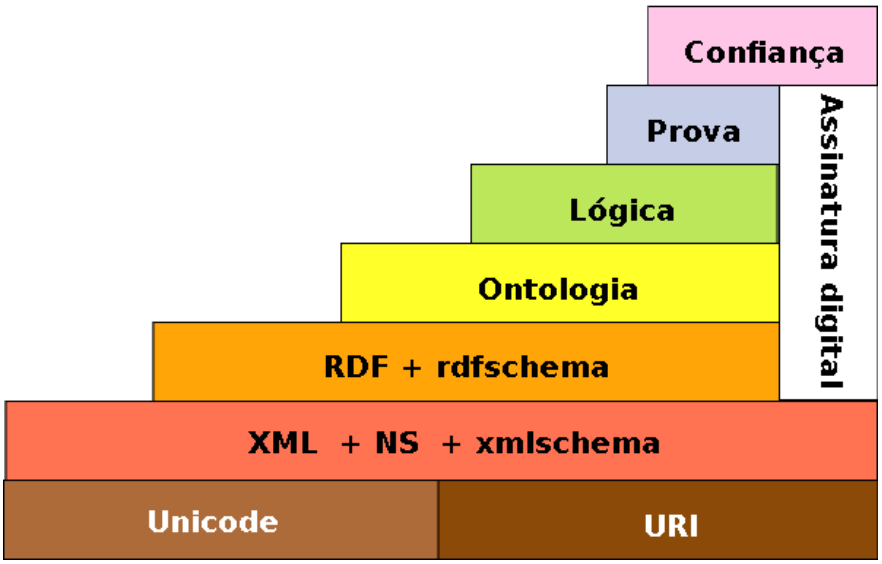

Figura 2.3: Arquitetura em camadas da Web Semântica, adaptado de [2]

descrição sintática dos elementos da Web Semântica, de modo que os torne consistentes e padronizados para permitir o processamento por sistemas computacionais. O padrão XML descreve uma classe de objetos chamada documentos XML e o comportamento de aplicações computacionais que processam esses documentos W3C [44]. Assim, com o XML é possível definir uma série de marcações, as quais permitem descrever, ou representar domínio ou gerar sub-linguagens específicas. Dessa forma, XML é largamente utilizada para suporte às linguagens fundamentadas para descrição do conhecimento na Web Semântica.

A camada **RDF + rdfschema** (Figura 2.3) trata da representação dos recursos da Web Semântica. A linguagem RDF, como já foi citada neste capítulo, é definida como padrão pelo W3C para descrição dos recursos em conjunto com a linguagem XML. Já a linguagem RDF Schema (RDFS) é uma extensão da capacidade semântica do RDF, permitindo mecanismos para descrição de grupos de recursos relacionados e o relacionamento entre os recursos [45]. A camada superior,**Ontologia** (Figura 2.3), que está bastante relacionada ao uso de esquemas RDF(S). A representação de ontologias

na linguagem OWL amplia a capacidade de descrição semântica, pois RDF limita-se à criação de predicados binários para descrição de recursos e RDFS, embora já tenha alguns conceitos de ontologia, é limitado à hierarquia de classes e propriedades, com especificação de domínio e espaço de valores dessas propriedades [42].

A Figura 2.3 representa também a camada **Lógica**, que é responsável pela inserção de regras lógicas, que dada a descrição ontológica dos dados é possível inferir conhecimento por meio de sistemas computacionais. Essas regras lógicas são realizadas por meio da linguagem SWRL, a qual é baseada na combinação de OWL com a linguagem RuleML, para representação de regras lógicas e fatos [46]. Por fim, as camadas de **Prova** e **Confiança** (ainda na Figura 2.3) ainda não possuem definição estabelecida. Swartz [47] prevê a necessidade de inserir assinaturas digitais aos conteúdos publicados na Web Semântica, de modo a apresentar maior garantia e confiabilidade na sua origem e integridade.

Os desdobramentos das pesquisas sobre Web Semântica resultaram em iniciativas e estudos que visam associar a Web Semântica a outros elementos. Dentre esses estudos resultantes, o presente trabalho relaciona-se com os Serviços Web Semânticos, o qual é discorrido na Seção 2.1, apresentando os seus principais elementos, que são usados como fundamentação para este trabalho. Além disso, a Seção 2.2 apresenta os estudos referentes ao *Linked Data*, que também ampara os objetivos da presente pesquisa.

2.1 Serviços Web Semânticos

Os estudos sobre Serviços Web Semânticos (SWS) se tornaram mais incidentes a partir do ano de 2001, quando se buscou relacionar as propostas da Web Semântica juntamente com os Serviços Web. Nesse mesmo ano McIlraith et al. [20] propuseram a definição de uma linguagem para descrição de Serviços Web chamada DAML-S, que utiliza a linguagem DAML, através de marcações semânticas associadas a ontologias e bases de

conhecimento. O intuito era permitir descoberta, execução e composição automática de Serviços Web, isto é, que agentes de software fossem capazes, de modo independente, de processar Serviços Web.

A primeira geração de Serviços Web representa bem os aspectos sintáticos, como por exemplo um Serviço Web para obtenção da temperatura de uma cidade. A representação sintática do serviço descreve que a entrada é uma *string* e a saída um número *float*. Porém não há representação que a entrada tem um significado semântico que indica uma cidade e nem a saída que representa um valor de temperatura. Martin et al. [21] expõem que a ausência, ou a pouca descrição, do funcionamento semântico dos Serviços Web caracteriza-se como uma das suas principais limitações, pois sem representações semânticas as manipulações dos Serviços Web tornam-se dependentes da intervenção humana, em outras palavras, não sendo compreensíveis por agentes de *software*. Além disso, é crescente a necessidade por especificações semânticas mais ricas de Serviços Web que permitam automação mais completa e flexível da provisão de um serviço. Somente as descrições sintáticas dos serviços não são suficientes para prover uma compreensão semântica, que permita esse tipo de interpretação por agentes de *software*. Assim, permitindo a automação de tarefas como descoberta, composição e execução de Serviços Web [48].

Como evolução da linguagem DAML-S, pesquisadores desenvolveram a linguagem OWL-S, a qual é utilizada na presente pesquisa. Outros estudos desenvolveram-se buscando associar descrições semânticas aos serviços Web, dentre eles destacam-se: SWSF (*Semantic Web Services Framework*)[49], WSMO (*Web Service Modeling Ontology*) [22] e WSDL-S (*Web Service Semantics*)[50].

Esta seção aborda os conceitos e tecnologias dos Serviços Web. A Seção 2.1.1 apresenta os aspectos referentes à descrição sintática dos Serviços Web, tal como sua representação e arquiteturas mais utilizadas. Já na Seção 2.1.2 apresenta uma das principais linguagens de descrição de Serviços Web Semânticos (OWL-S), a qual é

objeto de estudo do presente trabalho. Em seguida (Seção 2.1.3), explana-se sobre a OWL-S API, principal biblioteca de suporte aos serviços OWL-S, discorrendo sobre os principais aspectos referentes à implementação e uso da biblioteca. Por fim, a Seção 2.1.4 faz uma breve apresentação de outras representações de Serviços Web Semânticos, que também são objetos de pesquisa.

2.1.1 Serviço Web

Serviço Web é um componente de software, que tem como objetivo prover a interoperabilidade entre aplicações que trocam informações através de uma linguagem comum para comunicação [51]. Segundo Stal et al. [52], no contexto atual da Web, os Serviços Web são de fundamental importância, pois o ambiente da Web é bastante heterogêneo, e diversas aplicações consomem informações de outras. Por exemplo, uma aplicação de venda *online* de produtos precisa se comunicar com a aplicação da empresa que faz a entrega dos seus produtos. O componente para essa comunicação deve usar uma linguagem comum, a qual ambos enviem/recebam mensagens através de um protocolo padronizado.

A importância dos Serviços Web fez surgir a necessidade por padrões que permitam utilizá-los. Dan et al. [53] afirmam que um dos padrões mais bem estabelecidos é *Service Oriented Architecture* (SOA) [54], que estabelece um modelo arquitetural para interação de aplicações o qual é aplicável para Serviços Web. A Figura 2.4 exibe um modelo da arquitetura SOA, em que é possível observar que a figura exibe a interação de um Serviço Web com uma aplicação através de um registro de Serviços Web. Primeiramente (Figura 2.4 item 1), o Serviço Web publica seus parâmetros de uso (descrição das entradas, saídas, URL de acesso, etc) em um registro de Serviços Web (repositório de Serviços Web). Assim, aplicações acessando o registro do Serviço Web obtêm a descrição necessária para utilizar o serviço (Figura 2.4 item 2). Por fim, com a descrição do serviço a aplicação torna-se habilitada a executá-lo (Figura 2.4 item 3).

A linguagem XML, por exemplo, é bastante utilizada nos padrões de descrição de serviços, pois além de prover capacidade de expressão é muito utilizada em contextos de interoperabilidade.

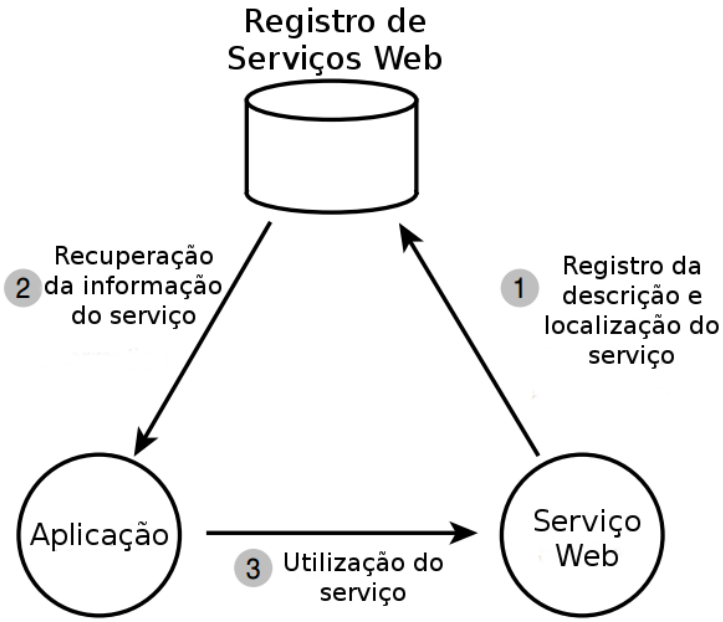

Figura 2.4: Modelo de arquitetura SOA, adaptado de [3]

O *REpresentational State Transfer* (REST) foi inicialmente introduzido como estilo arquitetural para construção distribuída em larga escala de aplicações hipermídia, todavia tal arquitetura passou a ser aplicada para Serviços Web baseados no protocolo HTTP [55]. A arquitetura REST é baseada em quatro princípios [56]: i) Identificação única, através de URIs; ii) Interface uniforme, onde cada operação utiliza um método HTTP diferente (GET, POST, PUT,DELETE); iii) Mensagens auto-descritivas, onde os recursos estão desassociados de suas representações, isto é podem ser acessados através de diferentes formatos de dados; iv) Interações de estado através de *hyperlinks*, onde as interações com os recursos são sem estado (como o protocolo HTTP). A aplicação dessa arquitetura para Serviços Web foi denominada de RESTful.

A Linguagem de descrição de Serviços Web *Web Services Description Language* (WSDL) [15] é um padrão bem estabelecido como parte importante da construção de Serviços Web. Foi desenvolvida e padronizada no grupo de trabalho de descrição de Serviços Web do W3C. Basicamente, o WSDL permite a especificação da sintaxe de entrada e saída de um serviço, bem como outros detalhes necessários para invocação do serviço. Os documentos WSDL são descritos em XML e fornecem informações de como interagir com o serviço, descrevendo os tipos de dados definidos, os métodos e parâmetros e a URI de acesso. A comunicação entre o Serviço Web e aplicação é feita utilizando o protocolo Simple Object Access Protocol (SOAP) [57], que especifica o encapsulamento dos dados trocados em um envelope que contém um cabeçalho, o qual representa os requisitos da aplicação e a segurança de acessibilidade aos dados, e o corpo que contém a mensagem em si. Assim, os Serviços Web descritos com WSDL especificam sintaticamente como utilizar os serviço, que utilizam a comunicação através do protocolo SOAP.

Similar ao WSDL, existe também a linguagem de descrição de Serviços Web chamada *Web Application Description Language* (WADL) [16], que através de documentos XML descreve serviços do tipo RESTful. A WADL foi desenvolvida pela W3C em conjunto com a Sun Microsystems com o objetivo de simplificar a reutilização de aplicações Web baseadas no protocolo HTTP, promovendo um melhor aproveitamento de tais serviços estendendo seu uso para além do navegador Web. Um número crescente de empresas baseadas na Web (Google, Yahoo, Amazon, Flickr, etc) estão desenvolvendo aplicativos baseados em requisições HTTP que fornecem acesso aos seus dados internos. Normalmente, esses aplicativos são descritos usando documentação textual, que em alguns casos é complementada com mais especificações formais, como esquema XML [16]. WADL é projetado para fornecer uma descrição capaz de permitir um processamento de máquina sobre tais aplicações através de uma descrição sintática desses Serviços Web.

As descrições sintáticas dos Serviços Web atendem bem o papel de apresentar para máquina os requisitos sintáticos de cada serviço. Contudo, se os Serviços Web são utilizados em um cenário de composição de serviços, tais descrições não são suficientes para viabilizar um processamento automático das composições [58]. O mesmo ocorre em um contexto de descoberta automática de Serviços Web. Dessa forma, uma possível solução é associar os Serviços Web aos conhecimentos relacionados a Web Semântica, com o uso de ontologias e regras de inferências, assim criando os chamados Serviços Web Semânticos [59].

Na seção seguinte (Seção 2.1.2) aborda-se mais detalhadamente a linguagem de descrição de Serviços Web Semânticos, OWL-S, que é objeto de estudo deste trabalho, posteriormente explana-se sobre outros padrões.

2.1.2 Serviço Web Semântico com OWL-S

Os Serviços Web Semânticos (SWS) visam proporcionar especificações semânticas de Serviços Web, de modo a permitir automação completa, flexibilidade na prestação de serviços, para apoiar a construção de ferramentas e metodologias mais poderosas [60]. Nesse contexto, a linguagem *Ontology Web Language for Services* (OWL-S) [4] surgiu, a partir da linguagem DAML-S, com o objetivo de fornecer uma linguagem de marcação para descrição de propriedades e capacidades de Serviços Web de forma inequívoca e com capacidade de ser interpretada por *software* [61].

A OWL-S é uma linguagem para descrição semântica de serviços, baseada em ontologias descritas em OWL (*Web Ontology Language*) [42], que tem como principais objetivos: (a) compor um *framework* representativo para descrição semântica do serviço; (b) viabilizar a automatização no gerenciamento de serviços, tal como sua utilização por agentes de sofwtare; (c) integrar os padrões existentes dos Serviços Web com a Web Semântica; (d) por fim, ser abrangente o suficiente para dar suporte a todo ciclo de vida do Serviço Web [62]. A OWL-S não tem como objetivo substituir os padrões de

Serviços Web e Web Semântica existentes, sua intenção é representar os Serviços Web com uma exposição semântica explícita, a qual se torne compreensível por programas.

Apesar dos autores referenciarem OWL-S como uma linguagem, a OWL-S é uma ontologia descrita com OWL, mais especificadamente OWL-DL, onde são representados axiomas, classes e relações da ontologia [62]. Assim, OWL-S foi projetada para descrever Serviços Web de modo independente da arquitetura que constitui os serviços.

De modo geral, a ontologia OWL-S é subdivida entre três sub-ontologias (como pode ser visto na Figura 2.5): *profile*, *process* e *grounding* [4]. A sub-ontologia *profile* apresenta o que o serviço faz, a sub-ontologia *process* descreve como o serviço é constituído e a *grounding* descreve como executar o serviço.

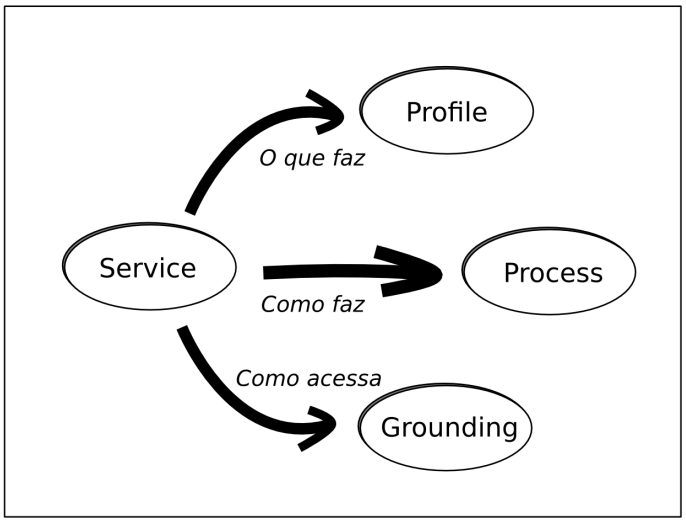

Figura 2.5: Descrição alto nível do OWL-S, adaptado de [4]

Sub-ontologia Profile

A sub-ontologia *profile* descreve o que o serviço faz. Provê um conjunto de conceitos para especificar a capacidade do serviço, com o objetivo de tornar possível a sua descoberta por agentes de *software*. Ou seja, permite que provedores de serviços anunciem o que

os seus serviços fazem, englobando informações como entradas e pré-condições exigidas pelo serviço e saídas e efeitos resultantes. Além disso, outras características podem ser especificadas no perfil de um Serviço Web Semântico, como categoria do serviço, tempo de resposta, qualidade de serviço, disponibilidade geográfica, etc [62].

As entradas são os objetos que representam o que serviço precisa para executar e as saídas são os objetos que são emitidos como resultado da execução. As pré-condições são proposições que precisam ser verdadeiras para o Serviço Web ser executado corretamente. Já os efeitos resultantes são as proposições que se tornam verdadeiras após o serviço ser executado. As entradas, precondições, saídas e efeitos, que são utilizados pelas três sub-ontologias da OWL-S, são representados respectivamente no OWL-S pelos elementos *input*, *precondition*, *output*, *result*.

A Figura 2.6 apresenta uma visão geral da sub-ontologia *profile* com a representação das classes e propriedades envolvidas. Note que a classe *profile* é definida como uma subclasse de *ServiceProfile*, que por sua vez define algumas propriedades de classe para expressar as entradas, saídas, pré-condições e resultados. Na classe *profile* também são definidas propriedades referentes a características e informações sobre o serviço.

Sub-ontologia Process

A sub-ontologia *process* descreve como o serviço é usado. Especifica os padrões de interação com o Serviço Web, que uma vez identificado pelo descrição do *profile* deve obsevar um modelo mais detalhado para determinar se o serviço realmente atende as necessidades do agente de *software*. Os processos podem ser do tipo *atomic* ou *composite*. O primeiro não possui estrutura interna e corresponde a um simples intercâmbio de entradas e saídas entre um provedor e um consumidor. Já o segundo consiste em um conjunto de componentes de processo interligados por estruturas de fluxo de controle e de dados, que descrevem como o serviço é provido a partir da composição de subelementos.

Segundo Prazeres et al.[63], o processo de descoberta de um serviço a partir de uma

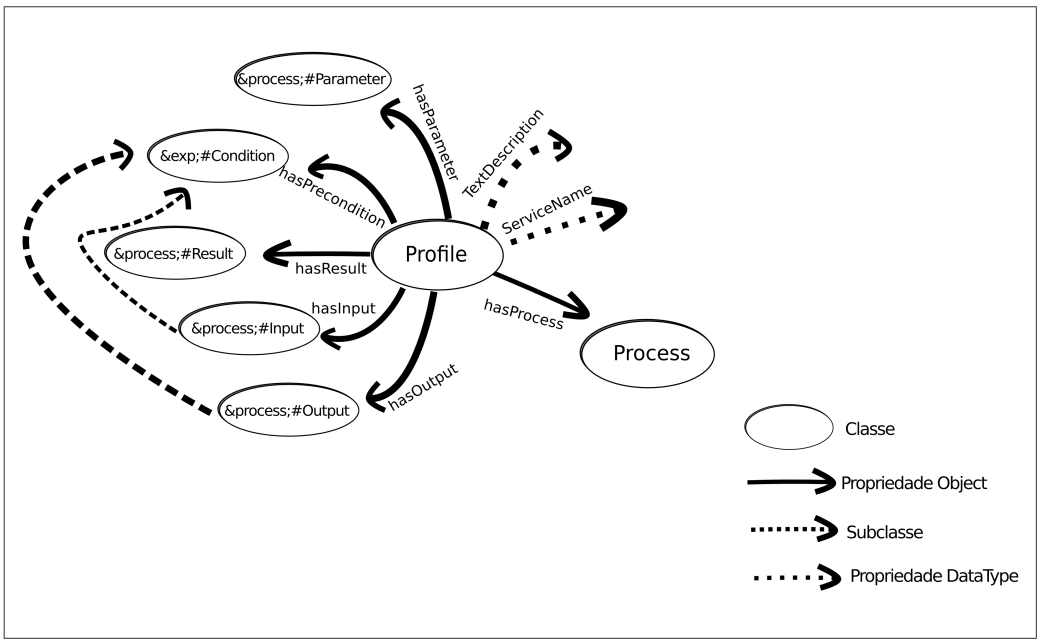

Figura 2.6: Demonstração da ontologia *profile* da OWL-S, adaptado de [4]

requisição semântica pode resultar em diferentes níveis de sucesso. Entre a possibilidade de um ou mais serviços atenderem totalmente a solicitação (melhor caso) e a possibilidade de nenhum serviço atender alguma parte da mesma (pior caso), existem outras alternativas: vários serviços podem atender parcialmente a requisição, e possivelmente podem respondê-la a partir de um composição de serviços parciais. Essa possibilidade motivou as buscas pela realização dessa composição de modo automático através das descrições semânticas.

Portanto, a linguagem OWL-S permite a geração de serviços atômicos ou compostos. Tal característica é definida na sub-ontologia *process* do serviço, no qual pode ser caracterizado como *atomic* (atômico) ou *composite* (composto). A Figura 2.7 apresenta uma visão alto nível da sub-ontologia *process*, exibindo os métodos de processos atômicos e compostos e também visualizações dos principais elementos da ontologia de processos da OWL-S. Os processo atômicos (*AtomicProcess* na Figura 2.7) são executados de

forma direta sem necessidade de composição ou processamentos internos, isto é, sua execução é realizada em um único passo, que consiste na invocação de um método do serviço. O processo atômico corresponde a uma simples troca de entradas e saídas entre o consumidor e o provedor do serviço, ou seja, é um serviço proveniente de uma única fonte.

Os processos compostos (*CompositeProcess* na Figura 2.7) são construídos através da composição de processos atômicos ou de outros processos compostos. Na composição os processo são conectados por meio de estruturas de controle de fluxo e de fluxo de dados (*ControlConstruct* na Figura 2.7). As estruturas de controle de fluxo são semelhantes às estruturas das linguagens de programação, como por exemplo [4]:

- **Sequence**: Os componentes são executados em uma ordem sequencial;
- **Split**: Componentes do processo são executado simultaneamente (em paralelo);
- **Any-Order**: Permite que um grupo de componentes sejam executados sem uma ordem definida, porém não de forma simultânea;
- **Choice**: Dado um grupo de componentes (com as mesmas propriedades) permite a escolha de um dos componentes para execução;
- **If-Then-Else**: Executa os componentes a partir de uma condição booleana;
- **Repeat-While**: Repete a execução de um grupo de componentes enquanto a condição for verdadeira.

A sub-ontologia de processo também possui o tipo de processo *SimpleProcess*. Trata-se de uma especialização para descrever processos simples, os quais são utilizados apenas como elemento de abstração. Processos simples podem ser utilizados com o intuito de fornecer versões simplificadas de processos compostos, para fins de planejamento.

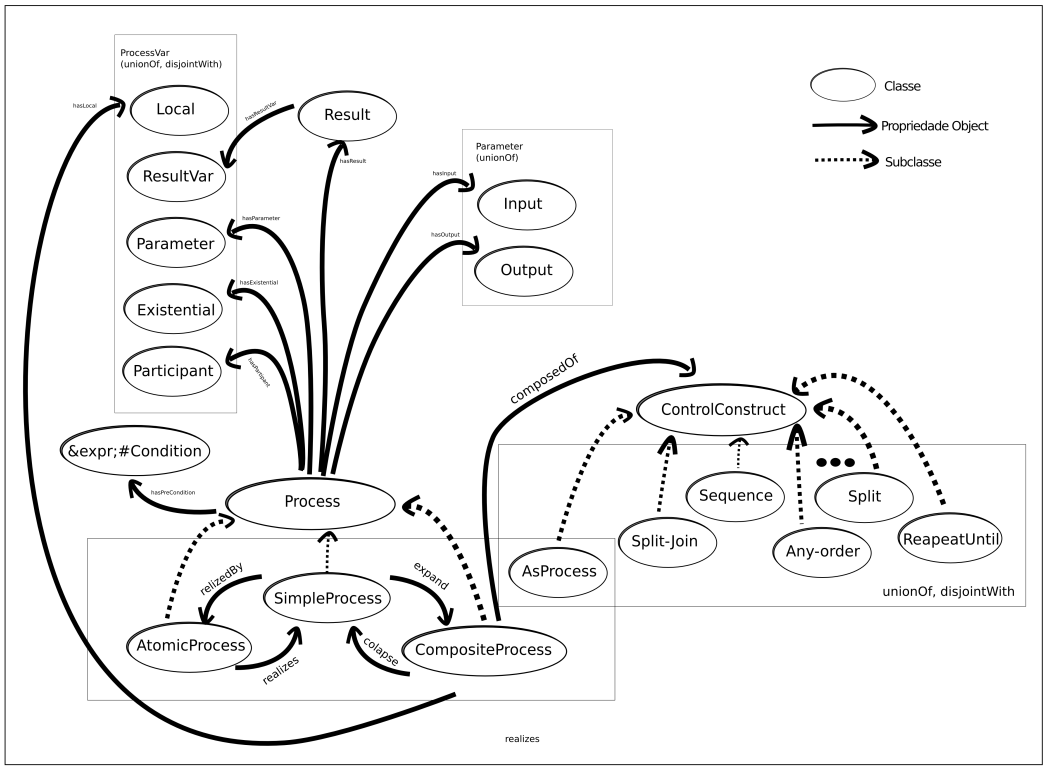

Figura 2.7: Demonstração da ontologia *process* da OWL-S[4]

Sub-ontologia Grounding

A sub-ontologia *grounding* representa a descrição de como interagir com Serviço Web. Em outras palavras essa sub-ontologia define a concretização do serviço através de especificações abstratas de características do Serviço Web que podem ser usadas por agentes software para executar o serviço. A *grounding* define elementos para aplicar uma conexão entre as classes semânticas definidas pela OWL-S (como *inputs* e *outputs*) com elementos referentes ao serviço a ser executado. Dessa forma, permite que a partir da descrição semântica do serviço seja possível executá-lo de modo automático [21].

O OWL-S não impõe o uso de arquiteturas especificas para implementação do *grounding*, isso é um ponto importante na capacidade de flexibilização e adaptação da linguagem para diferentes tecnologias. Assim, com tal definição é possível adaptar

a linguagem OWL-S para dar suporte a diferentes implementações de Serviços Web de diferentes arquiteturas. Uma das primeiras definições, foi o suporte para serviços WSDL/SOAP através do WSDLGrouding [62]. Paolucci et al. [64] propuseram uma descrição semântica com OWL-S para execução de serviços P2P. Masuoka et al. [65] utilizam OWL-S em sistemas para dispositivos sem fio com o conjunto de protocolos UPnP (*Universal Plug and Play*). Existem trabalhos também que buscam representar outras estruturas de SWS em OWL-S, como WSMO [66] e o SAWSDL [67]. Por fim, a pesquisas que implementaram uma extensão do *grounding* para suporte a serviços RESTful descritos com WADL [68, 69].

A sub-ontologia *grounding* define uma série de classes e propriedades que precisam ser estendidas para adaptar-se a tecnologia implementada para execução do Serviço Web. Assim, todas as implementações citadas tiveram que desenvolver uma extensão do *grounding*, que desse suporte as particularidades de cada tecnologia.

A Figura 2.8 apresenta um modelo de mapeamento da ontologia OWL-S com elementos de um Serviço Web WSDL/SOAP através de uma implementação do *grounding* para Serviços Web WSDL (WSDLGrouding). Primeiramente, um processo atômico do OWL-S (*AtomicProcess*) é relacionado com o elemento *operation* do WSDL, para construir uma execução do serviço referente ao WSDL relacionado. É possível notar também que a Figura 2.8 exibe a interação dos elementos OWL-S com o documento WSDL, onde existe uma correlação entre o elemento *message* do WSDL com os *inputs/outputs* do OWL-S. Nessa correlação, os elementos são relacionados com a entrada ou saída correspondente, de modo que para cada elemento de entrada/saída semanticamente descrito com OWL-S é relacionado com um elemento *message* do WSDL. Por fim, as operações e mensagens em WSDL são executadas utilizando o protocolo SOAP, que faz a comunicação entre a aplicação consumidora e o provedor do serviço.

A Seção 2.1.3 apresenta aspectos referentes a OWL-S API, que é uma biblioteca que implementa todo ciclo de vida de Serviços Web descritos com OWL-S e é bastante

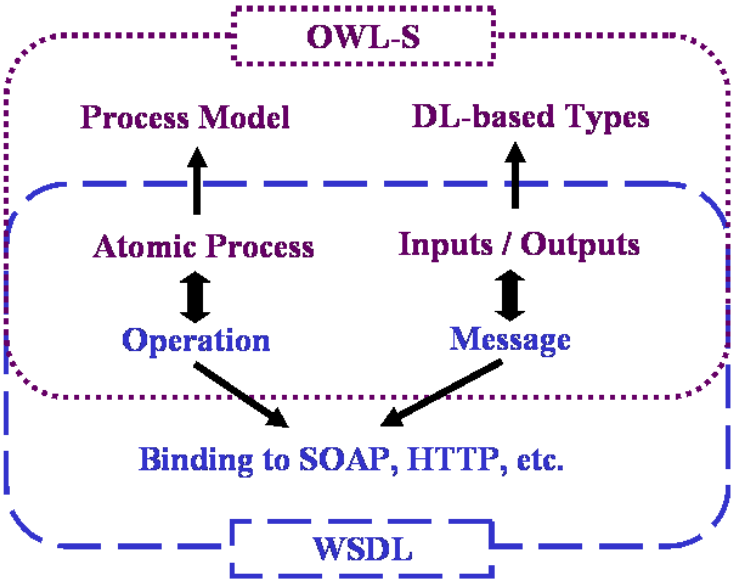

Figura 2.8: Modelo de interação dos elementos da sub-ontologia WSDLGrouding com elementos do Serviço Web WSDL [4]

utilizada para apresentar a proposta deste trabalho de dissertação.

2.1.3 OWL-S API

A linguagem OWL-S possui um conjunto de ontologias que descrevem Serviços Web de modo semântico, provendo condições para descoberta, composição e execução automática de serviços [4]. Apesar da sua estrutura formal, a OWL-S por si só não é capaz de plenamente prover a manipulação de Serviços Web [70]. Assim, para a geração programática das descrições, para validação, para certos tipos de raciocínio (por exemplo, planejamento) e para execução e monitoramento, é útil ter descrições de serviço representados em um nível mais alto de abstração. Deste modo, a OWL-S API[1] foi projetada para ser capaz de obter esse nível de abstração.

A OWL-S API é um biblioteca implementada com a linguagem Java e foi projetada

[1]Disponível em: http://on.cs.unibas.ch/owls-api/

com objetivo de [70]: i) Suportar múltiplas versões do OWL-S (atualmente possui suporte até a versão 1.2); ii) Executar SWS descritos com OWL-S, tanto para processos atômicos quanto compostos, implementando requisitos da ontologia de concretização; iii) Ser extensível para novas implementações de concretizações relacionadas a OWL-S (na sua versão oficial possui implementação para alguns *groundings* como WSDL, UPnP e Java). Ainda com a OWL-S API é possível ler e escrever serviços OWL-S, viabilizando processo de descoberta e composição automática de SWS.

Atualmente a OWL-S API está na versão 3.0 (estável), utiliza máquina virtual Java (JDK) versão 1.5 ou superiores e é um *software* de código-fonte aberto segundo a licença MIT[2]. Foi testada em sistemas operacionais Windows 2000/XP/Vista e sistemas baseados no Unix (Linux, Solaris e Mac OS X) [71]. A API foi construída utilizando o Jena *toolkit*[3] [70].

A estrutura de classes da OWL-S API foi definida para seguir a subdivisão das ontologias OWL-S. Desse modo, foram definidos quatro pacotes: `service, profile, process` e `grounding`. Para garantir a interoperabilidade, a API implementa polimorfismo das classes. Há interfaces que abstraem as funcionalidades previstas nas ontologias. Por exemplo, a classe ontológica `Grouding` possui uma interface que a representa [69]. Para cada tipo de fundamentação há uma classe que implementa tal interface, garantindo o polimorfismo e a interoperabilidade.

O Documento 2.1 exibe um trecho de código referente aos passos necessários para execução de um Serviço Web descrito com OWL-S através da OWL-S API. O primeiro passo é a criação de um motor de execução (`ProcessExecutionEngine` Documento 2.1 linha 2), que através dele o serviço OWL-S é invocado. Para leitura do serviço é preciso criar uma base de conhecimento (`OWLKnowledgeBase` Documento 2.1 linha 5), a qual por meio dela serão processados as informações referentes ao serviço (Documento 2.1 linha 8). Na linha 9 do Documento 2.1 é selecionada uma instância da classe `Process`

[2]Mais informações em: http://opensource.org/licenses/mit-license.php
[3]Disponível em: http://jena.apache.org/

do serviço, a qual é utilizada para executar o serviço (a classe `Grounding` é acionada internamente através do `Process`). Em seguida, são criados os elementos necessários para informar o valor de entrada e relacioná-lo com a descrição da entrada do documento OWL-S (Documento 2.1 linhas 11 a 17). Por fim, através do `ProcessExecutionEngine` é executado (Documento 2.1 linha 20), passando o processo, a entrada e a base de conhecimento. Essa função tem como retorno a saída, a qual pode ter seu valor acessado como demonstram as linhas 23 e 26 do Documento 2.1.

Documento 2.1: Exemplo execução de um Serviço Web com a OWL-S API [72]

```
1  // criacao do motor de execucao
2  ProcessExecutionEngine exec = OWLSFactory.createExecutionEngine();
3
4  // Criacao da base de conhecimento
5  OWLKnowledgeBase kb = OWLFactory.createKB();
6
7  // Leitura do servi o a partir de uma URI
8  Service aService = kb.readService("http://on.cs.unibas.ch/owl-s/1.2/
       Dictionary.owl");
9  Process aProcess = aService.getProcess();
10
11 // Iniciando a variavel que ira conter o valor de entrada
12 ValueMap<Input, OWLValue> inputs = new ValueMap<Input, OWLValue>();
13
14 // Especificando o dado de entrada
15 String inValue = "hello";
16 // Associando a entrada passada com a entrada do servi o OWL-S
17 inputs.setDataValue(aProcess.getInput("InputString"), inValue);
18
19 // Executando o servico
20 ValueMap<Output, OWLValue> outputs = exec.execute(aProcess, inputs, kb);
21
22 // Pegando os dados de saida
23 OWLDataValue out = outputs.getDataValue(aProcess.getOutput());
24
25 // Exibindo o valor de saida
26 System.err.println("Output␣=␣" + out.getValue());
```

Com a OWL-S API é possível também construir processos compostos de serviços utilizando os construtores previstos pela linguagem OWL-S [72]. Para o desenvolvimento de processos compostos deve-se utilizar a classe `CompositeProcess` e instanciar classes

referentes às estruturas de controle utilizada na composição (a OWL-S API implementa: `Choice`, `Sequence`, `AnyOrder`, `Split`, `Split-Join`, `IfThenElse`, `RepeatUntile` `RepeatWhile`).

Também é possível utilizar a biblioteca para monitoramento da execução de um serviço. `ProcessExecutionMonitor` define um conjunto mínimo de funções para essa finalidade, os quais são chamados quando o processo começa, termina ou falha de execução [72]. A OWL-S também possui um módulo específico para validação de documento OWL-S (através da classe `OWLSValidator`). Por fim, a API apresenta suporte às linguagens SPARQL e SWRL, que respectivamente permitem consultas em documentos RDF e construções de regras lógicas. No caso do suporte a SWRL, o suporte implementado pela OWL-S API é parcial.

Na seção a seguir (Seção 2.1.4) são apresentadas outras implementações de SWS, diferentes da OWL-S.

2.1.4 Outros padrões para Serviços Web Semânticos

Este trabalho faz uso direto da linguagem de descrição semântica de Serviços Web OWL-S. Todavia, outras abordagens buscam a associação de Serviços Web e Web Semântica.

SWSF

Semantic Web Services Framework (SWSF) foi proposto por Battle et al. [49] e, assim como o OWL-S, propõe desenvolver um *framework* para descrição semântica de Serviços Web.

O SWSF é composto por dois componentes básicos: a) *Semantic Web Services Language* (SWSL), que é usado para especificar caracterizações formais de conceitos de Serviços Web e descrições de serviços individuais. A linguagem SWSL usa um subcomponente baseado na lógica de primeira ordem para expressar a estrutura formal

do serviço e outro para apoiar a ontologia de serviço através do paradigma de programação lógica; b) Já o *Semantic Web Services Ontology* (SWSO) apresenta um modelo conceitual pelo qual os Serviços Web podem ser descritos, caracterizando formalmente o modelo.

WSMO

Web Service Modeling Ontology (WSMO) [22] é um modelo conceitual para formalização de aspectos relacionados a Serviços Web Semânticos. Ele é baseado em ontologias e possui suporte à entrega e interoperabilidade dos Serviços Web Semânticos.

O WSMO propõe um framework que é divido em quatro elementos principais [22]: 1) ontologias que fornecem a terminologia utilizada por outros elementos WSMO; 2) descrições de Serviços Web, que apresenta os aspectos funcionais e comportamentais de um Serviço Web; 3) objetivos que representam os desejos do usuário; 4) mediadores, que visam a manipulação dos problemas de interoperabilidade entre os diferentes elementos.

WSDL-S

O atual padrão WSDL opera no nível sintático e não tem a expressividade semântica necessária para representar as necessidades e capacidades dos Serviços Web Semânticos [50]. Como já foi abordado anteriormente, a semântica pode expandir a capacidade de desenvolvimento e interpretação dos Serviços Web por agentes de *software*. Foi neste sentido que Akkiraju et al. [50] propuseram uma extensão dos documentos WSDL adicionando anotações semânticas, denominada *Web Service Semantics*, WSDL-S.

Na sua proposta, Akkiraju et al. [50] permitem associar anotações semânticas a elementos WSDL (operações, entradas, saídas, tipos de dados e interfaces), utilizando conceitos bastantes similares ao OWL-S. Contudo, sua principal diferença do OWL-S é que seus conceitos semânticos são executados dentro da própria especificação do WSDL, enquanto o OWL-S executa-os dentro sa sub-ontologia de *grounding*.

SAWSDL

O *Semantic Annotations for WSDL and XML Schema* (SAWSDL) define mecanismos que utilizam anotações semânticas que podem ser adicionados aos componentes WSDL [73]. SAWSDL não especifica uma linguagem para descrição dos modelos semânticos, como por exemplo, ontologias. Em vez disso, o SAWSDL fornece mecanismos pelos quais os conceitos dos modelos semânticos sejam definidos dentro ou fora do documento WSDL. Estas semânticas quando expressas em linguagens formais podem ajudar a descrição dos Serviços Web durante a descoberta automática e composição de serviços.

Para expressar anotações semânticas, o SAWSDL estende três elementos da linguagem WSDL versão 2.0: o *modelReference*, para especificar a associação entre um componente WSDL e um conceito de algum modelo semântico; *liftingSchemaMapping* e *loweringSchemaMapping*, que são adicionados a declarações de elemento XMLSchema e a definições de tipo para especificar mapeamentos entre os dados semânticos e XML. Esses mapeamentos pode ser usado durante a invocação de serviços.

2.1.5 Padrão de descrição semântica de Serviços Web escolhido

Este trabalho optou por utilizar para a linguagem OWL-S para descrição semântica de Serviços Web, devido a proposta da linguagem em ser genérica o suficiente para abarcar diferentes arquiteturas de Serviços Web. Como esta pesquisa propôs um novo tipo de serviço, é muito importante a identificação de tal característica. Além disso, a OWL-S apresenta alguns recursos que foram suportes importantes para o desenvolvimento do trabalho, tal como, o fato de possuir uma biblioteca que implementa a linguagem (OWL-S API) e possuir uma grande coleção de Serviço Web para testes descritos com OWL-S.

2.2 Linked Data

A *World Wide Web* através dos *hyperlinks* revolucionou a forma de visitar conteúdos, permitindo que o usuário possa navegar sobre uma grande nuvem de documentos [9]. Bizer et al.[6] afirmam que a Web clássica é basicamente constituída sobre um pequeno conjunto de padrões:o *Uniform Resource Identifiers* (URIs), o *Hypertext Transfer Protocol* (HTTP) e o *Hypertext Markup Language* (HTML), estes porém não constituem descrição semântica dos documentos. Com a Web Semântica, Berners-Lee et al. [17] propuseram acrescentar à Web uma série de tecnologias que permitissem uma maior expressão semântica dos dados, mas sem abandonar a estrutura atual da Web. Todavia, posteriormente, Berners-Lee [74] acrescenta que, assim como na Web tradicional (Web dos Documentos), os dados da Web Semântica precisam ser interligados para permitir que pessoas e máquinas sejam capazes de explorá-los. Desse modo, tem-se a necessidade de seguir certos princípios para publicação de dados semânticos da Web para viabilizar seu alcance por potenciais consumidores.

Como a maioria das APIs Web não atribui identificadores globais exclusivos para itens de dados, geralmente não é possível estabelecer *links* entre itens de dados fornecidos por APIs diferentes. Assim, devido a esta fragmentação e ausência de semântica da descrição dos documentos, Berners Lee et al. [1] propuseram um conjunto de boas práticas para publicação e conexão de dados estruturados na Web denominada *Linked Data*.

Em linhas gerais, os princípios *Linked Data* visam, na publicação de dados na Web, padronizar a identificação, a recuperação e a representação dos dados, com uma característica diferente da Web tradicional, que permite a descoberta de novas fontes de dados em tempo de execução, seguindo novos *links* nas fontes já existentes [1]. Na Web do *Linked Data*, ou simplesmente Web de Dados, as URIs não mais identificam documentos, elas passam apontar para recursos, que representam entidades do mundo real,

como pessoas, países e categorias de objetos, usando a linguagem Resource Description Framework (RDF) [18].

Com base nesses princípios, Berners Lee [74] enunciou um conjunto de regras básicas para a publicação de dados na Web, que podem ser descritas como:

1. Usar URIs como nome para recursos.
2. Usar URIs HTTP para que as pessoas e aplicações possam encontrar esses nomes.
3. Quando alguém procurar por uma URI, garantir que as informações possam ser obtidas por meio dessa URI, as quais devem estar representadas no formato RDF.
4. Incluir links para outras URIs de forma que outros recursos possam ser descobertos.

O primeiro princípio defende a utilização de URIs para identificar um recurso na Web Semântica. Um recurso pode ser "concreto", como uma pessoa, uma imagem, um documento HTML, ou pode ser um conceito como a inteligência. Por exemplo, a URI `http://www.w3.org/People/Berners-Lee/card#i` identifica o pesquisador Tim Berners-Lee. Tal princípio garante que a Web Semântica seja uma extensão da Web atual, a qual também é endereçada através de URIs.

O segundo princípio fala sobre a utilização do protocolo HTTP, o mesmo usado na Web de Documentos, para referenciar os recursos na Web de Dados. O protocolo HTTP é utilizado como meio para processar requisições na Web de Dados.

Por sua vez, o terceiro princípio diz que quando um usuário ou aplicação procurar por uma URI, devem ser fornecidas informações sobre o recurso referenciado, e que estas informações devem ser representadas nos formatos RDF, RDFS e ontologias OWL. Essas três linguagens são um dos pilares da Web de Dados e constituem as tecnologias para expressão semântica dos dados. Muitos projetos de pesquisa organizam esforços

no desenvolvimento de ontologias para suporte aos dados semânticos. Porém, quando são publicados na Web sem seguir os princípios *Linked Data* tornam-se inacessíveis.

E por fim, o último princípio solicita a inclusão de links para outros recursos, garantindo que a navegação entre os recursos seja transparente como ocorre na Web de Documentos, o usuário navegará na Web de Dados como um único e grande banco de dados. Na Web de Documentos, nenhuma página HTML possui relevância se não estiver interconectada pela grande rede de dados da Web, do mesmo modo assemelha-se a Web de Dados, que ao gerar um grande interconexão de dados amplia o uso e a capacidade de toda a rede.

Esse conjunto de melhores práticas para publicação de dados na Web de Dados ficou conhecido como os "princípios *Linked Data*", e fornecem uma receita básica para a publicação e conexão de dados usando a infra-estrutura da Web, aderindo à sua arquitetura e padrões. Ao contrário da Web 2.0, as aplicações, que compõem diferentes fontes de dados (também conhecidas como *mashups*), trabalham com um conjunto fixo de fontes de dados. Porém aplicações que utilizam a Web de Dados podem descobrir novas fontes de dados em tempo de execução e podem, portanto, oferecer respostas mais completas com novas fontes de dados aparecem na Web [6].

O desenvolvimento do projeto *Linking Open Data* (discutido na Seção 2.2.2) motivou uma massiva publicação de dados adotando os princípios *Linked Data* criando, assim, um espaço global contendo bilhões inserções sobre pessoas, locais, livros, filmes, publicações científicas, remédios, dados estatísticos entre outros. Assim, com o grande volume de dados *Linked Data* sendo publicados na Web, diversos esforços foram reunidos no sentido de desenvolver meios para exploração da Web de Dados. Há pesquisas trabalhando no desenvolvimento de navegadores para dados *Linked Data*, como exemplo do Tabulator [75], que visa apresentar uma visão amigável dos documentos RDF. Também existem trabalhos relacionados a construção de motores de busca semânticos, como Falcons [76], que utiliza a semântica do *Linked Data* para pesquisa de recursos e

ontologias.

Existem estudos também que atuam na ligação entre a Web de Dados e os Serviços Web. Há pesquisas que visam utilizar os princípios *Linked Data* para publicações de Serviços Web Semânticos, utilizando RDF para descrição de entradas e saídas do Serviço Web e SPARQL para prover mecanismos de descoberta [27, 28]. Outros trabalhos buscam incorporar a Web de Serviços na Web de Dados, possibilitando que nela se possa encontrar dados semânticos como serviços [25]. Este trabalho também relaciona a Web de Dados com Serviços Web Semânticos, porém segue uma abordagem de utilizar os dados *Linked Data* para prover novos Serviços Web, possibilitando composições com os serviços já existentes (discutido mais amplamente no Capítulo 3).

A produção e o consumo de *Linked Data* está fortemente ligado ao RDF e a linguagem de consultas em RDF, tal como o SPARQL. O primeiro, previamente apresentado na Seção 2, é a base de armazenamento da Web de Dados. O segundo é apresentado na Seção 2.2.1, onde é relato os principais aspectos da linguagem, relacionando com os dados *Linked Data*. Por fim, na Seção 2.2.2 argumenta-se sobre *Linking Open Data* explanando sobre suas principais características.

2.2.1 SPARQL

O *Simple Protocol and RDF Query Language* (SPARQL) é a linguagem de consultas padrão para bases RDF [77]. Assim como o SQL (*Structured Query Language*) está para as bases de dados relacionais, o SPARQL está para as bases RDF. A linguagem SPARQL foi desenvolvida pela W3C e é considerada uma importante tecnologia da Web Semântica, pois permite a interação com documentos RDF através de consultas, que extraem informações das bases de acordo com os parâmetros selecionados.

Os parâmetros da consulta são definidos através de triplas do tipo sujeito, predicado e objeto, que coincide com a estrutura das triplas RDF. A Figura 2.9 exibe um exemplo de consulta SPARQL. Nessa figura é possível observar que no bloco inicial são definidos

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name "JoseSantos"@en.
  ?person foaf:mbox ?email.
}
```

Figura 2.9: Exemplos de consulta SPARQL

os PREFIX, o qual definem os *namespaces* para URIs utilizadas na consulta. Já na cláusula SELECT, são definidos os campos que serão selecionados para o resultado, no caso do exemplo, o *e-mail*. Por fim, a cláusula WHERE contém a lógica da consulta, isto é, triplas que definem as variáveis com a lógica sujeito-predicado-objeto. No caso do exemplo a linha *?person a foaf:Person* diz que a variável *person* é uma pessoa definida pela ontologia `http://xmlns.com/foaf/0.1/Person`, já a linha posterior diz que a pessoa selecionada possui nome "JoseSantos"no idioma inglês (@en). A última linha da cláusula WHERE associa a variável *?email* com o valor do atributo *foaf:mbox* da pessoa, o qual representa a conta de e-mail das pessoas que possuem nome "JoseSantos".

Quando direcionadas em bases *Linked Data*, as consultas SPARQL são aplicadas em URIs chamadas *endpoints*, que são aplicações que recebem requisições com consultas SPARQL. Nos *endpoints*, as consultas são aplicadas a partir da base de dados relacionada à aplicação.

A especificação da SPARQL é composta por um protocolo que define um meio que transporta as consultas SPARQL entre clientes e processadores [78]. Para isso, são usados Serviços Web que podem ser implementados em REST ou SOAP/WSDL. A partir desse protocolo é possível transmitir a consulta SPARQL para diferentes repositórios semânticos, transformando a Web em um grande e único repositório de dados.

No caso de consultas que lêem os dados do banco de dados, a linguagem SPARQL

especifica quatro variações de consulta diferentes para diferentes fins [77]:

- **SELECT**: Usado para extrair valores brutos de um *endpoint* SPARQL, os resultados são retornados em um formato de tabela;
- **CONSTRUCT**: Utilizado para extrair informações do *endpoint* SPARQL e transformar os resultados em RDF válido;
- **ASK**: Usado para fornecer um simples resultado *true* ou *false* para uma consulta em um endpoint SPARQL. Isto é, retorna *true* caso a base consultada possua dados que são capazes de atender a consulta, e retorna *false* caso contrário;
- **DESCRIBE**: Usado para extrair um grafo RDF do *endpoint* SPARQL, o conteúdo do que é deixado para o *endpoint* para decidir com base no que o mantenedor considerar informações como útil.

Com SPARQL é possível acrescentar parâmetros opcionais como LIMIT, ORDER BY, FILTER, que respectivamente limitam o número de resultados, ordenam a resposta segundo algum campo e restringem as soluções a um determinado padrão definido na consulta [77]. A linguagem SPARQL também permite o tipo de operação para aplicar modificações sobre a base de dados. Isto é feito através da cláusula UPDATE, que permite inserir, atualizar e remover dados de bases. Contudo, tal operação não se aplica às bases *Linked Data*, pois nelas só é permitido o consumo de informações.

2.2.2 O Projeto Linking Open Data

Em Janeiro de 2007, iniciou-se o projeto Linked Open Data (LOD) com base na comunidade com suporte do grupo *Semantic Web Education and Outreach* (SWEO) da W3C [79], que propôs uma estrutura básica de dados abertos seguindo os princípios *Linked Data* [1].

O objetivo do projeto Linked Open Data da SWEO é estender a Web com uma área comum de dados através da publicação de vários conjuntos de dados abertos na Web com RDF e definir os links RDF entre itens de diferentes fontes de dados [79]. Esse conjunto inicial de dados foi extraído basicamente do Wikipedia, Geonames, Musicbrainz dentre outros. Bizer et al.[1] expõem que os participantes nas fases iniciais do projeto foram principalmente pesquisadores e desenvolvedores de universidades, laboratórios de pesquisa e pequenas empresas. Desde então, o projeto tem crescido consideravelmente ao ponto de incluir um envolvimento significativo de grandes organizações como a BBC, Thomson Reuters e da Biblioteca do Congresso dos Estados Unidos. O sucesso do projeto pode ser justificado pela sua natureza, a qual permite que qualquer pessoa possa participar, bastando publicar dados seguindo os princípios *Linked Data* e interligá-los com dados pertencentes a nuvem LOD.

A Figura 2.10 exibe um grafo de relacionamento entre os provedores de dados da nuvem LOD em abril de 2014 [80], que nesta data atingiu cerca de 1,5 trilhões de triplas RDF interligadas por cerca de 909 milhões de documentos, que por sua vez descrevem pouco mais de 8 bilhões de recursos. A nuvem *Linked Data* agrega dados de diversos tipos, contudo, os principais são dados sociais, geográficos e governamentais.

Desde o início do desenvolvimento do projeto LOD faz-se uma categorização dos tipos de dados incluídos na nuvem, que podem variar de acordo com o domínio do conjunto de dados. Assim, pode-se classificar os conjuntos de dados nas seguintes categorias temáticas: Mídia, Governamentais, Publicações Científicas, Ciências Biológicas, Geográficos, Multi Domínio, Conteúdo Gerado pelo Usuário, e Conexões Sociais. Este esquema de categorização é o mesmo utilizado pelo relatório de estado da nuvem LOD de 2011, com a única diferença que foi acrescentada a categoria de Conexões Sociais, pois foi descoberto um grande número de conjuntos de dados que fornecem dados sobre as pessoas e os seus laços sociais [5].

A Tabela 2.1 apresenta uma visão geral do número de conjuntos de dados (*dataset*)

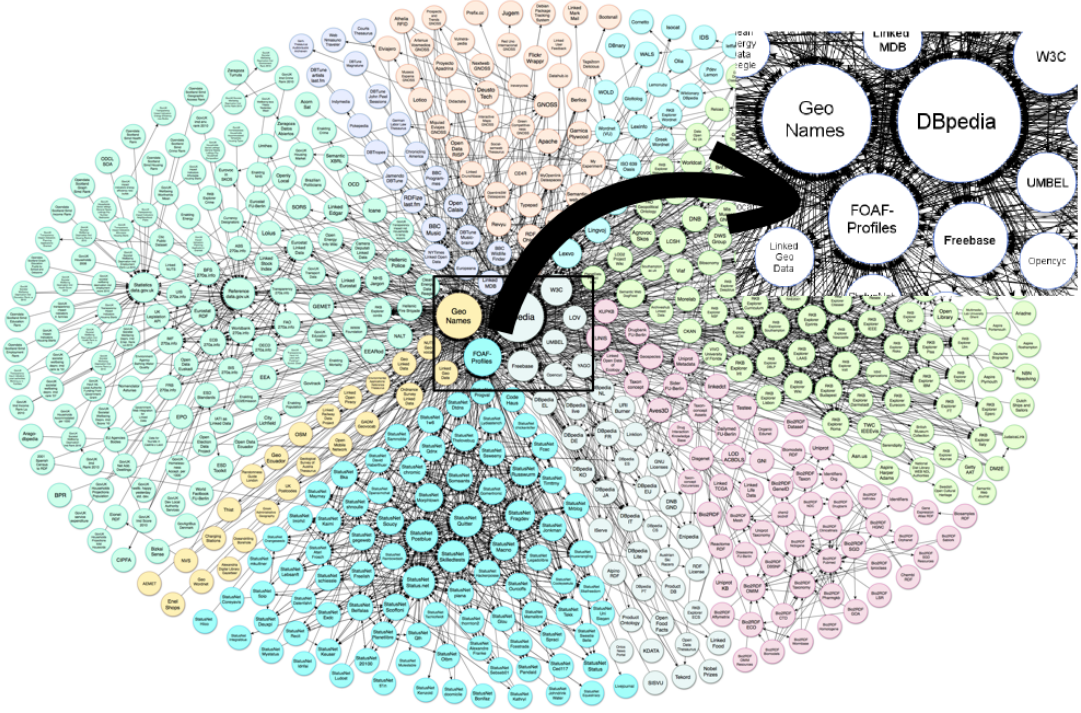

Figura 2.10: Nuvem do Linked Open Data com os relacionamentos entre as bases de dados, adaptado de [5]

em cada categoria, bem como o crescimento por categoria em comparação com o relatório de 2011. O conjunto de dados de maior percentual é categoria de conexões sociais com 520 conjuntos de dados (48% de todos os conjuntos de dados). A segunda maior categoria é o governo com 199 conjuntos de dados (18%), seguido por publicações com 138 conjuntos de dados (13%). Em comparação com o relatório do estado da nuvem LOD 2011, observa-se um maior número de conjuntos de dados em todas as categorias, exceto dados geográficos e de mídia. A categoria Governamentais mostra o maior crescimento, seguido pelas categorias de Conteúdo gerado por usuário e Ciências Biológicas. Excluindo-se a nova categoria de Conexões Sociais, o número total de conjuntos de dados referenciados tem aproximadamente o dobro de 2011 (294 conjuntos de dados) para 2014 (571 conjuntos de dados). Incluindo a nova categoria, observa-se um crescimento global de 271% (294-1091 conjuntos de dados) [5].

Tabela 2.1: Número de conjuntos de dados em cada categoria e de crescimento em relação a 2011 [5]

Categoria	*Dataset*/2014	%	*Dataset*/2011	Crescimento
Mídia	24	2%	25	-4%
Governamentais	199	18%	19	306%
Publicações Científicas	138	13%	87	59%
Geográficos	27	2%	31	-13%
Ciências Biológicas	85	8%	41	107%
Multi Domínio	47	4%	41	15%
Conteúdo gerado por usuário	51	5%	20	155%
Conexões Sociais	520	48%	-	-
Total	1091		294	271%

A Web de Dados através do projeto LOD apresenta forte crescimento e tem mostrado-se como a principal fonte de dados semânticos na Web. Por isso, exibe ainda um imensurável potencial para desenvolvimento de pesquisas, tais como projetos e aplicações.

2.3 Considerações Finais

A pesquisa abordada no presente trabalho visa explorar a relação entre os Serviços Web e a Web Semântica, através dos Serviços Web Semânticos. Em acréscimo, a presente pesquisa busca relacionar os Serviços Web Semânticos com dados publicados na Web de Dados (com os princípios *Linked Data*). Por esse motivo este capítulo tem o objetivo de fornecer subsídios necessários para compreensão dos conceitos relacionados à proposta do trabalho (Serviços Web, Web Semântica, Serviços Web Semânticos e *Linked Data*).

A Seção 2.1 apresenta os Serviços Web Semânticos, que são utilizados neste trabalho através da linguagem OWL-S discutida na Seção 2.1.2. A OWL-S apresenta grande potencial no desenvolvimento de descrições semânticas de Serviços Web e tem relevante presença nas pesquisas relacionadas ao tema.

Também destaca-se os recentes trabalhos relacionados à publicação de dados com

os princípios *Linked Data* (descritos na Seção 2.2), que deram origem a Web de Dados, a qual apresenta-se como uma grande nuvem de dados relacionados, formando a maior rede de informações semântica da Web. A exploração desses dados associados aos Serviços Web Semânticos motivou a presente pesquisa, de modo a buscar explorar oportunidades na correlação das duas áreas.

Dessa forma, as propostas de novos padrões, arquiteturas, tecnologias, e infraestruturas para a Web Semântica precisam levar em consideração a infraestrutura atual da Web. Nesse contexto, duas propostas atuais para a Web Semântica são: i) adicionar semântica aos Serviços Web, por meio dos Serviços Web Semânticos; ii) adicionar semântica aos dados da Web, por meio dos princípios do *Linked Data*. Este trabalho tem a proposta de integrar Serviços Web Semânticos e *Linked Data*, através da proposta *Semantic Web (of Data) Service* (SWoDS), no qual é apresentado no Capítulo 3.

Capítulo 3

Semantic Web (of Data) Service

O presente trabalho apresenta o *Semantic Web (of Data) Service* (SWoDS), cujo a proposta é de prover Serviços Web a partir de bases de dados *Linked Data*. A motivação para SWoDS incide principalmente em três pontos:

- A Web de Dados é um grande base de dados em que o acesso é restrito a consultas SPARQL, o que limita seu potencial, pois dificulta a integração (principalmente automática) com outras fontes de dados não *Linked Data*;
- Com SWoDS é possível interpretar a nuvem *Linked Data* como um provedor de serviços, que, sobe demanda das requisições, pode integrar seus dados aos Serviços Web e outras aplicações;
- SWoDS permite que dados Linked Data sejam integrados aos demais Serviços Web suportados pela linguagem OWL-S (por exemplo, Serviços Web SOAP e RESTful) de modo automático, possibilitando a interoperabilidade dos dados e reduzindo o esforço de programação para composição e descoberta de serviços.

Portanto, SWoDS tem como objetivo viabilizar a exploração, ou seja o acesso, dos dados semânticos da Web de Dados através de Serviços Web Semânticos. Sem essa

proposta, os dados *Linked Data* são geralmente acessados através de consultas SPARQL, que, por sua vez, limitam a capacidade de integração com outras fontes de dados, assim exigindo um maior esforço de programação. Porém, a exploração das informações publicadas na Web de Dados através de estruturas de Serviços Web Semânticos acrescem a capacidade de interoperabilidade e composição de dados *Linked Data* com outras fontes de dados acessíveis através de Serviços Web, como SOAP, RESTful.

Esta pesquisa optou pela utilização da linguagem OWL-S para descrição dos SWoDS, tendo como motivação a ampla utilização da linguagem em projetos de pesquisas na área de Serviços Web Semânticos [81]. Além disso, de modo especial, a linguagem OWL-S permite adaptações (através de extensões da linguagem), que propiciam o suporte da linguagem a diferentes tipos de arquiteturas de Serviços Web, como RESTFul, SOAP, UPnP, P2P dentre outros [21]. Assim, o presente trabalho propõe aplicar uma extensão à linguagem OWL-S para prover Serviços Web que exploram dados nas bases *Linked Data* para atender às solicitações requeridas.

Dessa forma, para dar sustentação a proposta apresentada, este trabalho desenvolveu uma extensão à linguagem OWL-S (discutida na Seção 3.1) de forma a permitir a descrição do SWoDS proposto nesta dissertação. Contudo, a proposta de uma nova extensão da OWL-S necessita de uma implementação que permita executar os SWoDS, portanto esta pesquisa optou utilizar a biblioteca OWL-S API, pois é a principal implementação para manipulação (incluindo operações de descoberta, composição e execução) de SWS com OWL-S. Assim, a presente pesquisa desenvolveu uma extensão da OWL-S API para suporte ao SWoDS (apresentada na Seção 3.2).

3.1 Extensão da sub-ontologia Grounding do OWL-S

A ontologia OWL-S é composta por três sub-ontologias: `Profile`, `Process` e `Grounding`. As duas primeiras são abstratas e genéricas para abranger qualquer implementação de

serviços (WSDL, WADL, etc.) e definem semanticamente os elementos de entrada e saída dos serviços [21]. A ontologia `Grounding`, que foi definida para ser a parte concreta da OWL-S, não define o tipo de implementação do serviço, porém é responsável pela descrição de como o serviço vai ser executado.

A OWL-S pode e deve ser estendida para qualquer tipo de serviço, ou seja é necessário implementar elementos ligados à ontologia `Grounding` que ofereçam suporte à tecnologia do Serviço Web desejado. A OWL-S já possui suporte a Serviços Web descritos com WSDL/SOAP, através do `WSDLGrounding` [21], também existem projetos em desenvolvimento propondo extensões de `Grounding` para serviços WADL/RESTful (`WADLGrounding`) [68], UPnP (`UPnPGrounding`) [65], dentre outros.

Dessa forma, a ontologia `SparqlGrounding`, proposta neste trabalho, é uma extensão à ontologia `Grounding` do OWL-S para permitir que serviços OWL-S utilizem a Web de Dados, ou seja, para de fato implementar a execução do SWoDS proposto neste trabalho.

Nesse contexto, a extensão proposta neste trabalho seguiu os seguintes requisitos:

- Permitir a execução de serviços baseados em consultas SPARQL, que possibilitem o mapeamento de recursos de entrada e saída em triplas SPARQL. Isto é, o `SparqlGrounding` deve expressar consultas SPARQL associadas aos elementos de entrada e saída presentes na ontologia de processos do OWL-S;

- Estar em consonância com as demais ontologias do OWL-S, isto é, que suas entradas e saídas sejam corretamente associadas aos elementos das ontologias `Process` e `Profile`;

- Ser dependente apenas do documento OWL-S e suas sub-ontologias para completa descrição do serviço;

- Oferecer descrição semântica, sem ambiguidades, para processos automatizados de busca, seleção e composição de serviços, executados por agentes de software.

Esses requisitos são necessários para o desenvolvimento de uma extensão do `Grounding`, onde seja possível integrar consultas SPARQL que atendam a requisições de Serviços Web. Por exemplo um Serviço Web, que requisita como entrada um nome de um filme e emite como saída o nome do diretor desse filme. É possível construir uma consulta SPARQL que atenda tal requisição. Porém, não ter um SWS, que atenda a solicitação, inviabiliza a composição automática com outros tipos de SWS, a qual visa atender solicitações mais complexas. Desde modo, é preciso construir mecanismos para descrever consultas SPARQL em bases Linked Data com interfaces de Serviços Web Semânticos com OWL-S.

Na Seção 3.1.1 são apresentados os mecanismos desenvolvidos, através da sub-ontologia `SparqlGrounding`, para associar as consultas SPARQL em bases Linked Data com a semântica dos serviços OWL-S.

3.1.1 SparqlGrounding e SparqlAtomicProcessGrounding

Para implementação de serviços OWL-S que são executados através de consultas SPARQL, é necessário desenvolver uma extensão da classe `Grounding` que contenha os elementos necessários para construir a consulta SPARQL que implementa o serviço e, consequentemente, as informações necessárias para executá-la.

O mapeamento do `Grounding` OWL-S para SPARQL especializa a camada abstrata composta pelas classes `Grounding` e `AtomicProcessGrounding`, ambas definidas pelo padrão OWL-S. A Figura 3.1 apresenta um diagrama de classes UML que exibe tal especialização.

Na Figura 3.1, os detalhes das ontologias `Service`, `Profile` e `Process` foram omitidos para evidenciar a especialização da ontologia `Grounding`. As classes `SparqlGrounding` e `SparqlAtomicProcessGrounding` na Figura 3.1 não fazem parte da ontologia `Grounding`. Essas classes são propostas neste trabalho com o objetivo de descrever a execução do SWoDS e respectivamente estendem as classes `Grounding` e `AtomicProcessGrounding`,

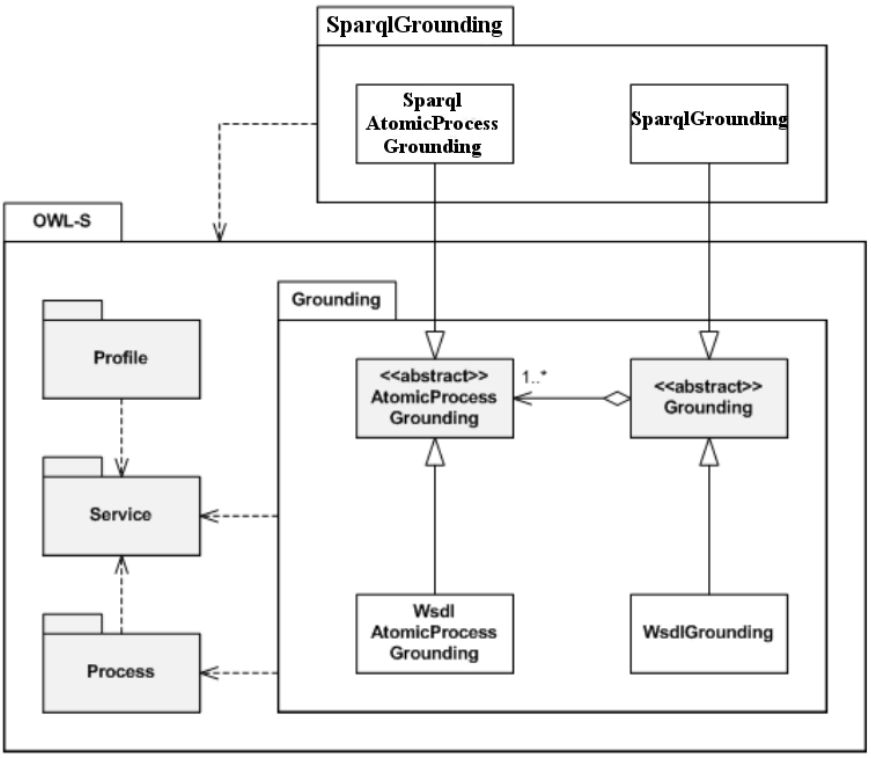

Figura 3.1: Modelo UML da extensão da classe `SparqlGrounding`.

mantendo a integração com os elementos do OWL-S.

O modelo de classes apresentado na Figura 3.1 propõe o agrupamento das novas classes em uma ontologia dedicada, chamada `SparqlGrounding`. Essa abordagem evita qualquer modificação na especificação OWL-S, o que facilitará a adoção da nova ontologia, sem interferir em serviços descritos com OWL-S anteriormente à incorporação da `SparqlGrounding`.

A classe `SparqlAtomicProcessGrounding`, que representa a execução de um processo atômico de um serviço OWL-S, deve implementar os mapeamentos necessários para construção e execução da consulta SPARQL que representa o serviço. Portanto, no domínio desta classe são definidos os elementos capazes de descrever uma consulta SPARQL, tal como prefixos, elementos selecionados e triplas na cláusula WHERE.

A Figura 3.2 apresenta esse mapeamento utilizando uma consulta SPARQL de

exemplo para um serviço que tem como entrada o ISBN (`?var_isbn`) e a saída um recurso livro (`?var_book`). Note, que são definidos para cada tipo de estrutura da consulta SPARQL elementos no `SparqlAtomicProcessGrounding`. Para os prefixos foram definidos a propriedade `SparqlPrefixes` e para as triplas a propriedade `SparqlTriples`. As entradas e saídas são respectivamente mapeadas pelas propriedades `SparqlinputParam` e `SparqlOutputParam`, que armazenam tanto elementos da consulta SPARQL, como as classes da sub-ontologia `Process` referentes a entrada e saída. A associação com classes do `Process` garante a conexão dos dados de entrada e saída com suas respectivas descrições semânticas.

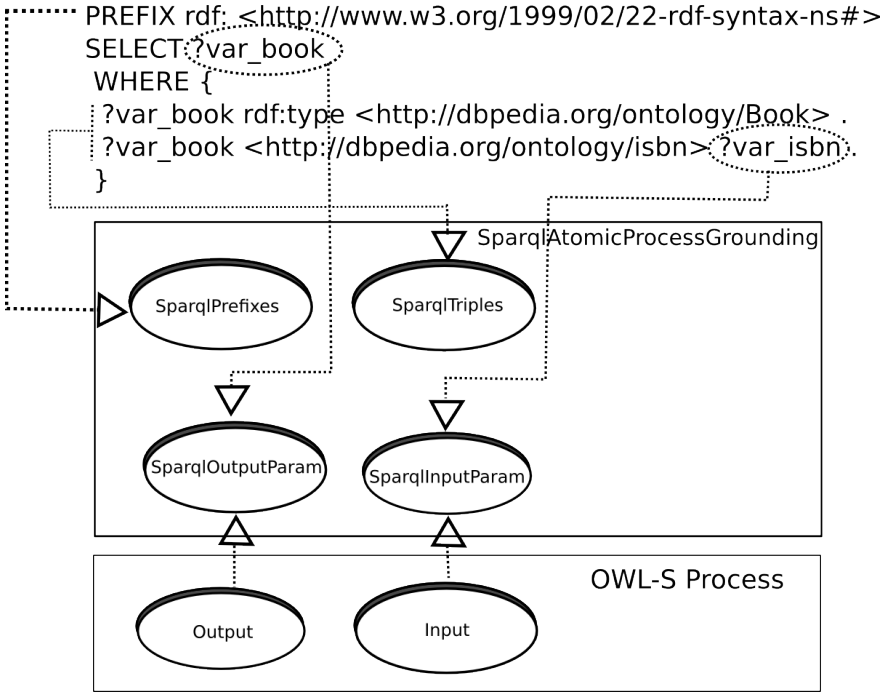

Figura 3.2: Demonstração do mapeamento de uma consulta SPARQL nos elementos do `AtomicProcessGrounding`.

Para a formalização da `SparqlGrounding` é necessário fazer a descrição de todos os seus elementos seguindo a sintaxe da linguagem OWL. O Documento 3.1 descreve a formalização em OWL da classe `SparqlGrounding` como uma subclasse da classe `Grounding` previamente definida pela OWL-S. Note também que a definição possui res-

trições referentes à existência de `AtomicProcessGrounding` (linha 5 do Documento 3.1) e que todos esses devem ser do tipo `SparqlAtomicProcessGrounding` (linha 6 do Documento 3.1). Tais restrições garantem que a instância da classe `SparqlGrounding` terá uma ou mais subclasses, as quais serão do tipo `SparqlAtomicProcessGrounding`, garantindo assim a construção exclusiva para as classes do `SparqlGrounding`.

Note também no Documento 3.1 a utilização do prefixo `&grounding` nas linhas 02 e 05 para referenciar classes definidas na ontologia OWL-S `Grounding`. O cabeçalho do documento que possui esse trecho de código deve necessariamente definir o URI completo da ontologia referenciada. Referências precisas entre os documentos que formam uma ontologia são requisitos fundamentais para que um agente de *software* da Web Semântica possa automaticamente encontrar a definição de todas as classes utilizadas.

Documento 3.1: Classe `SparqlGrounding`.

```
1  <owl:Class rdf:ID="SparqlGrounding">
2    <rdfs:subClassOf rdf:resource="&grounding;Grounding"/>
3    <rdfs:subClassOf>
4      <owl:Restriction>
5        <owl:onProperty rdf:resource="&grounding;hasAtomicProcessGrounding
           "/>
6        <owl:allValuesFrom rdf:resource="#SparqlAtomicProcessGrounding"/>
7      </owl:Restriction>
8    </rdfs:subClassOf>
9    <rdfs:comment>
10       The class that grounds every process to a SPARQL query.
11   </rdfs:comment>
12 </owl:Class>
```

A classe `SparqlAtomicProcessGrounding`, exibida no Documento 3.2, é uma subclasse de `AtomicProcessGrounding`, que é uma classe abstrata. A classe `SparqlAtomicProcessGrounding` possui a restrição de uma propriedade de dados (`DatatypeProperty`) chamada `sparqlEndPoint` (linha 5), a qual armazena a URI do ponto de acesso que a consulta SPARQL deve ser submetida na execução do serviço. A propriedade `sparqlEndPoint` possui restrição de cardinalidade de uma única ocorrência (Documento 3.2 linha 6). Os demais elementos associados ao SWoDS são pertencentes à essa

classe, logo a `SparqlAtomicProcessGrounding` representa uma instância de execução atômica de um serviço SWoDS.

Documento 3.2: Classe SparqlAtomicProcessGrounding.

```
1  <owl:Class rdf:ID="SparqlAtomicProcessGrounding">
2    <rdfs:subClassOf rdf:resource="&grounding;AtomicProcessGrounding"/>
3    <rdfs:subClassOf>
4      <owl:Restriction>
5    <owl:onProperty rdf:resource="#sparqlEndPoint"/>
6    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
       owl:cardinality>
7      </owl:Restriction>
8    </rdfs:subClassOf>
9    <rdfs:comment>
10       The class that relates elements of an OWL-S atomic process to a
           SPARQL query.
11   </rdfs:comment>
12 </owl:Class>
```

O Documento 3.3 exibe a definição das propriedades `sparqlEndPoint` e `sparqlVersion`, que pertencem ao domínio da classe `SparqlAtomicProcessGrounding`. A `sparqlEndPoint` é definida como uma propriedade de dados e tem seu valor referente a uma URI (Documento 3.3 linha 6). A propriedade `sparqlVersion` também possui valor definido como URI (Documento 3.3 linha 15), porém por não ter nenhuma restrição de cardinalidade, o que torna seu uso opcional. No caso de ausência da definição `sparqlVersion` assume-se a versão atual do SPARQL.

Documento 3.3: Propriedades sparqlEndPoint e sparqlVersion.

```
1  <owl:DatatypeProperty rdf:ID="sparqlEndPoint">
2    <rdfs:comment>
3        A URI for a SPARQL endpoint that provides the operation to which
           this atomic process is grounded.
4    </rdfs:comment>
5    <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
6    <rdfs:range rdf:resource="&xsd;anyURI"/>
7  </owl:DatatypeProperty>
8
9  <owl:DatatypeProperty rdf:ID="sparqlVersion">
10   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
11   <rdfs:comment>
12       The URI that indicates the SPARQL version being used.
13   </rdfs:comment>
```

```
14    <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
15    <rdfs:range rdf:resource="&xsd;anyURI"/>
16  </owl:DatatypeProperty>
```

Uma consulta SPARQL pode possuir prefixos, que eventualmente são usados nas triplas, portanto, no `SparqlGrounding` é definida uma propriedade do tipo *ObjectProperty*, seguida da classe OWL `SparqlPrefixMap` (Documento 3.4 linhas 9 a 25), que define o mapeamento do nome do prefixo (`PrefixName`) para a URI associada (`PrefixUri`). O Documento 3.4 apresenta um trecho de tal definição, note que o `SparqlPrefixMap` define a existência de um (e somente um) elemento `PrefixName` e `PrefixUri`, garantindo assim a integridade para formação dos prefixos. O Documento 3.4 também define, nas linhas 1 a 7 *ObjectProperty* `SparqlPrefixes`, que irá conter o conjunto de prefixos definidos.

Documento 3.4: Propriedade SparqlPrefixes e classe SparqlPrefixMap.

```
1   <owl:ObjectProperty rdf:ID="SparqlPrefixes">
2     <rdfs:comment>
3         There should list the prefixies of sparql query.
4     </rdfs:comment>
5     <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
6     <rdfs:range rdf:resource="#SparqlPrefixMap"/>
7   </owl:ObjectProperty>
8
9   <owl:Class rdf:ID="SparqlPrefixMap">
10    <rdfs:subClassOf>
11      <owl:Restriction>
12        <owl:onProperty rdf:resource="#PrefixName"/>
13        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
              owl:cardinality>
14      </owl:Restriction>
15    </rdfs:subClassOf>
16    <rdfs:subClassOf>
17      <owl:Restriction>
18        <owl:onProperty rdf:resource="#PrefixUri"/>
19        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
              owl:cardinality>
20      </owl:Restriction>
21    </rdfs:subClassOf>
22    <rdfs:comment>
23        Build a map of each prefix.
24    </rdfs:comment>
25  </owl:Class>
```

O Documento 3.5 descreve a definição das triplas pertencentes à clásula `WHERE` das consultas SPARQL (como pode ser exemplificado na Figura 3.2), mapeando o sujeito, o predicado e o objeto das triplas, respectivamente representados pelas propriedades `TripleSubject` (Documento 3.5 linhas 34 a 37), `TriplePredicate` (Documento 3.5 linhas 39 a 42) e `TripleObject` (Documento 3.5 linhas 44 a 47). A `SparqlTripleMap` (Documento 3.5 linhas 9 a 31) é responsável por mapear as propriedades que compõem a tripla e possui uma restrição referente à quantidade de elementos que compõem a tripla a fim de garantir sua integridade sintática. O *ObjectProperty* `SparqlTriples` (Documento 3.5 linhas 1 a 7) possui a classe de mapeamento de tripla RDF `SparqlTripleMap`.

Documento 3.5: Elementos OWL para descrever triplas SPARQL.

```
1  <owl:ObjectProperty rdf:ID="SparqlTriples">
2    <rdfs:comment>
3        There should list the triples in where clauses of sparql query.
4    </rdfs:comment>
5    <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
6    <rdfs:range rdf:resource="#SparqlTripleMap"/>
7  </owl:ObjectProperty>
8
9  <owl:Class rdf:ID="SparqlTripleMap">
10   <rdfs:subClassOf>
11     <owl:Restriction>
12   <owl:onProperty rdf:resource="#TriplePredicate"/>
13   <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
       owl:cardinality>
14     </owl:Restriction>
15   </rdfs:subClassOf>
16   <rdfs:subClassOf>
17     <owl:Restriction>
18   <owl:onProperty rdf:resource="#TripleSubject"/>
19   <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
       owl:cardinality>
20     </owl:Restriction>
21   </rdfs:subClassOf>
22   <rdfs:subClassOf>
23     <owl:Restriction>
24   <owl:onProperty rdf:resource="#TripleObject"/>
25   <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
       owl:cardinality>
26     </owl:Restriction>
27   </rdfs:subClassOf>
28   <rdfs:comment>
29       Build map of part of each triple.
```

```
30    </rdfs:comment>
31  </owl:Class>
32
33
34  <owl:DatatypeProperty rdf:ID="TripleSubject">
35    <rdfs:domain rdf:resource="#SparqlTripleMap"/>
36    <rdfs:range rdf:resource="&xsd;string"/>
37  </owl:DatatypeProperty>
38
39  <owl:DatatypeProperty rdf:ID="TriplePredicate">
40    <rdfs:domain rdf:resource="#SparqlTripleMap"/>
41    <rdfs:range rdf:resource="&xsd;string"/>
42  </owl:DatatypeProperty>
43
44  <owl:DatatypeProperty rdf:ID="TripleObject">
45    <rdfs:domain rdf:resource="#SparqlTripleMap"/>
46    <rdfs:range rdf:resource="&xsd;string"/>
47  </owl:DatatypeProperty>
```

3.1.2 Mapeamento das Entradas e Saídas de OWL-S para Sparql-Grounding

Como demonstrado na Figura 3.2, os elementos de entrada e saída do `SparqlGrounding` estão associados aos dados da consulta SPARQL e às classes `Input` e `output` da sub-ontologia `Process`.

O Documento 3.6 apresenta a definição da entrada de dados do `SparqlGrounding`. A classe `SparqlIntputParamMap` (Documento 3.6, linhas 6 a 24) é responsável por aplicar o mapeamento entre o elemento do `Process` (`Input`), através da propriedade `owlsParameter` (Documento 3.6, linha 17), e o dado associado à entrada do serviço (`SparqlDataParam` no Documento 3.6, linha 11). Nos `Grounding` WSDL e WADL esse último elemento é usado para conectar o ID da entrada do documento sintático do serviço. Porém, no contexto de SWoDS ele é utilizado para indicar qual variável na consulta será mapeada aos dados de entrada, pois o SWoDS não possui documento sintático do serviço e sim uma consulta SPARQL, que deve ser relacionada com a descrição OWL-S. Já a classe `SparqlInputParam` (Documento 3.6, linhas 1 a 4) é pertencente ao `SparqlAtomicProcessGrounding` e envolve a classe `SparqlIntputParamMap`.

Ainda no Documento 3.6 a classe `SparqlDataParamMap` (Documento 3.6, linhas 26 a 28) e a propriedade `SparqlDataParam` (Documento 3.6, linhas 30 a 36) são definidas para relacionar o dado da consulta com a entrada do serviço. A classe `Sparql- DataParamMap` possui herança dupla das classes `InputMessageMap` (Documento 3.6, linha 8) e `SparqlDataParamMap` (Documento 3.6, linha 7), onde a classe `InputMessageMap` (Documento 3.7, linhas 12 a 21) é definida pela sub-ontologia `Grounding` do OWL-S e mapeia os elementos de entrada associados do `Process` (Documento 3.7, linhas 28 e 29). Na prática, a propriedade `SparqlDataParam` irá armazenar a variável da consulta SPARQL (declarada na cláusula WHERE) para completar o mapeamento na classe `SparqlInputParamMap` juntamente com o *Input* do `Process`.

Documento 3.6: Elementos OWL para descrever entrada do serviço SWoDS.

```
1  <owl:ObjectProperty rdf:ID="SparqlInputParam">
2    <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
3    <rdfs:range rdf:resource="#SparqlInputParamMap"/>
4  </owl:ObjectProperty>
5
6  <owl:Class rdf:ID="SparqlInputParamMap">
7    <rdfs:subClassOf rdf:resource="#SparqlDataParamMap"/>
8    <rdfs:subClassOf rdf:resource="&grounding;InputMessageMap"/>
9    <rdfs:subClassOf>
10     <owl:Restriction>
11   <owl:onProperty rdf:resource="#SparqlDataParam"/>
12   <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
       owl:cardinality>
13     </owl:Restriction>
14   </rdfs:subClassOf>
15   <rdfs:subClassOf>
16     <owl:Restriction>
17       <owl:onProperty rdf:resource="&grounding;owlsParameter"/>
18       <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
           owl:cardinality>
19     </owl:Restriction>
20   </rdfs:subClassOf>
21   <rdfs:comment>
22       The mapping entity that shows how to derive a Sparql entry
           response message parameter from an OWL-S parameter value.
23   </rdfs:comment>
24 </owl:Class>
25
26 <owl:Class rdf:ID="SparqlDataParamMap">
27     <rdfs:subClassOf rdf:resource="&grounding;MessageMap"/>
```

```
28 </owl:Class>
29
30 <owl:DatatypeProperty rdf:ID="SparqlDataParam">
31   <rdfs:comment>
32       Message mapping for variables of SPARQL query.
33   </rdfs:comment>
34   <rdfs:domain rdf:resource="#SparqlDataParamMap"/>
35   <rdfs:range rdf:resource="&xsd;string"/>
36 </owl:DatatypeProperty>
```

Documento 3.7: Elementos da linguagem OWL-S para mapeamento de entradas e saídas.

```
1  <owl:Class rdf:ID="MessageMap">
2      <rdfs:comment> A class used to map parameters in OWL-S to the
           parameters in the grounded operation. owlsParamater property is
           used to specify the OWL-S parameter.
3    </rdfs:comment>
4  </owl:Class>
5
6  <owl:ObjectProperty rdf:ID="owlsParameter">
7      <rdfs:comment>An input or output property of an atomic process.</
           rdfs:comment>
8      <rdfs:domain rdf:resource="#MessageMap"/>
9      <rdfs:range rdf:resource="http://www.daml.org/services/owl-s/1.2/
           Process.owl#Parameter"/>
10     </owl:ObjectProperty>
11
12 <owl:Class rdf:ID="InputMessageMap">
13     <rdfs:comment>A MessageMap that maps inputs to grounding
           specification</rdfs:comment>
14     <rdfs:subClassOf rdf:resource="#MessageMap"/>
15     <rdfs:subClassOf>
16         <owl:Restriction>
17             <owl:onProperty rdf:resource="#owlsParameter"/>
18             <owl:allValuesFrom rdf:resource="http://www.daml.org/
                   services/owl-s/1.2/Process.owl#Input"/>
19         </owl:Restriction>
20     </rdfs:subClassOf>
21 </owl:Class>
22
23 <owl:Class rdf:ID="OutputMessageMap">
24     <rdfs:comment>A MessageMap that maps outputs to grounding
           specification</rdfs:comment>
25     <rdfs:subClassOf rdf:resource="#MessageMap"/>
26     <rdfs:subClassOf>
27         <owl:Restriction>
28             <owl:onProperty rdf:resource="#owlsParameter"/>
29             <owl:allValuesFrom rdf:resource="http://www.daml.org/
                   services/owl-s/1.2/Process.owl#Output"/>
30         </owl:Restriction>
```

```
31      </rdfs:subClassOf>
32  </owl:Class>
```

A definição dos dados de saída é análoga à entrada, também utilizando as propriedades `SparqlDataParam` (Documento 3.6, linhas 30 a 36) e `owlsParameter` (Documento 3.7, linhas 6 a 10). O Documento 3.8 apresenta a definição do mapeamento da saída dos serviços SWoDS. O mapeamento é feito pelo elemento `SparqlOutputParam` (Documento 3.8, linhas 1 a 7), que, de modo equivalente à entrada, faz a associação dos dados de saída através da classe `SparqlOutputParamMap` (Documento 3.8, linhas 9 a 27). As definições referentes à saída (*Outputs*) do serviço funcionam de modo análogo, com a única diferença que as variáveis da consulta SPARQL relacionadas são declaradas na cláusula SELECT. Assim, a classe `SparqlOutputParamMap` possui também herança dupla das classes `OutputMessageMap` (Documento 3.8, linha 10) e `SparqlDataParamMap` (Documento 3.8, linha 11), onde a classe `OutputMessageMap` (Documento 3.7, linhas 23 a 32) mapeia os elementos de saída do `Process`.

Documento 3.8: Elementos OWL para descrever as saídas do serviço SWoDS.

```
1   <owl:ObjectProperty rdf:ID="SparqlOutputParam">
2     <rdfs:comment>
3         There should be one instance of this property for each SPARQL
              select parameter.
4     </rdfs:comment>
5     <rdfs:domain rdf:resource="#SparqlAtomicProcessGrounding"/>
6     <rdfs:range rdf:resource="#SparqlOutputParamMap"/>
7   </owl:ObjectProperty>
8
9   <owl:Class rdf:ID="SparqlOutputParamMap">
10    <rdfs:subClassOf rdf:resource="#SparqlDataParamMap"/>
11    <rdfs:subClassOf rdf:resource="&grounding;OutputMessageMap"/>
12    <rdfs:subClassOf>
13      <owl:Restriction>
14        <owl:onProperty rdf:resource="#SparqlDataParam"/>
15        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
              owl:cardinality>
16      </owl:Restriction>
17    </rdfs:subClassOf>
18    <rdfs:subClassOf>
19        <owl:Restriction>
```

```
20    <owl:onProperty rdf:resource="&grounding;owlsParameter"/>
21    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</
          owl:cardinality>
22        </owl:Restriction>
23    </rdfs:subClassOf>
24    <rdfs:comment>
25    The mapping entity that shows how to relate a SPARQL select variable
          parameter from an OWL-S parameter value.
26    </rdfs:comment>
27  </owl:Class>
```

3.1.3 Exemplo de um serviço descrito com a SparqlGrounding

Explanado os elementos que compõem o `SparqlGrounding`, é importante apresentar uma instância de um serviço descrito com SWoDS. O Documento 3.9 exibe um exemplo de consulta SPARQL, onde são selecionados recursos referentes a livros (linha 2 do Documento 3.9) dado como entrada informações referentes ao ISBN (`?var_isbn` linha 5 do Documento 3.9). No Apêndice A o Documento A.1 apresenta o código completo (incluindo a instâncias de todas sub-ontologias do serviço) do serviço SWoDS referente à consulta SPARQL do Documento 3.9. Note que no `Profile` (Documento A.1, linhas 30 a 41) são declaradas a entrada `ISBN` (Documento A.1, linha 38) e saída `BOOK` (Documento A.1, linhas 39), que por sua vez são definidas no `Process` (Documento A.1, linhas 50 e 55) com tipo equivalente aos utilizados na consulta SPARQL. Na instância do `SparqlAtomicProcessGrounding` nota-se a referência ao *endpoint* do DBpedia, onde a consulta deve ser executada (propriedade `sparqlEndPoint`, Documento A.1, linha 72). Além disso, no `SparqlAtomicProcessGrounding` estão definidos os relacionamentos das entradas e saídas através dos elementos `SparqlInputParam` (Documento A.1, linhas 86 a 91) e `SparqlOutputParam` (Documento A.1, linhas 95 a 100), que respectivamente mapeiam o elemento do `Process` com a variável da consulta (com os elementos `grounding:owlsParameter` e `SparqlDataParam`). Por fim, as triplas da consulta SPARQL são definidas nas classes `SparqlTriples` (Documento A.1, linhas 104 a 118).

Portanto, através do `SparqlGrounding` é possível construir a consulta SPARQL, executá-la e relacioná-la com os elementos semânticos da ontologia OWL-S. Para a concretização dessa proposta, foi necessário desenvolver uma extensão da OWL-S API, que é descrita na Seção 3.2.

Documento 3.9: Exemplo de consulta SPARQL para geração de serviço SWoDS.

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 SELECT ?var_book
3   WHERE {
4     ?var_book rdf:type <http://dbpedia.org/ontology/Book> .
5     ?var_book <http://dbpedia.org/ontology/isbn> ?var_isbn .
6   }
```

3.2 Extensão da OWL-S API

Para implementação do `SparqlGrounding`, que possibilita a descrição da execução de SWoDS e a avaliação apresentada no Capítulo 5, foi necessário estender a OWL-S API, que é a principal biblioteca usada na fundamentação de todo o ciclo de vida dos serviços OWL-S. Dessa forma, a extensão foi desenvolvida a partir da versão estável (3.1) disponível no site oficial do projeto OSIRIS Next[1], do qual a OWL-S API faz parte.

No processo de desenvolvimento da extensão da OWL-API, para suporte ao SWoDS, levantou-se os principais requisitos que deveriam ser atendidos pela implementação. Os requisitos foram levantados com base nas funcionalidades necessárias para atingir os objetivos da proposta do SWoDS, que se concentra na possibilidade de descobrir Serviço Web na Web de Dados e permitir composição com outros tipos de serviços OWL-S. Deste modo, os principais requisitos são:

- **Estar em consonância com o sistema legado da OWL-S API.** Esse aspecto

[1] http://on.cs.unibas.ch/

é fundamental para o correto funcionamento da extensão, principalmente para a realização de operações de composição. A OWL-S API é projetada para ser estendida para novas implementações, portanto deve-se utilizar as interfaces e aplicar as especializações de acordo com o proposto pela API.

- **Prover a descoberta de SWoDS.** A descoberta automática de SWoD pode ocorrer de duas formas. Através de repositório de serviços (utilizando meios tradicionais praticados por serviços OWL-S), ou através da geração automática de consultas SPARQL, a partir de requisições de Serviços Web (discutido no Capítulo 4).

- **Prover a composição de SWoDS com outros serviços OWL-S.** A composição de SWoDS é uma das principais motivações da presente pesquisa, logo é indispensável que a extensão da API possibilite integração com outros tipos de serviços já suportados pela OWL-S.

- **Permitir a execução de SWoDS.** Para fins de experimentação e avaliação do SWoDS é importante que a extensão da API seja capaz de executar SWoDS de modo que torne possível a experimentação com utilização de dados reais das bases *Linked Data*.

Como pode ser visto na Figura 3.3, a OWL-S API foi projetada para permitir a extensão de novos `Grounding`, que por sua vez devem implementar certas interfaces específicas a fim de conectar a nova implementação com o restante da biblioteca. As classes referentes à nova implementação são divididas em dois pacotes: `br.ufba.dcc.linked_data` e `br.ufba.dcc.linked_data.impl`, que, respectivamente, implementam as interfaces e funcionalidades da extensão. O primeiro deles é responsável por declarar as classes abstratas que serão implementadas pelo segundo pacote. Além disso, as classes do pacote `br.ufba.dcc.linked_data` estendem as classes referentes ao `Grounding` definido na API.

Ainda na Figura 3.3, a classe `OWLSSparqlVocabulary` associa os vocábulos pertencentes ao `SparqlGrounding` a constantes que são utilizadas pelas demais classes. A interface `SparqlGroundingProvider` e a classe `SparqlGroundingProviderImpl` são responsáveis pela fundamentação do `SparqlGrounding` na API. Ou seja, aplicam a conversão dos elementos do `Grounding` para as classes da API previamente implementadas, utilizando outras classes (`SparqlPrefixesImpl`, `SparqlTriplesImpl` e `SparqlGroundingImpl`) que definem os elementos e propriedades do modelo. A classe `SparqlPrefixesImpl` é responsável por implementar os elementos para manipulação (inserir, remover, obter, etc) dos prefixos definidos no `SparqlGrounding`. A classe `SparqlTriplesImpl` é responsável pelo gerenciamento das triplas.

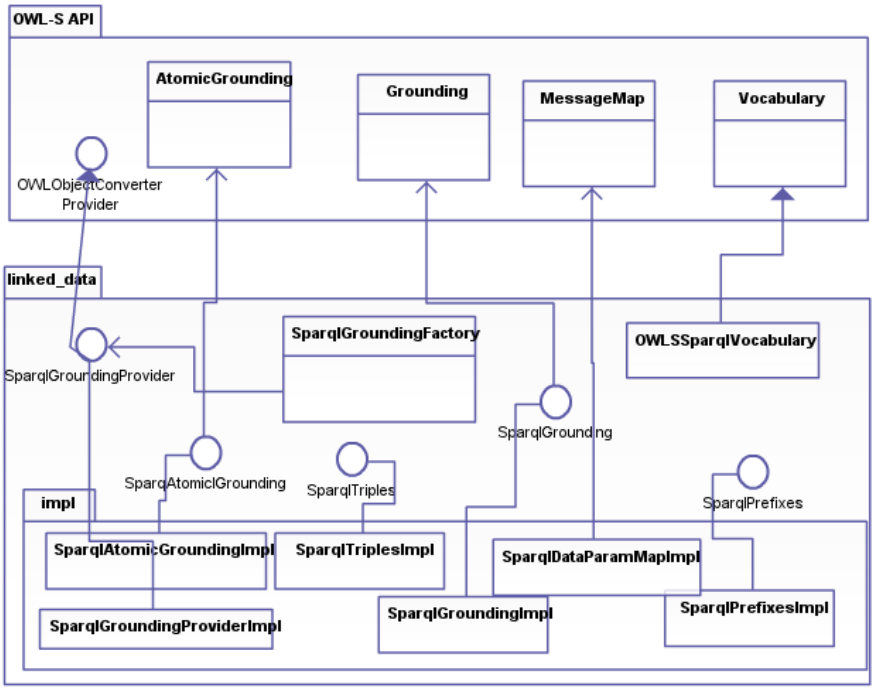

Figura 3.3: Diagrama de classe da OWL-S API estendida.

A classe `SparqlGroundingProviderImpl` é responsável pela conexão do `Process` com o `SparqlGrounding`. É através dela que o SWoDS é executado, onde é acionado o método `registerConverters`, descrito no Documento 3.10. Nesse método são conver-

tidos os objetos para introduzir a execução do `SparqlGrounding`, ou para criação do `SparqlGrounding` utilizando a OWL-S API. Cada um dos métodos do Documento 3.10 implementam conversão das classes genéricas do OWL-S `Grounding` às classes definidas no `SparqlGrounding`.

Documento 3.10: Trecho de código do método registerConverters da classe SparqlGroundingProviderImpl.

```
1  public void registerConverters(final OWLObjectConverterRegistry registry
       ){
2    final OWLObjectConverter<SparqlGrounding> gc = new GenericOWLConverter
         <SparqlGrounding>(
3      SparqlGroundingImpl.class, OWLSSparqlVocabulary.SparqlGrounding);
4
5    final OWLObjectConverter<SparqlAtomicGrounding> agc = new
         GenericOWLConverter<SparqlAtomicGrounding>(
6      SparqlAtomicGroundingImpl.class, OWLSSparqlVocabulary.
           SparqlAtomicProcessGrounding);
7
8    final OWLObjectConverter<SparqlDataParamMapImpl> immc =
9      new GenericOWLConverter<SparqlDataParamMapImpl>(
           SparqlDataParamMapImpl.class,
10         OWLSSparqlVocabulary.SparqlInputParamMap);
11
12   final OWLObjectConverter<SparqlDataParamMapImpl> ommc =
13     new GenericOWLConverter<SparqlDataParamMapImpl>(
           SparqlDataParamMapImpl.class,
14         OWLSSparqlVocabulary.SparqlOutputParamMap);
15
16   registry.registerConverter(SparqlPrefixes.class,
17     new GenericOWLConverter<SparqlPrefixes>(SparqlPrefixesImpl.class,
           OWLSSparqlVocabulary.SparqlPrefixMap));
18
19   registry.registerConverter(SparqlTriples.class,
20       new GenericOWLConverter<SparqlTriples>(SparqlTriplesImpl.class,
             OWLSSparqlVocabulary.SparqlTripleMap));
21
22 ...
23   }
```

A classe `SparqlAtomicGroundingImpl` (ver Figura 3.3) é a responsável pela execução de fato do serviço, através do método `invoke()` (Descrito no Documento 3.11). Esse método, com o auxílio de outros métodos da classe, tem a função de preparar e invocar o serviço. A Figura 3.4 ilustra os principais métodos da classe `SparqlAtomicGroundingImpl`.

Esses métodos, em sua maioria, são responsáveis em construir a consulta SPARQL. O `buildPrefixes` é responsável por extrair os dados referentes aos prefixos e gerá-los no formato utilizado pelas consultas SPARQL, utilizando de modo auxiliar o método `getPrefixies`. O método `buildTriples`, juntamente com o método `getTriples`, tem como funcionalidade a construção das triplas da consulta SPARQL e método `buildSelectStatement` constrói a cláusula SELECT da consulta.

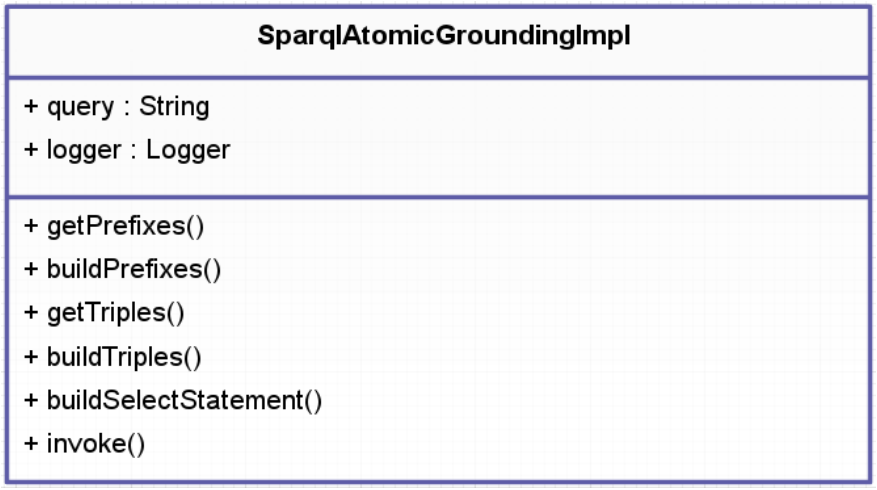

Figura 3.4: Diagrama com os métodos da classe SparqlAtomicGroundingImpl.

O Documento 3.11 apresenta o método `invoke()` da classe `SparqlAtomicGroundingImpl`. Esse método, inicialmente, invoca os métodos `buildPrexies`, `buildSelectStatement` e `buildTriples` para construção de parte da consulta SPARQL a ser executada (Documento 3.11, linhas 4 a 6). Em seguida, é executado um trecho de código para inserir os dados passados como entrada durante a execução do serviço nas triplas (Documento 3.11, linhas 9 a 25). Então, é concluída a construção da consulta SPARQL (Documento 3.11, linhas 28 a 36) e segue a consulta resultante para execução no SPARQL *endpoint* definido no documento (Documento 3.11, linhas 37 a 40). Por fim, o resultado é associado a cada saída definida no serviço (Documento 3.11, linhas 43 a 56) para finalmente retornar os dados para o usuário.

Documento 3.11: Trecho do código do método invoke() da classe SparqlAtomicGroundingImpl.

```
1  public ValueMap<Output, OWLValue> invoke(ValueMap<Input, OWLValue>
       inputs,  OWLKnowledgeBase env)
2      throws ExecutionException {
3
4    prefixes = buildPrexies();
5    select = buildSelectStatement();
6    l_triples = buildTriples();
7
8    //codigo para relacionar a entrada com as variaveis do SPARQL
9    Iterator<Entry<Input, OWLValue>> i = inputs.iterator();
10   while(i.hasNext()){
11     Entry<Input, OWLValue> input = i.next();
12     for (final Input inputParam : getProcess().getInputs()){
13       final MessageMap<String> mp = getMessageMap(inputParam);
14       final String var = mp.getGroundingParameter();
15       Iterator<Vector<String>> t = l_triples.iterator();
16       while(t.hasNext()){
17       Vector<String> triple = t.next();
18       for(int j=0; j<3; j++){
19         if (triple.get(j).contentEquals(var)){
20           triple.set(j, input.getValue().toString());
21         }
22       }
23   }
24     }
25   }
26
27   // construcao da consulta SPARQL e execucao
28   query = prefixes.toString();
29   query = query + select.toString();
30   query = query + "WHERE␣{\n";
31   Iterator<Vector<String>> t = l_triples.iterator();
32   while(t.hasNext()){
33     Vector<String> triple = t.next();
34     query = query + triple.get(0) + "␣" + triple.get(1) + "␣" + triple.
         get(2) + "␣.\n";
35   }
36   query = query + "}";
37   Query sparqlQuery = QueryFactory.create(query);
38   QueryExecution qexec = QueryExecutionFactory.sparqlService(
         getSparqlEndPoint().toString(), sparqlQuery);
39   ResultSet result = qexec.execSelect() ;
40   qexec.close() ;
41
42   //construcao da saida
43   final ValueMap<Output, OWLValue> results = new ValueMap<Output,
         OWLValue>();
44   for (final Output outputParam : getProcess().getOutputs()){
45     ResultSet tempResult = result;
```

```
46      final MessageMap<String> mp = getMessageMap(outputParam);
47      final String outputVar = mp.getGroundingParameter();
48      System.out.println("output: " + outputVar );
49      while(tempResult.hasNext()){
50        QuerySolution r = tempResult.next();
51        Object outputValue = r.get(outputVar);
52
53        if (outputParam.getParamType().isDataType())
54          results.setValue(outputParam, env.createDataValue(outputValue));
55        else
56          results.setValue(outputParam, env.parseLiteral(outputValue.
              toString()));
57      }
58
59    return results;
60  }
```

3.3 Considerações Finais

A proposta de definição do *Semantic Web (of Data) Service* (SWoDS) visa desenvolver Serviços Web Semânticos a partir de informações extraídas da Web de Dados. A descrição do SWoDS é desenvolvida com a linguagem OWL-S, através da proposta da extensão da sub-ontologia de concretização do Serviço Web, nomeada pelo autor de `SparqlGrounding`.

O desenvolvimento do `SparqlGrounding` consiste na definição de uma ontologia que dá suporte ao concretizar serviços SWoDS. Assim, a extensão da OWL-S API para suporte ao SWoDS permite a real execução de tais serviços viabilizando sua utilização em composições e descobertas de Serviços Web.

Capítulo 4

SWoDS: Requisição, Descoberta e Execução

Durante o desenvolvimento do SWoDS, esta pesquisa identificou a possibilidade de gerar Serviços Web (do tipo SWoDS) de modo automático a partir de requisições de Serviços Web descritos em OWL-S. Ou seja, a partir da descrição de uma requisição de serviço OWL-S (com as sub-ontologias `Process` e `Profile`) é possível, em alguns casos, construir automaticamente consultas SPARQL e consequentemente serviços SWoDS. Para outros casos a construção da consulta pode ser realizada de forma semi-automática. Entretanto, a geração semi-automática não foi tratada neste trabalho. Além disso, este trabalho não tem o objetivo de definir todos os casos em que se é possível ou não a geração automática das consultas SPARQL.

Assim, a Seção 4.1 apresenta o módulo de descoberta de SWoDS através da geração automática de consultas SPARQL a partir de requisições OWL-S, cujo é o primeiro passo para geração SWoDS. Isso possibilita que agentes de *software* baseados em serviços consigam acessar dados *Linked Data* de forma automática. Além disso, a Seção 4.2 apresenta a execução automática de SWoDS, sejam eles gerados através da descoberta automática, ou através de processos manuais.

4.1 Requisição e descoberta automáticas

As consultas SPARQL estão para bases *Linked Data* assim como o SQL está para a maioria dos bancos de dados relacionais. Desse modo, para explorar a Web de Dados é preciso utilizá-las, o que torna indispensável, para os objetivos deste trabalho, a necessidade de fazer uma tradução da semântica descrita pela linguagem OWL-S para uma consulta SPARQL. O SWoDS, descrito no Capítulo 3, formaliza a maneira de realizar essa tradução semântica, que pode ser manual, semi-automática (com apoio de ferramentas de edição OWL-S/SPARQL) ou automática. Esta Seção apresenta algumas possibilidades de tradução automática de OWL-S para SPARQL, e consequentemente SWoDs. Tal contribuição foi denominada por esta pesquisa de módulo de descoberta automática de SWoDS.

A Figura 4.1 apresenta uma visão geral do módulo de descoberta automática de SWoDS a partir de requisições de serviços descritas em OWL-S. A partir da requisição do serviço é desenvolvida uma consulta SPARQL do tipo *ASK*[1], gerada a partir das ontologias das entradas e saídas (Figura 4.1 passo 1). Optou-se por esse tipo consulta, pois ela tem um menor custo de execução, dado que, nessa etapa, não se deseja retornar dados e sim validar a consulta. A geração automática da consulta SPARQL é desenvolvida a partir de informações referentes às entradas e saídas do serviço obtidas na sub-ontologia de processos (Process).

Após isso, o próximo passo é a aplicação da consulta em um SPARQL *endpoint* para verificar se existem dados que correspondem à solicitação (Figura 4.1, passo 2).Assim, a execução da consulta SPARQL *endpoint* retornará um valor booleano, indicando a existência, ou não de dados que atendem a consulta (Figura 4.1, passo 3). Em caso de positivo (Figura 4.1, passo 4), pode-se criar um serviço OWL-S e utilizar essa base de dados para as respostas, isto é, pode-se seguir para implementação do `SparqlGrounding`

[1]Retorna *true* ou *false* se, respectivamente, existir ou não um ou mais dados de acordo com a requisição.

(descrito na Seção 3.1). O novo serviço SWoDS criado é armazenado em um repositório de Serviços Web, que pode ser consultado previamente, reduzindo o tempo de descoberta e geração do serviço. Caso o resultado da consulta retorne *false* (Figura 4.1, passo 5) indica que não existem conjuntos de dados capazes de atender esta solicitação. É possível ainda executar a consulta em outros *endpoints* buscando obter alguma fonte de dados compatível. Assim, pode-se desenvolver serviços SWoDS sob demanda de requisições.

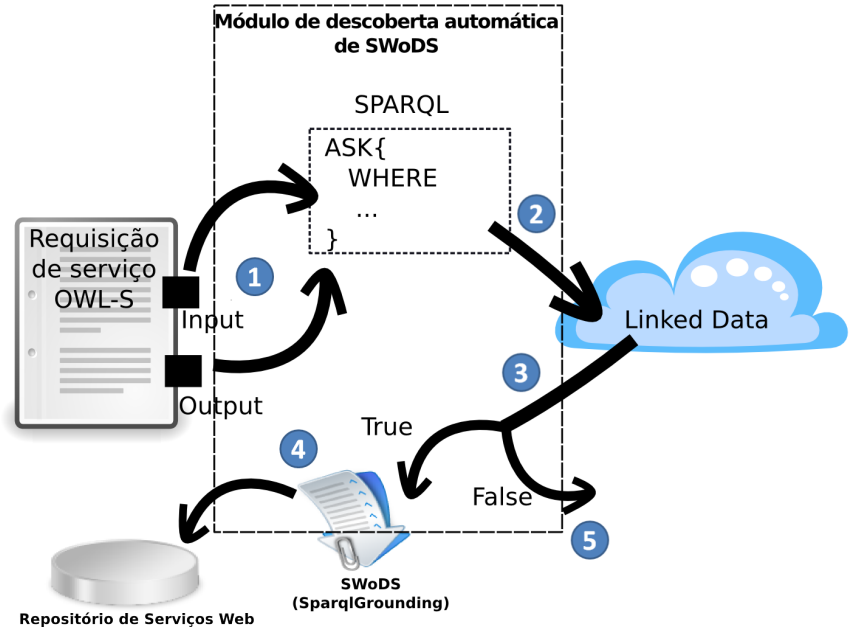

Figura 4.1: Módulo de geração automática de SWoDS a partir de requisições de serviço OWL-S.

Existem alguns desafios associados a esse mapeamento OWL-S para SPARQL apresentado na Figura 4.1. Inicialmente, a expressividade semântica da linguagem OWL-S é superior à capacidade de expressão da linguagem SPARQL. Isso acarreta a necessidade de identificar quais tipos de requisições estão habilitadas para realização automática da expressão de consulta, pois nem todas requisições de serviço OWL-S podem ser convertidas automaticamente para consultas SPARQL, seja pela falta de expressividade da linguagem de consulta *Linked Data*, ou pela falta de elementos que permitam a

inferência. Não faz parte do escopo desta dissertação uma avaliação precisa de quais requisições de serviço OWL-S podem ser convertidas automaticamente em consultas SPARQL.

Assim, para uma avaliação experimental, foi selecionado um perfil de requisição OWL-S simples, que permita a geração de consultas SPARQL. Dessa forma, foram selecionadas as solicitações de serviços onde o elemento de entrada é pertencente (é uma propriedade) do elemento de saída (ou o contrário). Por exemplo, uma requisição de serviço, em que a entrada é a latitude e a longitude de uma cidade e a saída é um recurso que representa a cidade. Nesse caso, a cidade é o recurso de saída que possui como propriedades os dados referentes a latitude e longitude, os quais são elementos de entrada.

A Figura 4.2 apresenta a descrição da requisição de um serviço (parte A da Figura 4.2), que tem como entrada o `ISBN` de um livro e a saída o recurso `Book`. As partes A e B da Figura 4.2 exibem trechos da sub-ontologia *Process*, os quais descrevem a definição dos tipos (*rdf:datatype*) dos recursos de entrada e saída. Através da descrição ontológica (parte B da Figura 4.2) é possível ver que `ISBN` (entrada) faz parte do domínio de `Book` (saída). Assim, esse tipo de serviço permite gerar a consulta SPARQL semelhante à consulta exibida na parte C da Figura 4.2.

A) Descrição do Process do serviço OWL-S

```
<process:Input rdf:ID="ISBN">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI">http://dbpedia.org/ontology/isbn</
process:parameterType>
<rdfs:label/>
</process:Input>

<process:Output rdf:ID="BOOK">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI">http://dbpedia.org/ontology/Book</
process:parameterType><rdfs:label/>
</process:Output>
```

B) RDF da ontologia http://dbpedia.org/ontology/Book

```
<rdf:RDF>
 ...
  <rdf:Description rdf:about="http://dbpedia.org/ontology/isbn">
<rdfs:domain rdf:resource="http://dbpedia.org/ontology/Book" />
  </rdf:Description>
  <rdf:Description rdf:about="http://dbpedia.org/ontology/
numberOfPages">   <rdfs:domain rdf:resource="http://dbpedia.org/
ontology/Book" />  </rdf:Description>
...
</rdf:RDF>
```

C) Consulta SPARQL gerada

```
ASK
WHERE{
?varbook rdf:type <http://dbpedia.org/ontology/Book>
?varbook <nhttp://dbpedia.org/ontology/isb> ?varisbn
}
```

Figura 4.2: Mapeamento de requisições de serviço OWL-S para geração de consulta SPARQL.

É possível mapear outros tipos de consultas, utilizando técnicas de similaridade de ontologias, ou até mesmo através de inferências que permitam relacionar os recursos de entrada com os de saída. Também é possível gerar consultas que atendam parcialmente a requisições, onde o serviço gerado poderia posteriormente ser composto com outros serviços para atender a requisição inicial. Entretanto, o mapeamento de consultas não é escopo deste trabalho, e sim a demonstração de que é possível extrair Serviços Web Semânticos em bases *Linked Data*.

É importante destacar que, dentro dos requisitos propostos, o SWoDs permite a execução de qualquer serviço descrito com uma consulta SPARQL, ou seja, não se restringe à execução de serviços com consultas geradas automaticamente. Assim, caso um desenvolvedor projete uma consulta SPARQL manualmente, ou de forma semi-automática, e queira defini-la como um serviço OWL-S, ele pode implementar o `SparqlGrounding` para tal consulta.

A Seção 4.2 discute a geração automática de serviço SWoDS a partir da consulta SPARQL mapeada por uma requisição de serviço OWL-S até os passos que antecedem a execução de fato.

4.2 Execução Automática

A Figura 4.3 ilustra uma visão geral da execução do SWoDS. O ponto de partida é a requisição OWL-S, onde, para construção do SWoDS, pode-se seguir por dois caminhos. Primeiro, o SWoDS pode ser desenvolvido através do módulo de descoberta automática (passo 1 da Figura 4.3). Nesta etapa, nem todas as requisições de serviço produzirão consultas SPARQL, pois nem todas elas referem-se a informações disponíveis na Web de Dados (esta questão foi tratada na Seção 4.1). Desta forma, ainda é possível desenvolver o `SparqlGrounding`, consequentemente o SWoDS (passo 2 da Figura 4.3), através de um processo manual (apresentado no Capítulo 3).

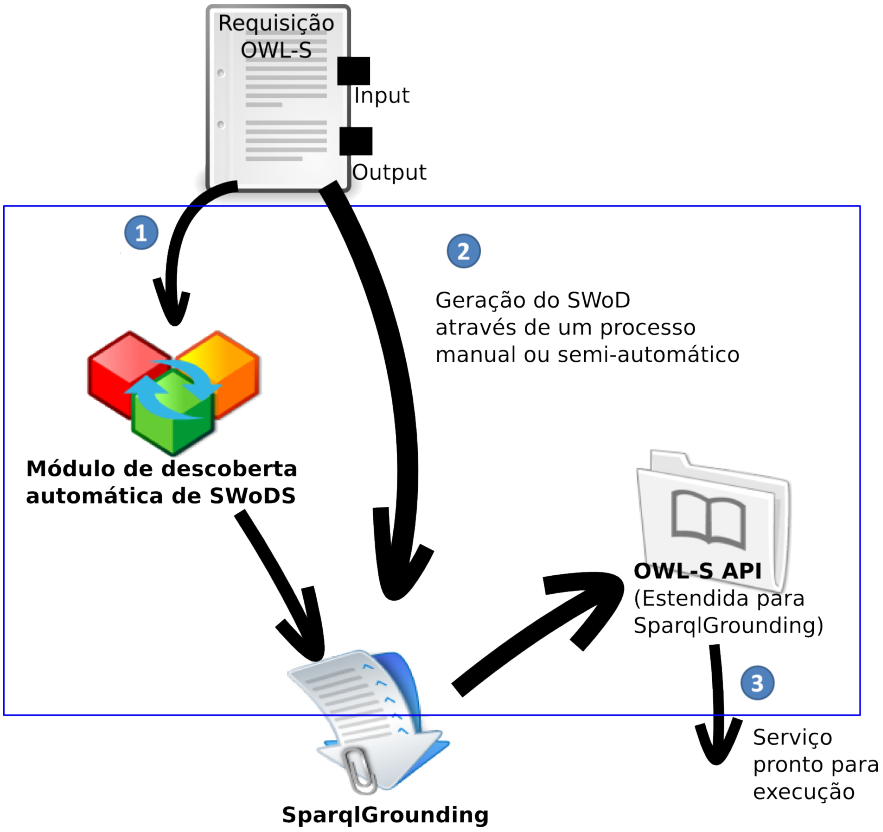

Figura 4.3: Visão geral da execução do SWoDS.

O `Grounding` resultante passa para a OWL-S API (passo 3 da Figura 4.3), que é uma biblioteca que implementa todo ciclo de vida de um serviço OWL-S, desde sua criação, descoberta, composição e execução. Todavia, foi necessário adaptar essa API para dar suporte ao `SparqlGrounding`, tornando o serviço disponível para execução (essa questão é discutida em detalhes na Seção 3.2). Assim, o serviço encontra-se pronto para execução de fato, a qual é realizada com o suporte da OWL-S API.

O Documento 4.1 exibe um trecho de código para execução de um serviço que utiliza o `SparqlGrounding`, que é equivalente ao passo 3 da Figura 4.3. Inicialmente, o serviço é carregado na base de conhecimento (linha 3 do Documento 4.1) para possibilitar o acesso ao `Process`, que, através dele, indica-se os dados de entrada (linhas 5 a 7 do Documento 4.1). Por fim, na linha 12 do Documento 4.1, o serviço é de fato executado, em

que, a partir de `Process`, é invocado a classe `SparqlGroundingProvider`, que iniciará toda a fundamentação das classes `SparqlGrounding` implementadas pelo serviço.

Documento 4.1: Trecho de código com um exemplo de execução do SWoDS.

```
1  // Carregando ontologias que descrevem o servi o
2  URI uri = new URI("http://localhost/services/isbn_book_sparql_grounding.
       owls");
3  Service service = kb.readService(uri);
4
5  Process process = service.getProcess();
6  // Definindo entradas
7  ValueMap<Input, OWLValue> inputs = new ValueMap<Input, OWLValue>();
8  inputs.setValue(process.getInput("ISBN"), kb.createDataValue("
       0-375-50137-1"));
9
10 // Criando conjunto de sa das e executando processo
11 ValueMap<Output, OWLValue> outputs = new ValueMap<Output, OWLValue>();
12 outputs = exec.execute(process, inputs, kb);
```

4.3 Considerações Finais

Na Seção 4.1 é apresentado o módulo de descoberta automática de SWoDS a partir de requisições de serviços descritas em OWL-S. Neste estudo, é possível perceber que o processo de geração de SWoDS, apesar de possuir a via de construção manual, adquire importantes oportunidades na sua descoberta automática. Isto porque a descoberta automática de SWoDS pode ser realizada em tempo de execução, em contextos mais complexos (envolvendo composições e sub-requisições de serviços), onde é possível preencher eventuais lacunas com dados oriundos da Web de Dados. A Seção 4.2 relata o processo de execução automática de SWoDS junto com a extensão da OWL-S API. Esse processo é fundamental para a concretização de fato dos SWoDS.

No Capítulo 5 são apresentadas as avaliações referentes às contribuições propostas com este trabalho. Além disso, na Seção 5.4 descreve um cenário de uso do SWoDS com composições de outros Serviços Web, o qual envolve um processo de descoberta automática.

Capítulo 5

Avaliação

O desenvolvimento do SWoDS tem como objetivo prover Serviços Web a partir da Web de Dados, utilizando as bases *Linked Data* e viabilizando a composição com outros Serviços Web. Como desdobramento desse objetivo, a presente pesquisa desenvolveu uma extensão da sub-ontologia de concretização da OWL-S cunhada de `SparqlGrounding` e uma extensão da OWL-S API para execução e manipulação do `SparqlGrounding`. Além disso, foi desenvolvido um módulo para descoberta automática de SWoDS a partir de requisições de serviços em OWL-S.

Para avaliação de tais contribuições, o presente trabalho desenvolveu experimentos, os quais visam abordar os diversos aspectos propostos nesta pesquisa. A Seção 5.1 relata as principais informações referentes a configuração do experimentos, citando em quais condições foram realizados. A Seção 5.2 aplica uma avaliação de desempenho do SWoDS em comparativo com serviços WSDL descritos com OWL-S, observando a avaliação da implementação da extensão da OWL-S API e, consequentemente, a ontologia `SparqlGrounding`. A Seção 5.3 analisa a descoberta automática de SWoDS a partir de requisições a serviços OWL-S, através de medições de desempenho. Por fim, a Seção 5.4 aplica um cenário de integração de Serviços Web descritos com OWL-S para avaliar a composição do SWoDS com outras arquiteturas de serviços e a descoberta

automática de SWoDS no contexto da composição.

5.1 Configuração dos experimentos

Para o desenvolvimento das avaliações foi utilizado um computador Intel Core i5 com 4 núcleos de 1,80GHz, 6GB de memória RAM, com o sistema operacional Debian/Linux 8.0 64-bits, na plataforma J2SE 1.7 e Eclipse 3.8.1. Os experimentos executados tem o objetivo de avaliar o desempenho das contribuições, seu correto funcionamento e a simulação de cenários para observação da aplicabilidade das propostas. Também foi utilizado um servidor Web Apache para acesso e armazenamento dos Serviços Web, além de utilizar dados acessados da base *Linked Data* do DBPedia[1].

Os serviços aplicados nas execuções foram extraídos do pacote *OWL-S Test Collection*[2], que foram adaptados para utilizar a ontologia do DBPedia e atualizados para versão 1.2 do OWL-S (mesma da versão OWL-S API utilizada neste trabalho).

Nos experimentos onde foram executadas medições de desempenho foram feitas 30 execuções para cada teste, onde foi observado o tempo consumido em cada execução e calculada a média aritmética e desvio padrão. Essa escolha foi feita para obter uma maior consistência dos resultados de cada avaliação.

5.2 SparqlGrounding: ontologia e API

A avaliação da implementação do `SparqlGrounding` (ontologia e API) consistiu em execuções, as quais verificam a capacidade do `SparqlGrounding` dar suporte a Serviços Web oriundos de consultas SPARQL.

Na avaliação da ontologia `SparqlGrounding` foi utilizada a extensão da OWL-S API desenvolvida neste trabalho. Para avaliar o correto funcionamento da ontolo-

[1] http://www.dbpedia.org
[2] http://projects.semwebcentral.org/projects/owls-tc/

gia `SparqlGrounding`, foi preciso executar Serviços Web Semânticos que utilizam o `SparqlGrounding`, ou seja, foi necessário usar a OWL-S API com suporte aos SWoDS para execução dos serviços. Assim, ao realizar tal tarefa, foi avaliado tanto a ontologia `SparqlGrounding` quanto a OWL-S API estendida para o suporte dos SWoDS.

Para melhor organização, dividiu-se a avaliação do `SparqlGrounding` e da extensão da OWL-S API em duas partes. A primeira faz uma aferição de desempenho em comparação com o WSDLGrounding (apresentado na Seção 5.2.1). No segundo é feita uma avaliação qualitativa, onde através de um situação de serviço mais complexa é executado o `SparqlGrounding` (descrito na Seção 5.2.2).

5.2.1 Avaliação de desempenho

A presente pesquisa considerou como relevante as medições referentes ao tempo de execução consumido para o processamento dos dados até o instante que antecede a execução de fato do serviço, a qual é equivalente à execução da consulta SPARQL. Este trabalho optou por não avaliar a execução do serviço em si, pois ele foge do escopo das contribuições e possui seu tempo de execução associado ao desempenho do serviço em si, por exemplo, o desempenho do *endpoint* utilizado. Para haver um melhor parâmetro da qualidade do desempenho do `SparqlGrounding`, optou-se por comparar os resultados medidos com a implementação OWL-S para serviços WSDL/SOAP, o `WSDLGrouding` (descrito na Seção 2.1.2), a qual é bem estabelecida e é referência nas execuções aplicadas pelos mantenedores da OWL-S API [72].

O serviço utilizado para medições de desempenho foi extraído do *OWL-S Test Collection* e consiste em um serviço que requisita como entrada o ISBN de um livro e a saída o livro que contém o ISBN requisitado. Esse serviço foi adaptado em duas versões: uma para utilizar o `SparqlGrounding` (Documento A.1) e outra para utilizar WSDL-Grounding (utiliza as mesmas sub-ontologias de perfil e processos do Documento A.1, modificando somente os aspectos referentes ao Grounding).

Portanto, a fim de avaliar o desempenho de execução do `SparqlGrounding` de modo mais detalhado, foi medido o tempo de execução nos seguintes pontos:

- **Leitura das ontologias antes de iniciar a execução do `Process`**. Tal medição, consiste na leitura das ontologias preliminares, as quais dão suporte a execução do OWL-S, incluindo leitura das ontologias `Profile` e `Process`, que respectivamente definem a apresentação do serviço e como compõe sua execução. Nesta medição não é incluída a sub-ontologia de concretização (WSDLGrounding e `SparqlGrounding`);

- **Tempo para mapeamento e leitura das classes do `Grounding`**. Nesse aspecto é medido o tempo consumido para ler a ontologia definida para cada tipo de serviço: `WSDLGrouding` e `SparqlGrounding`;

- **Tempo de preparação para execução do serviço**. Nesse ponto é mensurado o tempo gasto para os processamentos internos da concretização do serviço até instantes antes à sua execução de fato. Isto é, após a leitura da ontologia de concretização (WSDLGrouding ou `SparqlGrounding`) a OWL-S API inicia o processo para execução do serviço como descrito na Seção 3.2. Esse processo é finalizado no método `invoke()` (Documento 3.11) antes das instruções para execução da consulta SPARQL para o `SparqlGrounding` e antes da execução do documento WSDL para o WSDLGrounding.

As Figuras 5.1, 5.2 e 5.3 exibem os resultados da avaliação do `SparqlGrounding` em comparativo com `WSDLGrounding`, nos itens de aferições relacionados.

A Figura 5.1 apresenta um gráfico com as medições do tempo de leitura das ontologias antes de iniciar a execução do `Process` nas execuções com `SparqlGrounding` e `WSDLGrounding`. É possível perceber que não há diferenças substanciais de desempenho nesse quesito. Isso reflete o fato de que as ontologias lidas previamente entre os serviços

não possuem diferenças de desempenho, demonstrando assim, que a descrição semântica deles é quase equivalente. Isso também é observado nas medidas estatísticas da média aritmética e desvio padrão, onde a o `SparqlGrounding` possui média de `9206,8 ms` e desvio padrão de `517,374` e o `WSDLGrounding` possui média igual a `9235,63 ms` e desvio padrão de `457,980`.

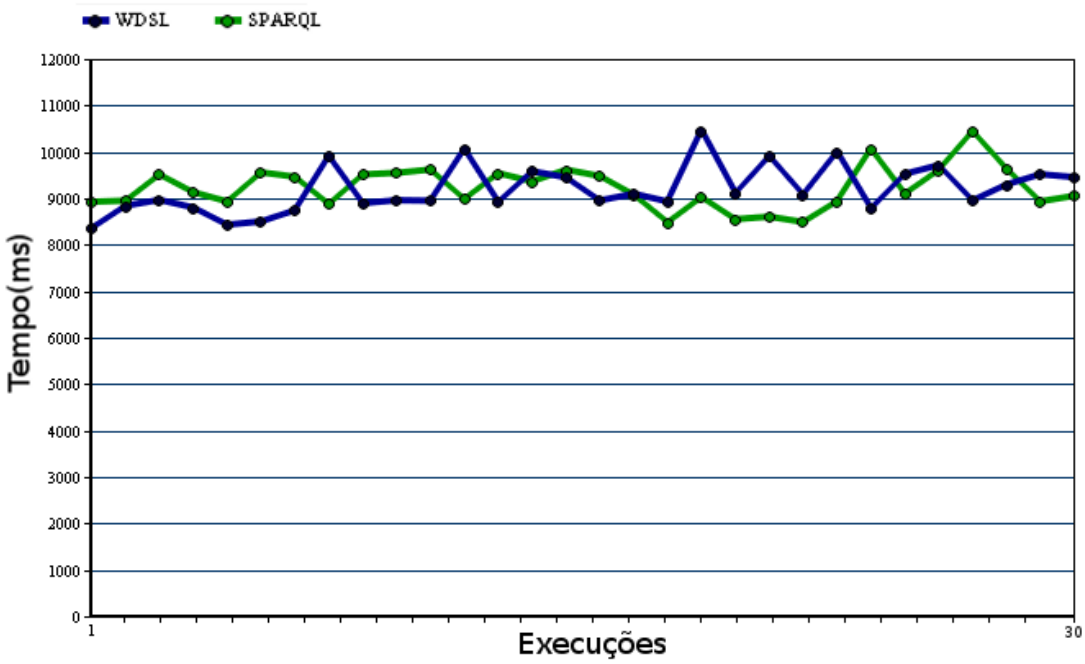

Figura 5.1: Leitura das ontologias antes de iniciar a execução do `Process`.

A Figura 5.2 apresenta um gráfico comparativo entre `SparqlGrounding` e `WSDLGrounding` na medição do tempo para mapeamento e leitura das classes do `Grounding` de cada serviço. Nesse quesito, o `SparqlGrounding` tem melhor desempenho que o `WSDLGrounding`. Esse resultado se deve ao fato de que a sub-ontologia `SparqlGrounding` possui menor número de elementos que a `WSDLGrounding`, assim tornando sua leitura mais rápida (na média). Neste experimento, obteve-se para `SparqlGrounding` média aritmética `13,1 ms` e desvio padrão `7,750` e para `WSDLGrounding` média aritmética `39,067 ms` e desvio padrão `5,833`.

Figura 5.2: Tempo para mapeamento e leitura das classes do `Grounding`.

A Figura 5.3 exibe o gráfico das execuções que medem o tempo de preparação para execução do serviço nos `SparqlGrounding` e `WSDLGrounding`. Essa medição inclui todo processamento feito pela OWL-S API para preparar o serviço para execução (iniciando-se após a a leitura da concretização de cada serviço). Assim, segundo o gráfico da Figura 5.3, observou-se que `SparqlGrounding` possui melhor desempenho que `WSDLGrounding` na preparação para execução do serviço. Isto acontece pelo fato de que o `SparqlGrounding` não possuir documento sintático, como o WSDL. A leitura do documento sintático (no caso do `WSDLGrounding`) acarreta um custo maior de desempenho, do que a construção da consulta SPARQL (no caso do `SparqlGrounding`).

Assim, através da análise dos gráficos verifica-se que o `SparqlGrounding` apresenta desempenho satisfatório na execução em conjunto com OWL-S API. A comparação com `WSDLGrounding` reforça isso, pois em todas as medições realizadas o `SparqlGrounding` teve desempenho igual ou superior ao `WSDLGrounding`.

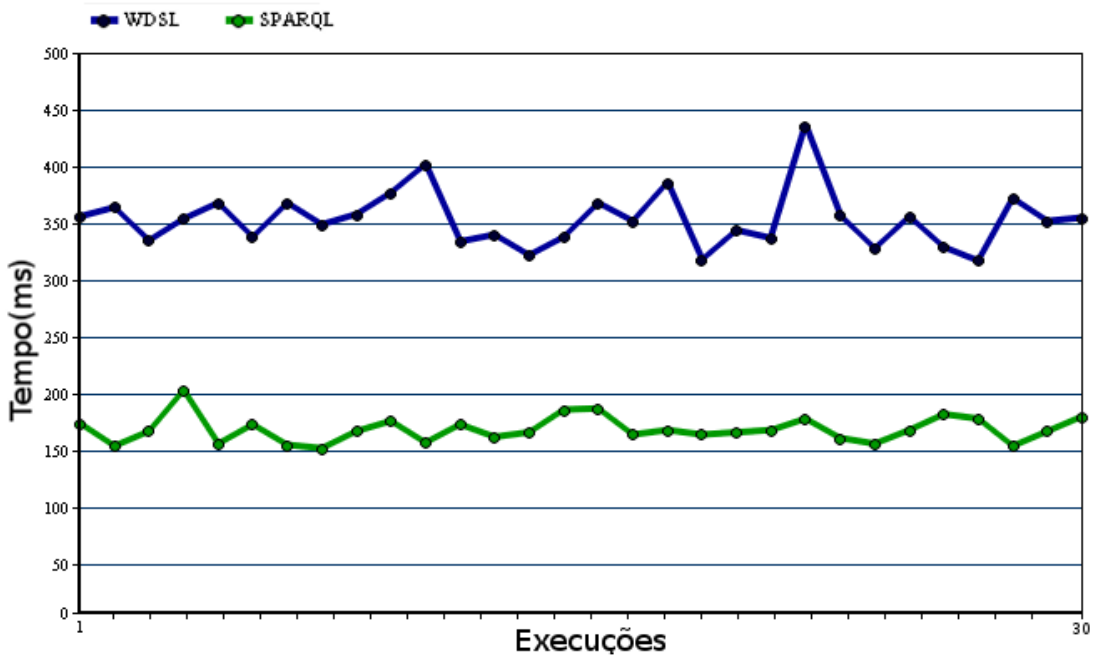

Figura 5.3: Tempo de preparação para execução do serviço.

5.2.2 Avaliação qualitativa

Avaliação qualitativa tem o objetivo de demonstrar a capacidade de execução do SWoDS para serviços que possam ser obtidos a partir de consultas SPARQL. Diferentemente da avaliação de desempenho (Seção 5.2.1), essa avaliação tem o intuito de mostrar a capacidade do SWoDS de prover serviços a partir consultas SPARQL que expressem Serviços Web.

Desse modo, tal experimento consistiu na elaboração de uma situação de requisição a qual seja possível atender através de uma consulta SPARQL. A partir dessa consulta, foi gerado manualmente um serviço SWoDS (que usa o `SparqlGrounding`), o qual é capaz de atender à requisição. Por fim, utilizou-se a OWL-S API para executar tal serviço e observar os resultados para uma situação de exemplo dentro da requisição.

O cenário de requisição escolhido foi uma solicitação em que, a partir da entrada de um nome de um país, fosse possível obter o nome da sua capital juntamente com

a população da mesma. Para tal requisição, foi desenvolvida uma consulta SPARQL utilizando o ponto de acesso do DBPedia. O Documento 5.1 apresenta a consulta desenvolvida para atender a solicitação. É possível notar que na cláusula SELECT (Documento 5.1, linha 4) são selecionados os dados de saída, que respectivamente são nome da capital (`?nome_capital`) e tamanho da população (`?populacao`). A informação de entrada na consulta está indicada na linha 6 do Documento 5.1, que se refere ao nome do país `"Brazil"@en` (o `@en` indica o idioma que a *string* está armazenada). As demais triplas da cláusula WHERE (Documento 5.1, linhas 7 a 11) são usadas para a construção do resultado obtido na consulta, relacionando as informações do país com a capital.

Documento 5.1: Consula SPARQL para requisição da avaliação qualitativa

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3  PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
4  SELECT ?nome_capital, ?populacao
5      WHERE {
6        ?pais foaf:name "Brazil"@en .
7        ?pais dbpedia-owl:capital ?capital .
8        ?capital rdf:type <http://dbpedia.org/ontology/Place> .
9        ?capital foaf:name ?nome_capital .
10       ?pais rdf:type <http://dbpedia.org/ontology/Country> .
11       ?capital dbpedia-owl:populationTotal ?populacao .
12
13     }
```

O Documento A.2 exibe o código do SWoDS criado a partir da consulta SPARQL do Documento 5.1. As linhas 75 a 172 do Documento A.2 consistem na sub-ontologia de concretização do serviço: `SparqlGrounding`. Nesse conjunto de linhas é possível observar a descrição da consulta armazenada através da estrutura definida na Seção 3.1. Os prefixos (`PREFIX`) são definidos nas linhas 85 a 102 do Documento A.2, as entradas e saídas das linhas 106 a 127 do Documento A.2 e as triplas das linhas 131 a 169 do Documento A.2.

Portanto, ao executar o serviço descrito no Documento A.2 obtêm-se o resultado

desejado. Utilizando a extensão da OWL-S API, realizou-se uma execução passando como entrada o país "Brazil". O Documento 5.2 apresenta o log da execução do serviço. É possível notar que nas linhas 2 e 3 são apresentadas as classes que descrevem a entrada e as saídas do serviço. Na linha 6 do Documento 5.2 é exibido o valor passado como entrada na execução: `"Brazil"@en`. A consulta SPARQL executada é descrita nas linhas 8 a 18 do Documento 5.2. Por fim, o resultado é indicado nas linhas 20 a 24 (Documento 5.2), onde se obtêm a população de `2789761` da capital `Brasilia`.

Documento 5.2: Log da execução do serviço SWoDs do Documento A.2.

```
1  Start executing process Atomic Process    http://127.0.0.1/owl_s/
       services/1.2/isbn_book_linked_data_grounding.owls#
       COUNTRY_CAPITAL_POP_PROCESS
2  Inputs            [name_country http://127.0.0.1/owl_s/services/1.2/
       isbn_book_linked_data_grounding.owls#_NAME_COUNTRY]
3  Outputs           [population_capital http://127.0.0.1/owl_s/services
       /1.2/isbn_book_linked_data_grounding.owls#_POPULATION_CAPITAL,
       name_capital http://127.0.0.1/owl_s/services/1.2/
       isbn_book_linked_data_grounding.owls#_NAME_CAPITAL]
4
5  Inputs:
6  name_country =  'Brazil'@en
7
8  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
9  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
10 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
11 SELECT ?populacao ?nome_capital
12 WHERE {
13 ?pais rdf:type <http://dbpedia.org/ontology/Country> .
14 ?capital foaf:name ?nome_capital .
15 ?pais dbpedia-owl:capital ?capital .
16 ?capital dbpedia-owl:populationTotal ?populacao .
17 ?pais foaf:name 'Brazil'@en .
18 }LIMIT 1
19
20 ----------------------------
21 | populacao | nome_capital  |
22 ============================
23 | 2789761   | "Brasilia"@en |
24 ----------------------------
```

5.3 Requisição e Descoberta Automática

Para avaliar a geração automática de consultas SPARQL, foi aplicado um cenário em que o módulo responsável por tal ação foi submetido a execuções utilizando requisições de serviço capazes de atender aos diferentes modelos de consultas implementados (Seção 4.1). As requisições são descritas utilizando a linguagem OWL-S e não possuem dados referentes à sub-ontologia de concretização (`Grounding`), pois delas só são utilizadas informações sobre a descrição semântica das entradas e saídas.

Os Documentos 5.3 e 5.4 apresentam trechos de códigos das requisições de serviços destacando os elementos de entrada e saída do `Process`, os quais são a principal fundamentação para geração das consultas.

No Documento 5.3 é descrito um serviço que possui como entrada um ISBN e saída um recurso livro. As ontologias que definem as saídas e as entradas estão descritas respectivamente nas linhas 3 e 10 do Documento 5.3. Essas ontologias serão usadas no processo de construção automática da consulta SPARQL. Conforme detalhado na Seção 4.1, para a construção automática de consultas SPARQL a partir de requisições OWL-S, é necessário identificar correlações entre as ontologias dos dados de entrada e saída. No exemplo do Documento 5.3, a ontologia que descreve a entrada (`ISBN`, linha 3 do Documento 5.3), que é propriedade dos recursos que são descritos pela ontologia de saída (`BOOK`, linha 10 do Documento 5.3). Dessa forma, é possível construir automaticamente uma consulta SPARQL que atenda os dados da requisição do Documento 5.3.

Documento 5.3: Trecho requisição de serviço ISBN-BOOK

```
1  <process:Input rdf:ID="_ISBN">
2    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
       anyURI">
3      http://dbpedia.org/ontology/isbn
4    </process:parameterType>
5    <rdfs:label>isbn</rdfs:label>
6  </process:Input>
7
8  <process:Output  rdf:ID="_BOOK">
```

```
9    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
         anyURI">
10     http://dbpedia.org/ontology/Book
11   </process:parameterType>
12   <rdfs:label>book</rdfs:label>
13 </process:Output>
```

O Documento 5.4 descreve um trecho da requisição referente a um serviço que, dada uma cidade como entrada, obtem a latitude e a longitude da mesma. A entrada possui ontologia definida na linha 3 do Documento 5.4 e seus recursos possuem como propriedades as ontologias de saída da requisição (latitude, na linha 10 do Documento 5.4, e longitude na linha 17 do Documento 5.4). Portanto, assim como no Documento 5.3, a partir de tal requisição é possível construir de modo automático uma consulta SPARQL atendendo à solicitação.

Documento 5.4: Trecho requisição de serviço City-Latitude/Longitude

```
1  <process:Input rdf:ID="_CITY">
2    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
         anyURI">
3      http://dbpedia.org/ontology/City
4    </process:parameterType>
5    <rdfs:label></rdfs:label>
6  </process:Input>
7
8  <process:Output  rdf:ID="_LAT">
9    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
         anyURI">
10     http://www.w3.org/2003/01/geo/wgs84_pos#lat
11   </process:parameterType>
12   <rdfs:label></rdfs:label>
13 </process:Output>
14
15 <process:Output  rdf:ID="_LON">
16   <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
         anyURI">
17     http://www.w3.org/2003/01/geo/wgs84_pos#long
18   </process:parameterType>
19   <rdfs:label></rdfs:label>
20 </process:Output>
```

Para uma medição mais adequada do desempenho do módulo gerador de consultas SPARQL, a avaliação foi dividida em três partes, as quais serão avaliada para as

requisições dos Documentos 5.3 e 5.4 :

- Medição do tempo de leitura das ontologias associadas à requisição de serviço OWL-S;
- Tempo de construção da consulta SPARQL a partir da requisição OWL-S;
- Tempo de execução da consulta (utilzando o DBPedia *endpoint*[3]);

As Figuras 5.4, 5.5 e 5.6 apresentam gráficos com os resultados das medições para as requisições de serviços OWL-S dos Documentos 5.3 e 5.4. A Figura 5.4 apresenta as medições referentes a leituras das ontologias da requisição. Se observada em comparação com as outras medições (Figuras 5.5 e 5.6), a leitura das ontologias consomem mais de 50% do custo total de execução, ponto que não está ligado à solução implementada, e sim ao acesso dos recursos da requisição de serviço em OWL-S, que seria consumido por qualquer solução (por exemplo, `WSDLGrounding`) que processasse a requisição.

A Figura 5.5 exibe o gráfico com as medições para a construção da consulta SPARQL das requisições dos Documentos 5.3 e 5.4. O tempo gasto para as duas requisições é similar, pois elas utilizam métodos semelhantes para construção da consulta SPARQL. É nesse processamento que de fato é medido o custo da solução desenvolvida, pois os outros dois (medição do tempo de leitura das ontologias associadas à requisição de serviço OWL-S e tempo de execução consulta) estão associados a elementos externos à solução.

Por fim, a Figura 5.6 apresenta o gráfico com os tempos de consumo para execução das consultas SPARQL resultantes. O tempo dessa medição também está associado ao desempenho do servidor que recebe as requisições HTTP com as consultas e ao acesso à Internet.

Dessa forma, os resultados apresentados referentes à medição do tempo de execução da criação e execução de consultas SPARQL demonstram que o tempo total

[3] http://www.dbpedia.org/sparql

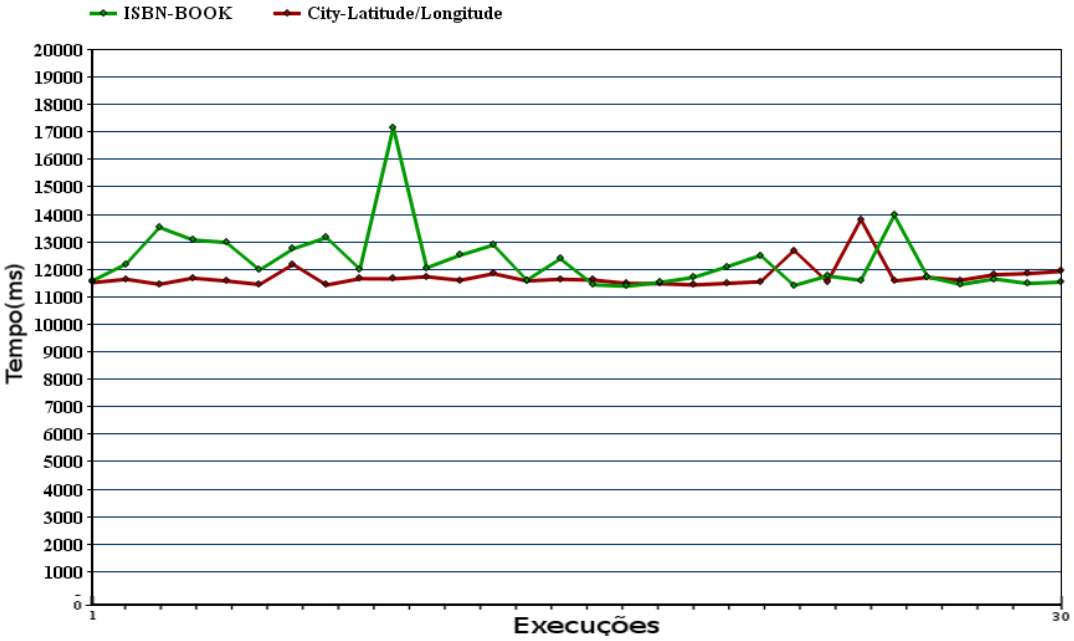

Figura 5.4: Tempo da leitura das ontologias que compõem a requisição OWL-S.

para a geração de consultas SPARQL automaticamente é em torno de 5 segundos. É importante ressaltar que esse tempo, em grande parte, está associado a velocidade de acesso a internet. O tempo é aceitável, considerando que um desenvolvedor de serviço não precisaria desenvolver a consulta manualmente e, além disso, esse processo não será repetido em casos de requisições de serviços OWL-S previamente convertidas em consultas SPARQL.

5.4 Cenário de Composição de SWS

Esta Seção apresenta a avaliação desenvolvida para simulação de um cenário de composição automática de Serviços Web Semânticos de diferentes fontes, de modo a utilizar diferentes arquiteturas de serviços através da linguagem de descrição OWL-S. Esta avaliação teve como objetivo compor Serviços OWL-S com os serviços propostos neste trabalho, o SWoDS. Além disso, buscou-se utilizar o módulo de descoberta automática

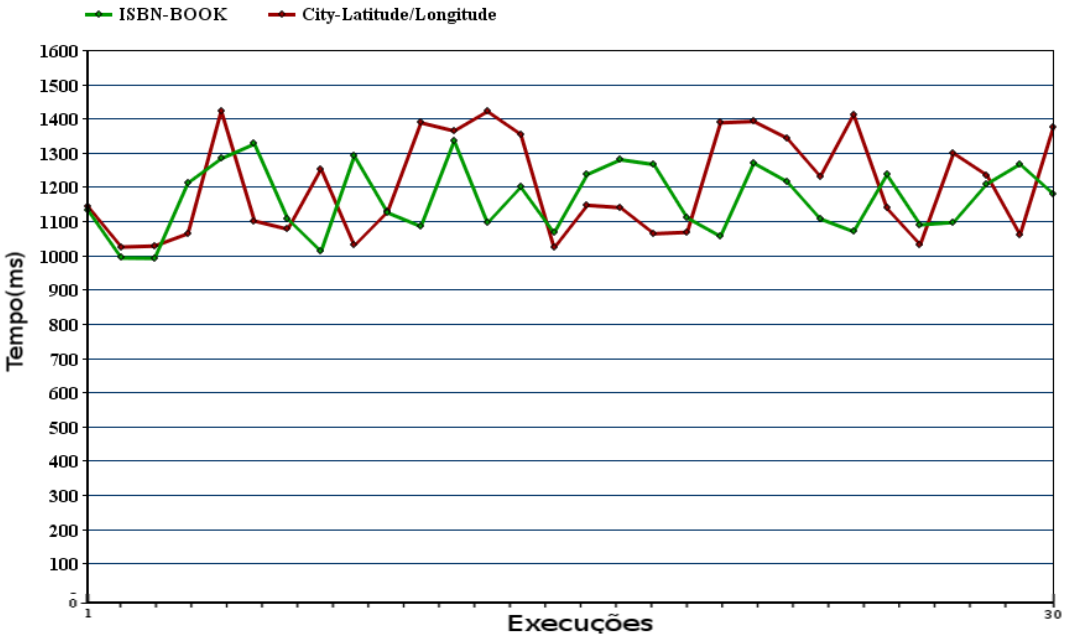

Figura 5.5: Tempo de criação da consulta SPARQL.

de serviços SWoDS para partes da requisição, em que não foram encontrados serviços disponíveis. Portanto, essa avaliação observa o aspecto referente à composição automática de dados provenientes da Web de Dados com outros tipos de serviços suportados pelo OWL-S.

Dessa forma, foi simulado um cenário de requisição, o qual envolvesse uma solicitação de Serviço Web que fosse capaz de ser atendida por diferentes serviços menores e que em composição atendessem toda a solicitação.

O cenário planejado consiste na requisição de um Serviço Web a fim de atender uma aplicação de uma hipotética agência de turismo. Nesse cenário, essa agência deseja um Serviço Web que, para cada cidade que a agência possui filial, esse serviço retorne uma série de informações sobre essa cidade, além da relação de hotéis conveniados com a agência. As informações referentes a cada cidade são: descrição da cidade, latitude e longitude, código de telefone e tamanho da população. Os desenvolvedores da aplicação da agência possuem um serviço OWL-S WSDL/SOAP, que atende à solicitação dos

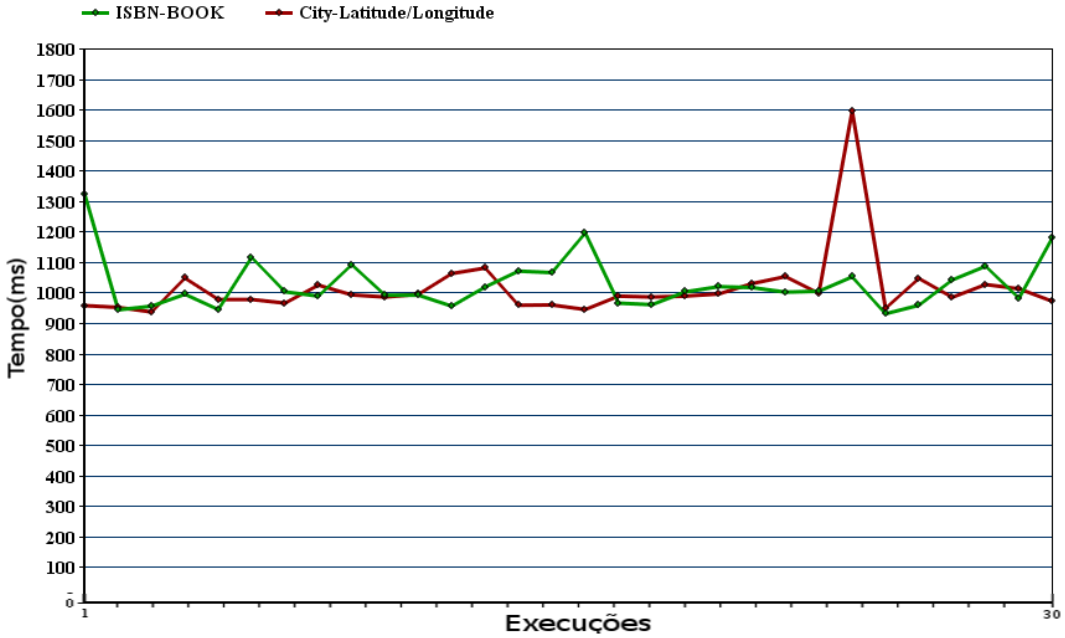

Figura 5.6: Tempo de execução da consulta SPARQL.

hotéis conveniados por cidade, além de um repositório de SWS em OWL-S, que atendem algumas das outras informações solicitas, porém não todas.

A Figura 5.7 ilustra o cenário de requisição de Serviços Web para aplicação da agência de turismo. Nesse cenário, não existe um Serviço Web que atenda à solicitação de modo exato, porém existem vários serviços que atendem a partes da requisição, exceto pelas informações de saída "descrição da cidade" e "tamanho da população", as quais não possuem nenhum serviço que atenda. Na Figura 5.7 é possível perceber também que existem dois serviços OWL-S/WSDL (serviços 2 e 5) atendendo os hotéis conveniados e o código de telefone, e um serviço SWoDS (serviço 1) que atende a solicitações referentes à latitude e longitude da cidade.

Ainda nesse cenário, a presente pesquisa desenvolveu um módulo de composição automática de serviço que, a partir de uma requisição de SWS em OWL-S, visa compor automaticamente SWS disponíveis em um repositório previamente alocado. Em acréscimo este módulo, busca-se descobrir novos serviços SWoDS para atender possíveis

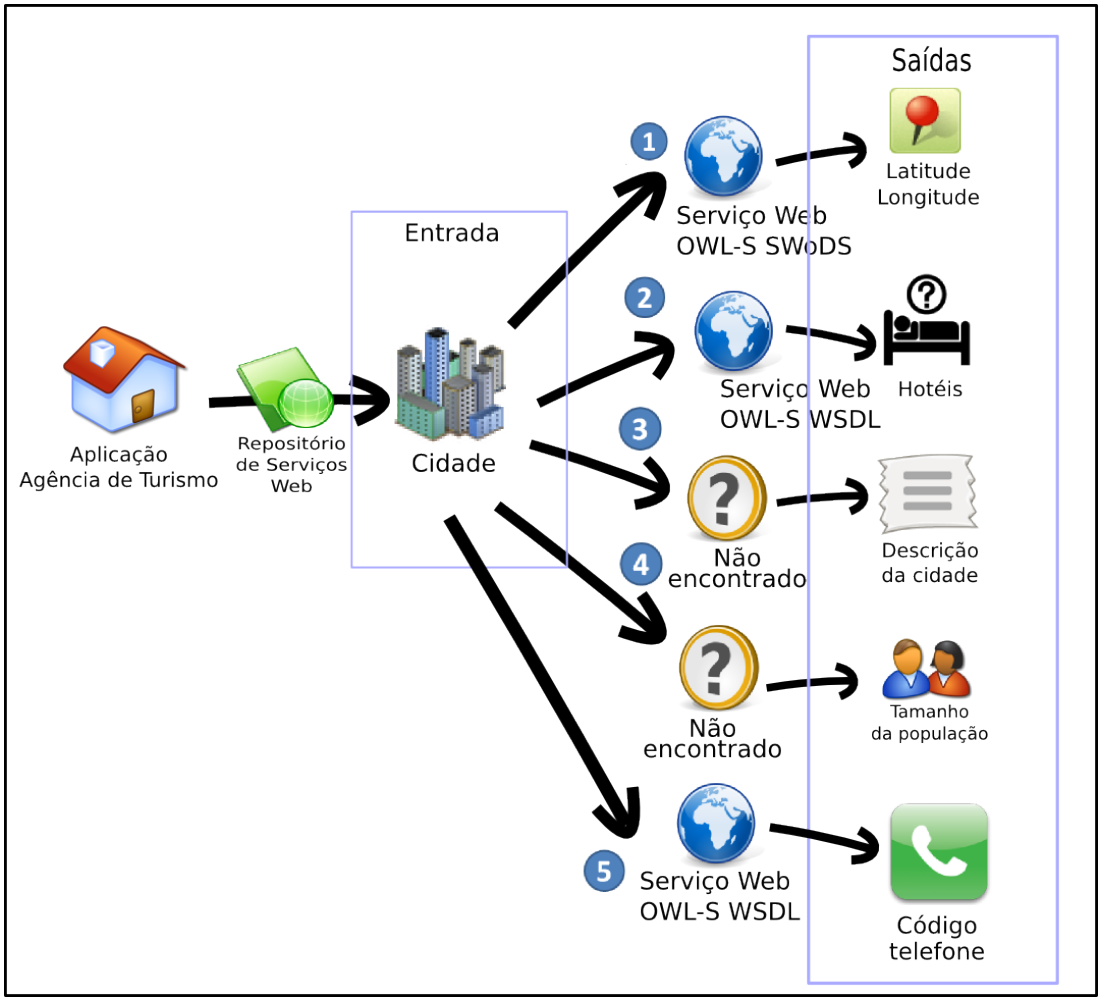

Figura 5.7: Cenário de requisição de Serviços Web para aplicação da agência de turismo.

partes da solicitação não atendidas por nenhum serviço.

Foi desenvolvido um código para aplicar a composição e descoberta dentro do cenário apresentado (é explicado mais a diante). Para suporte às descobertas, esse experimento utilizou o algoritmo proposto por Paolucci et al. [82] que, a partir das ontologias que descrevem as entradas e saídas da requisição, busca serviços que a atendam por completo ou parcialmente. Segundo Paolucci et al. [82], os serviços exatos em relação à requisição são chamados de *exact*, os serviços que atendam parcialmente a requisição são chamados de *plugin*. A partir desse algoritmo de descoberta é possível verificar se a requisição de

serviço é atendida totalmente, por um serviço *exact*, ou se é atendida totalmente por um conjunto de serviços *plugins*, ou se não é atendida por completo.

O Documento 5.5 apresenta um trecho de código utilizado para aplicar a composição e descoberta automática para o cenário da avaliação. O código aplicado nesse experimento foi desenvolvido especificamente para este cenário, contudo é possível utilizar algoritmos de composição genéricos [63, 83, 84] para qualquer tipo de composição. Inicialmente são armazenados em uma variável os serviços disponíveis no repositório (Documento 5.5, linha 2). Logo após, inicia-se a busca por um serviço que atenda exatamente a requisição de serviço (Documento 5.5, linhas 6 a 13). Caso nenhum serviço do tipo *exact* seja encontrado, busca-se por um serviço *plugin*, adicionando cada serviço encontrado a uma lista (Documento 5.5, linhas 15 a 23). Em seguida é verificado se alguma saída não é atendida por nenhum serviço *plugin* (Documento 5.5, linha 26). Caso haja saídas não atendidas, inicia-se o processo para descoberta de possíveis serviços SWoDS para as saídas não atendidas (Documento 5.5, linhas 27 a 33). Nesse processo, é feita a tentativa de construir uma consulta SPARQL do tipo ASK (como descrito na Figura 4.1) para verificação se existem dados que atendam à solicitação (Documento 5.5, linhas 28 e 29). Após isso, é verificado se a consulta SPARQL retornou *true* (indicando a presença de dados para solicitação). No caso positivo, é construído um serviço SWoDS a partir da consulta SPARQL gerada (Documento 5.5, linha 30), que é adicionado à lista de *plugins* e ao repositório.

Por fim, inicia-se o processo de composição automática de serviços utilizando a lista de *plugins* (Documento 5.5, linha 34). Nessa composição, é utilizado o construtor `Split` do OWL-S, no qual um conjunto de componentes de serviço devem ser executados de forma concorrente [4]. Esse construtor foi escolhido pois, no cenário em questão, não existem dependências entre os serviços *plugins*, permitindo assim, a execução deles em paralelo. Depois do processo de composição, o novo serviço composto é adicionado ao repositório e por fim executado (Documento 5.5, linha 38).

Documento 5.5: Trecho de código aplicado para composição de SWS e descoberta de SWoDS

```
1  //pega os servicos do repositorio
2  List<Service> services = getServicesRespository("http://localhost/owl_s/
       services/1.2/");
3
4  //busca por servicos exatos
5  Service exactService = null;
6  Iterator<Service> i_services = services.iterator();
7  while(i_services.hasNext()){
8      Service service = i_services.next();
9      org.mindswap.owls.process.Process s_process = service.getProcess();
10     if(isExactMatch(requestService, service){
11   exactService = service;
12     }
13 }
14 // busca por sub-servicos da requisicao
15 if(exactService == null){
16     List<Service> pluginServices = new ArrayList<Service>();
17     i_services = services.iterator();
18     while(i_services.hasNext()){
19         Service service = i_services.next();
20         if(isPluginMatch(requestService, service)){
21           pluginServices.add(service);
22         }
23     }
24
25 // verifica as saida nao atendidas para aplicar a descoberta de SWoDS
26     List<Output> noFoundOutputs = getOutputsWithoutPluginService(
           requestService, pluginServices);
27     if (noFoundOutputs.size() > 0){
28       RequestToSparqlGrounding SWoDS = new RequestToSparqlGrounding(
             requestService, noFoundOutputs);
29       if(SWoDS.runSparqlAskQuery()){
30     newService = SWoDS.buildSWoDS(serviceFileName, ont.getImports(false)
           );
31     pluginServices.add(newService);
32       }
33     }
34     exactService = createSplitService(pluginServices);
35 }
36
37 if(exactService != null)
38    execute(exactService);
```

Aplicando o código descrito no Documento 5.5 no cenário ilustrado na Figura 5.7, observa-se que inicialmente não há serviço exato que atenda à solicitação. Todavia, existem alguns serviços que são *plugins* em relação à solicitação: os que atendem as

saídas referentes a “hotéis”, “latitude longitude” e “código telefone”. A Figura 5.7 demonstra também que as saídas “descrição da cidade” e “total da população” não são atendidas. Assim, segundo o algoritmo, é aplicado um processo de descoberta que desenvolve uma consulta SPARQL do tipo ASK para verificar a disponibilidade de dados.

O Documento 5.6 exibe o *log* da execução do módulo de composição e descoberta automática aplicado no cenário avaliado. As linhas 5 a 14 do Documento 5.6 relacionam os serviços que foram encontrados como *plugin* e fazem um resumo sobre quais saídas foram encontradas ou não. As linhas 17 a 23 do Documento 5.6 exibem a consulta SPARQL gerada pelo módulo para buscar desenvolver um SWoDS para as saídas não atendidas. Por fim, a partir do SWoDS gerado é aplicada uma composição para executar o serviço (linhas 27 a 30 do Documento 5.6).

Documento 5.6: Log da execução da avaliação do cenário da agência de turismo

```
1  Request of service: http://localhost/owl_s/requests/1.2/
       city_tourism_request.owls
2
3  Searching a exact service...nothing found
4
5  Try to build a composition...
6  Plugin service: http://127.0.0.1/owl_s/services/1.2/city_hotel_service.
       owls#CITY_HOTEL_SERVICE
7  Plugin service: http://127.0.0.1/owl_s/services/1.2/city_lat_lon_service
       .owls#CITY_LAT_LON_SERVICE
8  Plugin service: http://127.0.0.1/owl_s/services/1.2/
       city_phone_code_service.owls#CITY_PHONE_CODE_SERVICE
9      Request output: http://dbpedia.org/ontology/abstract...no found
10     Request output: http://www.w3.org/2003/01/geo/wgs84_pos#long...found
11     Request output: http://www.w3.org/2003/01/geo/wgs84_pos#lat...found
12     Request output: http://127.0.0.1/owl_s/ontology/travel.owl#Hotel...
           found
13     Request output: http://dbpedia.org/ontology/areaCode...found
14     Request output: http://dbpedia.org/ontology/populationTotal...no
           found
15
16 Starting Web Data Service discovery module...
17 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
18 ASK
19 WHERE {
```

```
20    ?var__city rdf:type <http://dbpedia.org/ontology/Place> .
21    ?var__city <http://dbpedia.org/ontology/abstract> ?var__description .
22    ?var__city <http://dbpedia.org/ontology/populationTotal> ?
          var__population .
23  }
24  Runing SPARQL Ask query...
25  There are data in Linked Data cloud to meet the request
26
27  Building SWoDS...
28  Added a new service: /var/www/html/owl_s/services/1.2/
        SWoDS__CITY__DESCRIPTION__POPULATION.owls
29
30  Building Split composition...
```

Portanto, através dessa avaliação é possível observar a compatibilidade do SWoDS em composição com outros serviços OWL-S, além de demonstrar a aplicabilidade do módulo de descoberta, o qual pode atender partes de requisições complementando os serviços.

5.5 Considerações Finais

A avaliação apresentada neste Capítulo buscou examinar a aplicabilidade das contribuições apresentadas neste trabalho. A Seção 5.1 relatou em quais condições foram realizados os experimentos, a fim de descrever melhor o cenário.

Na Seção 5.2 é relatada a experimentação feita para avaliar a formação do `Sparql-Grounding` e o seu desempenho em conjunto com a OWL-S API estendida para suporte aos SWoDS. As execuções demonstraram que o `SparqlGrounding` apresenta tempo de execução consonante com uma implementação nativa do OWL-S `WSDLGrounding`. Em acréscimo, foi desenvolvida uma avaliação qualitativa a fim de demonstrar a capacidade do SWoDS de suportar Serviços Web que podem ser obtidos através de consultas SPARQL.

A Seção 5.3 discute a avaliação aplicada sobre o módulo de descoberta automática para a geração de consulta SPARQL a partir de requisições de serviço OWL-S. Como resultado, foi possível observar a possibilidade de explorar as bases *Linked Data* a partir

de requisições semânticas de Serviços Web.

A Seção 5.4 aplica um cenário de composição, no qual é possível ver a integração do SWoDS com outros serviços suportados pelo OWL-S. Nesse cenário de composição, pôde-se também observar a utilização do módulo de descoberta de SWoDS para suprir eventuais ausências de serviços para partes da requisição.

Capítulo 6

Trabalhos Relacionados

Este Capítulo visa discorrer sobre trabalhos relacionados ao trabalho desenvolvido pela presente pesquisa, os quais abordam de maneira correlata a utilização informações provenientes da Web de Dados com os Serviços Web. Esta pesquisa relacionou estudos que propõem a associação da Web de Dados com Serviços Web, seja para descoberta, composição ou publicação de serviços. Para facilitar a organização, os trabalhos relacionados estão divididos em dois grupos: a) Associação de Serviços Web com a Web de dados (Seção 6.1); b) Composição de aplicações (*mashups*) com dados e Serviços Web (Seção 6.2).

6.1 Serviços Web com a Web de Dados

Pedrinaci et al. [25], o denominado iServe, como uma plataforma para publicação e descoberta de Serviço Web baseado nos princípios de publicação *Linked Data*. O objetivo fundamental perseguido pelo iServe é fornecer uma plataforma capaz de publicar as anotações de serviços, de forma que permita razoáveis níveis de expressividade para viabilizar a descoberta de serviço para o uso de humanos e máquinas.

Como motivação da proposta, os autores salientam que a tecnologia de Serviços Web

reside no apoio que ela traz para o desenvolvimento de sistemas distribuídos de alta complexidade que maximizam a reutilização de componentes de baixo acoplamento [25]. Contudo o impacto da SWS na Web tem sido mínimo, pois embora as tecnologias de SWS já demonstraram benefícios, ainda existe resistência dos usuários na adoção na pilha de descrições semânticas na publicação de Serviços Web. Destaca-se também a necessidade da descoberta de Serviço Web ser capaz de lidar com a heterogeneidade de serviços (por exemplo, lidando com ambos os serviços WSDL e APIs Web).

A proposta do iServe observa que os Serviços Web Semânticos e a Web de dados podem mutuamente aliviar algumas das atuais limitações que dificultam o uso de ambos [23]. A publicação de Serviços Web seguindo os princípios *Linked Data* é capaz de unificar a descrições do serviços facilitando processos de descoberta automática. Por outro lado, os Serviços Web podem beneficiar a exploração de dados *Linked Data*, na associação em tempo de execução de modo que serviços sejam capazes de produzir e consumir dados em RDF.

Portanto, iServe transforma anotações de Serviços Web expressos em uma variedade de formatos para os chamados *Linked Service* [25]. O *Linked Service* é baseado em duas ideias básicas: publicação de anotações de Serviços Web na Web de Dados e criação de Serviço Web para Web de Dados, isto é, serviços que processem e gerem *Linked Data* [85]. Para isso os *Linked Services* a descrição das entradas e saídas do serviço utilizando RDF e para processos de descoberta utiliza-se SPARQL.

Desse modo, a proposta da iServe busca em outras palavras incluir a Web de Serviços na Web de Dados, de forma que os Serviços Web sejam publicados tal como os princípios *Linked Data* e a através de anotações uniformes utilizando RDF. Esta pesquisa trabalha em uma abordagem oposta ao SWoDS para integração de Serviços Web com Web de Dados. O SWoDS propõe a extensão da OWL-S para suporte a Serviços Web oriundos de consultas SPARQL, de modo que torna a Web de Dados um provedor de SWS.

Tanto o iServe (através do *Linked Services*) quanto SWoDS trabalham na associação

da Web de Dados com Serviços Web. Todavia, o SWoDS, na sua proposta de usar a OWL-S para prover serviços da Web de Dados, viabiliza uma estrutura mais robusta para composição automática de SWS. Isto porque a OWL-S possui maior expressividade semântica que RDF e possui definições de processos de construção para serviços compostos. Em contrapartida, o *Linked Service* apresenta estrutura de declaração do Serviço Web mais simples, o que pode tornar o processo de descoberta mais simples, por utilizar estruturas semânticas mais leves.

Assim, com o iServe, outros trabalhos visam relacionar a Web de Dados com os Serviços Web. De modo similar a maior parte dessas propostas utilizam RDF e SPARQL como estruturas básicas para a publicação, descoberta e composição de Serviços Web.

Segundo Speiser e Harth [86], a forte iniciativa de publicação de informações seguindo os princípios Linked Data dentro da nuvem de dados ligado no projeto Linking Open Data. Apesar do grande potencial dos dados integrados da nuvem LOD, existem diversas outras fontes de informações que não podem ter seus dados totalmente materializados abertamente na Web, pois em alguns casos estão em constantes mudanças, ou seus valores dependem de entradas específicas, ou por interesses comerciais. Esses tipos de dados normalmente estão disponíveis em Serviços Web ou em Web API's, os quais usualmente retornam suas requisições em formatos como JSON e XML. Assim com objetivo de descrever esses tipos de serviços semanticamente, eles propuseram a criação de um modelo formal para descrição semântica dos Serviços Web (RESTful) utilizando RDF e SPARQL. Esse modelo foi denominado de *Linked Data Services* (LIDS), onde tais serviços seguem os seguintes princípios:

- A entrada para uma chamada de serviço com determinadas ligações de parâmetro deve ser identificada por uma URI;
- A resolução da URI deve retornar a descrição dos dados de entrada e saída;
- A descrição da entrada e saída deve ser formatada em RDF.

Taheriyan et al. [26] discorrem sobre a nuvem de dados LOD, os quais não estão relacionados a informações temporais, como eventos em cidades e dados de clima-tempo. Ressalta-se também que apenas 1% das API's Web retornam seus dados em formatos RDF, isso consequentemente resulta no fato de que grande parte dessas informações estão sem conexões com a LOD. Desse modo, o modelo proposto pelos autores almeja conectar as Web API's à nuvem LOD, por meio de uma representação semântica, permitindo o consumo e a produção de RDF. Os autores citam algumas abordagens para integrar API's Web à Web de Dados, como a proposta de anotar o serviço com atributos usando conceitos de ontologias conhecidas e publicar as descrições de serviços na nuvem, e a tentativa de criar um recurso exclusivo de identificação para cada instância de uma chamada API e, em seguida, vincular esse recurso a outros conjuntos de dados. O principal obstáculo visto, que impede que estas abordagens ganhem uma ampla aceitação, é que a construção dos modelos requeridos é difícil e demorada, pois um desenvolvedor precisa criar um modelo que define o mapeamento da informação consumida e produzida por API's Web para vocabulários da Web Semântica. Segundo Taheriyan et al., a principal contribuição do trabalho é um método para construir modelos semânticos semi-automáticos de APIs Web, onde o usuário fornece exemplos de URLs para invocar um serviço e os vocabulários que deseja usar para modelá-lo. O modelo é construído automaticamente e pode ser modificado através de uma interface gráfica de fácil uso. Por fim, os modelos resultantes, representados em RDF usando vocabulários padrão, permitem a descoberta de serviço usando consultas SPARQL.

Norton e Stadtmüller [27, 28] ressaltam abordagens que buscam associar Serviços Web aos princípios Linked Data, que buscam expô-los utilizando recursos como HTTP, RDF e SPARQL. Eles observaram que trabalhos relacionados abordavam os *Linked Open Services* (LOS) [87] e *Linked Data Services* (LIDS) [88], que respectivamente referiam a serviços publicados e descritos semanticamente com Linked Data. Assim, es-

tendendo esses estudos, eles propuseram uma abordagem para descoberta e composição dos chamados Linked Services, que são Serviços Web descritos em RDF e SPARQL. Dessa forma, a proposta visa criar uma descrição semântica em serviços RESTful, de modo que seja possível aplicar operações de composição sobre eles a partir de tais interfaces, abstraindo suas particularidades de implementação. Com isso, as entradas e saídas são descritas em RDF utilizando ontologias populares na Web, em que se aplica algoritmos de similaridade para avaliar as operações de descoberta e composição.

Lanthaler e Gütl [89] destacam a grande popularidade que os serviços RESTful tem adquirido nos últimos anos, porém esses mesmos serviços possuem poucos elementos que os descrevem de modo legível para agentes de *software*. Por outro lado, observam o sucesso da expansão das bases *Linked Data*, que através de estruturas semânticas mais simples têm atraído desenvolvedores para publicação de dados. Assim, Lanthaler e Gütl [89] propõem uma nova abordagem para descrever semanticamente os serviços de dados RESTful, que permite a integração com a Web de Dados. Essa proposta é feita através da criação do SEREDASj, que é uma linguagem para descrição semântica de serviços RESTful, utilizando o formato de dados JSON. Com o uso do SEREDAj os autores afirmam que é possível integrar os serviços RESTful à Web de Dados de modo transparente, através de traduções envolvendo SPARQL, RDF e SEREDAj.

6.2 Composição de aplicações com dados e Serviços Web

Lorenzo et al.[29] ressalvam em seu trabalho que a composição de *mashups* possui diferenças em relação a composição de Serviços Web. A composição de *mashups* foca principalmente na integração de negócios (processos) dos serviços, além de agregarem dados com estruturas bastante heterogêneas. Os autores também argumentam que as *mashups* abrem uma grande oportunidade para consumo de dados e Serviços Web, contudo ainda precisa de muito investimento em pesquisas e desenvolvimento, pois

além da necessidade de conhecimento em programação é fundamental ter habilidade para realizar composições entre serviços. Assim, o trabalho analisa as principais ferramentas para autoria de aplicações Web, as quais buscam prover para o usuário meios facilitadores para a composição de Serviços Web. Na análise das ferramentas, foram observados diversos aspectos como mapeamento, formato e métodos de acesso aos dados, capacidade de extensão, dentre outras. Observa-se que as ferramentas apresentam fortes limitações de extensibilidade, de captar dados e na automatização do mapeamento das informações. Assim, dentre outros pontos, Lorenzo et al. concluem observando que a maioria das ferramentas não dão suporte à reutilização dos *mashups* criados nem à necessidade de adaptação a novas requisições dos usuários e novos dados.

De modo semelhante a Lorenzo et al., Bianchini et al. [30] argumentam também a diferença entre composição de aplicações Web e composição de Serviços Web. Nesse estudo também citam o grande volume de *mashups* que tem surgido, juntamente com ferramentas de autoria, que visam dar suporte facilitador aos usuários, todavia o processo de combinação e integração é fortemente manual. Assim, nesse contexto, técnicas de automatização, que buscam apoiar a seleção e combinação de componentes são muito importantes. Portanto, Bianchini et al. propõem métodos e técnicas para anotações semânticas dos componentes de *mashup*, visando abstrair a representação heterogênea subjacente; organização dos componentes semanticamente anotados em um repositório de componentes Mashup (MCR) com definição de ligações semânticas; e recomendação de componente dinâmico e coordenação baseada na exploração de organização MCR. As anotações semânticas usam modelos de ontologia específicos para serviços SOAP e RESTful.

Segundo Chowdhury et al. [31], as ferramentas para composição de *mashups*, apesar de disponibilizarem um ambiente gráfico e um paradigma mais simples, ainda exigem do usuário conhecimento especializado no desenvolvimento de aplicações. Analisando espe-

cialmente a ferramenta de composição de *mashups* Yahoo! Pipes[1], os autores observam que a realização de composição de componentes, além de ser um processo manual, não é trivial e nem intuitivo. Assim eles sustentam a necessidade de um sistema que auto complete a combinação de sentenças, a partir das escolhas do usuário, automatizando o processo de integração de componentes. Os autores ressalvam algumas abordagens que utilizam técnicas de análise sintática para sugestão de componentes a partir do tipo de conteúdo das entradas e saídas, tal como padrões de respostas pré-definido com base nos componentes selecionados. As limitações dessas abordagens reside no fato de que eles negligenciado as perspectivas para o desenvolvimento do usuário final, como eles ou ainda exigem habilidades avançadas de modelagem (que os usuários não têm), ou esperar que o usuário especificar regras complexas para a definição de metas (que eles não são capazes de), ou eles esperam especialistas do domínio para especificar e manter a semântica das construções de modelagem (que não o fazem). Portanto, Chowdhury et al. desenvolveram um *plugin* para o Yahoo! Pipes, que diferentemente das outras abordagens fornece recomendações interativas, contextuais da composição reusável de conhecimento. Em outras palavras, recomendam padrões de composição reutilizável, que são fragmentos de modelos quem levam o conhecimento sobre como compor *mashups*. Para isso, eles utilizam análise sintática dos padrões de conectores e componentes para compor os modelos.

6.3 Considerações Finais

Os trabalhos relacionados a integração de Serviços Web com a Web de dados (Seção 6.1) reforçam os benefícios de associar os dados aos Serviços Web. Diferentemente da proposta do SWoDS, as pesquisas apresentadas buscam incluir os Serviços Web na Web de Dados através de descrições semânticas com RDF e SPARQL. Contudo, a presente

[1] Yahoo! Pipes - http://pipes.yahoo.com

pesquisa segue uma linha oposta, onde utiliza descrições semânticas de Serviços Web para extrair dados das bases Linked Data.

As pesquisas relacionadas a composição de *mashups* (apresentadas na Seção 6.2) expõem a demanda por integração de dados e Serviços Web. A necessidade de grande intervenção do programador para desenvolver composições fortalece a necessidade de estruturas semânticas que visem trabalhar com dados e serviços de modo integrado. Assim a proposta do SWoDS ataca justamente esta lacuna, pois desenvolve condições para trabalhar com dados (*Linked Data*) e Serviços Web através de uma única interface.

Capítulo 7

Conclusão

Nesta dissertação de mestrado foram apresentados trabalhos que exploram a geração de Serviços Web através de dados oriundos da Web de Dados, viabilizando tarefas de descoberta e composição desses dados com outras fontes de Serviços Web.

A Seção 7.1 apresenta um resumo do trabalho desenvolvido com as principais contribuições produzidas. As Seções 7.2 e 7.3 relatam, respectivamente, as limitações da abordagem proposta e as oportunidades de trabalhos futuros dentro do trabalho produzido nesta dissertação.

7.1 Resumo do trabalho realizado

Motivado pela necessidade de integração da Web de Dados com os Serviços Web, esta dissertação teve como objetivo principal desenvolver uma proposta para permitir a geração de Serviços Web que utilizem informações da Web de Dados. Essa proposta resultou do desenvolvimento do *Semantic Web (of Data) Service* (SWoDS), que, através da extensão da linguagem de concretização da OWL-S, provê a criação de Serviços Web que utilizam consultas SPARQL em bases *Linked Data*. Com o uso do SWoDS é possível a realização de composição automática com outras arquiteturas de Serviços

Web descritos com OWL-S.

As principais contribuições referentes ao trabalho reportado nesta dissertação estão resumidas a seguir:

- Desenvolvimento da `SparqlGrounding`, que é uma extensão da ontologia de suporte à execução para serviços OWL-S. A `SparqlGrounding` viabiliza a concretização e descrição dos serviços SWoDS, que são desenvolvidos a partir de uma consulta SPARQL associada à estrutura semântica do OWL-S;
- Extensão da principal biblioteca de suporte a serviços OWL-S (OWL-S API) para dar sustentação do `SparqlGrounding`, permitindo a criação e execução de serviços SWoDS, tal como como a composição com outros serviços OWL-S;
- O desenvolvimento de um módulo para descoberta automática de SWoDs, que a partir de requisições de serviços descritas em OWL-S gera, automaticamente, consultas (na linguagem SPARQL e consequentemente SWoDS) para buscar na Web de Dados informações equivalentes à requisição do serviço;
- Desenvolvimento de um estudo de caso para avaliar a consistência dos Serviços Web de Dados, aplicando invocação de serviços, descoberta e composição, na qual é relacionado com outras fontes de serviços dentro de um cenário simulado de estudo.

7.2 Limitações

As limitações desse trabalho estão relacionadas à aspectos do SWoDS e do módulo de descoberta automática de SWoDS. As limitações referentes a esse trabalho podem ser tratadas como oportunidades de trabalhos futuros. Dentre elas destacam-se:

- O SWoDS não possui suporte a construções de serviços OWL-S que possuam

pré-condições e efeitos, que, respectivamente, são condições para execução dos serviços e consequências após a execução;

- Não existe suporte para criação de SWoDS que utilizem como base consultas SPARQL que façam uso da estrutura de sub-consultas. Em outras palavras, o `SparqlGrounding` não define suporte a sub-consultas SPARQL;
- O módulo de descoberta não analisa a estrutura hierárquica das ontologias que definem a entrada e saída das requisições de serviço para ampliar a capacidade de descoberta automática.

Como é citado na Seção 4.1, o módulo de descoberta automática de SWoDS não é o foco desta pesquisa, portanto, existem algumas limitações associadas a esses aspectos, as quais podem ser relevantes oportunidades de pesquisas.

7.3 Trabalhos futuros

O trabalho descrito nesta dissertação permite identificar alguns trabalhos futuros tanto para dar continuidade à pesquisa aqui reportada, como para avaliar alternativas às limitações citadas na Seção 7.2. Uma lista de algumas oportunidades de trabalhos futuros:

- Permitir a criação de SWoDS que utilizem estruturas de sub-consultas SPARQL;
- Estender o `SparqlGrounding`, para dar suporte as pré-condições e efeitos do OWL-S;
- Desenvolver interfaces gráfica para semi-automatizar o processo de criação e descoberta automática de SWoDS;
- Aprofundar os estudos referentes a descoberta automática de consultas SPARQL a partir de requisições semânticas de Serviços Web, visando definir os limites

e possibilidade de gerações automáticas de SWoDS a partir de requisições de serviços em OWL-S.

Referências Bibliográficas

[1] BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, v. 5, n. 3, p. 1–22, MarMar 2009.

[2] BERNERS-LEE, T. Semantic web on xml, 2000. Disponível na internet em http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html. Último acesso em 11 outubro 2014.

[3] BOND, M.; ROXBURGH, P.; LONGSHAW, A.; HAYWOOD, D.; LAW, D. *Sams teach yourself j2ee in 21 days*. Indianapolis, IN, USA: Sams, 2002.

[4] W3C. Owl-s: Semantic markup for web services. Disponível na internet em http://www.w3.org/Submission/OWL-S/. Último acesso em 14 outubro 2014.

[5] SCHMACHTENBERG, M.; BIZER, C.; PAULHEIM, H. Adoption of the linked data best practices in different topical domains. In: . Editors MIKA, P.; TUDORACHE, T.; BERNSTEIN, A.; WELTY, C.; KNOBLOCK, C.; VRANDEčIć, D.; GROTH, P.; NOY, N.; JANOWICZ, K.; GOBLE, C. Springer International Publishing, c2014. v. 8796 of *Lecture Notes in Computer Science*. p. 245–260.

[6] BIZER, C. The emerging web of linked data. *IEEE Intelligent Systems*, Piscataway, NJ, USA, v. 24, n. 5, p. 87–92, Sept. 2009.

[7] TSAI, C.-C.; LEE, C.-J.; TANG, S.-M. The web 2.0 movement: mashups driven and web services. *W. Trans. on Comp.*, Stevens Point, Wisconsin, USA, v. 8, n. 8, p. 1235–1244, aug 2009.

[8] OŔEILLY, T. What is web 2.0. design patterns and business models for the next generation of software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, September 2005.

[9] LYTRAS, M.; DAMIANI, E.; DE PABLOS, P. O. (Eds.). *Web 2.0 : the business model.* New York: Springer, 2009.

[10] BERNERS-LEE, T. Services and semantics: Web architecture. W3c recommendation, World Wide Web Consortium, 2001.

[11] MEDJAHED, B.; BOUGUETTAYA, A.; ELMAGARMID, A. K. Composing web services on the semantic web. *The VLDB Journal*, Secaucus, NJ, USA, v. 12, n. 4, p. 333–351, Nov. 2003.

[12] BENSLIMANE, D.; DUSTDAR, S.; SHETH, A. Services mashups: The new generation of web applications. *Internet Computing, IEEE*, v. 12, n. 5, p. 13–15, 2008.

[13] BOX, D.; EHNEBUSKE, D.; KAKIVAYA, G.; LAYMAN, A.; MENDELSOHN, N.; NIELSEN, H. F.; THATTE, S.; WINER, D. Simple Object Access Protocol (SOAP) 1.1. W3c note, World Wide Web Consortium, 2000. See `http://www.w3.org/TR/SOAP/`.

[14] RICHARDSON, L.; RUBY, S. *Restful web services.* First. ed. OŔeilly, 2007.

[15] ERIK CHRISTENSEN, FRANCISCO CURBERA, G. M.; WEERAWARANA,

S. Web service definition language (wsdl), 2001. Disponível na internet em http://www.w3.org/TR/wsdl. Último acesso em 14 outubro 2014.

[16] HADLEY, M. J. Web application description language (wadl). In: . Mountain View, CA, USA: Sun Microsystems, Inc., c2006.

[17] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, v. 284, n. 5, p. 34–43, May 2001.

[18] GIBBINS, N.; SHADBOLT, N. Resource description framework (rdf). In: *Encyclopedia of Library and Information Sciences*. 2009.

[19] BIZER, C.; HEATH, T.; IDEHEN, K.; BERNERS-LEE, T. Linked data on the web (ldow2008). In: . WWW '08. New York, NY, USA: ACM, c2008. p. 1265–1266.

[20] MCILRAITH, S.; SON, T. C.; ZENG, H. Semantic web services. *Intelligent Systems, IEEE*, v. 16, n. 2, p. 46–53, 2001.

[21] MARTIN, D.; BURSTEIN, M.; MCDERMOTT, D.; MCILRAITH, S.; PAOLUCCI, M.; SYCARA, K.; MCGUINNESS, D. L.; SIRIN, E.; SRINIVASAN, N. Bringing semantics to web services with owl-s. *World Wide Web*, Hingham, MA, USA, v. 10, n. 3, p. 243–277, Sept. 2007.

[22] ROMAN, D.; KELLER, U.; LAUSEN, H.; DE BRUIJN, J.; LARA, R.; STOLLBERG, M.; POLLERES, A.; FEIER, C.; BUSSLER, C.; FENSEL, D. Web service modeling ontology. *Appl. Ontol.*, Amsterdam, The Netherlands, The Netherlands, v. 1, n. 1, p. 77–106, jan 2005.

[23] PEDRINACI, C.; DOMINGUE, J.; KRUMMENACHER, R. Services and the web of data: An unexploited symbiosis. In: . AAAI, c2010.

[24] DAVIES, J.; POTTER, M. R. M. S. S. D. J. P. C. F. D.; GONZáLEZ-CABERO, R. Towards the open service web. In: . c2009. v. 26.

[25] PEDRINACI, C.; LIU, D.; MALESHKOVA, M.; LAMBERT, D.; KOPECKY, J.; DOMINGUE, J. iserve: a linked services publishing platform, 2010.

[26] TAHERIYAN, M.; KNOBLOCK, C. A.; SZEKELY, P.; AMBITE, J. L. Rapidly integrating services into the linked data cloud. In: . ISWC'12. Berlin, Heidelberg: Springer-Verlag, c2012. p. 559–574.

[27] NORTON, B.; STADTMULLER, S. Scalable discovery of linked servicess. In: . CEUR-WS.org,, c2011.

[28] STADTMULLER, S. Composition of linked data-based restful services. In: . ISWC'12. Berlin, Heidelberg: Springer-Verlag, c2012. p. 461–464.

[29] DI LORENZO, G.; HACID, H.; PAIK, H.-Y.; BENATALLAH, B. Data integration in mashups. *SIGMOD Rec.*, New York, NY, USA, v. 38, n. 1, p. 59–66, jun 2009.

[30] BIANCHINI, D.; DE ANTONELLIS, V.; MELCHIORI, M. Semantic-driven mashup design. In: . iiWAS '10. New York, NY, USA: ACM, c2010. p. 247–254.

[31] ROY CHOWDHURY, S.; RODRÍGUEZ, C.; DANIEL, F.; CASATI, F. Baya: assisted mashup development as a service. In: . WWW '12 Companion. New York, NY, USA: ACM, c2012. p. 409–412.

[32] BERNERS-LEE, T.; CAILLIAU, R.; LUOTONEN, A.; NIELSEN, H. F.; SECRET, A. The world-wide web. *Commun. ACM*, New York, NY, USA, v. 37, n. 8, p. 76–82, Aug. 1994.

[33] Editors FENSEL, D.; HENDLER, J. A.; LIEBERMAN, H.; WAHLSTER, W. MIT Press, c2003.

[34] LACY, L. W. *Owl : representing information using the web ontology language.* Victoria BC, Canada: Trafford Publishing, 2005.

[35] ANTONIOU, G.; HARMELEN, F. V. *A semantic web primer, 2nd edition (cooperative information systems)*. 2. ed. The MIT Press, 2008.

[36] KLYNE, G.; CARROLL, J. J. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[37] SWARTZ, A. application/rdf+xml Media Type Registration. RFC 3870 (Informational), September 2004.

[38] BERNERS-LEE, T.; CONNOLLY, D. Notation3 (n3): A readable rdf syntax. Technical report, W3C, 1 2008.

[39] GRUBER, T. R. A translation approach to portable ontology specifications. *Knowl. Acquis.*, London, UK, UK, v. 5, n. 2, p. 199–220, June 1993.

[40] FENSEL, D. Ontologies: A silver bullet for knowledge management and electronic commerce. Secaucus, NJ, USA, 2003.

[41] HENDLER, J. Agents and the semantic web. *Intelligent Systems, IEEE*, v. 16, n. 2, p. 30–37, Mar-Apr 2001.

[42] ANTONIOU, G.; ; ANTONIOU, G.; ANTONIOU, G.; HARMELEN, F. V.; HARMELEN, F. V. Web ontology language: Owl. In: . Springer, c2003. p. 67–92.

[43] CONSORTIUM, T. U. What is unicode?, 2013. Disponível na internet em http://www.unicode.org/standard/WhatIsUnicode.html. Último acesso em 11 outubro 2014.

[44] W3C. Extensible markup language (xml) 1.1, 2004. Disponível na internet em http://www.w3.org/TR/2004/REC-xml11-20040204/. Último acesso em 13 outubro 2014.

[45] W3C. Rdf schema 1.1, 2004. Disponível na internet em http://www.w3.org/TR/rdf-schema/. Último acesso em 13 outubro 2014.

[46] W3C. Swrl: A semantic web rule language combining owl and ruleml, 2004. Disponível na internet em http://www.w3.org/Submission/SWRL/. Último acesso em 14 outubro 2014.

[47] SWARTZ, A. The semantic web in breadth, 2001. `http://logicerror.com/semanticWeb-long` [Last access: 2004-02-18].

[48] PAYNE, T.; LASSILA, O. Guest editors' introduction: Semantic web services. *IEEE Intelligent Systems*, Piscataway, NJ, USA, v. 19, n. 4, p. 14–15, jul 2004.

[49] BATTLE, S.; BERNSTEIN, A.; BOLEY, H.; GROSOF, B.; GRUNIGER, M.; HULL, R.; KIFER, M.; MARTIN, D.; MCILRAITH, S.; MCGUINNESS, D.; SU, J.; TABET, S. Semantic web services framework (swsf). Technical Report IFI-2008.0008, APR 2005.

[50] AKKIRAJU, R.; FARRELL, J.; J.MILLER; NAGARAJAN, M.; SCHMIDT, M.; SHETH, A.; VERMA, K. Web Service Semantics - WSDL-S, "A joint UGA-IBM Technical Note, version 1.0,. Technical report, April 2005.

[51] BERNERS-LEE, T. Web services: Program integration across application and organization boundaries. W3c recommendation, World Wide Web Consortium, 2002.

[52] STAL, M. Web services: beyond component-based computing. *Commun. ACM*, New York, NY, USA, v. 45, n. 10, p. 71–76, Oct. 2002.

[53] DAN, A.; JOHNSON, R. D.; CARRATO, T. Soa service reuse by design. In: . SDSOA '08. New York, NY, USA: ACM, c2008. p. 25–28.

[54] W3C. Web services architecture, 2003. Disponível na internet em http://www.w3.org/TR/2003/WD-ws-arch-20030514/. Último acesso em 14 outubro 2014.

[55] PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. "big"' web services: Making the right architectural decision. In: . WWW '08. New York, NY, USA: ACM, c2008. p. 805–814.

[56] FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. 2000. Tese (Doutorado em Física) - 2000. AAI9980887.

[57] GUDGIN, M.; HADLEY, M.; MENDELSOHN, N.; MOREAU, J.-J.; Frystyk Nielsen, H.; KARMARKAR, A.; LAFON, Y. Soap version 1.2 part 1: Messaging framework (second edition). World Wide Web Consortium, Recommendation REC-soap12-part1-20070427, April 2007.

[58] ARROYO, S., L. R. G. J. M. B. D. D. Y.; FENSEL, D. Semantic aspects of web services. *The Practical Handbook of Internet Computing*, p. 17–31, 2005.

[59] SIRIN, E.; PARSIA, B.; HENDLER, J. Filtering and selecting semantic web services with interactive composition techniques. *Intelligent Systems, IEEE*, v. 19, n. 4, p. 42–49, Jul-Aug 2004.

[60] MARTIN, D.; PAOLUCCI, M.; WAGNER, M. Bringing semantic annotations to web services: Owl-s from the sawsdl perspective. In: . ISWC'07/ASWC'07. Berlin, Heidelberg: Springer-Verlag, c2007. p. 340–352.

[61] DAML. Daml services, 2007. Disponível na internet em http://www.daml.org/services/owl-s/. Último acesso em 18 outubro 2014.

[62] MARTIN, D.; BURSTEIN, M.; MCDERMOTT, D.; MCILRAITH, S.; PAOLUCCI, M.; SYCARA, K.; MCGUINNESS, D.; SIRIN, E.; SRINIVASAN, N. Bringing

semantics to web services with owl-s. *World Wide Web*, v. 10, n. 3, p. 243–277, 2007.

[63] PRAZERES, C.; TEIXEIRA, C.; PIMENTEL, M. Semantic web services discovery and composition: Paths along workflows. In: . c2009. p. 58–65.

[64] PAOLUCCI, M.; SYCARA, K. P.; NISHIMURA, T.; SRINIVASAN, N. Using daml-s for p2p discovery. In: . Editor ZHANG, L.-J. CSREA Press, c2003. p. 203–207.

[65] MASUOKA, R.; PARSIA, B.; LABROU, Y. Task computing - the semantic web meets pervasive computing. In: . Editors FENSEL, D.; SYCARA, K. P.; MYLOPOULOS, J. Springer, c2003. v. 2870 of *Lecture Notes in Computer Science*. p. 866–881.

[66] PAOLUCCI, M.; SRINIVASAN, N.; SYCARA, K. Expressing wsmo mediators in owls. In: . c2004.

[67] PAOLUCCI, M.; 0001, M. W.; MARTIN, D. Grounding owl-s in sawsdl. In: . Editors KRäMER, B. J.; LIN, K.-J.; NARASIMHAN, P. Springer, c2007. v. 4749 of *Lecture Notes in Computer Science*. p. 416–421.

[68] FILHO, O. F. F. *Serviços semânticos: Uma abordagem restful*. 2010. Dissertação (Mestrado em Física) - 2010. Universidade de São Paulo, Brasil.

[69] XAVIER, O. C. *Serviços web semânticos baseados em restful: Um estudo de caso em redes sociais online*. 2011. Dissertação (Mestrado em Física) - 2011. Universidade Federal de Goiás, Brasil.

[70] SIRIN, E.; PARSIA, B. The owl-s java api. In: . c2004.

[71] EVREN SIRIN, M. D.; MöLLER, T. Installation of the owl-s api, 2012. Disponível

na internet em http://on.cs.unibas.ch/owls-api/installation.html. Último acesso em 21 outubro 2014.

[72] EVREN SIRIN, M. D.; MöLLER, T. Documentation of the owl-s api, 2012. Disponível na internet em http://on.cs.unibas.ch/owls-api/documentation.html. Último acesso em 21 outubro 2014.

[73] KOPECKý, J.; VITVAR, T.; BOURNEZ, C.; FARRELL, J. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, Piscataway, NJ, USA, v. 11, n. 6, p. 60–67, Nov. 2007.

[74] BERNERS-LEE, T. Linked data - design issues. *W3C*, , n. 09/20, 2006.

[75] BERNERS-LEE, T.; CHEN, Y.; CHILTON, L.; CONNOLLY, D.; DHANARAJ, R.; HOLLENBACH, J.; LERER, A.; SHEETS, D. Tabulator: Exploring and analyzing linked data on the semantic web. In: . c2006.

[76] CHENG, G.; QU, Y. Searching linked objects with falcons: Approach, implementation and evaluation. *Int. J. Semantic Web Inf. Syst.*, v. 5, n. 3, p. 49–70, 2009.

[77] PRUD'HOMMEAUX, E.; SEABORNE, A. Sparql query language for rdf. W3c recommendation, W3C, Jan. 2008.

[78] CLARK, K. G.; FEIGENBAUM, L.; TORRES, E. Sparql protocol for rdf. World Wide Web Consortium, Recommendation REC-rdf-sparql-protocol-20080115, January 2008.

[79] W3C. Linking Open Data, August. Disponível na internet em http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData. Último acesso em 14 outubro 2014.

[80] MAX SCHMACHTENBERG, C. B.; PAULHEIM, H. State of the lod cloud 2014, 2014. Disponível na internet em http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/. Último acesso em 24 outubro 2014.

[81] CARDOSO, J.; SHETH, A. P. (Eds.). *Semantic web services, processes and applications*. Springer, 2006. v. 3 of *Semantic Web And Beyond Computing for Human Experience*.

[82] PAOLUCCI, M.; KAWAMURA, T.; PAYNE, T. R.; SYCARA, K. P. Semantic matching of web services capabilities. In: . ISWC '02. London, UK, UK: Springer-Verlag, c2002. p. 333–347.

[83] MITRA, S.; KUMAR, R.; BASU, S. Automated choreographer synthesis for web services composition using i/o automata. In: . c2007. p. 364–371.

[84] Goncalves da Silva, E. M.; Ferreira Pires, L.; van Sinderen, M. J. An algorithm for automatic service composition. In: . Editors Goncalves da Silva, E. M.; Ferreira Pires, L.; van Sinderen, M. J. Portugal: INSTICC Press, c2007. p. 65–74.

[85] PEDRINACI, C.; DOMINGUE, J. Toward the next wave of services: Linked services for the web of data. *J. UCS*, v. 16, n. 13, p. 1694–1719, 2010.

[86] SPEISER, S.; HARTH, A. Integrating linked data and services with linked data services. In: . ESWC'11. Berlin, Heidelberg: Springer-Verlag, c2011. p. 170–184.

[87] KRUMMENACHER, R.; NORTON, B.; MARTE, A. Towards linked open services and processes. In: . FIS'10. Berlin, Heidelberg: Springer-Verlag, c2010. p. 68–77.

[88] SPEISER, S.; HARTH, A. Taking the lids off data silos. In: . I-SEMANTICS '10. New York, NY, USA: ACM, c2010. p. 44:1–44:4.

[89] LANTHALER, M.; GüTL, C. Seamless integration of restful services into the web

of data. *Adv. MultiMedia*, New York, NY, United States, v. 2012, p. 1:1–1:14, Jan. 2012.

Apêndice A

Apêndice 1

Documento A.1: Exemplo de serviço SWoDS (Entrada: ISBN Saída: Livro).

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:owl       = "http://www.w3.org/2002/07/owl#"
xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:service   = "http://www.daml.org/services/owl-s/1.2/Service.owl#"
xmlns:process   = "http://www.daml.org/services/owl-s/1.2/Process.owl#"
xmlns:profile   = "http://www.daml.org/services/owl-s/1.2/Profile.owl#"
xmlns:grounding = "http://www.daml.org/services/owl-s/1.2/Grounding.owl#"
xmlns:sparql    = "http://localhost/owl_s/ontology/sparqlGrounding.owl#"
xml:base        = "http://127.0.0.1/owl_s/services/1.2/isbn_book_linked_data_grounding.owls">

<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Service.owl" />
<owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl" />
<owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Profile.owl" />
<owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Grounding.owl" />
<owl:imports rdf:resource="http://127.0.0.1/owl_s/ontology/portal.owl" />
<owl:imports rdf:resource="http://127.0.0.1/owl_s/ontology/dbpedia_3.9.owl" />
<owl:imports rdf:resource="http://localhost/owl_s/ontology/sparqlGrounding.owl" />

```

```
22 </owl:Ontology>
23
24 <service:Service rdf:ID="ISBN_BOOK_SERVICE">
25 <service:presents rdf:resource="#ISBN_BOOK_PROFILE"/>
26 <service:describedBy rdf:resource="#ISBN_BOOK_PROCESS"/>
27 <service:supports rdf:resource="#ISBN_BOOK_GROUNDING"/>
28 </service:Service>
29
30 <profile:Profile rdf:ID="ISBN_BOOK_PROFILE">
31 <service:isPresentedBy rdf:resource="#ISBN_BOOK_SERVICE"/>
32 <profile:serviceName xml:lang="en">
33     BookProviderService
34 </profile:serviceName>
35 <profile:textDescription xml:lang="en">
36     This service provides a book title on the given ISBN.
37 </profile:textDescription>
38 <profile:hasInput  rdf:resource="#_ISBN"/>
39 <profile:hasOutput rdf:resource="#_BOOK"/>
40 <profile:has_process rdf:resource="ISBN_BOOK_PROCESS" />
41 </profile:Profile>
42
43 <process:AtomicProcess rdf:ID="ISBN_BOOK_PROCESS">
44 <service:describes rdf:resource="#ISBN_BOOK_SERVICE"/>
45 <process:hasInput  rdf:resource="#_ISBN"/>
46 <process:hasOutput rdf:resource="#_BOOK"/>
47 </process:AtomicProcess>
48
49 <process:Input rdf:ID="_ISBN">
50 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
       anyURI">http://dbpedia.org/ontology/isbn</process:parameterType>
51 <rdfs:label>isbn</rdfs:label>
52 </process:Input>
53
54 <process:Output  rdf:ID="_BOOK">
55 <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
       anyURI">http://dbpedia.org/ontology/Book</process:parameterType>
56 <rdfs:label>book</rdfs:label>
57 </process:Output>
58
59 <sparql:SparqlGrounding rdf:ID="ISBN_BOOK_GROUNDING">
60 <service:supportedBy rdf:resource="#ISBN_BOOK_SERVICE"/>
61 <grounding:hasAtomicProcessGrounding rdf:resource="#
       ISBN_BOOK_AtomicProcessGrounding"/>
62 </sparql:SparqlGrounding>
63
64 <!-- SPARQL Atomic Process Grounding -->
65
66 <sparql:SparqlAtomicProcessGrounding rdf:ID="
       ISBN_BOOK_AtomicProcessGrounding">
67 <grounding:owlsProcess rdf:resource="#ISBN_BOOK_PROCESS"/>
68
69 <sparql:sparqlVersion rdf:datatype="http://www.w3.org/2001/XMLSchema#
       anyURI">
```

```
70  http://www.w3.org/TR/rdf-sparql-query/
71  </sparql:sparqlVersion>
72  <sparql:sparqlEndPoint rdf:datatype="http://www.w3.org/2001/XMLSchema#
        anyURI">http://dbpedia.org/sparql</sparql:sparqlEndPoint>
73
74  <!-- Prefixies -->
75
76  <sparql:SparqlPrefixes>
77  <sparql:SparqlPrefixMap>
78      <sparql:PrefixName>rdf</sparql:PrefixName>
79      <sparql:PrefixUri>http://www.w3.org/1999/02/22-rdf-syntax-ns#</
            sparql:PrefixUri>
80  </sparql:SparqlPrefixMap>
81  </sparql:SparqlPrefixes>
82
83
84  <!-- Input Parameters -->
85
86  <sparql:SparqlInputParam>
87  <sparql:SparqlInputParamMap>
88    <grounding:owlsParameter rdf:resource="#_ISBN"/>
89    <sparql:SparqlDataParam rdf:datatype="http://www.w3.org/2001/XMLSchema
          #string">?var_isbn</sparql:SparqlDataParam>
90  </sparql:SparqlInputParamMap>
91  </sparql:SparqlInputParam>
92
93  <!-- Ouput Parameters -->
94
95  <sparql:SparqlOutputParam>
96  <sparql:SparqlOutputParamMap>
97    <grounding:owlsParameter rdf:resource="#_BOOK"/>
98    <sparql:SparqlDataParam rdf:datatype="http://www.w3.org/2001/XMLSchema
          #string">?var_book</sparql:SparqlDataParam>
99  </sparql:SparqlOutputParamMap>
100 </sparql:SparqlOutputParam>
101
102 <!-- Triples -->
103
104 <sparql:SparqlTriples>
105 <sparql:SparqlTripleMap>
106     <sparql:TripleSubject>?var_book</sparql:TripleSubject>
107     <sparql:TriplePredicate>rdf:type</sparql:TriplePredicate>
108     <sparql:TripleObject>http://dbpedia.org/ontology/Book</
            sparql:TripleObject>
109 </sparql:SparqlTripleMap>
110 </sparql:SparqlTriples>
111
112 <sparql:SparqlTriples>
113 <sparql:SparqlTripleMap>
114     <sparql:TripleSubject>?var_book</sparql:TripleSubject>
115     <sparql:TriplePredicate>http://dbpedia.org/ontology/isbn</
            sparql:TriplePredicate>
116     <sparql:TripleObject>?var_isbn</sparql:TripleObject>
```

```
117 </sparql:SparqlTripleMap>
118 </sparql:SparqlTriples>
119
120 </sparql:SparqlAtomicProcessGrounding>
121 </rdf:RDF>
```

Documento A.2: SWoDS para serviço com entrada um nome de um país e saída a população e nome da capital

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rdf:RDF
3
4  xmlns:owl       = "http://www.w3.org/2002/07/owl#"
5  xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
6  xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
7  xmlns:service   = "http://www.daml.org/services/owl-s/1.2/Service.owl#"
8  xmlns:process   = "http://www.daml.org/services/owl-s/1.2/Process.owl#"
9  xmlns:profile   = "http://www.daml.org/services/owl-s/1.2/Profile.owl#"
10 xmlns:grounding = "http://www.daml.org/services/owl-s/1.2/Grounding.owl#
       "
11 xmlns:sparql    = "http://localhost/owl_s/ontology/sparqlGrounding.owl#"
12 xml:base        = "http://127.0.0.1/owl_s/services/1.2/
       isbn_book_linked_data_grounding.owls">
13
14 <owl:Ontology rdf:about="">
15   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
         Service.owl" />
16   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
         Process.owl" />
17   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
         Profile.owl" />
18   <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
         Grounding.owl" />
19   <owl:imports rdf:resource="http://127.0.0.1/owl_s/ontology/portal.owl"
          />
20   <owl:imports rdf:resource="http://127.0.0.1/owl_s/ontology/dbpedia_3
         .9.owl" />
21   <owl:imports rdf:resource="http://localhost/owl_s/ontology/
         sparqlGrounding.owl" />
22
23 </owl:Ontology>
24
25 <service:Service rdf:ID="COUNTRY_CAPITAL_POP_SERVICE">
26   <service:presents rdf:resource="#COUNTRY_CAPITAL_POP_PROFILE"/>
27   <service:describedBy rdf:resource="#COUNTRY_CAPITAL_POP_PROCESS"/>
28   <service:supports rdf:resource="#COUNTRY_CAPITAL_POP_GROUNDING"/>
29 </service:Service>
30
31 <profile:Profile rdf:ID="COUNTRY_CAPITAL_POP_PROFILE">
32   <service:isPresentedBy rdf:resource="#COUNTRY_CAPITAL_POP_SERVICE"/>
33   <profile:serviceName xml:lang="en">
```

```
34        CountryCapitalPopulationProvider
35    </profile:serviceName>
36    <profile:textDescription xml:lang="en">
37        Service thats given a country name return the population and name
              of its capital.
38    </profile:textDescription>
39    <profile:hasInput  rdf:resource="#_NAME_COUNTRY"/>
40    <profile:hasOutput rdf:resource="#_NAME_CAPITAL"/>
41    <profile:hasOutput rdf:resource="#_POPULATION_CAPITAL"/>
42    <profile:has_process rdf:resource="COUNTRY_CAPITAL_POP_PROCESS" />
43  </profile:Profile>
44
45  <process:AtomicProcess rdf:ID="COUNTRY_CAPITAL_POP_PROCESS">
46    <service:describes rdf:resource="#COUNTRY_CAPITAL_POP_SERVICE"/>
47    <process:hasInput  rdf:resource="#_NAME_COUNTRY"/>
48    <process:hasOutput rdf:resource="#_NAME_CAPITAL"/>
49    <process:hasOutput rdf:resource="#_POPULATION_CAPITAL"/>
50  </process:AtomicProcess>
51
52  <process:Input rdf:ID="_NAME_COUNTRY">
53    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
          anyURI">http://xmlns.com/foaf/0.1/name</process:parameterType>
54    <rdfs:label>name_country</rdfs:label>
55  </process:Input>
56
57
58  <process:Output  rdf:ID="_NAME_CAPITAL">
59    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
          anyURI">http://xmlns.com/foaf/0.1/name</process:parameterType>
60    <rdfs:label>name_capital</rdfs:label>
61  </process:Output>
62
63  <process:Output  rdf:ID="_POPULATION_CAPITAL">
64    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#
          anyURI">http://dbpedia.org/ontology/populationTotal</
          process:parameterType>
65    <rdfs:label>population_capital</rdfs:label>
66  </process:Output>
67
68  <sparql:SparqlGrounding rdf:ID="COUNTRY_CAPITAL_POP_GROUNDING">
69  <service:supportedBy rdf:resource="#COUNTRY_CAPITAL_POP_SERVICE"/>
70  <grounding:hasAtomicProcessGrounding rdf:resource="#
        COUNTRY_CAPITAL_POP_AtomicProcessGrounding"/>
71  </sparql:SparqlGrounding>
72
73  <!-- SPARQL Atomic Process Grounding -->
74
75  <sparql:SparqlAtomicProcessGrounding rdf:ID="
        COUNTRY_CAPITAL_POP_AtomicProcessGrounding">
76  <grounding:owlsProcess rdf:resource="#COUNTRY_CAPITAL_POP_PROCESS"/>
77
78  <sparql:sparqlVersion rdf:datatype="http://www.w3.org/2001/XMLSchema#
        anyURI">
```

```
79    http://www.w3.org/TR/rdf-sparql-query/
80  </sparql:sparqlVersion>
81  <sparql:sparqlEndPoint rdf:datatype="http://www.w3.org/2001/XMLSchema#
      anyURI">http://dbpedia.org/sparql</sparql:sparqlEndPoint>
82
83  <!-- Prefixies -->
84
85  <sparql:SparqlPrefixes>
86    <sparql:SparqlPrefixMap>
87        <sparql:PrefixName>rdf</sparql:PrefixName>
88        <sparql:PrefixUri>http://www.w3.org/1999/02/22-rdf-syntax-ns#</
            sparql:PrefixUri>
89    </sparql:SparqlPrefixMap>
90  </sparql:SparqlPrefixes>
91  <sparql:SparqlPrefixes>
92    <sparql:SparqlPrefixMap>
93        <sparql:PrefixName>foaf</sparql:PrefixName>
94        <sparql:PrefixUri>http://xmlns.com/foaf/0.1/</sparql:PrefixUri>
95    </sparql:SparqlPrefixMap>
96  </sparql:SparqlPrefixes>
97  <sparql:SparqlPrefixes>
98    <sparql:SparqlPrefixMap>
99        <sparql:PrefixName>dbpedia-owl</sparql:PrefixName>
100       <sparql:PrefixUri>http://dbpedia.org/ontology/</sparql:PrefixUri>
101   </sparql:SparqlPrefixMap>
102 </sparql:SparqlPrefixes>
103
104 <!-- Input Parameters -->
105
106 <sparql:SparqlInputParam>
107   <sparql:SparqlInputParamMap>
108     <grounding:owlsParameter rdf:resource="#_NAME_COUNTRY"/>
109     <sparql:SparqlDataParam rdf:datatype="http://www.w3.org/2001/
          XMLSchema#string">?pais_nome</sparql:SparqlDataParam>
110   </sparql:SparqlInputParamMap>
111 </sparql:SparqlInputParam>
112
113 <!-- Ouput Parameters -->
114
115 <sparql:SparqlOutputParam>
116   <sparql:SparqlOutputParamMap>
117     <grounding:owlsParameter rdf:resource="#_NAME_CAPITAL"/>
118     <sparql:SparqlDataParam rdf:datatype="http://www.w3.org/2001/
          XMLSchema#string">?nome_capital</sparql:SparqlDataParam>
119   </sparql:SparqlOutputParamMap>
120 </sparql:SparqlOutputParam>
121
122 <sparql:SparqlOutputParam>
123   <sparql:SparqlOutputParamMap>
124     <grounding:owlsParameter rdf:resource="#_POPULATION_CAPITAL"/>
125     <sparql:SparqlDataParam rdf:datatype="http://www.w3.org/2001/
          XMLSchema#string">?populacao</sparql:SparqlDataParam>
126   </sparql:SparqlOutputParamMap>
```

```
127 </sparql:SparqlOutputParam>
128
129 <!-- Triples -->
130
131 <sparql:SparqlTriples>
132   <sparql:SparqlTripleMap>
133       <sparql:TripleSubject>?pais</sparql:TripleSubject>
134       <sparql:TriplePredicate>foaf:name</sparql:TriplePredicate>
135       <sparql:TripleObject>?pais_nome</sparql:TripleObject>
136   </sparql:SparqlTripleMap>
137 </sparql:SparqlTriples>
138
139 <sparql:SparqlTriples>
140   <sparql:SparqlTripleMap>
141       <sparql:TripleSubject>?pais</sparql:TripleSubject>
142       <sparql:TriplePredicate>dbpedia-owl:capital</
          sparql:TriplePredicate>
143       <sparql:TripleObject>?capital</sparql:TripleObject>
144   </sparql:SparqlTripleMap>
145 </sparql:SparqlTriples>
146
147 <sparql:SparqlTriples>
148   <sparql:SparqlTripleMap>
149       <sparql:TripleSubject>?capital</sparql:TripleSubject>
150       <sparql:TriplePredicate>foaf:name</sparql:TriplePredicate>
151       <sparql:TripleObject>http://dbpedia.org/ontology/Country</
          sparql:TripleObject>
152   </sparql:SparqlTripleMap>
153 </sparql:SparqlTriples>
154
155 <sparql:SparqlTriples>
156   <sparql:SparqlTripleMap>
157       <sparql:TripleSubject>?pais</sparql:TripleSubject>
158       <sparql:TriplePredicate>rdf:type</sparql:TriplePredicate>
159       <sparql:TripleObject>http://dbpedia.org/ontology/Country</
          sparql:TripleObject>
160   </sparql:SparqlTripleMap>
161 </sparql:SparqlTriples>
162
163 <sparql:SparqlTriples>
164   <sparql:SparqlTripleMap>
165       <sparql:TripleSubject>?capital</sparql:TripleSubject>
166       <sparql:TriplePredicate>dbpedia-owl:populationTotal</
          sparql:TriplePredicate>
167       <sparql:TripleObject>?populacao</sparql:TripleObject>
168   </sparql:SparqlTripleMap>
169 </sparql:SparqlTriples>
170
171
172 </sparql:SparqlAtomicProcessGrounding>
173
174 </rdf:RDF>
```