OTÁVIO FREITAS FERREIRA FILHO

SERVIÇOS SEMÂNTICOS: UMA ABORDAGEM RESTFUL

São Paulo

2009

OTÁVIO FREITAS FERREIRA FILHO

SERVIÇOS SEMÂNTICOS: UMA ABORDAGEM RESTFUL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia Elétrica

São Paulo

2009

OTÁVIO FREITAS FERREIRA FILHO

SERVIÇOS SEMÂNTICOS: UMA ABORDAGEM RESTFUL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia Elétrica

Área de Concentração:
Sistemas Digitais

Orientador:
Profª. Drª. Maria Alice Grigas Verella Ferreira

São Paulo

2009

FICHA CATALOGRÁFICA

Ferreira Filho, Otávio Freitas
Serviços semânticos: uma abordagem RESTful / O.F. Ferreira
Filho. -- São Paulo, 2009.
103 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. WEB semântica 2. WEB 2.0 3. XML 4. Ontologia I. Universidade de São Paulo. Escola Politécnica. Departamento de Enge – nharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

Dedico este trabalho aos meus pais, Vera e Otávio, e àqueles que são meu porto-seguro, Marília e Martin.

AGRADECIMENTOS

À professora Maria Alice Grigas Varella Ferreira, pela orientação e pelo constante estímulo transmitido durante todo o trabalho.

Aos amigos Stephen Thomas, Ítalo Santiago Vega, Jacques van Niekerk e a todos que colaboraram, direta ou indiretamente, na execução deste trabalho.

Gênio é um por cento inspiração e noventa e nove por cento transpiração.

(Thomas Alva Edson)

RESUMO

Este trabalho foca na viabilização do desenvolvimento de serviços semânticos de acordo com o estilo arquitetural REST. Mais especificamente, considera-se a realização REST baseada no protocolo HTTP, resultando em serviços semânticos RESTful. A viabilização de serviços semânticos tem sido tema de diversas publicações no meio acadêmico. Porém, a grande maioria dos esforços considera apenas os serviços desenvolvidos sob o estilo arquitetural RPC, através do protocolo SOAP. A abordagem RPC, fortemente incentivada pela indústria de software, é perfeitamente realizável em termos tecnológicos, mas agrega computações e definições desnecessárias, o que resulta em serviços mais complexos, com baixo desempenho e pouca escalabilidade. O fato é que serviços REST compõem a maioria dos serviços disponibilizados na Web 2.0 – nome amplamente adotado para referenciar a atual fase da Web –, notoriamente focada na geração colaborativa de conteúdo. A proposta oferecida por este trabalho utiliza uma seleção específica de linguagens e protocolos já existentes, reforçando sua realizabilidade. Utiliza-se a linguagem OWL-S como ontologia de serviços e a linguagem WADL para a descrição sintática dos mesmos. O protocolo HTTP é utilizado na transferência das mensagens, na definição da ação a ser executada e no escopo de execução desta ação. Identificadores URI são utilizados na definição da interface de acesso ao serviço. A compilação final dá origem à ontologia `RESTfulGrounding`, uma especialização de OWL-S.

Palavras-chave: Web Semântica. Web 2.0. Ontologia. XML.

ABSTRACT

The proposal is to allow the development of semantic Web services according to an architectural style called REST. More specifically, it considers a REST implementation based on the HTTP protocol, resulting in RESTful Semantic Web Services. The development of semantic Web services has been the subject of various academic papers. However, the predominant effort considers Web services designed according to another architectural style named RPC, mainly through the SOAP protocol. The RPC approach, strongly stimulated by the software industry, aggregates unnecessary processing and definitions that make Web services more complex than desired. Therefore, services end up being not as scalable and fast as possible. In fact, REST services form the majority of Web services developed within the Web 2.0 context, an environment clearly focused on user-generated content and social aspects. The proposal presented here makes use of a specific selection of existing languages and protocols, reinforcing its feasibility. Firstly, OWL-S is used as the base ontology for services, whereas WADL is for syntactically describing them. Secondly, the HTTP protocol is used for transferring messages; defining the action to be executed; and also defining the execution scope. Finally, URI identifiers are responsible for specifying the service interface. The final compilation proposed results in an ontology named `RESTfulGrounding`, which extends OWL-S.

Keywords: Semantic Web. Web 2.0. Ontology. XML.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BBC	British Broadcasting Corporation
CNN	Cable News Network
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DOD	United States Department of Defense
FOAF	Friend of a Friend
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IOPE	Inputs, Outputs, Preconditions and Effects
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Ontology Web Language
OWL-DL	Ontology Web Language Description Logic
OWL-S	Ontology Web Language for Services
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RIF	Rule Interchange Format
ROA	Resource Oriented Architecture
RPC	Remote Procedure Call
RuleML	Rule Modeling Language
S3	Simple Storage Service
SMW	Semantic Media Wiki
SOAP	Simple Object Access Protocol
SPARQL	Simple Protocol and RDF Query Language
SQL	Structures Query Language
SWRL	Semantic Web Rule Language
UDDI	Universal Description, Discovery and Integration
UMBC	University of Maryland Baltimore County
UML	Unified Modeling Language
UOL	Universo Online

URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WS-I	Web Service Interoperability Organization
WSDL	Web Service Description Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformation
YNS	Yahoo’s News Search

SUMÁRIO

1 INTRODUÇÃO

Este capítulo introdutório tem por objetivo inicial apresentar a contextualização do tema pesquisado, as justificativas para a elaboração da pesquisa, e as motivações que levaram à escolha do tema. Contudo, seu principal foco está na apresentação dos objetivos e das contribuições acadêmicas propostas. Adicionalmente, também são expostas a metodologia e a organização do trabalho.

1.1 Contextualização

Notoriamente, a Web tem sido a principal fonte de informação da última década, continuamente transformando e acelerando os processos de consumo e publicação de conteúdo. Ao longo de sua evolução, a Web vivenciou três fases distintas, principalmente marcadas por mudanças conceituais, e não tecnológicas.

A primeira fase focou principalmente no consumo de conteúdo. A maior parte das informações era, geralmente, disponibilizada por provedores coorporativos, como empresas com foco publicitário, organizações e serviços noticiários (KOLBITSCH; MAURER, 2006). Páginas textuais estáticas eram oferecidas por este grupo limitado de provedores, que detinham o conhecimento necessário para a publicação de conteúdo eletrônico.

Com a vasta adoção da Web ao redor do mundo, as técnicas de publicação tornaram-se populares, e ferramentas de fácil manipulação permitiram que usuários comuns da Web publicassem seu próprio conteúdo. Esta nova e atual fase – marcada pela interação, colaboração e comunicação entre os usuários – foi nomeada “Web 2.0”.

Aplicações desenvolvidas de acordo com os princípios da Web 2.0 são aquelas continuadamente atualizadas, que se tornam melhores quanto maior o número de usuários; que incentivam estes usuários a consumir, produzir e combinar conteúdo; que oferecem seu próprio conteúdo ao mesmo tempo em que oferecem serviços para extração e reutilização deste conteúdo; que criam efeitos de rede através de uma arquitetura de participação (O'REILLY, 2007).

Como descrito, aplicações Web 2.0 tipicamente oferecem um serviço responsável por expor o conteúdo gerado por seus usuários. Desta forma, outras aplicações têm a possibilidade de acessar e reutilizar este conteúdo para outras finalidades, geralmente diferentes das intenções originais da aplicação provedora. Existem inúmeras definições para o termo Serviço Web, geralmente conectadas pelo grupo de linguagens e protocolos citados. A seguir, colocam-se três delas:

- Um Serviço Web é um sistema de software projetado para suportar a interação interoperável entre máquinas em uma rede. Este serviço possui uma interface descrita em um formato processável por máquina (especificamente o WSDL[1]). Outros sistemas interagem com o serviço Web de acordo com esta interface, utilizando mensagens SOAP[2], tipicamente enviadas via HTTP[3] com uma serialização em XML[4] (HAAS; BROWN, 2004).

- Serviços Web são componentes de software que são desenvolvidos usando tecnologias específicas a partir de três categorias de tecnologias primárias: um formato de descrição baseado em XML (por exemplo, WSDL), um protocolo de mensagem de aplicação (por exemplo, SOAP) e um protocolo de transporte (por exemplo, HTTP) (ADAMS et al., 2002).

- Um Serviço Web é um componente de software descrito via WSDL e capaz de ser acessado via protocolos de rede padrão, como – mas não limitado ao – SOAP e HTTP (BROBERG, 2002).

[1] Web Service Description Language

[2] Simple Object Access Protocol

[3] Hypertext Transfer Protocol

[4] Extensible Markup Language

A expressiva publicação de conteúdo existente nas aplicações Web 2.0 e a comunicação interoperável viabilizada pelos serviços Web podem ser consideradas importantes catalisadores no crescimento exponencial da Web. Um estudo relativamente recente, realizado por Gulli e Signorini (2005), estimava que a Web teria por volta de 11,5 bilhões de páginas indexadas por mecanismos de busca, naquela ocasião. Neste contexto, é fundamental que existam critérios rigorosos a serem aplicados na busca por informação. Filtros baseados em similaridade de palavras-chave são imprecisos e suas deficiências são observadas há anos. Segundo Pretschner e Gauch (1999), “termos de busca são ambíguos; seus significados dependem do contexto e, de forma mais importante, do significado que o usuário atribui a eles”.

Diante de tais limitações, torna-se necessária a anotação semântica tanto dos dados já existentes, quanto dos dados a serem publicados. Esta é a força condutora da terceira fase da Web, nomeada Web Semântica, e introduzida por Berners-Lee, Hendler e Lassila (2001). Seus idealizadores a definem não como uma Web separada, mas como uma extensão da existente, na qual a informação recebe um significado bem definido, melhor capacitando os computadores e as pessoas a trabalharem em cooperação.

Muitos autores já tratam da necessidade do estabelecimento de conexões entre a Web 2.0 e a Web Semântica. Ankolekar et al. (2008) defendem que ambas as abordagens são complementares, e que a Web 2.0 continuará focada nos aspectos sociais e de usabilidade, enquanto que a infra-estrutura disponibilizada pela Web Semântica seria utilizada para a combinação e compartilhamento dos dados. Bojars et al. (2008) defendem a utilização da Web Semântica como ferramenta para a ligação e reutilização de dados entre as redes sociais existentes na Web 2.0. Gruber (2008) une a também chamada Web Social com a Web Semântica para a criação de sistemas de conhecimento coletivo. Neste trabalho, o autor utiliza as técnicas de representação de conhecimento oferecidas pela Web Semântica para representar a inteligência coletiva que emana da Web 2.0. Por fim, Hendler e Golbeck (2008) buscam extrair valor da combinação entre as redes sociais e as redes de conceitos semânticos.

Ao passo que a Web 2.0 é uma realidade vivenciada por milhões de usuários na Internet, a Web Semântica ainda encontra barreiras, inclusive tecnológicas. Porém, existe claramente uma forte tendência acadêmica que atua no sentido de aproximar as duas áreas, o que provavelmente irá acelerar o processo de adoção do novo paradigma.

Evidentemente, serviços Web também são afetados pela abordagem semântica. Como visto anteriormente nesta seção, serviços Web são freqüentemente definidos de acordo com um grupo específico de linguagens e protocolos. As definições citadas apresentam tamanha similaridade porque definem serviços pertencentes à mesma classificação, conhecida como serviços RPC (*Remote Procedure Call*), onde o SOAP é o protocolo predominante. Por ser uma classificação intensamente explorada em pesquisas e pela indústria, as primeiras iniciativas de viabilização de serviços semânticos trataram exclusivamente de serviços RPC, especialmente os serviços SOAP. Entretanto, a grande maioria das aplicações Web 2.0 oferece serviços classificados como REST (*Representational State Transfer*), onde o HTTP é o principal protocolo.

Dada esta lacuna nas pesquisas sobre serviços semânticos, este trabalho propõe uma combinação específica de linguagens e protocolos já existentes para possibilitar a criação de serviços semânticos de acordo com o paradigma REST. Tal combinação possibilitará que as inúmeras aplicações sociais existentes atualmente possam migrar suas interfaces para o mundo semântico.

1.2 Justificativas

Serviços Web desenvolvidos de acordo com a abordagem RPC/SOAP adicionam inúmeras abstrações como, por exemplo, a especificação WS-Security (OASIS, 2006).

Focada em requisitos não-funcionais relacionados à segurança, a especificação WS-Security ainda inclui outras abstrações como WS-Policy, WS-Trust, WS-SecureConversation, entre outras.

Apesar de cumprir com seu objetivo original, tais abstrações desfavorecem outros requisitos não-funcionais igualmente importantes, pois geram pontos de falha e problemas de escalabilidade, além de prejudicar a interoperabilidade (RICHARDSON; RUBY, 2007). Adicionalmente, processamento excessivo e baixo desempenho são apontados como conseqüências comuns da utilização de serviços RPC, principalmente os que adotam o protocolo SOAP e suas extensões focadas em segurança (LIU; PALLICKARA; FOX, 2005; MAKINO et al., 2005).

Segundo classificação apresentada por Richardson e Ruby (2007), existem três grupos distintos de serviços Web: serviços RPC, serviços REST-RPC, e serviços RESTful. Este terceiro grupo, um dos principais objetos de estudo desta pesquisa, contém os serviços que melhor realizam o estilo arquitetural REST, além de atrelar a implementação desses serviços ao protocolo HTTP. Todos os serviços, independentemente do grupo ao qual pertencem, recebem requisições e enviam respostas. Portanto, cada grupo é formado de acordo com as respostas para as seguintes questões:

- A forma como a ação a ser executada é discriminada na requisição;
- A forma como o escopo de execução desta ação é discriminado na requisição.

Abreviadamente, serviços RPC são aqueles que recebem todos os detalhes sobre a requisição no corpo da mensagem. De acordo com esta abordagem, a ação a ser executada, assim como o escopo de execução desta ação, são informações enviadas como atributos do documento presente no corpo da requisição. Este documento é tipicamente enviado no formato SOAP, uma sintaxe derivada do padrão XML.

O exemplo na Figura 1 representa uma requisição a ser enviada ao popular serviço de busca disponibilizado pelo Google. Esta API (*Application Programming Interface*) – como também são chamados os serviços Web – foi desenvolvida de acordo com o protocolo SOAP, e não é mais suportada pelo Google desde dezembro de 2006.

```
01 POST /search/beta2/ HTTP/1.1
02 Host: api.google.com
03 Content-Type: application/soap+xml
04 SOAPAction: urn:GoogleSearchAction
05
06 <?xml version="1.0" encoding="UTF-8"?>
07 <soap:Envelope xlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
08   <soap:Body>
09     <gs:doGoogleSearch xmlns:gs="urn:GoogleSearch">
10       <q>universidade</q>
11     </gs:doGoogleSearch>
12   </soap:Body>
13 </soap:Envelope>
```

Figura 1 – Requisição SOAP ao serviço de busca disponibilizado pelo Google

Como pode ser claramente observado, a ação a ser executada (`doGoogleSearch`, linha 9) é discriminada pelo próprio documento SOAP (linhas 7 a 13). O mesmo documento ainda define o escopo de atuação desta ação, ou seja, páginas que contenham a palavra-chave "universidade" (linha 10). Desta forma, todas as requisições ao serviço são enviadas para o mesmo endereço, independentemente da ação a ser executada. Neste exemplo, o endereço completo da API é `http://api.google.com/search/beta2/`.

Alternativamente, serviços pertencentes ao segundo grupo, nomeado REST-RPC, utilizam o próprio endereço da requisição para definir tanto a ação a ser executada, quanto o escopo de execução desta ação. Esta abordagem utiliza o protocolo HTTP apenas como um formato para a transferência de envelopes, ou seja, esta ainda não extrai todo o poder e flexibilidade que são oferecidos pelo protocolo. Por outro lado, toda a complexidade existente no SOAP é descartada, uma vez que este protocolo não é utilizado.

O exemplo da Figura 2 apresenta uma requisição a ser enviada ao serviço de manipulação de fotos chamado Flickr, muito popular entre as aplicações Web 2.0. Este serviço é disponibilizado pelo Yahoo.

```
01 GET /services/rest/?method=flickr.photos.search&tags=Brasil HTTP/1.1
02 Host: www.flickr.com
```

Figura 2 – Requisição REST-RPC ao serviço de fotos chamado Flickr

Neste exemplo da Figura 2, o consumidor do serviço está solicitando uma coleção de fotos marcadas com a palavra-chave “Brasil” (na linha 01). Como pode ser observado, o método a ser executado (`flickr.photos.search`, na linha 01) é discriminado pelo próprio endereço da requisição, ou seja, o consumidor claramente solicita a execução de um procedimento remoto, como acontece em serviços RPC. O escopo de execução também é definido no endereço da requisição, o que torna o serviço semelhante aos serviços RESTful, como será descrito a seguir. Afirma-se, portanto, que este segundo grupo reúne serviços híbridos.

Finalmente, o terceiro grupo de serviços Web – nomeado RESTful – representa uma fiel realização do estilo arquitetural REST, paradigma proposto por Fielding (2000). Serviços RESTful são aqueles que melhor utilizam todo o potencial oferecido pelo protocolo HTTP. O capítulo 2 apresenta um extenso detalhamento sobre este grupo de serviços Web, uma vez que se encontra no núcleo da proposta deste trabalho. Resumidamente, serviços RESTful utilizam o próprio método HTTP para discriminar o método remoto a ser executado, ao passo que o endereço da requisição define o escopo da execução.

O serviço S3 (*Simple Storage Service*) é integrante do novo grupo de serviços de infra-estrutura disponibilizado pela empresa Amazon, tradicionalmente reconhecida por seu serviço de comércio eletrônico. O S3 foi projetado de acordo com o estilo arquitetural REST, e oferece uma interface extremamente simples que viabiliza o armazenamento de arquivos binários. As Figuras 3, 4 e 5 apresentam exemplos de uso.

```
01 PUT /object-id HTTP/1.1
02 Host: bucket-id.s3.amazonaws.com
```

Figura 3 – Requisição RESTful para criação de um novo objeto no repositório S3

```
01 GET /object-id HTTP/1.1
02 Host: bucket-id.s3.amazonaws.com
```

Figura 4 – Requisição RESTful para recuperação do objeto armazenado no repositório S3

```
01 DELETE /object-id HTTP/1.1
02 Host: bucket-id.s3.amazonaws.com
```

Figura 5 – Requisição RESTful para deleção do objeto armazenado no repositório S3

Os exemplos apresentam diversas ações realizadas sobre o mesmo objeto: criação, recuperação e exclusão, correspondendo às operações PUT (Figura 3), GET (Figura 4) e DELETE (Figura 5). Por seguir um estilo arquitetural orientado a recursos, cada objeto recebe seu próprio identificador, ou URI (*Uniform Resource Identifier*). Este é o escopo da execução. Desta forma, o método HTTP torna-se o responsável por definir a ação a ser executada (GET, HEAD, POST, PUT, DELETE). O capítulo 2 detalha cada componente existente em um serviço RESTful.

As concisas apresentações dos grupos, assim como os exemplos ilustrativos, formam a base para a listagem das justificativas sobre a escolha do tema deste trabalho. Inicialmente, constata-se que novos protocolos tentam contornar a Web, ao invés de utilizá-la em todo o seu potencial. O protocolo HTTP/1.1 (IETF, 1999) e o identificador URI (IETF, 2005) formam a base dos serviços RESTful, assim como a base de todo o restante da Web. Na realidade, a portabilidade, a flexibilidade e a simplicidade do protocolo HTTP podem tornar os serviços Web tão populares quanto as próprias páginas Web. A única diferença efetiva entre serviços e páginas Web é o fato das páginas serem projetadas para serem apresentadas graficamente e consumidas por leitores humanos, enquanto que os serviços são projetados para o consumo automatizado via software. De uma forma ou de outra, as tecnologias e os padrões devem ser exatamente os mesmos.

O processo de popularização já é uma realidade, em vista da grande quantidade de APIs RESTful disponibilizadas pelas aplicações Web 2.0. Porém, na área dos serviços semânticos, observa-se uma grande escassez de publicações sobre serviços RESTful. Praticamente todos os esforços são direcionados para os serviços SOAP, o que exclui grande parte do desenvolvimento atual da Internet, além de agregar uma indesejável sobrecarga de processamento e definições. Grupos como W3C (*World Wide Web Consortiu*m), OASIS (*Organization for the Advancement of Structured Information Standards*) e WS-I (*Web Services Interoperability Organization*) têm produzido uma grande quantidade de novas extensões para o protocolo SOAP, ao passo que nenhuma contribuição efetiva ao paradigma REST foi apresentada.

O presente estudo aproxima o desenvolvimento de serviços semânticos do atual estado da Web, permitindo a migração dos serviços RESTful ao mundo semântico.

1.3 Objetivos

O objetivo principal desde trabalho é estabelecer um conjunto específico de padrões que possibilitem a existência de serviços semânticos projetados de acordo com o estilo arquitetural REST. Para o estabelecimento deste conjunto de padrões, esta pesquisa se propõe a reutilizar protocolos e linguagens já definidas pelas entidades competentes.

Adicionalmente, este trabalho defende o HTTP como um protocolo suficientemente capaz de lidar com a significativa complexidade agregada aos serviços semânticos.

A fim de atingir estes objetivos, pretende-se também:

- Apresentar todos os conceitos e princípios que formam um serviço RESTful.
- Apresentar uma visão geral sobre a Web Semântica, principalmente no que diz respeito aos conceitos Ontologia e Agente, além da arquitetura baseada em camadas que guia a evolução da terceira fase da Web, chamada por alguns de Web 3.0.
- Apresentar o estado da arte em serviços semânticos, principalmente em relação ao padrão de notação OWL-S (*Ontology Web Language for Services*).
- Investigar alternativas bem documentadas para protocolos e linguagens fortemente atreladas ao SOAP.
- Definir e documentar uma ontologia Web para a descrição semântica de serviços RESTful.
- Aplicar a ontologia resultante em algum serviço bem estabelecido no contexto da Web 2.0, validando a proposta.

1.4 Contribuições

A contribuição inicial está relacionada à mudança de foco no estudo de serviços semânticos, pois evidencia a importância da inclusão da abordagem RESTful numa área predominantemente ocupada por serviços SOAP.

De forma concreta, o presente estudo apresenta as seguintes contribuições:

- A utilização da linguagem WADL (*Web Application Description Language*) para a descrição sintática de serviços semânticos RESTful. Paralelamente, a linguagem WSDL continuaria sendo utilizada para a descrição de serviços semânticos SOAP. Ao descrever sintaticamente um serviço RESTful, a linguagem WADL permite o consumo automático deste serviço pelos agentes da Web Semântica.
- O suporte à linguagem OWL-S como ontologia para a descrição semântica de serviços Web.
- A proposta da ontologia `RESTfulGrounding` como extensão da linguagem OWL-S. Tal especialização seria projetada a partir da classe abstrata OWL-S `ServiceGrounding`, principal responsável por descrever como um serviço semântico qualquer realiza, tecnologicamente, o comportamento descrito.
- A definição formal de cada uma das classes que compõem a ontologia `RESTfulGrounding`, como por exemplo `WadlServiceGrounding`, classe concreta que referencia um documento WADL. Outro exemplo igualmente importante é a classe `WadlAtomicServiceGrounding`, responsável por descrever o mapeamento semântico-sintático de processos atômicos. Assim como geralmente é feito para as demais ontologias da Web Semântica, as definições serão oferecidas na sintaxe OWL (*Ontology Web Language*).
- A aplicação da nova ontologia `RESTfulGrounding` para exemplificar sua utilização em um contexto factível.

1.5 Metodologia

A metodologia utilizada durante a pesquisa compreendeu os seguintes passos:

- Investigação sobre o real poder dos componentes que formam um serviço RESTful, cobrindo conceitos como Recurso, Representação, Identificador Uniforme, Interface Unificada e Escopo de Execução, além dos princípios da Endereçabilidade, do Estado Não-Persistente e da Conectividade;
- Compreensão das tecnologias que apóiam o desenvolvimento de serviços semânticos, como as linguagens XML e WSDL, os protocolos HTTP e SOAP, o registro UDDI (*Universal Description, Discovery and Integration*) e as ontologias OWL e OWL-S;
- Seleção e agrupamento dos padrões a serem utilizados na definição formal, descobrimento e consumo de serviços semânticos RESTful;
- Aplicação de conceitos a fim de testar a viabilidade das contribuições propostas;
- Formalização dos resultados obtidos;
- Avaliação dos resultados.

1.6 Organização

Além desta introdução, o trabalho encontra-se segmentado nos capítulos descritos resumidamente a seguir:

O capítulo 2 – Serviços Web – apresenta o conceito geral sobre serviços Web, além de uma descrição detalhada sobre serviços RESTful.

O capítulo 3 – Web Semântica – apresenta a terceira geração da Web, cobrindo os principais conceitos, arquitetura, protocolos e tecnologias.

O capítulo 4 – Serviços Semânticos – trata do atual entendimento sobre serviços na Web Semântica.

O capítulo 5 – Serviços Semânticos RESTful – foca no estabelecimento de um grupo de linguagens e protocolos para suportar o desenvolvimento de serviços Web de acordo com o paradigma RESTful. Este capítulo também oferece um exercício de aplicação de conceitos para o desenvolvimento de serviços semânticos RESTful.

O capítulo 6 – Considerações Finais – lista e descreve conclusões, limitações e pontos a serem aprimorados em trabalhos futuros.

2 SERVIÇOS WEB

Os Serviços Web garantem a comunicação entre aplicações de forma interoperável e independente de plataforma. Eles representam uma das abordagens mais promissoras para o reuso de software em ambientes distribuídos, valorizando os ativos de software já existentes.

Através de Serviços Web, qualquer aplicação de software na Web tem o potencial para alcançar qualquer outra. As aplicações que trocam mensagens de forma compatível aos padrões estabelecidos para serviços Web podem se comunicar independentemente do sistema operacional, linguagem de programação, protocolos internos e processador (BREITMAN; CASANOVA; TRUSZKOWSKI, 2007).

Neste capítulo discutem-se os Serviços Web desenvolvidos segundo o paradigma RESTful, seus conceitos e princípios.

2.1 Serviços RESTful

REST é um estilo arquitetural para sistemas hipermídia distribuídos que enfatiza a generalização das interfaces, a escalabilidade da interação entre os componentes e a instalação independente dos mesmos (FIELDING, 2000). O paradigma REST reúne um grupo de critérios a serem incorporados ao projeto de aplicações distribuídas, porém não existe qualquer conexão direta entre este paradigma e algum protocolo em específico.

Esta seção apóia-se principalmente nos conceitos apresentados por Richardson e Ruby (2007), que cunharam o termo “Serviços RESTful” para designar Serviços Web que seguem os critérios defendidos pelo paradigma REST. Adicionalmente, os

autores conectaram REST ao protocolo HTTP, trazendo um aspecto tecnológico concreto àquele grupo de critérios.

Pouco tempo após sua definição, o HTTP foi revisado e definido pelo W3C como o protocolo padrão para a transmissão de mensagens na Web. Tal compatibilidade faz com que todo sistema projetado de acordo com o paradigma RESTful tenha uma potencial audiência composta por todos os dispositivos conectados à Web, ou seja, um grupo virtualmente infinito. Outra conseqüência igualmente benéfica é a utilização da própria Web como infra-estrutura de distribuição e acesso, o que torna o sistema inteiramente portável, sem qualquer dependência adicional em termos que hardware ou software.

Ao conectar REST ao HTTP, Richardson e Ruby (2007) definem ROA (*Resource-Oriented Architecture*), uma arquitetura que segue fielmente o estilo arquitetural REST, ao mesmo tempo em que utiliza o HTTP para preencher as principais lacunas teóricas com realizações concretas. A arquitetura ROA nos mostra que o protocolo HTTP vinha sendo subaproveitado até então, e que protocolos como o SOAP trazem uma camada de complexidade, geralmente, desnecessária.

A lista a seguir apresenta os principais conceitos e princípios que formam a arquitetura ROA:

- Recurso
- Representação
- Identificador Uniforme
- Interface Unificada
- Escopo de Execução
- Princípio da Endereçabilidade
- Princípio do Estado Não-Persistente
- Princípio da Conectividade

Como todo Serviço Web, um Serviço RESTful recebe uma requisição discriminando a computação a ser executada, e retorna uma resposta, detalhando o resultado obtido. Porém, duas diferenças fundamentais são observadas nas requisições RESTful – já mencionadas anteriormente –, como segue:

- A forma como a ação a ser executada é discriminada na requisição;
- A forma como o escopo de execução desta ação é discriminado na requisição.

As próximas subseções detalham ambos os pontos, respectivamente, através dos conceitos Interface Unificada e Escopo de Execução. Entretanto, para que tal detalhamento seja possível, as subseções que seguem detalham, na realidade, todos os conceitos e princípios que compõem uma arquitetura ROA.

2.1.1 Conceitos

Serviços RESTful são compostos por cinco conceitos – Recurso, Representação, Identificador Uniforme, Interface Unificada e Escopo de Execução. As próximas seções detalham cada um desses conceitos de forma individual.

2.1.1.1 Recurso

Um recurso é uma abstração ou conceito relevante que existe no domínio tratado pelo serviço em questão. O projetista deste serviço tem plena liberdade para selecionar qualquer objeto do domínio, seja este objeto real ou fictício, concreto ou abstrato. Alguns exemplos práticos são listados abaixo:

- O livro “Utilizando UML e Padrões”, ISBN 85-363-0358-1;
- A Escola Politécnica da Universidade de São Paulo;
- O autor Ivar Jacobson;
- O álbum “Help”, dos Beatles;

- O mapa de pontos turísticos da cidade de Santos;
- A coleção de todas as composições do músico Damien Rice;
- O humor atual do participante Daniel Dias na rede social “Facebook”.

Como observado, existe uma enorme flexibilidade na definição dos recursos gerenciados pelo serviço. Um livro, uma universidade, uma pessoa ou um álbum são recursos materiais, concretos. Por outro lado, é possível que um serviço trate especificamente do humor dos usuários de uma rede social, fazendo com que o humor passe a ser o próprio recurso gerenciado, ou seja, um recurso abstrato.

Adicionalmente, um recurso pode compreender um grupo – ao invés de um único objeto – como exemplificado pela coleção de composições de Damien Rice. Mesmo nestes casos, é correto tratar a coleção como um único recurso.

Alguns serviços lidam com mais de um recurso e isso não afeta a qualidade RESTful deste componente de forma alguma. Apenas para ilustrar este caso, podemos imaginar um serviço que gerencia tanto arquivos de vídeo quanto arquivos de áudio, produzidos pelos usuários em uma aplicação multimídia.

2.1.1.2 Representação

Os serviços manipulam as representações dos recursos e não os recursos propriamente ditos, pois estes são apenas abstrações. Seria impossível, por exemplo, o tráfego, via rede, de um objeto material como um livro, ou de um não-material, como o humor.

Conceitualmente, uma representação é qualquer informação útil sobre o estado do recurso (RICHARDSON; RUBY, 2007). Cada serviço atua em um determinado nível de abstração, oferecendo uma quantidade maior ou menor de informação sobre o recurso.

Tecnicamente, uma representação é uma serialização do recurso na sintaxe escolhida. Alguns formatos comumente utilizados são: XML, XHTML (*Extensible Hypertext Markup Language*), JSON (*JavaScript Object Notation*), RDF (*Resource Description Framework*), entre outros. Linguagens derivadas do padrão XML, como RDF, ganham força num contexto semântico, uma vez que permitem anotações sobre os dados.

É perfeitamente viável que um determinado serviço ofereça mais de um tipo de serialização para seus recursos. Neste caso, a requisição deve ser clara quanto ao formato desejado. Esta informação é freqüentemente enviada através de um cabeçalho HTTP, nomeado `Accept`. O Quadro 1 apresenta a lista deles para as linguagens mencionadas.

Formato	Cabeçalho
XML	`application/xml`
XHTML	`application/xhtml+xml`
RDF	`application/rdf+xml`
JSON	`application/json`

Quadro 1 – Formatos para representação de recurso e respectivos valores para o cabeçalho HTTP

2.1.1.3 Identificador Uniforme

Cada recurso está necessariamente associado a pelo menos um identificador uniforme (URI) que atua simultaneamente como nome e localizador deste recurso. Caso um determinado objeto não tenha um URI associado, não poderá ser considerado um recurso. Entretanto, um único recurso pode ser referenciado por um número ilimitado de URIs.

Pode-se afirma que o endereço http://www.poli.usp.br/ é um URI válido para o recurso “Escola Politécnica da Universidade de São Paulo”. Em um exemplo imaginário, poderíamos afirmar que o endereço http://mapas.com.br/turistico/santos/ é um URI válido para o recurso “Mapa turístico da cidade de Santos”.

Note-se que identificadores devem ser descritivos. Imagine-se que o mapa mencionado é um recurso gerenciado por um serviço que também oferece mapas rodoviários e geográficos. Pois bem, a abordagem RESTful defende a correlação intuitiva entre os identificadores, como exemplificado a seguir:

- Mapa turístico de Salvador: http://www.mapas.com.br/turistico/salvador/
- Mapa rodoviário de São Paulo: http://www.mapas.com/rodoviario/sao+paulo/
- Mapa geográfico de Recife: http://www.mapas.com.br/geografico/recife/
- Coleção de todos os mapas de Manaus: http://www.mapas.com.br/manaus/
- Coleção dos mapas geográficos: http://www.mapas.com.br/geografico/

Os identificadores devem respeitar uma estrutura, uma hierarquia e um padrão de notação, o que facilita a análise do URI recebido pelo serviço. Todas as regras são definidas pelo próprio serviço. Uma estrutura planejada de URI faz com que os consumidores do serviço possam criar os seus próprios pontos de entrada através de alterações nos argumentos que compõem o identificador. Um URI tem o poder de conectar dois recursos distintos, aspecto fundamental para a criação de redes ou grafos de recursos.

2.1.1.4 Interface Unificada

Segundo o paradigma RESTful, a ação a ser executada é definida diretamente pelo método HTTP, também conhecido como “verbo HTTP”. O protocolo oferece cinco métodos principais: GET, HEAD, POST, PUT e DELETE. Todos os métodos são aplicados nos recursos, ou seja, nos objetos gerenciados pelo serviço. Mais especificamente, um método HTTP é executado para um determinado URI.

O protocolo HTTP ainda define os métodos OPTIONS, TRACE e CONNECT, mas estes não foram incorporados (até agora) pela arquitetura ROA.

Apesar da simplicidade, esta característica é extremamente poderosa, pois define uma interface unificada para todos os serviços. Caso um determinado consumidor conheça os recursos oferecidos por um determinado serviço, ele automaticamente conhece os processos de recuperação, criação, modificação e remoção destes recursos. Isso é possível graças à existência de uma interface unificada. Adicionalmente, esta característica fortalece o caráter interoperável dos serviços.

O método GET trata da recuperação real da representação de um recurso, ao passo que HEAD retorna apenas os meta-dados que descrevem o recurso afetado pela requisição. Meta-dados são retornados através de cabeçalhos HTTP.

O método HEAD é geralmente aplicado para reduzir a quantidade de bytes que trafegam como resposta à requisição por um recurso binário, potencialmente muito grande. Em algumas situações, o consumidor do serviço está interessado unicamente nos meta-dados do recurso, e não no recurso propriamente dito. O método HEAD torna-se um grande aliado na busca por escalabilidade e desempenho.

Já os métodos POST e PUT tratam da inserção e atualização de recursos. A especificação HTTP é um tanto abrangente na comparação entre estes dois métodos, mas a abordagem RESTful é absolutamente clara. Caso exista um recurso gerador de outros recursos, uma mensagem POST enviada ao gerador deve provocar a criação de um recurso aninhado.

Por exemplo, considere-se um recurso chamado “Lista de assinantes do jornal Metro”. Pois bem, uma requisição POST enviada a este recurso deve resultar na criação de um novo assinante para o jornal. Conseqüentemente, o recurso resultante é uma instância do tipo “Assinante do jornal Metro”. A partir deste momento, qualquer requisição PUT enviada diretamente à instância criada deve provocar a atualização dos dados dessa instância.

Por outro lado, caso o consumidor do serviço tenha total conhecimento e controle sobre o recurso a ser criado, uma requisição PUT deve ser enviada diretamente à instância, assim como na alteração/atualização do recurso.

Em outras palavras, nos casos onde o URI do novo recurso é gerado pelo próprio serviço, o consumidor deve enviar uma requisição POST ao URI do gerador, ou

recurso-fábrica. Por outro lado, caso o consumidor possa gerar o URI deste novo recurso, o próprio consumidor deve enviar uma requisição PUT para o URI da nova instância, diretamente.

Por fim, o método DELETE deve ser enviado para a remoção do recurso. Alguns serviços preferem executar uma técnica conhecida como *soft-delete*, onde o recurso não é efetivamente removido do repositório, mas apenas marcado como inativo. Outros preferem executar o *hard-delete*, o que faz com que o recurso nunca mais possa ser acessado, pois já não existe na base de dados. A especificação do serviço deve cobrir esta decisão.

2.1.1.5 Escopo de Execução

Sobre a definição do escopo de execução do método, o paradigma RESTful defende a utilização do URI presente na requisição HTTP para este fim. Nesta abordagem, o URI contém não apenas o caminho do serviço em questão, mas também qualquer parâmetro necessário para identificação única do recurso afetado.

Observe-se o primeiro exemplo na Figura 6, onde uma aplicação cliente envia uma mensagem HTTP utilizando o método GET, e um URI que identifica a foto de número 123. Neste cenário fictício, `galeria.com` seria uma aplicação de gerenciamento de fotos, que oferece uma API pública de acordo com os princípios RESTful. Neste momento, a aplicação quer recuperar a foto armazenada em `galeria.com`.

```
01 GET /foto/123 HTTP/1.1
02 Host: galeria.com
```

Figura 6 – Requisição RESTful utilizando o método HTTP GET

No segundo exemplo (Figura 7), a requisição HTTP enviada utiliza o método DELETE, o que significa que a foto 123 será removida da base.

```
01 DELETE /foto/123 HTTP/1.1
02 Host: galeria.com
```

Figura 7 – Requisição RESTful utilizando o método HTTP DELETE

Note-se que apenas o método HTTP foi alterado, enquanto que o URI permaneceu intacto. Esta é uma característica marcante dos serviços RESTful: aquele URI sempre identifica o mesmo recurso, ou seja, a foto 123; este é o escopo de execução. Já o método HTTP apenas diz qual a ação a ser executada: recuperação, no primeiro exemplo e exclusão, no segundo.

2.1.2 Princípios

Adicionalmente, serviços RESTful devem realizar os princípios da Endereçabilidade, do Estado Não-Persistente e da Conectividade. As próximas seções detalham cada princípio individualmente.

2.1.2.1 Endereçabilidade

Serviços são classificados como endereçáveis quando seu conjunto de dados é exposto como uma série de recursos, cada um com seu respectivo URI (RICHARDSON; RUBY, 2007). Este é o primeiro passo que permite ao consumidor do serviço acessar e compartilhar um determinado recurso diretamente.

Antagonicamente, serviços não-endereçáveis tipicamente apresentam apenas um URI para todo o seu conjunto de dados. Neste caso, o URI está na realidade

nomeando e endereçando o próprio serviço, e não as porções de dados significativas para o consumidor. Este é o caso dos serviços RPC.

Aplicações Web também podem ser classificadas de acordo com esta propriedade, e não apenas serviços. As que cumprem o requisito expõem cada uma de suas páginas como um recurso endereçável, ou seja, uma porção de dados identificada por um URI. Usuários da Internet, intuitivamente, se familiarizaram e tiraram bom proveito de aplicações Web endereçáveis. O envio de identificadores URI é prática comum entre os usuários, que encontraram nos recursos endereçáveis uma excelente forma para compartilhar suas experiências na Internet.

Porém, existe uma variação de aplicação Web que expõe apenas um URI para todo o conteúdo. Estas aplicações utilizam um grupo de tecnologias bem estabelecidas, conhecido como *Asynchronous JavaScript and XML* (AJAX). Com o passar do tempo, os desenvolvedores perceberam que outras tecnologias igualmente bem estabelecidas também eram capazes de oferecer o mesmo comportamento. Portanto, estas aplicações passaram a ser referenciadas como aplicações Ajax, e não mais através do acrônimo.

Estas aplicações não utilizam a técnica do redirecionamento para apresentar suas diversas páginas (ou recursos), mas requisições assíncronas emitidas pelo código executado no navegador Web, e enviadas ao servidor da aplicação. Chamadas assíncronas aprimoram a experiência do usuário, mas inviabilizam recursos endereçáveis. Desta forma, o usuário já não é mais capaz de compartilhar um endereço (URI) a fim de referenciar uma determinada página (recurso).

2.1.2.2 Estado Não-Persistente

Serviços com estado não-persistente são aqueles cujas requisições acontecem em total isolamento, ou seja, o consumidor do serviço envia todas as informações necessárias para a correta execução de um determinado método. E isto acontece

em todos os momentos em que o método é solicitado. (RICHARDSON; RUBY, 2007). Serviços RESTful não persistem estado, pois o próprio protocolo HTTP não o persiste. Outros protocolos – como o FTP (*File Transfer Protocol*) – atuam com estado persistente, o que resulta numa especificação mais complexa e restrita.

Este princípio atua em conjunto com o Princípio da Endereçabilidade. Ou seja, enquanto que cada recurso gerenciado por um serviço deve ser endereçado por um URI, o serviço deve permitir, aos seus consumidores, acesso direto a este recurso. Nenhuma requisição anterior deve ser necessária para o estabelecimento de um determinado estado. Infelizmente, a técnica de gerenciamento de estado é utilizada na Web com recorrência significativa e os consumidores, muitas vezes, apenas conseguem alcançar um recurso C, qualquer, após ter acessado os recursos A e B.

O Princípio do Estado Não-Persistente também é válido no contexto das aplicações Web. Algumas violam este princípio ao utilizar a biblioteca de estado promovida por uma determinada linguagem de programação. Praticamente todas as linguagens modernas com potencial para a Internet – como, por exemplo, Java, PHP e C# – oferecem alguma estrutura de dados capaz de persistir o estado de uma seção. Apesar de ser uma estrutura que facilita o desenvolvimento da aplicação, ao mesmo tempo ela faz com que o usuário perca um pouco de sua liberdade de navegação.

Estas bibliotecas atuam de forma muito simples, como se segue. Cada requisição enviada ao servidor da aplicação carrega o identificador da seção. Este valor identifica o estado em que um determinado usuário se encontra. Ao receber a requisição, o servidor seleciona um determinado subconjunto do estado global da aplicação. Este é justamente o subconjunto de interesse do usuário que iniciou a requisição, e que responde ao identificador recebido. Um recurso com estado persistente não pode ser armazenado ou compartilhado, pois é o resultado de uma série de requisições.

2.1.2.3 Conectividade

Um recurso deve apontar para outros em sua representação (RICHARDSON; RUBY, 2007). Serviços que seguem o Princípio da Conectividade servem representações que contêm apontadores para outras representações. Desta forma, podemos dizer que os recursos gerenciados pelo serviço estão conectados entre si.

Este princípio favorece o descobrimento de recursos a partir da representação de outros recursos. Este fenômeno é comum nas aplicações Web, onde usuários navegam entre diversas páginas através dos *links* oferecidos pelas próprias páginas.

A conectividade é um aspecto fundamental da Web. Num espaço onde a heterogeneidade existente é tão evidente (plataformas, tecnologias, linguagens e paradigmas), o conceito URI ganha ainda mais importância, pois foi o único capaz de estabelecer a conectividade entre peças que não se encaixariam naturalmente.

2.1.3 Descrição Sintática de Serviços RESTful: O Padrão WADL

Inúmeras empresas que atuam na Web 2.0 oferecem serviços Web de acordo com o estilo arquitetural REST; alguns exemplos incluem Amazon, Google e Yahoo. Com o objetivo de facilitar o consumo destes serviços, essas empresas ainda oferecem diversas opções de bibliotecas-cliente, uma para cada linguagem de programação.

Por exemplo, ao disponibilizar o seu serviço de busca[5], o Yahoo também disponibilizou uma série de bibliotecas[6] nas seguintes linguagens: ActionScript, C#, ColdFusion, Java, JavaScript, Lua, Perl, PHP, Python, Ruby e VB.NET.

[5] http://developer.yahoo.com/search/

Desta forma, ao utilizar uma biblioteca, um desenvolvedor de software tem a habilidade de acessar um serviço Web remoto como se este fosse um recurso local, ou seja, uma classe pertencente à arquitetura de seu software.

Esta estratégia abstrai boa parte da complexidade existente em um ambiente distribuído, principalmente no que se refere à composição da requisição, à interpretação da resposta e ao protocolo de transporte. Estes e outros requisitos não-funcionais são realizados pela própria biblioteca.

Porém, é fácil perceber que uma nova coleção de bibliotecas se faz necessária para cada serviço RESTful existente na Web. Por exemplo, similarmente ao Yahoo, o Google poderia oferecer a mesma quantidade de bibliotecas para o seu serviço busca. O mesmo pode ser dito em relação à Amazon, e assim sucessivamente. Note-se o caráter exponencial na quantidade de bibliotecas a serem desenvolvidas.

Portanto, é desejável a existência de um padrão único para a descrição de serviços RESTful, o que conseqüentemente permitiria a existência de uma biblioteca única por linguagem de programação, independentemente do serviço Web em questão. A biblioteca PHP seria capaz de acessar qualquer serviço RESTful na Web, e o mesmo aconteceria para a biblioteca Java, por exemplo. Note-se que o número de bibliotecas cairia consideravelmente. A dinâmica seria simples, pois uma biblioteca utilizaria apenas este arquivo de descrição do serviço para preparar as mensagens de requisição.

Infelizmente, ainda não existe um padrão bem estabelecido para a descrição sintática de serviços RESTful. Como descrito anteriormente, praticamente todas as publicações acadêmicas sobre o tema tratam exclusivamente de serviços SOAP. Entretanto, Hadley (2009), integrante do grupo de pesquisas da Sun Microsystems, propôs uma abordagem baseada em XML, que aos poucos desponta como padrão definitivo: a linguagem WADL. Como conseqüência da utilização deste padrão, o proponente apresenta os seguintes benefícios:

- Suporte ao desenvolvimento de ferramentas para a modelagem de recursos;
- Geração automática de código para a manipulação de recursos;
- Configuração de cliente e servidor a partir de um único formato portável.

[6] http://developer.yahoo.com/download/download.html

Como observado no segundo ponto da listagem acima, Hadley defende a geração automática de bibliotecas-cliente para a manipulação e acesso aos serviços remotos. Porém, na realidade, os benefícios são ainda maiores, pois seria possível a existência de uma única biblioteca por linguagem de programação para todos os serviços RESTful, como explicado anteriormente nessa seção Esta afirmação é especialmente válida para linguagens com aspectos dinâmicos, como Ruby, Python e PHP, por exemplo.

O padrão WADL é composto por diversos elementos XML, entre os quais os principais são diretamente mapeáveis aos conceitos que formam o paradigma RESTful, como os elementos Recurso (`Resource`), Método (`Method`), Requisição (`Request`), Resposta (`Response`) e Representação (`Representation`).

Por fim, é interessante notar que o padrão WADL descreve sintaticamente serviços RESTful assim como o padrão WSDL descreve serviços SOAP. A Figura 8 apresenta um exemplo de aplicação da linguagem WADL no serviço de busca disponibilizado pelo Yahoo. O capítulo 5 contém uma explicação detalhada deste documento na seção 5.2.

```
01 <?xml version="1.0"?>
02 <application
03  xmlns:xsi          = "http://www.w3.org/2001/XMLSchema-instance"
04  xmlns:xsd          = "http://www.w3.org/2001/XMLSchema"
05  xsi:schemaLocation = "http://wadl.dev.java.net/2009/02 wadl.xsd"
06  xmlns              = "http://wadl.dev.java.net/2009/02"
07  xmlns:tns          = "urn:yahoo:yn"
08  xmlns:yn           = "urn:yahoo:yn"
09  xmlns:ya           = "urn:yahoo:api">
10 <grammars>
11  <include href="NewsSearchResponse.xsd"/>
12  <include href="Error.xsd"/>
13 </grammars>
14
15 <resources base="http://search.yahooapis.com/NewsSearchService/V1/">
16  <resource path="newsSearch" id="NewsResource">
17   <method name="GET" id="NewsMethodGET">
18    <request>
19    <param name="appid" type="xsd:string" style="query" required="true"/>
20    <param name="query" type="xsd:string" style="query" required="true"/>
21    <param name="type" style="query" default="all">
22     <option value="all"/>
23     <option value="any"/>
24     <option value="phrase"/>
25    </param>
26    <param name="results" style="query" type="xsd:int" default="10"/>
27    <param name="start" style="query" type="xsd:int" default="1"/>
28    <param name="sort" style="query" default="rank">
29     <option value="rank"/>
30     <option value="date"/>
31    </param>
32    <param name="language" style="query" type="xsd:string"/>
33    </request>
34    <response status="200">
35     <representation mediaType="application/xml" element="yn:ResultSet"/>
36    </response>
37    <response status="400">
38     <representation mediaType="application/xml" element="ya:Error"/>
39    </response>
40   </method>
41  </resource>
42 </resources>
43 </application>
```

Figura 8 – Descrição sintática em WADL do serviço de buscas Yahoo, adaptado de (HADLEY, 2009)

2.2 Discussão

Os conceitos e princípios que formam serviços RESTful referem-se, na realidade, ao próprio comportamento da Web. Ou seja, não são introduzidos novos protocolos a serem aplicados sobre o HTTP.

Por outro lado, no desenvolvimento de serviços RPC, o protocolo SOAP impõe uma sintaxe formal e restrita para a especificação da ação a ser executada, assim como do escopo de execução dessa ação. Tal imposição tende a resultar em uma interpretação equivocada, na qual o protocolo HTTP não seria flexível o suficiente para lidar com a complexidade existente no desenvolvimento de serviços Web. De fato, o termo serviço Web é comumente utilizado como sinônimo para serviço RPC.

Entretanto, a teoria apresentada nessa seção evidencia a existência de outro grupo, muito comum no contexto da Web 2.0, composto por serviços que seguem uma arquitetura orientada a recursos (ROA). Mais especificamente, são serviços que aplicam o protocolo HTTP ao paradigma REST.

WADL é um protocolo novo, realmente, porém que não faz parte da teoria RESTful propriamente dita. Esse padrão foi apresentado neste capítulo porque a definição completa de um serviço semântico também inclui uma descrição sintática. De qualquer forma, o protocolo HTTP nunca se propôs a descrever sintaticamente qualquer objeto na Web, inclusive serviços.

3 WEB SEMÂNTICA

A maior parte do conteúdo disponibilizado na Web é destinada ao consumo humano, e não ao processamento e interpretação por computadores através de processos automatizados. Desta forma, o significado do conteúdo publicado não é considerado em tarefas comuns executadas na Web, como a busca, por exemplo. Tais atividades acabam sofrendo as conseqüências da ambigüidade e da baixa relevância.

Proposta por Berners-Lee, Hendler e Lassila (2001), a Web Semântica surge com o propósito de decorar o conteúdo publicado na Web através de anotações semânticas formais, o que irá minimizar as limitações da Web como a conhecemos atualmente. Neste novo contexto, sintaxe e semântica passam a compartilhar o mesmo grau de importância nos processos de publicação, busca e recuperação de informações.

Inúmeras pesquisas acadêmicas sobre a Web Semântica têm sido publicadas nos últimos anos, e padrões para notação semântica já foram aplicados com sucesso em projetos renomados, como nos exemplos listados a seguir.

Primeiramente, FOAF[7] (*Friend of a Friend*) é um padrão para descrever pessoas e as conexões entre elas, permitindo o desacoplamento de qualquer grafo social com uma aplicação Web em específico. Tecnicamente, FOAF é baseado em RDF, o principal padrão para a notação formal de relacionamentos entre objetos na Web Semântica. Sobre sua adoção, o Google utiliza o padrão FOAF em um serviço[8] que permite o descobrimento de grafos sociais públicos na Internet, o que confirma sua aplicabilidade.

[7] http://foaf-project.org/

[8] http://code.google.com/apis/socialgraph/

Outro exemplo é uma plataforma colaborativa de gerenciamento de conteúdo nomeada SMW[9] (*Semantic Media Wiki*), na qual meta-dados são associados aos artigos publicados com o objetivo de descrevê-los semanticamente. Esta plataforma permite a importação e a exportação de conteúdo não apenas em RDF, mas também em OWL, o principal formato para a descrição formal de ontologias. A base desta plataforma é o pacote não-proprietário MediaWiki[10], a mesma da enciclopédia virtual Wikipedia[11].

Por fim, Swoogle[12] é um mecanismo de busca semântico desenvolvido pela UMBC (*University of Maryland Baltimore County*). Atualmente, mais de 10000 ontologias já foram indexadas por este mecanismo, que ainda aceita a sintaxe RDF e outros meta-dados a fim de refinar os resultados obtidos nas consultas.

Estes são apenas alguns exemplos que ilustram a flexibilidade na aplicação dos padrões que formam a terceira fase da Web. Ao serem considerados em um único conjunto, os padrões compõem uma arquitetura que atua como principal fundação para o desenvolvimento da Web Semântica, como exibido na seção 3.3. Entretanto, faz-se necessária a apresentação prévia dos conceitos Ontologia e Agente, nas seções 3.1 e 3.2, respectivamente, para que a arquitetura possa ser compreendia.

3.1 Ontologia

A definição do termo “ontologia” varia de acordo com o contexto. O conceito foi introduzido pelo filósofo grego Aristóteles (384 – 322 a.C.) como uma teoria sobre a natureza da existência. Enquanto disciplina filosófica, o estudo sobre ontologias preocupa-se em prover um sistema de categorização do conhecimento.

[9] http://semantic-mediawiki.org/

[10] http://mediawiki.org/

[11] http://wikipedia.org/

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

Em uma das referências bibliográficas mais citadas em publicações relacionadas à Web Semântica, Gruber (1995) define uma ontologia como uma especificação formal e explícita de uma conceitualização compartilhada. Nessa definição, “formal” significa que a especificação da ontologia deve ser passível de interpretação por máquinas e “explícita” significa que os elementos devem ser claramente definidos. A “conceitualização” mencionada é simplesmente um modelo abstrato de algum fenômeno identificado por seus conceitos relevantes. Por fim, tal conceitualização deve ser “compartilhada” no sentido que uma ontologia captura um conhecimento consensual, ou seja, de grupo.

De forma mais pragmática, inserido no contexto da Web Semântica, Hendler (2001) define uma ontologia como um conjunto de termos do conhecimento, incluindo o vocabulário, as interconexões semânticas e algumas regras simples de inferência e lógica para algum tópico em particular.

Ontologias podem aprimorar o funcionamento da Web de diversas maneiras. Elas podem ser utilizadas, por exemplo, para melhorar a precisão dos mecanismos de busca, ou ainda para relacionar o conteúdo de uma página Web ao conhecimento real sobre os dados nela contidos (BERNERS-LEE; HENDLER; LASSILA, 2001).

3.2 Agente

Berners-Lee, Hendler e Lassila (2001, p.36) descrevem agentes de software como peças fundamentais para o estabelecimento da Web Semântica, como segue:

> O real poder da Web Semântica será realizado quando as pessoas criarem programas que coletam conteúdo Web das mais diversas fontes, que processam essas informações e que compartilham os resultados com outros programas. A efetividade de tais agentes de software irá aprimorar-se exponencialmente quanto mais conteúdo Web passível de interpretação por máquinas e serviços Web tornarem-se disponíveis. A Web Semântica promove essa sinergia: até mesmo agentes que não foram projetados para trabalharem em cooperação podem transferir dados entre eles quando tais dados são decorados semanticamente. (BERNERS-LEE; HENDLER; LASSILA, 2001, p. 36)

Observa-se a importância dos serviços Web como fonte de dados para os agentes de software da Web Semântica. Portanto, para o real estabelecimento da terceira fase da Web, é necessário que os diversos serviços Web já disponibilizados sejam descritos semanticamente. Somente desta forma os agentes seriam capazes de encontrar e executar serviços automaticamente, ou seja, sem intervenção humana. Os autores descrevem tal necessidade como segue:

> Muitos serviços Web já existem sem semântica, mas outros programas como os agentes não são capazes de localizar o serviço que irá executar uma função em específico. Esse processo, chamado descobrimento de serviço, pode acontecer apenas quando há uma linguagem comum que permita aos agentes entender tanto a função oferecida quanto o processo de consumo dessa função. (BERNERS-LEE; HENDLER; LASSILA, 2001, p. 36)

De acordo com essa estratégia, é razoável a conclusão que qualquer serviço com pretensão de integrar-se ao contexto semântico seja descrito através de ontologias. Isso compreende, obviamente, ambas as categorias de serviços: RPC e RESTful.

A linguagem comum mencionada no fragmento de texto acima se refere ao padrão de notação formal de ontologias, ou OWL, como já concretizado na arquitetura da Web Semântica, apresentada na seção 3.3. Ao descrever serviços, utiliza-se uma extensão do padrão OWL chamada OWL-S. Essa extensão foi criada originalmente para permitir a descrição semântica de serviços RPC realizados através do protocolo SOAP. Entretanto, ainda é necessária a descrição semântica de serviços RESTful. Caso contrário, serviços integrantes a este segundo grupo não estariam disponíveis para os agentes, o que os excluiria da Web Semântica como um todo.

A proposta apresentada ao longo do presente trabalho, especialmente no capítulo 5, define uma especialização de OWL-S para permitir a descrição semântica de serviços RESTful. Pretende-se, portanto, preencher a lacuna identificada.

3.3 Arquitetura

O desenvolvimento da Web Semântica evolui de forma segmentada, baseada em camadas. Segundo Antoniou e van Harmelen (2008), uma justificativa pragmática para essa abordagem é o fato que as pessoas tendem a chegar ao consenso mais facilmente em decisões menores. Porém, os mesmos autores lembram que, tipicamente, diferentes grupos de pesquisa caminham em diferentes direções. De fato, o atual entendimento sobre a arquitetura da Web Semântica é o resultado de modificações feitas em quatro versões anteriores, e o debate ainda existe (GERBER, 2007). A quinta e mais recente versão é apresentada na Figura 9.

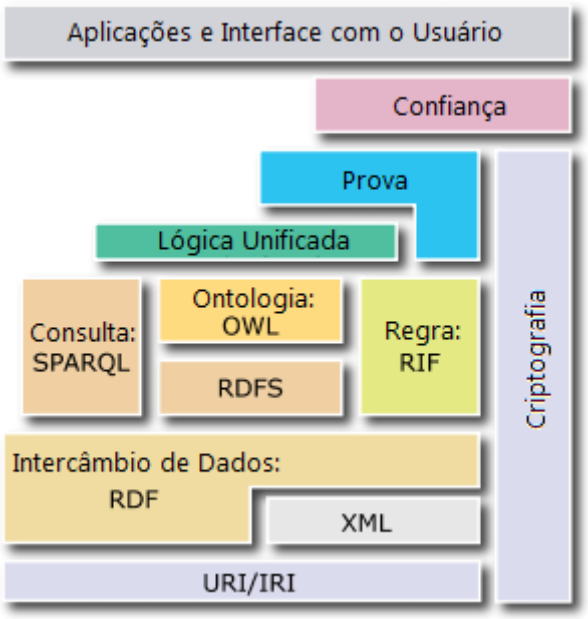

Figura 9 – Arquitetura da Web Semântica, adaptada de (W3C, 2007)

A abordagem baseada em camadas permite que padrões já estabelecidos possam ser aplicados antes mesmo que toda a visão da Web Semântica seja realizada de forma concreta, como nos exemplos listados anteriormente neste capítulo. Espera-se, portanto, que tais iniciativas acelerem o processo de adoção do novo paradigma semântico como um todo.

A primeira camada da arquitetura define o conceito de identificador uniforme para recursos na Web. O padrão URI, ou IRI (*Internationalized Resource Identifier*) (IETF, 2005), não apenas fornece o identificador, mas também o localizador do recurso, o

que permite o estabelecimento de redes ou grafos de recursos, como descrito na seção 2.1.1.3 (Identificador Uniforme). Geralmente, os grafos são compostos por tipos de recursos distintos, como uma imagem e um documento textual, por exemplo. Por seu caráter genérico, interoperável e altamente reutilizável, esta camada estabelece a fundação da arquitetura da Web Semântica.

Já a segunda camada refere-se ao XML, a base sintática dos demais padrões definidos em camadas superiores. O mais expressivo poder dessa linguagem é que os elementos que a compõem são definidos pelo próprio autor do documento XML em questão. Mais precisamente, cada documento XML define seu próprio vocabulário. Entretanto, com o intuito de possibilitar o reuso de vocabulário, documentos XML podem referenciar espaços de nomes remotos (*namespaces*) através de identificadores URI. Desta forma, muitos documentos XML encontram-se conectados entre si.

Conceitos semânticos são introduzidos na terceira camada, composta pelos padrões RDF e RDFS (*Resource Description Framework Schema*). RDF é uma linguagem para representar formalmente meta-dados sobre recursos na Web. Conceitualmente, é considerado recurso qualquer objeto passível de identificação na Web, mesmo que este não possa ser recuperado diretamente a partir da Web (MANOLA; MILLER, 2004). Até certo ponto, RDF pode ser considerado um padrão simplificado para a definição de ontologias, uma linguagem que apóia a interoperabilidade entre aplicações que trocam informações interpretáveis por máquinas (BREITMAN; CASANOVA; TRUSZKOWSKI, 2007).

Um documento RDF é sintaticamente estruturado em declarações (*statements*). Cada declaração é, por sua vez, composta por três elementos: sujeito (*subject*), predicado (*predicate*) e objeto (*object*). O sujeito é simplesmente um recurso identificado por URI; este é o objeto que se pretende descrever semanticamente. Predicado é uma propriedade do sujeito também identificada por um URI. Por fim, o objeto é o valor atribuído à propriedade do sujeito. Diferentemente dos outros dois elementos, o objeto pode ser tanto um recurso identificado por URI, como um valor literal simples. O segundo caso é utilizado para datas e nomes, por exemplo.

Tal abordagem baseada em declarações provou-se genérica o suficiente para mapear os mais diversos domínios. Um documento RDF é geralmente serializado

na sintaxe XML, cuja estrutura em árvore possibilita a existência de declarações aninhadas em quantidade ilimitada.

Autores de documentos RDF devem ter a habilidade de definir o vocabulário que pretendem utilizar nas declarações (MANOLA; MILLER, 2004). A linguagem RDF oferece enorme flexibilidade, mas não provê meios para a definição de classes, propriedades e hierarquias específicas de um determinado domínio (BREITMAN; CASANOVA; TRUSZKOWSKI, 2007). Portanto, um padrão conhecido como RDFS surge para permitir a definição personalizada de modelos de classes e propriedades. Uma das principais características do padrão RDFS é a capacidade de estabelecer hierarquias de classes, assim como no paradigma da orientação a objetos existente no contexto da Engenharia de Software. O elemento que realiza esta característica é nomeado `rdfs:subClassOf`. Adicionalmente, o padrão RDFS ainda permite o estabelecimento de hierarquias de propriedades através da aplicação do elemento `rdfs:subPropertyOf`.

As classes e propriedades definidas em um documento RDFS são instanciadas em documentos RDF. Desta forma, dois efeitos muito comuns na Web Semântica são observados. Primeiramente, nota-se o reuso de vocabulário, pois o mesmo modelo de classes é consumido por qualquer documento RDF interessado. Em segundo lugar, observa-se a geração de declarações sensíveis ao domínio, o que é fundamental na definição de ontologias.

A quarta camada é justamente iniciada pelo padrão responsável pela definição de ontologias, ou seja, a linguagem OWL. Este padrão é composto pelas três sub-linguagens listadas e brevemente descritas a seguir:

- **OWL Lite:** Versão simplificada da linguagem. Permite hierarquia de classes e restrições simples em propriedades, como cardinalidade binária. Possui poder semântico suficiente para a especificação de ontologias simples e eficientes.

- **OWL DL:** Agrega as construções da versão simplificada com a expressividade da lógica descritiva. A versão DL (*Description Logic*) permite operações baseadas na Teoria de Conjuntos – como união, intersecção e complemento –, além de restrições mais complexas em propriedades.

- **OWL Full:** Tem a expressividade da versão DL, mas não impõe limitações sobre como os recursos se relacionam, ou seja, uma classe por ser uma instância de outra classe, por exemplo. Nas versões anteriores, classes, propriedades e instâncias são desconexas. Portanto, a versão completa da linguagem OWL pode ser considerada tão expressiva quanto o próprio padrão RDF, não havendo, porém, garantias sobre eficiência computacional.

Como visto, linguagens de representação do conhecimento como RDFS e OWL são aplicadas na descrição semântica de domínios, provendo classes, relacionamentos e hierarquias entre essas entidades. Linguagens de representação de regras, por outro lado, oferecem suporte à descrição de regras de transformação de dados. Integrantes da quarta camada da Web Semântica, as regras são necessárias na geração de novos fatos a partir de bases de conhecimento. Em outras palavras, regras tratam de deduções sobre os dados, ou seja, da derivação de informações a partir de dados existentes.

Existem inúmeras linguagens para a notação formal de regras. Portanto, o padrão RIF (*Rule Interchange Format*) foi estabelecido com o objetivo de focar na troca e não na unificação das linguagens (BOLEY et al., 2009). Em contraste com RDF e OWL, uma única linguagem de regras não seria capaz de atender a todos os paradigmas de utilização, que compartilham pouco em termos de sintaxe e semântica. Os três principais paradigmas são: lógica de primeira ordem, lógica de programação e regra de ação.

Assim, o padrão RIF permite o compartilhamento de regras criadas em diferentes linguagens. Como exemplo, RuleML (*Rule Modeling Language*) é uma linguagem de marcação para a publicação e o compartilhamento de bases de conhecimento na Web. Já a linguagem SWRL (*Semantic Web Rule Language*) combina a RuleML com a OWL, incluindo uma sintaxe abstrata de alto nível, além de serialização XML.

A quarta camada da Web Semântica é finalizada pela funcionalidade de consulta, realizada pelo padrão SPARQL. Essa linguagem permite a construção de comandos de consulta simples a serem aplicados em declarações RDF para a seleção de subconjuntos. O padrão SPARQL possui sintaxe semelhante ao SQL (*Structured Query Language*), a linguagem padrão para a manipulação de dados em bases relacionais.

As demais camadas da arquitetura ainda encontram barreiras tecnológicas e a falta de padrões bem estabelecidos faz com que seu papel na Web Semântica ainda seja abstrato. A quinta camada – Lógica Unificada – seria a primeira representante deste grupo. Teoricamente, ela seria responsável por descrever uma lógica matemática formal com o objetivo de conciliar todos os diferentes modelos semânticos das camadas inferiores – RDF, RDFS, OWL, SPARQL e RIF – em um modelo holístico e consistente. A idéia é prover uma interface lógica única para dados e regras, permitindo então o desenvolvimento de aplicações a partir de uma base comum e não a partir das partes individuais, potencialmente, heterogêneas.

Também integrantes do grupo teórico, as próximas três camadas lidam basicamente com confiabilidade. Esse requisito não-funcional é de extrema importância no contexto semântico, uma vez que aplicações serão capazes de analisar bases de conhecimento e regras associadas a fim de automatizar a tomada de decisão. A severidade das conseqüências de uma determinada decisão varia de acordo com o domínio em questão, mas é fácil perceber que na área da saúde ou do direito, por exemplo, é extremamente importante que cada uma das decisões tenha um grau de confiabilidade alto.

A sexta camada – Prova – provê explicação sobre quais inferências e regras foram executadas para uma determinada conclusão ou recomendação. Esta camada irá possibilitar, portanto, auditoria sobre as inferências. Os passos percorridos poderão ser armazenados como atalho, o que facilitaria futuras tomadas de decisão.

Já a sétima camada – Confiança – seria responsável basicamente por validar o grau de confiabilidade das provas produzidas na camada anterior.

Finalmente, a oitava camada – Criptografia – provê técnicas para a identificação única das fontes de informação. Assinatura digital, por exemplo, será uma das técnicas presentes nessa camada. Observa-se o posicionamento ortogonal da oitava camada em relação às demais, justamente pelo fato da criptografia poder ser aplicada em objetos pertencentes a camadas inferiores da arquitetura.

A nona e última camada na arquitetura, nomeada Aplicações e Interface com o Usuário, representa o meio pelo qual as pessoas entrarão em contato com a Web Semântica. Espera-se que as aplicações a serem desenvolvidas sejam intuitivas o

suficiente para permitir que usuários sem experiência com os conceitos que formam a abordagem semântica possam usufruir das capacidades oriundas da terceira fase da Web.

3.4 Discussão

Observa-se que as camadas inferiores na arquitetura da Web Semântica possuem padrões com considerável nível de maturidade, inclusive com aplicações práticas em projetos proprietários ou de código-aberto. Entretanto, como descrito por Gerber (2007), argumenta-se na comunidade acadêmica se o esquema baseado em camadas apresentado na Figura 9 poderia realmente ser classificado como arquitetura, termo com aplicações variáveis na Engenharia de Software. A mesma autora ainda apresenta uma revisão bibliográfica sobre se determinadas camadas seriam de fato necessárias para o estabelecimento do contexto semântico. De qualquer forma, é inegável que a organização posposta por Tim Berners-Lee, Hendler e Lassila (2001) tem realmente guiado o desenvolvimento da Web Semântica. É clara a evolução, camada após camada, rumo ao topo da arquitetura.

A ausência de padrões bem estabelecidos nas camadas superiores dificulta o desenvolvimento de agentes semânticos e, conseqüentemente, o processo de validação de ontologias. Portanto, a ontologia `RESTfulGrounding` proposta neste trabalho será testada por validadores RDF e OWL, além de ser aplicada em um estudo de caso. Trata-se de um serviço RESTful real, disponibilizado pelo Yahoo, que será descrito sintaticamente através do padrão WADL e semanticamente através da linguagem OWL-S.

4 SERVIÇOS SEMÂNTICOS

Este capítulo descreve o estado da arte em serviços semânticos, apresentando os principais padrões já estabelecidos para o seu desenvolvimento. Portanto, apenas os serviços RPC/SOAP são consideramos neste momento.

A tecnologia de serviços Web traz o aspecto dinâmico à utilização da Web. Já a tecnologia de serviços semânticos facilita e aumenta a precisão dos processos de busca, recuperação, representação, extração, interpretação e manutenção da informação (BREITMAN; CASANOVA; TRUSZKOWSKI, 2007).

Portanto, serviços semânticos desempenham um papel fundamental no processo de viabilização da Web Semântica como um todo. Como descrito por Berners-Lee, Hendler e Lassila (2001), a Web Semântica traria estrutura ao rico conteúdo já disponibilizado pelas páginas Web, criando um ambiente no qual os agentes de software poderiam navegar de página em página, executando tarefas sofisticadas para seus usuários. Para que esta dinâmica se torne possível, os agentes deverão ser capazes de encontrar e solicitar a execução dos serviços de forma autônoma, sem a intervenção humana.

O problema é que nenhum agente de software seria capaz de ler e utilizar uma interface WSDL sem a assistência humana. A especificação WSDL não provê meios para a inclusão da representação semântica de uma operação disponibilizada por um serviço Web. Além disso, esta especificação também não suporta anotações semânticas sobre os elementos que compõem uma requisição ao serviço Web (MARTIN et al., 2004).

Conseqüentemente, faz-se necessária a adesão de ontologias que possam suprir tal ausência de anotações semânticas. A OWL é projetada para aplicações que necessitam processar o conteúdo da informação, ao invés de apenas apresentá-lo aos humanos, provendo vocabulário e semântica formal (MC GUINNESS; VAN HARMELEN, 2004). Já a OWL-S é uma ontologia genérica que estende a OWL com o objetivo específico de permitir anotações semânticas em serviços Web.

A seção 4.1 apresenta a linguagem OWL-S em detalhe. São cobertas não somente as características desse padrão, mas também as quatro ontologias internas que o compõem. Fragmentos do padrão OWL-S são apresentados na linguagem OWL a fim de facilitar a compreensão da descrição textual. Já a seção 4.2 trata especificamente do mapeamento semântico-sintático baseado nos padrões WSDL e SOAP. Por fim, a seção 4.3 apresenta um exemplo completo, no qual um serviço RPC é descrito sintática e semanticamente através dos conceitos apresentados.

4.1 OWL-S

Serviços semânticos lidam com as limitações existentes nos atuais serviços Web através do aprimoramento da descrição dos serviços, definindo uma camada semântica com o objetivo de alcançar processos automáticos de descobrimento, composição, monitoramento e execução (ANTONIOU; VAN HARMELEN, 2008).

A camada semântica mencionada é geralmente definida pela linguagem OWL-S, que oferece uma ontologia principal (`Service`[13]) e outras três subordinadas à primeira (`Profile`[14], `Process`[15] e `Grounding`[16]). A justificativa para tal estruturação baseia-se no fato que qualquer consumidor em potencial deve ter acesso a três tipos de conhecimento sobre o serviço, como listado na proposta ao W3C (MARTIN et al., 2004):

[13] http://www.daml.org/services/owl-s/1.2/Service.owl

[14] http://www.daml.org/services/owl-s/1.2/Profile.owl

[15] http://www.daml.org/services/owl-s/1.2/Process.owl

[16] http://www.daml.org/services/owl-s/1.2/Grounding.owl

- **O que o serviço provê ao seu consumidor**

  A ontologia `Profile` é responsável por definir as funcionalidades oferecidas pelo serviço. Esta ontologia assiste, portanto, aos processos de publicação e de promoção do serviço. Considerando a utilização de repositórios ou mecanismos de busca semântica, a ontologia `Profile` permite aos agentes de software descobrir automaticamente o serviço em questão.

- **Como o serviço funciona**

  A resposta a esta questão encontra-se na ontologia `Process`, responsável por descrever todos os passos necessários para a correta execução de uma determinada funcionalidade do serviço. Esta ontologia permite a composição de serviços para a geração de um serviço complexo. Um agente de software pode utilizar as informações providas pela ontologia `Process` a fim de decidir se o serviço realmente atende às necessidades do consumidor que solicitou a tarefa.

- **Como acessar o serviço**

  Por fim, a ontologia `Grounding` é a responsável por descrever como o serviço pode ser acessado. Esta ontologia discrimina, inclusive, os protocolos de transporte e comunicação aceitos pelo serviço. Considerando o contexto da Web Semântica, a ontologia `Grounding` possibilita a solicitação automática do serviço por um agente de software.

A ontologia `Service`, principal integrante do padrão OWL-S, atua simplesmente como ponto de ligação entre as três ontologias listadas acima. Mais especificamente, ela define as classes abstratas que são posteriormente especializadas pelas demais ontologias. As próximas seções detalham, em maior profundidade, cada uma das quatro ontologias que formam a OWL-S.

4.1.1 A Ontologia do Serviço (`Service`)

Como representado pela Figura 10, a ontologia `Service` define as quatro classes principais: `Service`, `ServiceProfile`, `ServiceModel` e `ServiceGrounding`. As três últimas são classes abstratas, posteriormente estendidas pelas classes `Profile`, `Process` e `Grounding`, respectivamente, integrantes das ontologias de mesmo nome. Estas ontologias serão discutidas nas seções seguintes (4.1.2, 4.1.3 e 4.1.4). A Figura 10 ainda apresenta as propriedades que conectam as classes.

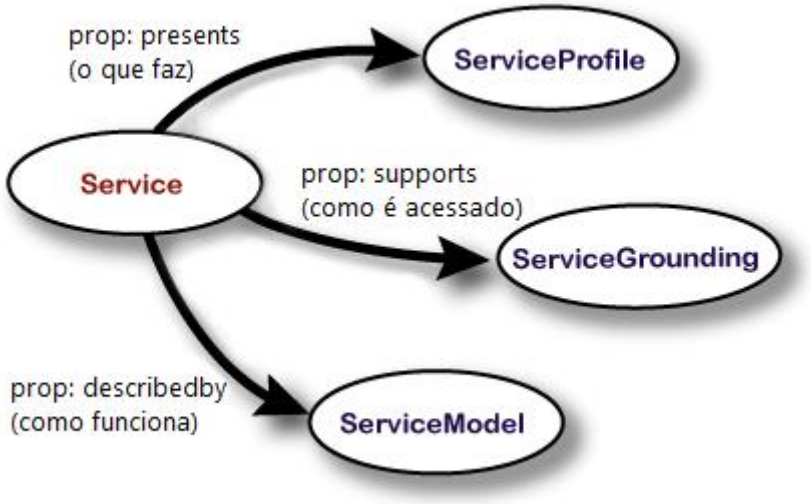

Figura 10 – As quatro principais classes da ontologia Service, adaptada de Martin et al. (2004)

Realizações concretas de `ServiceProfile` devem descrever as funcionalidades oferecidas pelo serviço em questão. As realizações concretas de `ServiceModel` devem descrever o processo de execução das funcionalidade do serviço. Já as realizações concretas de `ServiceGrounding` devem descrever o acesso ao serviço, principalmente em relação aos protocolos suportados.

Finalmente, a classe `Service` provê um meio simples para a organização das partes que formam um serviço semântico. É a classe responsável por agrupar as demais classes, compondo a descrição completa do serviço.

A Figura 11 apresenta a definição das classes `Service`, `ServiceProfile`, `ServiceModel` e `ServiceGrounding`.

```
01 <!-- Service -->
02 <owl:Class rdf:ID="Service">
03   <rdfs:label>Service</rdfs:label>
04 </owl:Class>
05
06 <!-- Service Profile -->
07 <owl:Class rdf:ID="ServiceProfile">
08   <rdfs:label>ServiceProfile</rdfs:label>
09 </owl:Class>
10
11 <!-- Service Model -->
12 <owl:Class rdf:ID="ServiceModel">
13   <rdfs:label>ServiceModel</rdfs:label>
14 </owl:Class>
15
16 <!-- Service Grounding -->
17 <owl:Class rdf:ID="ServiceGrounding">
18   <rdfs:label>ServiceGrounding</rdfs:label>
19 </owl:Class>
```

Figura 11 – Definição das principais classes na ontologia `Service`
(fragmento extraído e adaptado de http://www.daml.org/services/owl-s/1.2/Service.owl)

Sobre a cardinalidade existente nos relacionamentos entre essas quatro classes, as seguintes restrições são impostas:

- Cada instância da classe `Service` apresenta zero ou mais instâncias da classe `ServiceProfile`, uma para cada funcionalidade oferecida pelo serviço.
- Cada instância da classe `Service` pode ser descrita por uma instância da classe `ServiceModel`.
  - É permitido que uma instância de `Service` não possua nenhuma instância de `ServiceModel` associada. Neste caso, a descrição do serviço torna-se disponível apenas para o propósito de seu descobrimento.
  - Caso exista uma instância de `ServiceModel` associada, a instância de `Service` deve obrigatoriamente suportar uma ou mais instâncias da classe `ServiceGrounding`. Espera-se que a existência de diversas instâncias de `ServiceGrounding` possa trazer um alto grau de flexibilidade ao serviço, uma vez que estariam disponíveis diversas formas de acesso, possivelmente através de diversos protocolos distintos.

4.1.2 A Ontologia do Perfil de um Serviço (`Profile`)

O perfil de um serviço no padrão OWL-S refere-se a uma funcionalidade oferecida. Na Web Semântica, agentes de software examinam os perfis com o objetivo de decidir se o serviço é adequado ou não à tarefa solicitada. Perfis são os primeiros pontos de contato no processo de descobrimento.

Utilizando a técnica de especialização de classes oferecida pela RDFS, a classe `Profile` estende a classe abstrata `ServiceProfile`. Essa técnica é muito semelhante àquela oferecida pelo paradigma da orientação a objetos, onde classes pertencentes a diferentes níveis de abstração associam-se através do relacionamento de generalização. Na linguagem RDFS, porém, este relacionamento é chamado `subClassOf`, como apresentado pela Figura 12, linha 03.

```
01 <owl:Class rdf:ID="Profile">
02   <rdfs:label>Profile</rdfs:label>
03   <rdfs:subClassOf rdf:resource="&service;#ServiceProfile" />
04   <rdfs:comment>Definition of Profile</rdfs:comment>
05 </owl:Class>
```

Figura 12 – Definição da classe `Profile`, pertencente à ontologia `Profile`
(fragmento extraído e adaptado de http://www.daml.org/services/owl-s/1.2/profile.owl)

O perfil de um serviço descreve uma funcionalidade de acordo com três tipos de informação, como se apresenta a seguir.

- **Provedor do serviço**

  O provedor é a organização que disponibiliza o serviço na Web. Esta categoria compreende simplesmente algumas informações de contato, geralmente referentes ao técnico responsável por operar o serviço. Essas informações são destinadas ao leitor humano e não aos agentes de software.

- **Funcionalidades oferecidas pelo serviço**

  As informações funcionais são expressas em termos das transformações produzidas pelo serviço. Mais especificamente, o perfil define os parâmetros de entrada e saída, as pré-condições para a execução e os efeitos que dela resultam. A partir desse conjunto de informações, gerou-se o acrônimo IOPE (*Inputs, Outputs, Preconditions, Effects*).

Considere-se o exemplo apresentado por Martin et al. (2004), no qual um serviço de vendas define como pré-condição um cartão de crédito válido, e como parâmetros de entrada o número e a data de expiração desse cartão. O recibo gerado seria o parâmetro de saída, e o débito na conta-corrente associada ao cartão seria o efeito resultante do processamento.

- **Características do serviço**
  Esse terceiro grupo contém informações extras como, por exemplo, a categoria à qual o serviço pertence e o índice de qualidade garantido. O fornecedor do serviço deve utilizar métricas padronizadas para a definição da categoria e índice de qualidade. Porém, a OWL-S não provê tais métricas, delegando ao provedor a escolha do padrão que melhor se enquadre em seu contexto. Adicionalmente, este terceiro grupo de informações compreende quaisquer outros parâmetros que se façam necessários para a execução do serviço.

4.1.3 A Ontologia do Processo de um Serviço (`Process`)

No padrão OWL-S, um processo é responsável por definir como um determinado serviço é executado, ou seja, quais os passos que compõem este fluxo de execução. No contexto da Web Semântica, um agente de software pode utilizar essas informações para decidir se o serviço realmente cumpre com os requisitos impostos pela parte que iniciou a requisição, geralmente referenciada como o "cliente do processo". Quando analisadas em conjunto, as classes `Profile` e `Process` apresentam todas as informações necessárias para uma tomada de decisão.

Novamente, utilizando a técnica de especialização de classes oferecida pela RDFS, a classe `Process` estende a classe abstrata `ServiceModel`, como apresentado na Figura 13, linha 03.

```
01 <owl:Class rdf:ID="Process">
02   <rdfs:comment>The most general class of processes</rdfs:comment>
03   <rdfs:subClassOf rdf:resource="&service;#ServiceModel"/>
04   <owl:unionOf rdf:parseType="Collection">
05     <owl:Class rdf:about="#AtomicProcess"/>
06     <owl:Class rdf:about="#SimpleProcess"/>
07     <owl:Class rdf:about="#CompositeProcess"/>
08   </owl:unionOf>
09 </owl:Class>
```

Figura 13 – Definição da classe `Process`, pertencente à ontologia `Process`
(fragmento extraído e adaptado de http://www.daml.org/services/owl-s/1.2/process.owl)

Cada processo depende de um grupo IOPE, ou seja, parâmetros de entrada e saída, pré-condições e efeitos resultantes. Parâmetros de entrada e saída especificam a transformação de dados produzida pelo processo. Os parâmetros de entrada fornecem as informações necessárias para o início da execução do processo, ao passo que os de saída representam os resultados da execução.

Um processo se relaciona com seus parâmetros de acordo com as propriedades `hasParameter` (linhas 01-04), `hasInput` (linhas 06-09) e `hasOutput` (linhas 11-14), como apresentado pela Figura 14. Note-se que o primeiro representa um relacionamento abstrato entre as classes `Process` e `Parameter`, de tal forma que os demais relacionamentos apenas estendem estes conceitos mais abstratos através do atributo `subPropertyOf`. Diferentemente do paradigma da orientação a objetos, o padrão RDFS também permite hierarquia entre relacionamentos, e não apenas entre classes.

Efeitos resultantes do processo (`result`) são definidos de forma similar, mas não se encontram exibidos na figura. Esses efeitos indicam situações que podem acontecer em um processamento específico e são mutuamente exclusivos. Eles condicionam a saída do processo.

De forma complementar, a Figura 15 apresenta uma versão simplificada da definição das classes `Parameter` (linhas 01-13), `Input` (linhas 25-27) e `Output` (linhas 29-31). Note-se que a definição completa da classe `Parameter` ainda inclui as propriedades `ParameterType` (linhas 15-18) e `ParameterValue` (linhas 20-23).

```
01 <owl:ObjectProperty rdf:ID="hasParameter">
02   <rdfs:domain rdf:resource="#Process"/>
03   <rdfs:range rdf:resource="#Parameter"/>
04 </owl:ObjectProperty>
05
06 <owl:ObjectProperty rdf:ID="hasInput">
07   <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
08   <rdfs:range rdf:resource="#Input"/>
09 </owl:ObjectProperty>
10
11 <owl:ObjectProperty rdf:ID="hasOutput">
12   <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
13   <rdfs:range rdf:resource="#Output"/>
14 </owl:ObjectProperty>
```

Figura 14 – Relacionamento entre processo (classe `Process`) e parâmetro (classe `Parameter`)
(fragmento extraído e adaptado de http://www.daml.org/services/owl-s/1.2/process.owl)

```
01 <owl:Class rdf:ID="Parameter">
02   <rdfs:subClassOf>
03     <owl:Restriction>
04       <owl:onProperty rdf:resource="#parameterType"/>
05       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
06       </owl:cardinality>
07     </owl:Restriction>
08   </rdfs:subClassOf>
09 </owl:Class>
10
11 <owl:Class rdf:about="#Parameter">
12   <rdfs:subClassOf rdf:resource="&swrl;#Variable"/>
13 </owl:Class>
14
15 <owl:DatatypeProperty rdf:ID="parameterType">
16   <rdfs:domain rdf:resource="#Parameter"/>
17   <rdfs:range rdf:resource="&xsd;#anyURI"/>
18 </owl:DatatypeProperty>
19
20 <owl:DatatypeProperty rdf:ID="parameterValue">
21   <rdfs:domain rdf:resource="#Parameter"/>
22   <rdfs:range rdf:resource="&rdf;#XMLLiteral"/>
23 </owl:DatatypeProperty>
24
25 <owl:Class rdf:ID="Input">
26   <rdfs:subClassOf rdf:resource="#Parameter"/>
27 </owl:Class>
28
29 <owl:Class rdf:ID="Output">
30   <rdfs:subClassOf rdf:resource="#Parameter"/>
31 </owl:Class>
```

Figura 15 – Definição das classes `Parameter`, `Input` e `Output`
(extraído e adaptado de http://www.daml.org/services/owl-s/1.2/process.owl)

Por fim, é importante notar a existência de três tipos primitivos de processo, como apresentado anteriormente na definição da classe `Process` (Figura 13, linhas 04-08), são eles: `AtomicProcess`, `SimpleProcess` e `CompositeProcess`.

Um processo atômico é uma ação passível de ser executada em uma única interação, ou seja, não existem sub-processos, ou dependências entre serviços. Toda a ação é realizada em um único passo, pelo menos do ponto de vista do cliente do processo.

Um serviço composto, por outro lado, é uma ação que requer um protocolo baseado em múltiplos passos. Estes serviços são passíveis de decomposição em processos menores, ou seja, que oferecem um processamento mais específico. O padrão OWL-S permite um número ilimitado de processos compostos aninhados.

Já os serviços simples não são chamados diretamente, ou seja, não recebem requisições de algum cliente de processo. Tais serviços são utilizados apenas como elementos de abstração. Um serviço simples pode ser criado para representar uma visão de algum processo atômico, ou seja, um modo especial de utilização do serviço atômico referenciado. Alternativamente, um serviço simples pode oferecer uma versão simplificada de algum processo composto, potencialmente útil para fins de planejamento e raciocínio.

4.1.4 A Ontologia de Acesso ao Serviço (`Grounding`)

A ontologia `Grounding` especifica os detalhes de acesso ao serviço, principalmente em relação aos protocolos, formatos de mensagem, serialização, transporte e endereçamento. Enquanto `Profile` e `Process` são ontologias que descrevem o serviço de forma abstrata, a ontologia `Grounding` é responsável pela descrição concreta do serviço, ou seja, trata de aspectos tecnológicos reais.

Na Web Semântica, agentes de software utilizam esta ontologia para preparar a requisição a ser enviada ao serviço. Neste momento, toda a verificação de compatibilidade entre o requisitante e o provedor já foi realizada e o agente está pronto para se comunicar diretamente com o serviço.

A Figura 16 apresenta um fragmento da ontologia `Grounding`, mostrando a definição da classe `Grounding`, uma realização concreta da classe abstrata `ServiceGrounding`. A especificação do conceito processo atômico é também mostrada na figura. Essa ontologia nada mais é do que uma coleção de instâncias de processos atômicos, uma para cada processo no modelo. Os últimos comandos na Figura 16 mostram essa ligação.

```
01 <owl:Class rdf:ID="Grounding">
02   <rdfs:subClassOf rdf:resource="&service;#ServiceGrounding"/>
03 </owl:Class>
04
05 <owl:ObjectProperty rdf:ID="hasAtomicProcessGrounding">
06   <rdfs:domain rdf:resource="#Grounding"/>
07   <rdfs:range rdf:resource="#AtomicProcessGrounding"/>
08 </owl:ObjectProperty>
09
10 <owl:Class rdf:ID="AtomicProcessGrounding">
11   <rdfs:subClassOf>
12     <owl:Restriction>
13       <owl:onProperty rdf:resource="#owlsProcess"/>
14       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
15       </owl:cardinality>
16     </owl:Restriction>
17   </rdfs:subClassOf>
18 </owl:Class>
19
20 <owl:ObjectProperty rdf:ID="owlsProcess">
21   <rdfs:domain rdf:resource="#AtomicProcessGrounding"/>
22   <rdfs:range rdf:resource="&process;#AtomicProcess"/>
23 </owl:ObjectProperty>
```

Figura 16 – Definição da classe `Grounding`, pertencente a ontologia `Grounding`
(fragmento extraído e adaptado de http://www.daml.org/services/owl-s/1.2/process.owl)

Mensagens concretas são explicitamente especificadas na ontologia `Grounding`. Desta forma, os parâmetros de entrada e saída discriminados de forma abstrata na ontologia `Process` são realizados concretamente na ontologia `Grounding`, que os representa em algum formato específico para a transmissão da mensagem.

Ao propor o padrão OWL-S, Martin et al. (2004) selecionam a linguagem WSDL como sintaxe inicial na descrição concreta de serviços, mas também são explícitos quanto à abertura para novas linguagens: “nossa intenção não é prescrever a única abordagem possível para *grounding* a ser usada com todos os serviços, mas sim prover uma abordagem geral, canônica e largamente aplicável, que será útil na grande maioria dos casos”.

A despeito das desvantagens existentes em serviços SOAP em termos de desempenho e escalabilidade, como mencionado anteriormente, a comunhão entre OWL-S é WSDL é realmente valiosa, pois permite aos inúmeros serviços SOAP já disponíveis na Web migrar ao mundo semântico. Entretanto, este mesmo caminho de migração deve ser disponibilizado aos serviços RESTful, que realmente despontam como a abordagem mais adotada na Web 2.0. Essa dissertação tem como principal objetivo viabilizar o desenvolvimento de serviços semânticos de acordo com a abordagem RESTful.

4.2 OWL-S Service Grounding baseado em WSDL e SOAP

A descrição completa de um serviço compreende tanto os elementos semânticos, quanto os elementos sintáticos. Como apresentado anteriormente, a semântica é fundamental para a existência de processos automáticos de descobrimento e composição de serviços Web. Tais elementos semânticos são providos pela ontologia OWL-S. Entretanto, ainda são necessários os elementos sintáticos, que possibilitam que o serviço descoberto possa também ser acessado de forma automática. O acesso depende de aspectos tecnológicos como protocolos de transporte e comunicação, sintaxe de serialização e assim por diante. Serviços SOAP são sintaticamente descritos através do padrão WSDL.

A ontologia `Grounding`, integrante do padrão OWL-S, é a responsável pelo mapeamento semântico-sintático. A principal classe, homônima à própria ontologia e apresentada na Figura 16, é um elemento abstrato e, conseqüentemente, passível de realizações concretas. Nesta seção, a realização adotada será referenciada como Grounding OWL-S/WSDL.

O relacionamento entre OWL-S e WSDL acontece de acordo com as regras que se seguem e que estão apresentadas na Figura 17 para melhor visualização (MARTIN et al., 2004). A seta bidirecional nessa figura representa a correspondência entre os padrões. Note-se que os padrões OWL-S e WSDL cobrem espaços conceituais diferentes, mas se sobrepõem nos pontos em que o mapeamento é necessário.

Figura 17 – Grounding OWL-S / WSDL, adaptada de Martin et al. (2004)

1. Um Processo Atômico OWL-S corresponde ao elemento WSDL Operação (`operation`). Diferentes tipos de Operações são disponibilizados pelo padrão WSDL e o mapeamento se dá de acordo com seguintes pontos:
    a. Um Processo Atômico OWL-S, definido com parâmetros de entrada e saída, corresponde a uma Operação WSDL do tipo `request-response`.

   b. Um Processo Atômico OWL-S, definido somente com parâmetros de entrada, corresponde a uma Operação WSDL do tipo `one-way`.
   c. Um Processo Atômico OWL-S, definido somente com parâmetros de saída, corresponde a uma Operação WSDL do tipo `notification`.
   d. Um Processo Composto OWL-S, que possui parâmetros de entrada e saída, porém com o envio dos parâmetros de saída acontecendo antes do recebimento dos parâmetros de entrada, corresponde a uma Operação WSDL do tipo `solicit-response`.
2. Cada grupo de parâmetros de entrada corresponde a um elemento WSDL chamado Mensagem (`message`). O mesmo acontece com cada grupo de parâmetros de saída. O padrão WSDL define um elemento para mensagens de entrada e outro elemento para mensagens de saída.
3. Os tipos (classes OWL) dos parâmetros de entrada e saída em um Processo Atômico OWL-S correspondem a um tipo abstrato no padrão WSDL.

Resumidamente, o processo de mapeamento consiste na identificação das mensagens e operações em um documento WSDL, identificação do Processo Atômico em questão no documento OWL-S e especificação das correspondências conforme listado nos passos 1, 2 e 3.

Portanto, Grounding OWL-S/WSDL inclui a criação de um documento WSDL com suas partes usuais: `types`, `message`, `operation`, `port-type`, `binding`, `port` e `service`. Adicionalmente, cada uma das partes de uma Mensagem (`message`) é mapeada em um parâmetro OWL-S (Entradas/Saídas ou `Inputs/Outputs`), que por sua vez tem seu tipo definido através de uma classe OWL. Esta classe pode ser definida internamente no documento WSDL (seção `types`) ou em um documento independente. No segundo caso, a classe OWL é referenciada no documento WSDL através do elemento `owl-s-parameter`, que na realidade representa apenas uma das três extensões OWL-S que afetam documentos WSDL:

1. Uma parte (`part`) de uma Mensagem WSDL (`message`) pode utilizar o atributo `owl-s-parameter` a fim de indicar o nome de um parâmetro OWL-S de entrada ou saída. Este nome deve ser apresentado de forma totalmente qualificada e ser uma instância da classe `Parameter`, pertencente à ontologia

OWL-S `Process`. Neste caso, o tipo do parâmetro não está definido na seção `types` do documento WSDL, mas pode ser obtido através da inspeção da propriedade `parameterType` do parâmetro referenciado.

2. Mensagens da WSDL possuem partes (`part`), que constituem um mecanismo flexível para descrever o conteúdo lógico/abstrato da mensagem e que são usadas pelo mecanismo de ligação para determinar o conteúdo da Mensagem WSDL. Quando uma parte (`part`) de uma Mensagem WSDL utiliza um tipo OWL diretamente, o atributo `encodingStyle` do elemento `binding` em um documento WSDL pode receber o endereço da definição do padrão OWL[17]. Isto indica que as partes da mensagem serão serializadas normalmente de acordo com o tipo definido.

3. Em cada elemento Operação de um documento WSDL (`operation`), um novo atributo chamado `owl-s-process` pode ser utilizado para indicar o nome de um processo atômico OWL-S. Resumidamente, esta abordagem relaciona uma operação com um processo atômico de forma direta.

As regras de mapeamento apresentadas até este ponto mostram apenas como elementos WSDL referenciam os elementos OWL-S correspondentes. Porém, é importante que o relacionamento inverso também seja estabelecido, fazendo com que elementos WSDL possam ser referenciados em um documento OWL-S. Apesar deste relacionamento bidirecional entre os dois documentos não ser obrigatório segundo a especificação, os autores do padrão OWL-S acreditam que isto seja uma boa prática, uma vez que são definidas construções em ambas as linguagens. A flexibilidade agregada é inegável.

A classe `WsdlGrounding`, realização concreta da classe abstrata `Grounding`, atende a este propósito. Cada instância desta classe contém uma lista de objetos `WsdlAtomicProcessGrounding`, representando os processos atômicos do serviço descrito. Um objeto `WsdlAtomicProcessGrounding` referencia elementos WSDL através das seguintes propriedades descritas no Quadro 2.

[17] http://www.w3.org/2002/07/owl

Propriedade	Descrição
`wsdlVersion`	URI que indica a versão do padrão WSDL em uso.
`wsdlDocument`	URI que indica o documento WSDL referenciado.
`wsdlOperation`	URI que indica a operação WSDL mapeada neste serviço atômico.
`wsdlService`	URI que indica o serviço WSDL que oferece a operação referenciada.
`wsdlInputMessage`	Um objeto que contém o URI da especificação da mensagem que carrega os parâmetros de entrada deste serviço atômico.
`wsdlOutputMessage`	Um objeto que contém o URI da especificação da mensagem que carrega os parâmetros de saída deste serviço atômico.
`wsdlInput`	Um objeto que contém o mapeamento entre um parâmetro de entrada OWL-S e uma parte de uma mensagem de entrada WSDL. Cada par é representado por um elemento `WsdlInputMessageMap`.
`wsdlOutput`	Um objeto que contém o mapeamento entre um parâmetro de saída OWL-S e uma parte de uma mensagem de saída WSDL. Desta vez, o par é representado pelo elemento `WsdlOutputMessageMap`.

Quadro 2 – Propriedades do objeto `WsdlAtomicProcessGrounding`, adaptada de Martin et al. (2004)

O diagrama de classes UML (*Unified Modeling Language*) na Figura 18 complementa graficamente o que já foi descrito de forma textual sobre Grounding OWL-S/WSDL. A ilustração facilita a identificação das ontologias, das classes e dos relacionamentos envolvidos neste mapeamento, bem como expõe os valores de cardinalidade.

Como pode ser observado, `WsdlGrounding` e `WsdlAtomicProcessGrounding` são as principais classes responsáveis pelo mapeamento OWL-S/WSDL. Ambas especializam uma camada abstrata definida respectivamente por `Grounding` e `AtomicProcessGrounding` da ontologia `Grounding`. Esta estrutura realiza a intenção original dos autores da ontologia OWL-S, ou seja, permite que novas especializações para a camada abstrata sejam desenvolvidas.

O padrão WSDL realmente tornou-se a principal abordagem para a descrição sintática de serviços RPC, especialmente SOAP, o que torna a especialização ilustrada na Figura 18 muito valiosa. Porém, no Capítulo 5, se propõe uma nova especialização, focada em serviços RESTful e baseada em documentos WADL. Esta nova abordagem será referenciada como Grounding OWL-S/WADL.

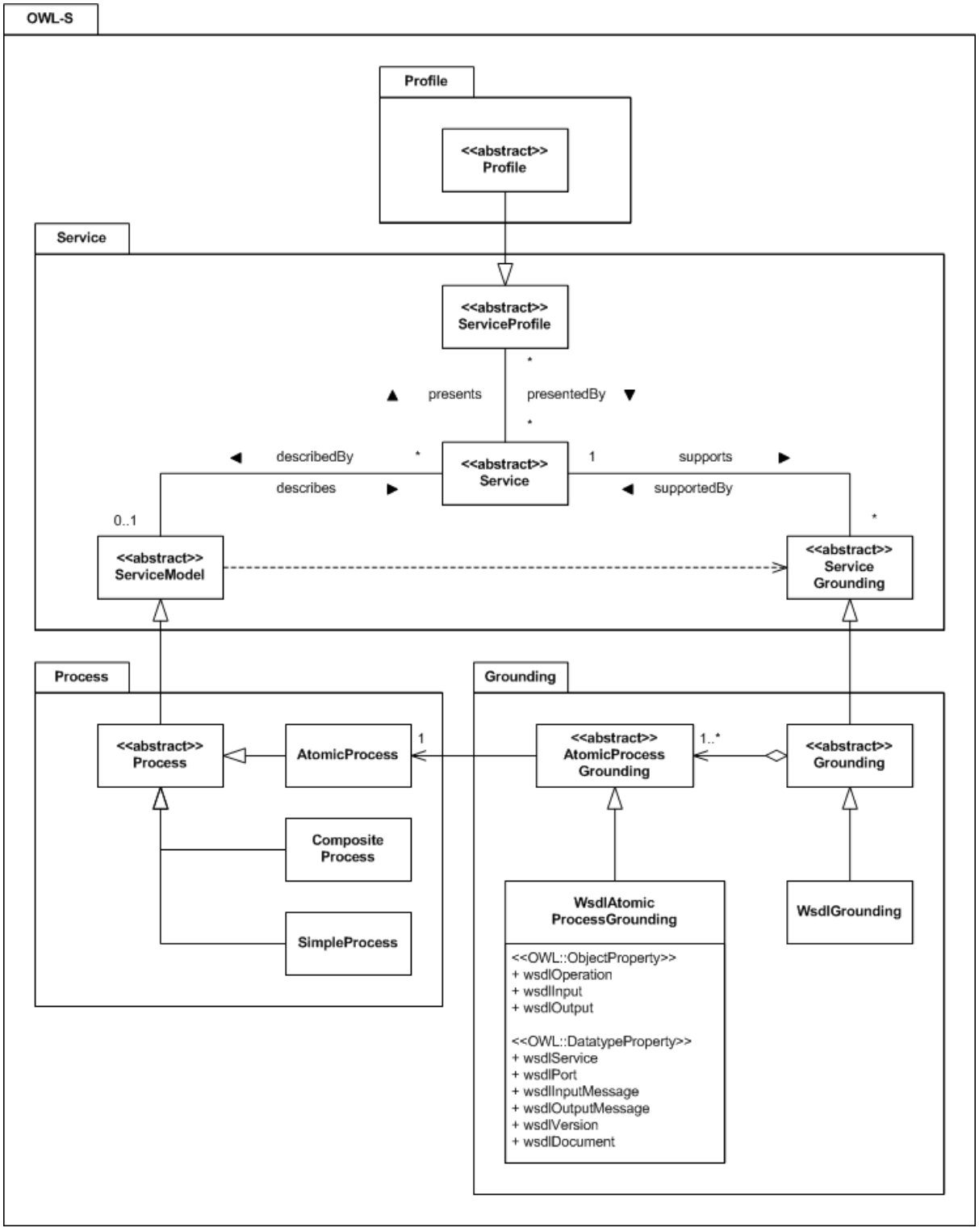

Figura 18 – Versão simplificada da ontologia OWL-S em um diagrama de classes UML

4.3 Exemplo

O exemplo apresentado nessa seção descreve um serviço RPC/SOAP de forma completa, ou seja, são cobertos os elementos semânticos e sintáticos. O domínio em questão trata da venda online de livros.

Para contextualizar o exemplo, nota-se que os fragmentos OWL-S (Figuras 19 e 20) e WSDL (Figura 21) apresentados foram extraídos de um exercício[18] conduzido pela equipe do DAML[19], um programa da agência DARPA[20]. O programa foi mantido pelo DOD[21], nos Estados Unidos, entre agosto de 2000 e janeiro de 2006.

Pois bem, o primeiro trecho da descrição semântica (Figura 19) contém, entre as linhas 01 e 18, referências para espaços de nomes (*namespaces*) definidos em documentos remotos. Dessa forma, qualquer documento XML é capaz de importar e utilizar as classes e propriedades definidas em outros documentos, sejam eles formatados na própria sintaxe XML ou em padrões derivados, como OWL ou WSDL.

No exemplo corrente, são importadas ontologias que pertencem ao padrão OWL-S (linhas 06 e 07), além da descrição sintática do serviço Web em questão (linha 08). Como normalmente acontece em qualquer documento de descrição semântica de serviços, ainda são importados os documentos que formam a base do padrão OWL-S, ou seja, XSD[22], RDF, RDFS e OWL (linhas 02-05, respectivamente).

Entretanto, a porção mais significativa deste documento encontra-se entre as linhas 26 e 42, onde é definido o processo atômico executado para vendas online de livros. Nota-se a declaração de dois parâmetros de entrada, o primeiro para o nome do livro a ser comprado (`In-BookName`, linhas 27-31) e o segundo para os dados de autenticação do usuário (`In-SignInInfo`, linhas 32-36).

[18] http://www.daml.org/services/owl-s/1.1/owl-s-wsdl.html

[19] DARPA Agent Markup Language: http://www.daml.org/

[20] Defense Advanced Research Projects Agency: http://www.darpa.mil/

[21] United States Department of Defense: http://www.defense.gov/

[22] XML Schema Definition

Adicionalmente, observa-se a declaração do parâmetro de saída (`Out-Confirmation`, linhas 37-41), que nesse caso representa a mensagem de confirmação da compra.

Já o segundo trecho da descrição semântica (Figura 20) apresenta o mapeamento do processo atômico para uma determinada descrição semântica. Observa-se a criação do objeto `FullCongoBuyGrounding` como instância da classe OWL-S `WsdlGrounding`, entre as linhas 03 e 05. Adicionalmente, observa-se, entre as linhas 07 e 52, a criação do objeto `CongoBuyGrounding` como instância da classe OWL-S `WsdlAtomicProcessGrounding`.

```
01 <!DOCTYPE uridef[
02   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
03   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
04   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
05   <!ENTITY owl "http://www.w3.org/2002/07/owl">
06   <!ENTITY grounding "http://daml.org/services/owl-s/1.1/Grounding.owl">
07   <!ENTITY process "http://daml.org/services/owl-s/1.1/Process.owl">
08   <!ENTITY wsdl "http://example.com/congo/congobuy.wsdl ">
09   <!ENTITY DEFAULT "http://example.com/congo/CongoBuy.owl">
10 ]>
11 <rdf:RDF xmlns:rdf="&rdf;#"
12          xmlns:rdfs="&rdfs;#"
13          xmlns:owl="&owl;#"
14          xmlns:xsd="&xsd;#"
15          xmlns:process="&process;#"
16          xmlns:grounding="&process;#"
17          xmlns:wsdl="&wsdl;#"
18          xmlns="&DEFAULT;#">
19
20 <!-- OWL-S Atomic Process Instance -->
21
22 <owl:Class rdf:ID="SignInInfo">
23   <!-- details omitted -->
24 </owl:Class>
25
26 <process:AtomicProcess rdf:ID="CongoBuy">
27   <process:hasInput>
28     <process:Input ref:ID="In-BookName">
29       <process:parameterType rdf:about="&xsd;#string">
30     </process:Input>
31   </process:hasInput>
32   <process:hasInput>
33     <process:Input ref:ID="In-SignInInfo">
34       <process:parameterType rdf:resource="#SignInInfo">
35     </process:Input>
36   </process:hasInput>
37   <process:hasOutput>
38     <process:Output ref:ID="Out-Confirmation">
39       <process:parameterType rdf:resource="&xsd;#string">
40     </process:Output>
41   </process:hasOutput>
42 </process:AtomicProcess>
```

Figura 19 – Exemplo de documento OWL-S: Processo Atômico

Ainda sobre Figura 20, cada parâmetro do processo é mapeado para seu correspondente no documento de descrição sintática WSDL. Portanto, o parâmetro de entrada `In-BookName` é mapeado para `&wsdl;#BookName` (linhas 21-28), ao passo que `In-SignInInfo` é mapeado para `&wsdl;#SignInInfo` (linhas 29-36). Finalmente, o parâmetro de saída `Out-Confirmation` é mapeado para o seu correspondente `&wsdl;#Confirmation` (linhas 39-46).

```
<!-- OWL-S Grounding Instance -->

<grounding:WsdlGrounding rdf:ID="FullCongoBuyGrounding">
<grounding:hasAtomicProcessGrounding rdf:resource="#CongoBuyGrounding"/>
</grounding:WsdlGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="CongoBuyGrounding">
  <grounding:owlsProcess rdf:resource="#congoBuy"/>
  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef>
      <grounding:portType>
        <xsd:uriReference rdf:value="&wsdl;#CongoBuyPortType"/>
      </grounding:portType>
      <grounding:operation>
        <xsd:uriReference rdf:value="&wsdl;#BuyBook"/>
      </grounding:operation>
    </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>

  <grounding:wsdlInputMessage rdf:resource="&wsdl;#CongoBuyInput"/>
  <grounding:wsdlInput>
    <grounding:wsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#In-BookName">
      <grounding:wsdlMessagePart>
        <xsd:uriReference rdf:value="&wsdl;#BookName">
      </grounding:wsdlMessagePart>
    </grounding:wsdlInputMessageMap>
  </grounding:wsdlInput>
  <grounding:wsdlInput>
    <grounding:wsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#In-SignInInfo">
      <grounding:wsdlMessagePart>
        <xsd:uriReference rdf:value="&wsdl;#SignInInfo">
      </grounding:wsdlMessagePart>
    </grounding:wsdlInputMessageMap>
  </grounding:wsdlInput>

  <grounding:wsdlOutputMessage rdf:resource="&wsdl;#CongoBuyOutput"/>
  <grounding:wsdlOutput>
    <grounding:wsdlOutputMessageMap>
      <grounding:owlsParameter rdf:resource="#Out-Confirmation">
      <grounding:wsdlMessagePart>
        <xsd:uriReference rdf:value="&wsdl;#Confirmation">
      </grounding:wsdlMessagePart>
    </grounding:wsdlOutputMessageMap>
  </grounding:wsdlOutput>

  <grounding:wsdlVersion rdf:resource="http://www.w3.org/TR/wsdl">
  <grounding:wsdlDocument>
    "http://example.com/congo/congobuy.wsdl"
  </grounding:wsdlDocument>
</grounding:WsdlAtomicProcessGrounding>
```

Figura 20 – Exemplo de documento OWL-S: Grounding OWL-S / WSDL

```
01 <?xml version="1.0"?>
02 <definitions name  = "CongoBuy"
03   targetNamespace  = "http://example.com/congo/congobuy.wsdl"
04   xmlns:tns        = "http://example.com/congo/congobuy.wsdl"
05   xmlns:congoOwl   = "http://example.com/congo/CongoBuy.owl#"
06   xmlns:soap       = "http://schemas.xmlsoap.org/wsdl/soap/"
07   xmlns:owl-s-wsdl = "http://www.daml.org/services/owl-s/wsdl/"
08   xmlns            = "http://schemas.xmlsoap.org/wsdl/">
09
10 <message name="CongoBuyInput">
11   <part name="BookName"
12         owl-s-wsdl:owl-s-parameter="congoOwl:In-BookName"/>
13   <part name="SignInInfo"
14         owl-s-wsdl:owl-s-parameter="congoOwl:In-SignInInfo"/>
15 </message>
16 <message name="CongoBuyOutput">
17   <part name="Confirmation"
18         owl-s-wsdl:owl-s-parameter="congoOwl:Out-Confirmation"/>
19 </message>
20
21 <portType name="CongoBuyPortType">
22  <operation name="BuyBook" owl-s-wsdl:owl-s-process="congoOwl:CongoBuy">
23    <input message="tns:CongoBuyInput"/>
24    <output message="tns:CongoBuyOutput"/>
25  </operation>
26 </portType>
27
28 <binding name="CongoBuySoapBinding" type="tns:CongoBuyPortType">
29   <soap:binding style="document"
30                transport="http://schemas.xmlsoap.org/soap/http"/>
31   <operation name="BuyBook">
32     <soap:operation
33       soapAction="http://example.com/congo/CongoBuy.owl#BuyBook"/>
34     <input>
35       <soap:body parts="BookName SignInInfo" use="encoded"
36                  namespace="http://example.com/congo/"
37                  encodingStyle="http://www.daml.org/2001/03/"/>
38     </input>
39     <output>
40       <soap:body parts="Confirmation" use="encoded"
41                  namespace="http://example.com/congo/"
42                  encodingStyle="http://www.daml.org/2001/03/"/>
43     </output>
44   </operation>
45 </binding>
46
47 <service name="CongoBuyService">
48   <port name="CongoBuyPort" binding="tns:CongoBuySoapBinding">
49     <soap:address location="http://example.com/congo/"/>
50   </port>
51 </service>
52 </definitions>
```

Figura 21 – Exemplo de documento WSDL: Descrição Sintática

A descrição sintática é provida pelo documento WSDL apresentado na Figura 21. Observa-se a definição da mensagem de entrada com seus dois parâmetros entre as linhas 10 e 15, além da definição da mensagem de saída entre as linhas 16 e 19. Cada um dos parâmetros em um documento WSDL referencia, opcionalmente, seu correspondente no documento OWL-S, criando o relacionamento bi-direcional entre as descrições sintática e semântica do serviço.

Nota-se, também, a utilização do protocolo SOAP entre as linhas 28 e 45. São considerados aspectos tecnológicos como, por exemplo, transporte e codificação (*encoding*) da mensagem.

Teoricamente, serviços RPC podem utilizar outros protocolos para essa ligação (*binding*), mas SOAP é certamente a escolha primária nesse grupo.

4.4 Discussão

A extensa teoria e o exemplo apresentados nesse capítulo permitem a conclusão que a realização concreta de serviços semânticos mostra-se plenamente possível. Entretanto, ao oferecer mapeamento semântico-sintático baseado no padrão WSDL somente, a abordagem apresentada limita-se a descrição de serviços RPC.

Felizmente, o padrão OWL-S define uma camada abstrata formada pelas classes `Grounding` e `AtomicProcessGrounding`, permitindo a realização concreta baseada em outros padrões sintáticos. Portanto, a abordagem apresentada no capítulo 5 utiliza-se do padrão WADL a fim de possibilitar o desenvolvimento de serviços semânticos de acordo com o estilo arquitetural REST.

O estabelecimento de padrões democráticos tende a acelerar o processo de adoção da Web Semântica pela comunidade de desenvolvedores e, conseqüentemente, de usuários. Espera-se que o suporte a diferentes estilos arquiteturais de serviços Web produza exatamente esse efeito.

5 SERVIÇOS SEMÂNTICOS RESTFUL

Esta nova classificação compreende serviços semânticos projetados de acordo com o paradigma RESTful, uma combinação pouco explorada pelo meio acadêmico e que portanto representa a maior contribuição desta pesquisa. O objetivo principal é possibilitar a inclusão dos serviços RESTful da Web 2.0 no contexto semântico, potencialmente facilitando a adoção da Web Semântica como um todo.

No sentido de iniciar a descrição formal desta categoria, os pontos a seguir representam os princípios estabelecidos para a criação de serviços semânticos RESTful. Obviamente, os princípios que definem os serviços RESTful, assim como aqueles que definem os serviços semânticos, foram incluídos na listagem. É importante que a integração entre os dois grupos aconteça sem provocar qualquer tipo de alteração nos padrões e protocolos já estabelecidos anteriormente pelos órgãos competentes. Os princípios são:

- Realizar o princípio da endereçabilidade através da atribuição de um identificador URI para cada recurso gerenciado;
- Realizar o princípio do estado não-persistente, permitindo apenas requisições que contenham todos os dados necessários para a correta execução do serviço;
- Realizar o princípio da conectividade, retornando representações que contenham apontadores para outras representações. Os recursos devem estar conectados entre si;
- Estar disponível para acesso HTTP, além de utilizar a interface unificada deste protocolo para permitir que um consumidor qualquer possa manipular os recursos gerenciados. A interface mínima compreende os métodos GET, HEAD, POST, PUT e DELETE;
- Oferecer sua descrição sintática para acesso automatizado, um processo a ser realizado por agentes de software da Web Semântica;
- Oferecer sua descrição semântica, sem ambigüidades, para processos automatizados de busca, seleção e composição. Estes são processos executados pelos agentes semânticos.

A primeira decisão sobre os protocolos a serem adotados refere-se àquele que descreverá sintaticamente os serviços desta nova categoria. O presente trabalho apóia a utilização da linguagem WADL para este fim, pois sua sintaxe XML favorece a integração com os demais padrões da Web Semântica. Segundo Richardson e Ruby (2007) – autores que cunharam o termo Serviços RESTful – o padrão WADL é realmente “a solução mais simples e elegante” para resolver o problema da descrição sintática.

Já a segunda decisão refere-se ao protocolo de descrição semântica. Neste caso, será adotada a linguagem OWL-S, uma ontologia genérica para serviços. A OWL-S é uma extensão de OWL, que por sua vez é a principal linguagem para a notação formal de ontologias, padronizada pelo W3C em 2004. Adicionalmente, o padrão OWL-S define uma camada abstrata que possibilita novas realizações do mapeamento semântico-sintático, ou seja, define uma arquitetura que é suficientemente flexível para possibilitar a inclusão dos serviços RESTful, e não apenas RPC/SOAP. Assim, a Figura 22 apresenta a estrutura a ser utilizada na descrição completa de serviços semânticos RESTful.

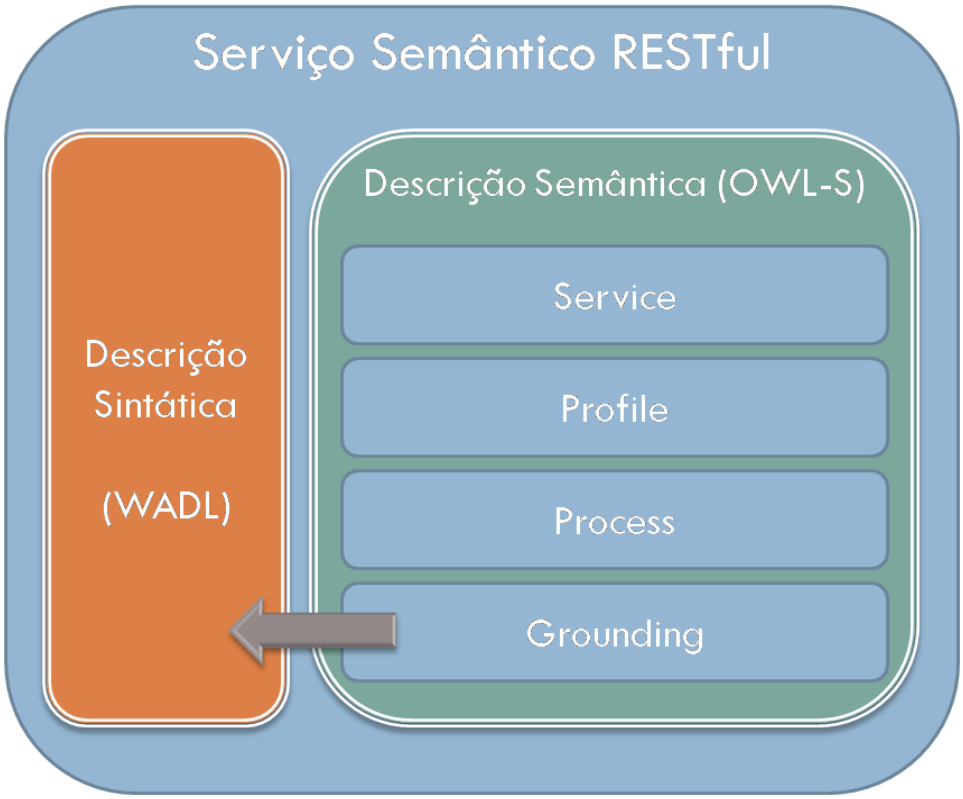

Figura 22 – Estrutura da descrição completa de um serviço semântico RESTful

Na abordagem proposta, as ontologias `Service`, `Profile` e `Process`, definidas pelo padrão OWL-S, não receberão qualquer tipo de extensão. Esta premissa reforça a noção de elementos abstratos e concretos na definição de um serviço. Como detalhado anteriormente, a ontologia `Grounding` é a única que lida com o mapeamento do serviço semântico para uma determinada descrição sintática, concreta, baseada em tecnologias especificas. Considerando os protocolos adotados nesta pesquisa, tal mapeamento será referenciado ao longo do texto como Grounding OWL-S/WADL. A ontologia `RESTfulGrounding` proposta é a responsável por realizar este mapeamento e será apresentada formalmente na seção 5.1.

5.1 OWL-S Service Grounding baseado em WADL e REST

O mapeamento Grounding OWL-S/WADL especializa a camada abstrata composta pelas classes `Grounding` e `AtomicProcessGrounding`, ambas definidas pelo padrão OWL-S. A Figura 23 apresenta um diagrama de classes UML que retrata tal especialização. Os pacotes que representam as ontologias `Service`, `Profile` e `Process` foram suprimidos para que o diagrama pudesse evidenciar a especialização da camada abstrata.

Note-se que as classes `WadlGrounding` e `WadlAtomicProcessGrounding` não fazem parte da ontologia OWL-S `Grounding`, mas teriam potencial para tanto. A arquitetura apresentada propõe o agrupamento das novas classes em uma ontologia dedicada, nomeada `RESTfulGrounding`. Esta abordagem evita qualquer modificação na especificação OWL-S, o que a princípio facilitará a adoção da nova ontologia.

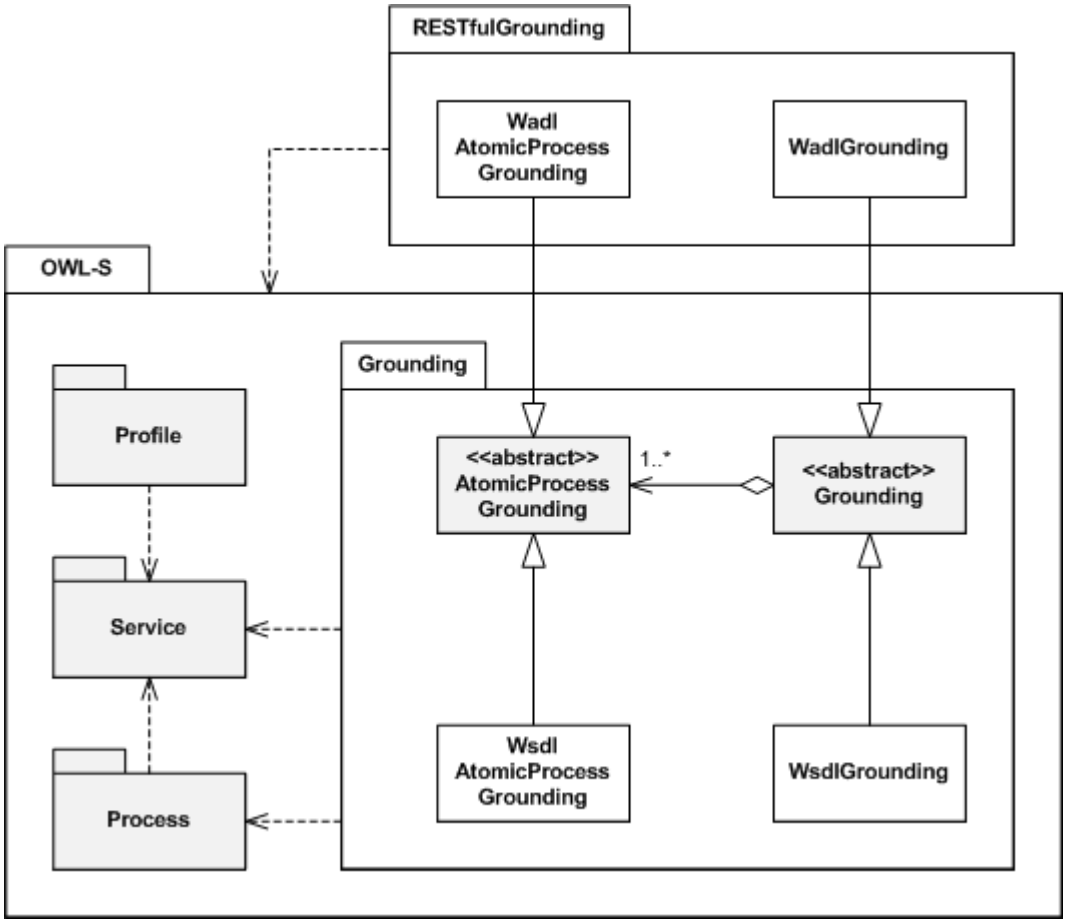

Figura 23 – `RESTfulGrounding`: extensão do padrão OWL-S para Grounding OWL-S/WADL

Para a formalização da proposta, faz-se necessária a descrição da ontologia `RESTfulGrounding` na sintaxe OWL, principal notação para ontologias na Web Semântica. Primeiramente, a Figura 24 apresenta a classe `WadlGrounding`, uma especialização da classe abstrata `Grounding`, conforme definido na instrução da linha 05. O código ainda define uma restrição nas linhas 07-12, discriminando que a propriedade `hasAtomicProcessGrounding` poderá receber instâncias da classe `WadlAtomicProcessGrounding`. Por não haver cardinalidade, esta restrição define uma coleção de instâncias e não apenas um objeto.

Note-se a utilização do prefixo `&grounding` nas linhas 05 e 09 para referenciar classes definidas na ontologia OWL-S `Grounding`. O cabeçalho do documento que possui este trecho de código deve necessariamente definir o URI completo da ontologia referenciada.

Referências precisas entre os documentos que formam uma ontologia é um requisito fundamental para que um agente de software da Web Semântica possa automaticamente encontrar a definição de todas as classes utilizadas.

```
01 <owl:Class rdf:ID="WadlGrounding">
02   <rdfs:comment>
03     The class that grounds every process to a WADL document.
04   </rdfs:comment>
05   <rdfs:subClassOf rdf:resource="&grounding;#Grounding"/>
06   <rdfs:subClassOf>
07     <owl:Restriction>
08       <owl:onProperty
09         rdf:resource="&grounding;#hasAtomicProcessGrounding"/>
10       <owl:allValuesFrom
11         rdf:resource="#WadlAtomicProcessGrounding"/>
12     </owl:Restriction>
13   </rdfs:subClassOf>
14 </owl:Class>
```

Figura 24 – Definição OWL da classe `WadlGrounding`

Já a Figura 25 apresenta a definição da segunda principal entidade para a ontologia `RESTfulGrounding`, ou seja, a classe `WadlAtomicProcessGrounding`. Esta entidade é responsável por mapear um determinado processo atômico OWL-S em um par formado por duas entidades do padrão WADL: o recurso afetado pela requisição e o método HTTP a ser utilizado.

Pois bem, o processo atômico é referenciado pela propriedade `owlsProcess`, pertencente à classe `AtomicProcessGrounding`. Esta é a classe abstrata estendida por `WadlAtomicProcessGrounding`, conforme especificado pela instrução na linha 06 da Figura 25. Qualquer processo atômico é uma instância de `AtomicProcess`, classe integrante da ontologia OWL-S `Process`.

Já sobre o par WADL referenciado, o recurso é uma instância da classe `Resource`, ao passo que e o método HTTP é uma instância da classe `Method`. Obviamente, ambas as classes estão definidas no padrão WADL. Conforme a cardinalidade imposta pela restrição definida nas linhas 07-12 da Figura 25, somente um par recurso/método é aceito por processo atômico.

```
01 <owl:Class rdf:ID="WadlAtomicProcessGrounding">
02   <rdfs:comment>
03      The class that relates elements of an OWL-S atomic process to a
04      WADL document.
05   </rdfs:comment>
06   <rdfs:subClassOf rdf:resource="&grounding;#AtomicProcessGrounding"/>
07   <rdfs:subClassOf>
08     <owl:Restriction>
09       <owl:onProperty rdf:resource="#wadlResourceMethod"/>
10       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
11       </owl:cardinality>
12     </owl:Restriction>
13   </rdfs:subClassOf>
14 </owl:Class>
15
16 <owl:ObjectProperty rdf:ID="wadlResourceMethod">
17   <rdfs:comment>
18     The WADL resource and method to which the atomic process refers to.
19     The process is defined by the property "owlsProcess".
20   </rdfs:comment>
21   <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
22   <rdfs:range rdf:resource="#WadlResourceMethodRef"/>
23 </owl:ObjectProperty>
```

Figura 25 – Definição OWL da classe `WadlAtomicProcessGrounding`

A responsável por conectar o processo atômico ao par WADL recurso/método é a propriedade `wadlResourceMethod`. Esta propriedade aceita instâncias da classe `WadlResourceMethodRef`, como apresentado pela Figura 25, linha 22. Caso fosse possível referenciar um par recurso/método através de um único URI, a propriedade `wadlResourceMethod` seria definida com o tipo de dados `anyURI`, pertencente ao padrão XSD. Entretanto, isso não é permitido pela versão 20090202 do padrão WADL, a mais recente até a data de redação deste trabalho. Portanto, faz-se necessária a definição de uma nova classe para agrupar os elementos deste par.

Este é exatamente o papel da classe `WadlResourceMethodRef`, cuja definição é apresentada pela Figura 26, linhas 01-06. Essa classe possui duas propriedades: `resource` (linhas 08-11) e `method` (linhas 13-16). Ambas são definidas com o tipo `anyURI`, e referenciam, respectivamente, um recurso e um método HTTP em um documento WADL.

Observa-se também que ambas as propriedades foram definidas com a mesma cardinalidade, onde apenas uma instância é permitida por propriedade (linhas 18-29 e 31-42).

```
01 <owl:Class rdf:ID="WadlResourceMethodRef">
02   <rdfs:comment>
03     The class that provides a unique specification of the
04     pair resource/method (both defined by the WADL standard).
05   </rdfs:comment>
06 </owl:Class>
07
08 <owl:DatatypeProperty rdf:ID="resource">
09   <rdfs:domain rdf:resource="#WadlResourceMethodRef"/>
10   <rdfs:range rdf:resource="&xsd;#anyURI"/>
11 </owl:DatatypeProperty>
12
13 <owl:DatatypeProperty rdf:ID="method">
14   <rdfs:domain rdf:resource="#WadlResourceMethodRef"/>
15   <rdfs:range rdf:resource="&xsd;#anyURI"/>
16 </owl:DatatypeProperty>
17
18 <owl:Class rdf:about="#WadlResourceMethodRef">
19   <rdfs:comment>
20     Restricting the cardinality of resource to one.
21   </rdfs:comment>
22   <rdfs:subClassOf>
23     <owl:Restriction>
24       <owl:onProperty rdf:resource="#resource"/>
25       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
26       </owl:cardinality>
27     </owl:Restriction>
28   </rdfs:subClassOf>
29 </owl:Class>
30
31 <owl:Class rdf:about="#WadlResourceMethodRef">
32   <rdfs:comment>
33     Restricting the cardinality of method to one.
34   </rdfs:comment>
35   <rdfs:subClassOf>
36     <owl:Restriction>
37       <owl:onProperty rdf:resource="#method"/>
38       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
39       </owl:cardinality>
40     </owl:Restriction>
41   </rdfs:subClassOf>
42 </owl:Class>
```

Figura 26 – Definição OWL da classe `WadlResourceMethodRef`

A classe `WadlAtomicProcessGrounding` ainda possui mais quatro propriedades que merecem certo destaque neste longo exercício de definição da ontologia `RESTfulGrounding`. A propriedade `wadlResourceMethod`, única apresentada até este momento, é declarada através da estrutura OWL `ObjectProperty`. Pois bem, outras duas propriedades são definidas através desta mesma estrutura: `wadlRequestParam` (Figura 27 e 28) e `wadlResponseParam` (Figura 29 e 30).

```
01 <owl:ObjectProperty rdf:ID="wadlRequestParam">
02   <rdfs:comment>
03     There should be one instance of this property for each WADL request
04     message parameter.
05   </rdfs:comment>
06   <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
07   <rdfs:range rdf:resource="#WadlRequestParamMap"/>
08 </owl:ObjectProperty>
```

Figura 27 – Descrição OWL da propriedade `wadlRequestParam`

Uma instância da classe `WadlAtomicProcessGrounding` deve possuir uma instância da propriedade `wadlRequestParam` para cada parâmetro existente na mensagem de requisição HTTP a ser enviada ao serviço. Essa propriedade oferece um mapeamento definido pela classe `WadlRequestParamMap`, que relaciona um parâmetro de entrada OWL-S a um parâmetro de requisição WADL.

Essa entidade de mapeamento, definida entre as linhas 01 e 08 da Figura 28, estende exatamente duas classes: `WadlMessageParamMap` (linhas 10-29) e `InputMessageMap` (definida na ontologia OWL-S `Grounding`). Esta dupla especialização é expressa nas linhas 06 e 07. Diferentemente da maioria das linguagens de programação, a linguagem para notação de ontologias OWL permite a herança múltipla. Na realidade, esta habilidade é oriunda do padrão RDFS, que compõe a base do padrão OWL.

Para finalizar a descrição da classe `WadlRequestParamMap`, observa-se a declaração de sua propriedade `wadlMessageParam` (linhas 31-37), que discrimina o parâmetro de requisição (classe WADL `Param`) referenciado no mapeamento. Já o parâmetro de entrada OWL-S é referenciado por `MessageMap` (a superclasse de `WadlMessageParamMap`) através da propriedade `owlsParameter`, conforme definido na própria ontologia OWL-S `Grounding`.

Apenas uma instância de `wadlMessageParam` é aceita por mapeamento, conforme restrição nas linhas 16-20. A mesma cardinalidade é imposta para a propriedade `owlsParameter`, conforme as linhas 23-27.

```
01 <owl:Class rdf:ID="WadlRequestParamMap">
02   <rdfs:comment>
03     The mapping entity that shows how to derive a WADL request
04     message parameter from an OWL-S parameter value.
05   </rdfs:comment>
06   <rdfs:subClassOf rdf:resource="#WadlMessageParamMap"/>
07   <rdfs:subClassOf rdf:resource="&grounding;#InputMessageMap"/>
08 </owl:Class>
09
10 <owl:Class rdf:ID="WadlMessageParamMap">
11   <rdfs:comment>
12     Message mapping for both WADL request and response parameters.
13   </rdfs:comment>
14   <rdfs:subClassOf rdf:resource="&grounding;#MessageMap"/>
15   <rdfs:subClassOf>
16     <owl:Restriction>
17       <owl:onProperty rdf:resource="#wadlMessageParam"/>
18       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
19       </owl:cardinality>
20     </owl:Restriction>
21   </rdfs:subClassOf>
22   <rdfs:subClassOf>
23     <owl:Restriction>
24       <owl:onProperty rdf:resource="&grounding;#owlsParameter"/>
25       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
26       </owl:cardinality>
27     </owl:Restriction>
28   </rdfs:subClassOf>
29 </owl:Class>
30
31 <owl:DatatypeProperty rdf:ID="wadlMessageParam">
32   <rdfs:comment>
33     WADL message parameter URI.
34   </rdfs:comment>
35   <rdfs:domain rdf:resource="#WadlMessageParamMap"/>
36   <rdfs:range rdf:resource="&xsd;#anyURI"/>
37 </owl:DatatypeProperty>
```

Figura 28 – Definição OWL da classe `WadlRequestParamMap`

A terceira propriedade da classe `WadlAtomicProcessGrounding`, nomeada `wadlResponseParam`, tem por objetivo mapear os parâmetros da mensagem de resposta HTTP. A descrição em OWL dessa propriedade pode ser encontrada na Figura 29.

```
01 <owl:ObjectProperty rdf:ID="wadlResponseParam">
02   <rdfs:comment>
03     There should be one instance of this property for each WADL response
04     message parameter.
05   </rdfs:comment>
06   <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
07   <rdfs:range rdf:resource="#WadlResponseParamMap"/>
08 </owl:ObjectProperty>
```

Figura 29 – Definição OWL da propriedade `wadlResponseParam`

Da mesma forma como `wadlRequestParam`, a propriedade `wadlResponseParam` é auxiliada por uma classe de mapeamento, neste caso `WadlResponseParamMap` (Figura 30).

Como visto, as duas entidades de mapeamento (`WadlRequestParamMap` e `WadlResponseParamMap`) estendem a classe abstrata `WadlMessageParamMap`. Conseqüentemente, ambas recebem a propriedade `wadlMessageParam` para referenciar um parâmetro de mensagem WADL. Para mensagens de resposta HTTP, o parâmetro referenciado poderá ser uma representação (classe WADL `Representation`) ou até mesmo um cabeçalho HTTP (classe WADL `Param`).

```
01 <owl:Class rdf:ID="WadlResponseParamMap">
02   <rdfs:comment>
03     The mapping entity that shows how to derive a WADL response
04     message parameter from an OWL-S parameter value.
05   </rdfs:comment>
06   <rdfs:subClassOf rdf:resource="#WadlMessageParamMap"/>
07   <rdfs:subClassOf rdf:resource="&grounding;#OutputMessageMap"/>
08 </owl:Class>
```

Figura 30 – Definição OWL da classe `WadlResponseParamMap`

Por fim, as últimas duas propriedades da classe `WadlAtomicProcessGrounding`, totalizando cinco, são: `wadlVersion` (Figura 31) e `wadlDocument` (Figura 32). A primeira apenas discrimina qual a versão do padrão WADL utilizada na descrição do serviço, ao passo em que a segunda discrimina qual o documento WADL referenciado. Diferentemente das demais, essas duas propriedades são definidas através da estrutura OWL `DatatypeProperty`, pois oferecem apenas um URI para o seu valor, ao invés de referenciarem outra classe.

```
01 <owl:DatatypeProperty rdf:ID="wadlVersion">
02   <rdfs:comment>
03     The URI that indicates the WADL version being used.
04   </rdfs:comment>
05   <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
06   <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
07   <rdfs:range rdf:resource="&xsd;#anyURI"/>
08 </owl:DatatypeProperty>
```

Figura 31 – Definição OWL da propriedade `wadlVersion`

```
01 <owl:DatatypeProperty rdf:ID="wadlDocument">
02   <rdfs:comment>
03     The URI that indicates the WADL document to which
04     this grounding refers to.
05   </rdfs:comment>
06 <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
07 <rdfs:range rdf:resource="&xsd;#anyURI"/>
08 </owl:DatatypeProperty>
```

Figura 32 – Definição OWL da propriedade `wadlDocument`

O conjunto de todas as classes e propriedades apresentado neste capítulo formaliza a ontologia proposta por este trabalho: `RESTfulGrounding`[23].

5.2 Exemplo

Com o objetivo de validar a ontologia proposta, esta seção apresenta um exemplo de aplicação sobre um serviço RESTful bem estabelecido no contexto da Web 2.0. Trata-se do serviço de busca de notícias disponibilizado pelo Yahoo, no qual qualquer aplicação consumidora torna-se capaz de receber noticias sobre uma determinada palavra-chave publicada nas mais diversas fontes eletrônicas, incluindo portais como CNN, BBC, UOL e o próprio Yahoo. Ao longo desta seção, o exemplo também se refere ao serviço através do acrônimo YNS (*Yahoo's News Search*) [24].

[23] http://fullsemanticweb.com/ontology/RESTfulGrounding/v1.0/RESTfulGrounding.owl

[24] http://developer.yahoo.com/search/news/V1/newsSearch.html

Muitos serviços RESTful não são formalmente descritos por seus provedores, o que impossibilita processos automatizados de descoberta, seleção e invocação. Desta forma, os agentes de software da Web Semântica não serão capazes de lidar com estes serviços, que por conseqüência seriam completamente excluídos da terceira fase da Web.

Esta ausência de formalismo também é realidade para o serviço considerado neste exemplo. Portanto, a primeira descrição a ser apresentada é essencialmente informal, ou seja, passível de interpretação apenas por humanos.

A Figura 33 define a mensagem de requisição a ser enviada ao serviço.

```
01 http://search.yahooapis.com/NewsSearchService/V1/newsSearch
```

Figura 33 – Base do endereço de requisição ao serviço YNS

Adicionalmente, esta mensagem deve ser decorada de acordo com os parâmetros apresentados no Quadro 3. Note-se que apenas os dois primeiros parâmetros são obrigatórios.

Parâmetro	Tipo	Restrições	Descrição
`appid`	Texto	-	Identificador da aplicação consumidora
`query`	Texto	-	Palavra chave a ser buscada
`type`	Texto	`all` (padrão), `any`, `phrase`	Tipo da busca
`results`	Inteiro	`10` (padrão), `50` (máximo)	Quantidade máxima de notícias
`start`	Inteiro	`1` (padrão)	Posição da primeira notícia na listagem
`sort`	Texto	`rank` (padrão), `date`	Ordenação dos itens resultantes
`language`	Texto	37 idiomas suportados[25]	Idioma no qual as noticias foram escritas

Quadro 3 – Parâmetros da mensagem de requisição ao serviço YNS

A Figura 34 apresenta um exemplo válido de requisição, no qual são solicitadas até duas notícias sobre automobilismo, ordenadas de forma decrescente por data e escritas em português. Os parâmetros opcionais não utilizados neste exemplo receberão os valores-padrão para a realização da busca.

[25] http://developer.yahoo.com/search/languages.html

```
01 http://search.yahooapis.com/NewsSearchService/V1/newsSearch
02   ?appid=YahooDemo
03   &query=automobilismo
04   &results=2
05   &sort=date
06   &language=pt
```

Figura 34 – Exemplo válido de requisição ao serviço YNS

O resultado desta busca é um documento XML estruturado de acordo com o Quadro 4. Obviamente, o conteúdo deste documento é temporal, portanto diferentes notícias serão retornadas para requisições realizadas em diferentes instantes no tempo. Um exemplo concreto é apresentado pela Figura 35.

Para consumir[26] este serviço, uma aplicação cliente deverá apoiar-se em uma biblioteca de acesso especializada, codificada em uma linguagem de programação compatível com o restante da aplicação. Ou seja, o processo de invocação[27] do serviço só é possível graças ao auxilio de um desenvolvedor humano, que estudou a descrição informal deste serviço e então a traduziu para código-fonte. Tal processo anula o propósito dos agentes de software da Web Semântica.

Parâmetro	Descrição
`ResultSet`	Elemento raiz do documento XML, que agrupa as notícias retornadas
`Result`	Elemento que contém uma notícia individualmente
`Title`	Título da notícia
`Summary`	Resumo da notícia
`Url`	Endereço eletrônico da notícia
`NewsSource`	Fonte que publicou a notícia
`NewsSourceUrl`	Endereço eletrônico da fonte que publicou a notícia
`Language`	Idioma no qual a notícia foi escrita
`PublishDate`	Data original de publicação da notícia
`ModificationDate`	Data na qual a notícia foi modificada, quando aplicável
`Thumbnail`	Endereço eletrônico para uma ilustração associada à notícia

Quadro 4 – Parâmetros da mensagem de resposta enviada pelo serviço YNS

[26] Consumo é o processo que inclui o envio da mensagem de requisição e a leitura da resposta.

[27] Invocação é simplesmente o processo de envio da mensagem de requisição ao serviço.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ResultSet xsi:schemaLocation="urn:yahoo:yn
03 http://api.search.yahoo.com/NewsSearchService/V1/NewsSearchResponse.xsd"
04 totalResultsAvailable="123" totalResultsReturned="2"
05 firstResultPosition="1">
06 <Result>
07   <Title>Villeneuve elogia Massa e fala sobre retornar</Title>
08   <Summary>O canadense, campeão mundial em 1997, vem ocupando o
09   noticiário especializado com os rumores envolvendo um possível retorno
10   à categoria máxima do automobilismo.</Summary>
11   <Url>http://tazio.uol.com.br/f-1/textos/12092/</Url>
12   <NewsSource>UOL Esporte</NewsSource>
13   <NewsSourceUrl>http://esporte.uol.com.br/</NewsSourceUrl>
14   <Language>pt</Language>
15   <PublishDate>1247944583</PublishDate>
16   <ModificationDate>1247944585</ModificationDate>
17 </Result>
18 <Result>
19   <Title>López segue invicto nos treinos da Top Race em SP</Title>
20   <Summary>A segunda bateria de treinos livres da Top Race em Interlagos
21   voltou a ter José María López na primeira posição.</Summary>
22   <Url>http://tazio.uol.com.br/outros/textos/12087/</Url>
23   <NewsSource>UOL Esporte</NewsSource>
24   <NewsSourceUrl>http://esporte.uol.com.br/</NewsSourceUrl>
25   <Language>pt</Language>
26   <PublishDate>1247931689</PublishDate>
27   <ModificationDate>1247931956</ModificationDate>
28 </Result>
29 </ResultSet>
```

Figura 35 – Exemplo concreto da mensagem de resposta enviada pelo serviço YNS

Para permitir a invocação automatizada, é necessário que o serviço em questão seja descrito sintaticamente através de alguma linguagem formal. Como descrito anteriormente, esta pesquisa adota a linguagem WADL para este fim. Portanto, a Figura 36 apresenta uma possível descrição sintática em WADL para o serviço de busca de notícias disponibilizado pelo Yahoo.

```
01 <?xml version="1.0"?>
02 <application
03  xsi:schemaLocation = "http://wadl.dev.java.net/2009/02 wadl.xsd"
04  xmlns              = "http://wadl.dev.java.net/2009/02"
05  xmlns:xsi          = "http://www.w3.org/2001/XMLSchema-instance"
06  xmlns:xsd          = "http://www.w3.org/2001/XMLSchema"
07  xmlns:tns          = "urn:yahoo:yn"
08  xmlns:yn           = "urn:yahoo:yn"
09  xmlns:ya           = "urn:yahoo:api">
10 <grammars>
11   <include href="NewsSearchResponse.xsd"/>
12   <include href="Error.xsd"/>
13 </grammars>
14 <representation mediaType="application/xml" element="yn:ResultSet"
15                 id="YNS-Rep-Success"/>
16 <representation mediaType="application/xml" element="ya:Error"
17                 id="YNS-Rep-Failure"/>
18 <method name="GET" id="YNS-Method-GET">
19   <request>
20    <param name="appid" style="query" type="xsd:string" required="true"/>
21    <param name="query" style="query" type="xsd:string" required="true"/>
22    <param name="type" style="query" default="all">
23      <option value="all"/>
24      <option value="any"/>
25      <option value="phrase"/>
26    </param>
27    <param name="results" style="query" type="xsd:int" default="10"/>
28    <param name="start" style="query" type="xsd:int" default="1"/>
29    <param name="sort" style="query" default="rank">
30      <option value="rank"/>
31      <option value="date"/>
32    </param>
33    <param name="language" style="query" type="xsd:string"/>
34   </request>
35   <response status="200"><representation href="#YNS-Rep-Success"/>
36   </response>
37   <response status="400"><representation href="#YNS-Rep-Failure"/>
38   </response>
39 </method>
40 <resources base="http://search.yahooapis.com/NewsSearchService/V1/">
41   <resource path="newsSearch" id="YNS-Resource">
42     <method href="#YNS-Method-GET"/>
43   </resource>
44 </resources>
45 </application>
```

Figura 36 – Descrição sintática em WADL para o serviço YNS

O único recurso gerenciado por este serviço é uma coleção de notícias associadas a uma determinada palavra-chave. A definição deste recurso é, por sua vez, segmentada pelos métodos HTTP. Neste caso, entretanto, apenas o método GET é suportado (linhas 18-39, Figura 36).

Internamente à definição do método, são apresentadas a requisição e as possíveis respostas do serviço. Primeiramente, a requisição é composta por uma série de parâmetros, conforme apresentado entre as linhas 19 e 34. Note-se que os 7 parâmetros apresentados na Figura 36 são os mesmos listados no Quadro 3. Os parâmetros `type` e `sort` enumeram seus possíveis valores através do elemento `option`, como exibido respectivamente nas linhas 22-26 e 29-32.

Duas mensagens de resposta são previstas pelo serviço. A primeira delas trata de operações concluídas com sucesso (linhas 35-36). Neste caso, o código HTTP 200 é utilizado e o objeto XML resultante é serializado de acordo com a classe `ResultSet`, uma entidade definida no documento XSD `NewsSearchResponse`. Este documento é referenciado no cabeçalho da descrição WADL na linha 11.

Já a segunda mensagem de resposta trata de operações que falharam durante sua execução (linhas 37-38). Neste caso, o código HTTP 400 é utilizado, e o objeto XML resultante é definido pela classe `Error`, uma entidade pertencente ao documento XSD `Error`. Este documento está referenciado na linha 12.

Desta forma, espera-se que um agente de software da Web Semântica possa invocar este serviço sem a intervenção humana, pois toda a estrutura de requisição e os possíveis objetos de resposta foram sintaticamente descritos em linguagem formal, passível de interpretação por máquinas.

Porém, para que todo o potencial da Web Semântica seja realizado, é necessário que os agentes de software possam encontrar e selecionar os serviços relevantes ao seu objetivo de forma autônoma. Portanto, uma camada semântica deve ser incluída na descrição do serviço. Novamente, a abordagem proposta baseia-se no padrão OWL-S em conjunto com sua extensão `RESTfulGrounding`.

A descrição semântica do serviço será dividida em quatro documentos OWL. Para fins didáticos, os quatro documentos OWL compartilharão o cabeçalho da Figura 37.

A Figura 38 apresenta a descrição do serviço baseada na ontologia `Service`, ou seja, uma descrição mais abstrata, que referencia os demais documentos que formam a descrição semântica. A Figura 39 apresenta o perfil do serviço com base na ontologia `Profile`. Já o único processo atômico no serviço de busca de notícias

do Yahoo é definido nas Figuras 40 e 41 através da ontologia `Process`. Por fim, a Figura 42 apresenta os elementos concretos que definem o acesso ao serviço através da ontologia `Grounding` e também das construções WADL exibidas na Figura 36.

A Figura 37 referencia documentos OWL no endereço real do serviço disponibilizado pelo Yahoo (linhas 11-15), embora esses documentos não existam fisicamente no local especificado. O código a seguir pretende apenas exemplificar um provável local para o armazenamento dos documentos, caso os mesmos existissem.

```
01 <?xml version="1.0" encoding="ISO-8859-1"?>
02
03 <!DOCTYPE uridef[
04 <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns">
05 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
06 <!ENTITY owl  "http://www.w3.org/2002/07/owl">
07 <!ENTITY service   "http://daml.org/services/owl-s/1.2/Service.owl">
08 <!ENTITY profile   "http://daml.org/services/owl-s/1.2/Profile.owl">
09 <!ENTITY process   "http://daml.org/services/owl-s/1.2/Process.owl">
10 <!ENTITY grounding "http://daml.org/services/owl-s/1.2/Grounding.owl">
11 <!ENTITY ypf "http://search.yahooapis.com/NewsSearchService/V1/YPf.owl">
12 <!ENTITY ypc "http://search.yahooapis.com/NewsSearchService/V1/YPc.owl">
13 <!ENTITY ygr "http://search.yahooapis.com/NewsSearchService/V1/YGr.owl">
14 <!ENTITY
15      DEFAULT "http://search.yahooapis.com/NewsSearchService/V1/YSv.owl">
16 ]>
17
18 <rdf:RDF
19  xmlns:rdf       = "&rdf;#"
20  xmlns:rdfs      = "&rdfs;#"
21  xmlns:owl       = "&owl;#"
22  xmlns:service   = "&service;#"
23  xmlns:profile   = "&profile;#"
24  xmlns:process   = "&process;#"
25  xmlns:grounding = "&grounding;#"
26  xmlns:ypf       = "&ypf;#"
27  xmlns:ypc       = "&ypc;#"
28  xmlns:ygr       = "&ygr;#"
29  xmlns           = "&DEFAULT;#"
30  xml:base        = "&DEFAULT;"
31 >
```

Figura 37 – Cabeçalho OWL compartilhado para o serviço YNS

```
01 <!-- Ontology -->
02
03 <owl:Ontology rdf:about="">
04   <rdfs:comment>
05     This ontology represents the OWL-S service description
06     for Yahoo's News Search Web Service.
07   </rdfs:comment>
08   <owl:imports rdf:resource="&service;"/>
09   <owl:imports rdf:resource="&profile;"/>
10   <owl:imports rdf:resource="&process;"/>
11   <owl:imports rdf:resource="&restful;"/>
12   <owl:imports rdf:resource="&ypf;"/>
13   <owl:imports rdf:resource="&ypc;"/>
14   <owl:imports rdf:resource="&ygr;"/>
15 </owl:Ontology>
16
17 <!-- Direct references to the nested ontologies -->
18
19 <service:Service rdf:ID="YNS-Service">
20   <!-- Reference to the Service Profile -->
21   <service:presents rdf:resource="&ypf;#YNS-Profile"/>
22
23   <!-- Reference to the Service Process Model -->
24   <service:describedBy rdf:resource="&ypc;#YNS-Process"/>
25
26   <!-- Reference to the Service Grounding -->
27   <service:supports rdf:resource="&ygr;#YNS-Grounding"/>
28 </service:Service>
29
30 <!-- Inverse links -->
31
32 <profile:Profile rdf:about="&ypf;#YNS-Profile">
33   <service:presentedBy rdf:resource="#YNS-Service"/>
34 </profile:Profile>
35
36 <process:AtomicProcess rdf:about="&ypc;#YNS-Process">
37   <service:describes rdf:resource="#YNS-Service"/>
38 </process:AtomicProcess>
39
40 <restful:WadlGrounding rdf:about="&ygr;#YNS-Grounding">
41   <service:supportedBy rdf:resource="#YNS-Service"/>
42 </restful:WadlGrounding>
43
44 </rdf:RDF>
```

Figura 38 – Parte da descrição semântica do serviço YNS baseada na ontologia OWL-S `Service`

Um detalhe fundamental na Figura 38 é a utilização da classe `WadlGrounding` (linhas 40-42). Como visto anteriormente neste capítulo, esta é a principal entidade da ontologia `RESTfulGrounding` proposta, e relaciona a descrição semântica com a sintática em serviços RESTful.

```
01 <owl:Ontology rdf:about="">
02   <rdfs:comment>
03     This ontology represents the OWL-S service profile description
04     for Yahoo's News Search Web Service.
05   </rdfs:comment>
06   <owl:imports rdf:resource="&service;"/>
07   <owl:imports rdf:resource="&profile;"/>
08   <owl:imports rdf:resource="&process;"/>
09   <owl:imports rdf:resource="&ysv;"/>
10   <owl:imports rdf:resource="&ypc;"/>
11 </owl:Ontology>
12
13 <profile:Profile rdf:ID="YNS-Profile">
14   <profile:has_process rdf:resource="&ypc;#YNS-Process"/>
15   <profile:serviceName>Yahoo News Search</profile:serviceName>
16   <profile:textDescription>
17     This service searches for news across a wide range of web portals.
18   </profile:textDescription>
19   <profile:contactInformation>
20     <actor:Actor rdf:ID="YNS-Contact">
21       <actor:name>Yahoo Inc.</actor:name>
22       <actor:title>Copyright Agent</actor:title>
23       <actor:phone>(408) 349-5080</actor:phone>
25       <actor:fax>(408) 349-7821</actor:fax>
26       <actor:email>copyright@yahoo-inc.com</actor:email>
27       <actor:physicalAddress>Sunnyvale, CA, USA</actor:physicalAddress>
28       <actor:webURL>http://info.yahoo.com/copyright/us/</actor:webURL>
29     </actor:Actor>
30   </profile:contactInformation>
31   <profile:hasInput  rdf:resource="&ypc;#YNS-AppID"/>
32   <profile:hasInput  rdf:resource="&ypc;#YNS-Query"/>
33   <profile:hasInput  rdf:resource="&ypc;#YNS-Type"/>
34   <profile:hasInput  rdf:resource="&ypc;#YNS-Results"/>
35   <profile:hasInput  rdf:resource="&ypc;#YNS-Start"/>
36   <profile:hasInput  rdf:resource="&ypc;#YNS-Sort"/>
37   <profile:hasInput  rdf:resource="&ypc;#YNS-Language"/>
38   <profile:hasOutput rdf:resource="&ypc;#YNS-Output"/>
39 </profile:Profile>
40
41 </rdf:RDF>
```

Figura 39 – Parte da descrição semântica do serviço YNS baseada na ontologia OWL-S `Profile`

As informações de contato listadas como atributos do elemento `Actor` (linhas 20-29) formam a única porção da descrição semântica endereçada para consumo humano.

Ao analisar um documento baseado na ontologia OWL-S `Profile`, como é no caso da Figura 39, um agente de software da Web Semântica está mais interessado nos parâmetros de entrada e saída que definem o processo atômico de execução do serviço. Tais parâmetros podem ser utilizados na verificação de compatibilidade entre o requisitante do serviço e o provedor.

As Figuras 40 e 41 compõem a parte da descrição semântica que se apóia na ontologia OWL-S `Process`. Inicialmente, na Figura 40, são definidos os tipos de dados utilizados na definição dos parâmetros de entrada do processo atômico: `SearchType` (linhas 13-19), `SearchResultSort` (linhas 21-26) e, resumidamente, `SearchResultLanguage` (linhas 28-42). Nota-se que cada um dos tipos já define seus valores possíveis através de coleções declaradas pelo comando OWL `oneOf`.

```
<!-- Ontology -->

<owl:Ontology rdf:about="">
  <rdfs:comment>
    This ontology represents the OWL-S service process description
    for Yahoo's News Search Web Service.
  </rdfs:comment>
  <owl:imports rdf:resource="&process;"/>
</owl:Ontology>

<!-- Basic Data Types -->

<owl:Class rdf:ID="SearchType">
  <owl:oneOf rdf:parseType="Collection">
    <SearchType rdf:ID="all"/>
    <SearchType rdf:ID="any"/>
    <SearchType rdf:ID="phrase"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="SearchResultSort">
  <owl:oneOf rdf:parseType="Collection">
    <SearchResultSort rdf:ID="rank"/>
    <SearchResultSort rdf:ID="date"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="SearchResultLanguage">
  <rdfs:comment>
    Yahoo's News Search Web Service supports 37 languages,
    although only 5 of them have been listed here for the sake
    of brevity, namely English, Spanish, French, Italian, and
    Portuguese.
  </rdfs:comment>
  <owl:oneOf rdf:parseType="Collection">
    <SearchResultLanguage rdf:ID="en"/>
    <SearchResultLanguage rdf:ID="es"/>
    <SearchResultLanguage rdf:ID="fr"/>
    <SearchResultLanguage rdf:ID="it"/>
    <SearchResultLanguage rdf:ID="pt"/>
  </owl:oneOf>
</owl:Class>
```

Figura 40 – Parte da descrição semântica do serviço YNS baseada na ontologia OWL-S `Process`

```
<!-- Atomic Process -->
<process:AtomicProcess rdf:ID="YNS-AtomicProcess">
  <!-- Input Parameters -->
  <process:hasInput>
    <process:Input rdf:ID="YNS-AppID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#ID
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Query">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Type">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SearchType
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Results">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        &xsd;#positiveInteger
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Start">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        &xsd;#positiveInteger
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Sort">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        #SearchResultSort
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="YNS-Language">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        #SearchResultLanguage
      </process:parameterType>
    </process:Input>
  </process:hasInput>
  <!-- Output Parameters -->
  <process:hasOutput>
    <process:Output rdf:ID="YNS-Output">
     <process:parameterType rdf:datatype="&xsd;#anyURI">#YSN-Output-Type
     </process:parameterType>
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
</rdf:RDF>
```

Figura 41 – Parâmetros de entrada e saída do processo atômico relativo ao serviço YNS

Já a Figura 41 apresenta o processo atômico propriamente dito. São definidos os parâmetros de entrada (linhas 03-49) e saída (linhas 50-56), o que exprime a transformação de dados suportada pelo serviço.

Por fim, a Figura 42 refere-se à porção da descrição semântica que se apóia na ontologia OWL-S `Grounding`. Mais especificamente, o documento a seguir utiliza as classes que foram definidas na ontologia `RESTfulGrounding` proposta. Nota-se que este é o único documento que requer a aplicação da nova ontologia para permitir a descrição de serviços RESTful. Os demais documentos apresentados durante esse capítulo baseiam-se simplesmente nas classes e propriedades já definidas originalmente pelo padrão OWL-S. Tal observação confirma a proposta de organização ilustrada pela Figura 22.

```
01 <!-- Ontology -->
02 <owl:Ontology rdf:about="">
03   <rdfs:comment>
04     This ontology represents the OWL-S service grounding description
05     for Yahoo's News Search Web Service.
06   </rdfs:comment>
07   <owl:imports rdf:resource="&service;"/>
08   <owl:imports rdf:resource="&process;"/>
09   <owl:imports rdf:resource="&grounding;"/>
10   <owl:imports rdf:resource="&restful;"/>
11   <owl:imports rdf:resource="&ypc;"/>
12 </owl:Ontology>
13
14 <!-- WADL Grounding -->
15 <restful:WadlGrounding rdf:ID="YNS-Grounding">
16   <grounding:hasAtomicProcessGrounding
17     rdf:resource="#YNS-AtomicProcessGrounding"/>
18 </restful:WadlGrounding>
19
20 <!-- WADL Atomic Process Grounding -->
21 <restful:WadlAtomicProcessGrounding rdf:ID="YNS-AtomicProcessGrounding">
22   <grounding:owlsProcess rdf:resource="&ypc;#YNS-AtomicProcess"/>
23   <restful:wadlResourceMethod>
24     <restful:WadlResourceMethodRef>
25       <restful:resource rdf:datatype="&xsd;#anyURI">
26         &wadl;#NewsResource
27       </restful:resource>
28       <restful:method rdf:datatype="&xsd;#anyURI">
29         &wadl;#NewsMethodGET
30       </restful:method>
31     </restful:WadlResourceMethodRef>
32   </restful:wadlResourceMethod>
33   <restful:wadlVersion rdf:datatype="&xsd;#anyURI">
34    https://wadl.dev.java.net/wadl20090202.xsd
35   </restful:wadlVersion>
```

```
36   <restful:wadlDocument rdf:datatype="&xsd;#anyURI">
37    http://search.yahooapis.com/NewsSearchService/V1/YahooNewsSearch.wadl
38   </restful:wadlDocument>
39
40   <!-- Request Parameters -->
41   <restful:wadlRequestParam>
42     <restful:WadlRequestParamMap>
43       <grounding:owlsParameter rdf:resource="&ypc;#YNS-AppID"/>
44       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
45         &wadl;#appid
46       </restful:wadlMessageParam>
47     </restful:WadlRequestParamMap>
48   </restful:wadlRequestParam>
49   <restful:wadlRequestParam>
50     <restful:WadlRequestParamMap>
51       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Query"/>
52       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
53         &wadl;#query
54       </restful:wadlMessageParam>
55     </restful:WadlRequestParamMap>
56   </restful:wadlRequestParam>
57   <restful:wadlRequestParam>
58     <restful:WadlRequestParamMap>
59       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Type"/>
60       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
61         &wadl;#type
62       </restful:wadlMessageParam>
63     </restful:WadlRequestParamMap>
64   </restful:wadlRequestParam>
65   <restful:wadlRequestParam>
66     <restful:WadlRequestParamMap>
67       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Query"/>
68       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
69         &wadl;#query
70       </restful:wadlMessageParam>
71     </restful:WadlRequestParamMap>
72   </restful:wadlRequestParam>
73   <restful:wadlRequestParam>
74     <restful:WadlRequestParamMap>
75       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Results"/>
76       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
77         &wadl;#results
78       </restful:wadlMessageParam>
79     </restful:WadlRequestParamMap>
80   </restful:wadlRequestParam>
81   <restful:wadlRequestParam>
82     <restful:WadlRequestParamMap>
83       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Start"/>
84       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
85         &wadl;#start
86       </restful:wadlMessageParam>
87     </restful:WadlRequestParamMap>
88   </restful:wadlRequestParam>
```

```
89    <restful:wadlRequestParam>
90      <restful:WadlRequestParamMap>
91        <grounding:owlsParameter rdf:resource="&ypc;#YNS-Sort"/>
92        <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
93          &wadl;#sort
94        </restful:wadlMessageParam>
95      </restful:WadlRequestParamMap>
96    </restful:wadlRequestParam>
97    <restful:wadlRequestParam>
98      <restful:WadlRequestParamMap>
99        <grounding:owlsParameter rdf:resource="&ypc;#YNS-Language"/>
100       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
101         &wadl;#language
101       </restful:wadlMessageParam>
102     </restful:WadlRequestParamMap>
103   </restful:wadlRequestParam>
104
105   <!-- Response Parameters -->
106   <restful:wadlResponseParam>
107     <restful:WadResponseParamMap>
108       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Out-Success"/>
109       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
110         &wadl;#YNS-Rep-Success
111       </restful:wadlMessageParam>
112     </restful:WadResponseParamMap>
113   </restful:wadlResponseParam>
114   <restful:wadlResponseParam>
115     <restful:WadResponseParamMap>
116       <grounding:owlsParameter rdf:resource="&ypc;#YNS-Out-Failure"/>
117       <restful:wadlMessageParam rdf:datatype="&xsd;#anyURI">
118         &wadl;#YNS-Rep-Failure
119       </restful:wadlMessageParam>
120     </restful:WadResponseParamMap>
121   </restful:wadlResponseParam>
122</restful:WadlAtomicProcessGrounding>
123</rdf:RDF>
```

Figura 42 – Parte da descrição semântica do serviço YNS baseada na ontologia `RESTfulGrounding`

A aplicação da ontologia `RESTfulGrounding` inicia-se entre as linhas 15 e 18, onde a classe `WadlGrounding` é instanciada. Torna-se explícito, portanto, que o serviço Web em questão não somente é classificado como RESTful, como já teve sua sintaxe descrita formalmente através de um documento WADL. Neste exemplo, a instância de `WadlGrounding` foi nomeada `YNS-Grounding`.

Observa-se que o serviço YNS é simples o suficiente para ser descrito através de um único processo atômico. Na Figura 42, a associação entre as instâncias de `WadlGrounding` e `WadlAtomicProcessGrounding` acontece nas linhas 16 e 17.

Já a definição completa do mapeamento do processo, cuja instância foi nomeada `YNS-AtomicProcessGrounding`, encontra-se entre as linhas 21 e 122.

Obviamente, todo mapeamento deve apontar para o processo descrito. No exemplo corrente, o processo em questão foi nomeado `YNS-AtomicProcess`. Tal processo foi apresentado na Figura 41, e referenciado no mapeamento da Figura 42 na linha 22.

Como detalhado na seção 5.1, cada mapeamento de processo atômico deve referenciar um recurso e um método HTTP, ambos definidos no documento WADL. Este comportamento foi realizado entre as linhas 23 e 32 através da propriedade `wadlResourceMethod`, pertencente à classe `WadlAtomicProcessGrounding`, como definido pela ontologia `RESTfulGrounding`. Pois bem, o recurso e o método referenciados foram, respectivamente, `NewsResource` e `NewsMethodGET`. Nota-se a utilização do prefixo `&wadl` para ambas as referências. Tal notação é aplicada ao longo do documento para todas as entidades WADL referenciadas.

O documento WADL é discriminado pela propriedade `wadlDocument` (linhas 36-38), ao passo que a versão deste padrão é expressa através da propriedade `wadlVersion` (linhas 33-35). Nota-se que o documento WADL referenciado é justamente aquele apresentado na Figura 36.

Finalmente, cada parâmetro de entrada é realizado através de uma instância da propriedade `wadlRequestParam`, como no bloco entre as linhas 41 e 48. Já os parâmetros de saída são mapeados através de instâncias da propriedade `wadlResponseParam`, como no bloco das linhas 106-113.

Considerando os documentos apresentados nessa seção, espera-se que um agente de software da Web Semântica possa, automaticamente, descobrir e executar o serviço de busca de notícias disponibilizado pelo Yahoo.

5.3 Discussão

A ontologia `RESTfulGrounding` apresentada, ao especializar a camada abstrata definida pela ontologia OWL-S `Grounding`, permite o desenvolvimento de serviços semânticos RESTful.

A partir dessa contribuição, pode-se concluir que a teoria sobre serviços semânticos torna-se mais democrática, pois agora oferece meios de mapeamento semântico-sintático para ambos os grupos de serviços Web.

Com o objetivo de facilitar a adoção da especialização proposta, nenhuma alteração é feita sobre os demais padrões já estabelecidos na arquitetura da Web Semântica. Assim, serviços semânticos RPC legados continuarão corretos, pois as demais classes OWL-S pertencentes ao pacote `Grounding` permaneceram inalteradas.

6 CONSIDERAÇÕES FINAIS

A realização completa da Web Semântica ainda depende de um considerável esforço da comunidade acadêmica. Novas iniciativas continuarão sendo propostas para que padrões sejam estabelecidos nas camadas menos exploradas de sua arquitetura.

O presente trabalho foca especificamente na anotação semântica de serviços Web, que por sua vez são fundamentais para o estabelecimento da Web Semântica como um todo. São entidades que atuam como fonte de dados primária para agentes de software e demais consumidores.

Porém, para que todo o potencial seja extraído, serviços Web devem ser descritos semântica e sintaticamente. Essa é a única forma conhecida de permitir que esses componentes sejam descobertos e executados automaticamente, ou seja, sem a assistência humana.

É desejável, portanto, que a maior quantidade possível de serviços seja descrita de forma completa, em notação formal. De acordo com essa constatação, não seria o suficiente a definição de padrões semânticos para apenas um grupo de serviços. Ou seja, um posicionamento unilateral não seria o mais adequado.

Entretanto, essa é a realidade. O caminho de migração ao contexto semântico foi oferecido somente aos serviços RPC/SOAP, fazendo com que o grupo mais popular na Web 2.0 seja impossibilitado de ser acessado na Web Semântica. De fato, os padrões definidos até então simplesmente não são adequados ou compatíveis aos serviços REST/HTTP.

Observa-se que não é relevante a opinião pessoal de qualquer Engenheiro de Software quanto à preferência pelo grupo RPC ou REST. Assim como em outras especialidades de engenharia, um mesmo problema pode ser perfeitamente resolvido por diferentes abordagens. É fato que existem diferentes grupos de serviços disponíveis na Web. Então, considerando o mapeamento semântico-

sintático com o problema em questão, o importante na realidade é que ele possa ser resolvido por ambos os grupos, cada um com sua própria abordagem.

Assim, espera-se alcançar um número maior de serviços e, conseqüentemente, promover a Web Semântica de forma mais expressiva. Seria muito valioso poder contar com a vibrante e ativa comunidade da Web 2.0 para acelerar os processos de evolução e adoção do paradigma semântico. A solução proposta nesse trabalho pode colaborar para a existência desse apoio.

Pois bem, faz-se necessária uma avaliação técnica da solução apresentada. Primeiramente, pode-se perceber que o mapeamento Grounding OWL-S/WADL, concretizado pela nova ontologia `RESTfulGrounding`, foi claramente inspirado em Grounding OWL-S/WSDL, proposto por Martin et al. (2004) quando especificaram o próprio padrão OWL-S. A semelhança existente entre as duas abordagens, mesmo tratando de estilos arquiteturais tão distintos, confirma o caráter estendível da camada abstrata definida pela ontologia OWL-S `Grounding`.

Realmente, novas especializações são incentivadas pelos autores, que são claros ao relatar que “a intenção não é prescrever uma única abordagem possível para *grounding* a ser usada com todos os serviços, mas sim prover uma abordagem geral, canônica e largamente aplicável, que será útil na grande maioria dos casos”.

Agora, com a definição da ontologia `RESTfulGrounding`, serviços projetados de acordo com o paradigma RESTful também poderão ser descritos semanticamente e integrar a iminente terceira fase da Web.

Por se basear em protocolos já estabelecidos e não estipular qualquer alteração nos mesmos, a ontologia proposta mostra-se plenamente realizável. Assim, aplicações imediatas seriam possíveis nos inúmeros serviços RESTful já disponibilizados na Web 2.0. Como exemplo, a seção 5.2 apresentou um exercício para a descrição completa do serviço de busca de notícias disponibilizado pelo Yahoo.

Obviamente, outra conclusão importante derivada da ausência de modificações nos protocolos já existentes é que a coexistência de mapeamentos também é possível.

6.1 Limitações

Duas limitações existem no modelo proposto. Primeiramente, apenas os serviços descritos sintaticamente por um documento WADL poderão ser decorados semanticamente através da nova ontologia.

Apesar da crescente adoção da linguagem WADL, um número certamente elevado de serviços não é descrito por qualquer notação formal. O ponto positivo é que empresas influentes como Yahoo e Sun já optaram pela descrição em WADL. Além disso, em outubro de 2009, o padrão WADL passou a ser uma submissão oficial para integrar a base de padrões do W3C. Ambos os pontos mencionados reforçam a validade da proposta apresentada.

Já a segunda limitação refere-se ao mapeamento entre os parâmetros OWL-S e os parâmetros WADL. A ontologia proposta suporta, atualmente, apenas o mapeamento direto, ou seja, com cardinalidade 1:1.

6.2 Trabalhos Futuros

Trabalhos futuros poderiam investigar mapeamentos mais complexos, de tal forma a permitir que um único parâmetro semântico em OWL-S possa ser derivado em um conjunto de parâmetros sintáticos em WADL. Tais transformações poderiam ser realizadas, por exemplo, através de comandos em XSLT (*Extensible Stylesheet Language Transformation*).

Tal capacidade seria utilizada para que parâmetros muito técnicos pudessem ser ignorados na descrição semântica do serviço, que tipicamente reside em um nível de abstração mais elevado.

Adicionalmente, o futuro estabelecimento de padrões para o desenvolvimento de agentes de software semânticos possibilitará uma validação mais precisa do potencial de descobrimento oferecido pela ontologia `RESTfulGrounding`. Os agentes deverão ser capazes de acessar automaticamente serviços descritos sintaticamente em WADL.

Por fim, seria interessante que pesquisas futuras aplicassem a nova ontologia nos mais variados domínios, propondo ajustes se necessário. Casos de uso devem focar, preferencialmente, em serviços populares da Web 2.0, o que traria mais sinergia entre o contexto social e o semântico.

REFERÊNCIAS

ADAMS, H.; GISOLFI, D.; SNELL, J.; VARADAN, R. Best Practices for Web Services: Part 1, Back to the Basics. IBM Developer Works Library. 2002. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-best1/>. Acesso em: 14 fev. 2009.

ANKOLEKAR, A.; KROTZSCH, M.; TRAN, T.; VRANDECIC, D. The Two Cultures: Mashing up Web 2.0 and the Semantic Web. Journal of Web Semantics, v. 6, n. 1, p. 70-75, 2008.

ANTONIOU, G.; VAN HARMELEN, F. A Semantic Web Primer. Cambridge: The MIT Press, 2008. 264 p.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. Scientific American, v. 284, n. 5, p. 28-37, 2001.

BOJARS, U.; BRESLIN, J.; FINN, A.; DECKER, S. Using the Semantic Web for Linking and Reusing Data Across Web 2.0 Communities. Journal of Web Semantics, v. 6, n. 1, p. 21-28, 2008.

BOLEY, H.; HALLMARK, G.; KIFER, M.; PASCHKE, A.; POLLERES, A.; REYNOLDS, D. RIF Core Dialect, W3C Candidate Recommendation. 2009. Disponível em: <http://www.w3.org/TR/rif-core/>. Acesso em: 27 out. 2009.

BREITMAN, K.; CASANOVA, M. A.; TRUSZKOWSKI, W. Semantic Web: Concepts, Technologies and Applications. Londres: Springer, 2007. 327 p.

BROBERG, J. Glossary for the OASIS Web Service Interactive Applications (WSIA / WSRP). OASIS Consortium. 2002. Disponível em: <http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm>. Acesso em: 15 fev. 2009.

FIELDING, T. Architectural Styles and the Design of Network-based Software Architectures. 2000. 180 p. Tese (Doutorado). University of California, Irvine, 2000.

GERBER, A. Towards a Comprehensive Functional Layered Architecture for the Semantic Web. In: IASTED International Multi-Conference: Software Engineering, 25., 2007, Innsbruck.

GRUBER, T. Toward Principles for the Design of Ontologies used for Knowledge Sharing. International Journal of Human-Computer Studies, v. 43, p. 907-928, 1995.

GRUBER, T. Collective knowledge systems: Where the Social Web meets the Semantic Web. Journal of Web Semantics, v. 6, n. 1, p. 4-13, 2008.

GULLI, A.; SIGNORINI, A. The Indexable Web is more than 11.5 Billion Pages. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE, 14., 2005, Chiba.

HAAS, H.; BROWN, A. Web Services Glossary, W3C Working Group Note. 2004. Disponível em: <http://www.w3.org/TR/ws-gloss/>. Acesso em: 14 fev. 2009.

HADLEY, M. Web Application Description Language, W3C Member Submission. 2009. Disponível em: <http://www.w3.org/Submission/wadl/>. Acesso em: 10 set. 2009.

HENDLER, J. Agents and the Semantic Web. IEEE Intelligent Systems Journal, v. 16, n. 2, p. 30-37, 2001.

HENDLER, J.; GOLBECK, J. Metcalfe's Law, Web 2.0, and the Semantic Web. Journal of Web Semantics, v. 6, n. 1, p. 14-20, 2008.

HEYLIGHEN, F. Collective Intelligence and its Implementation on the Web: Algorithms to Develop a Collective Mental Map. Computational & Mathematical Organization Theory, v. 5, n. 3, p. 253-280, 1999.

IETF: Internet Engineering Task Force. Hypertext Transfer Protocol – HTTP/1.1. 1999. Disponível em: <http://tools.ietf.org/html/rfc2616>. Acesso em: 14 fev. 2009.

IETF: Internet Engineering Task Force. Internationalized Resource Identifiers (IRIs). 2005. Disponível em: <http://tools.ietf.org/html/rfc3987>. Acesso em: 30 nov. 2009.

IETF: Internet Engineering Task Force. Uniform Resource Identifier (URI): Generic Syntax. 2005. Disponível em: <http://tools.ietf.org/html/rfc3986>. Acesso em: 14 fev. 2009.

ISOTANI, S.; MIZOGUCHI, R.; BITTENCOURT, I.; COSTA, E. Estado da Arte em Web Semântica e Web 2.0: Potencialidades e Tendências da Nova Geração de Ambientes de Ensino na Internet. Revista Brasileira de Informática na Educação, v. 17, n. 1, p. 30-42, 2009.

KOLBITSCH, J.; MAURER, H. The Transformation of the Web: How Emerging Communities Shape the Information We Consume. Journal of Universal Computer Science, v. 12, n. 2, p. 187-213, 2006.

LIU, H.; PALLICKARA, S.; FOX, G. Performance of Web Services Security. In: ANNUAL MARDI GRAS CONFERENCE, 13., 2005, Baton Rouge.

MAKINO, S.; TATSUBORI, M.; TAMURA, K.; NAKAMURA, Y. Improving WS-Security Performance with a Template-Based Approach. In: IEEE INTERNATIONAL CONFERENCE ON WEB SERVICES, 3., 2005, Orlando.

MANOLA, F.; MILLER, E. RDF Primer, W3C Recommendation. 2004. Disponível em: <http://www.w3.org/TR/rdf-primer/>. Acesso em: 30 nov. 2009.

MARTIN, D.; BURSTEIN, M.; HOBBS, J.; LASSILA, O.; MC DERMOTT, D.; MC ILRAITH, S.; NARAYANAN, S.; PAOLUCCI, M.; PARSIA, B.; Payne, T.; SIRIN, E.; SRINIVASAN, N.; SYCARA, K. OWL-S: Semantic Markup for Web Services, W3C Member Submission. 2004. Disponível em: <http://www.w3.org/Submission/OWL-S/>. Acesso em: 26 mai. 2009.

MC GUINNESS, D.; VAN HARMELEN, F. OWL Web Ontology Language Overview, W3C Recommendation. 2004. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: 30 abr. 2009.

OASIS: Organization for the Advancement of Structured Information Standards. Web Services Security: SOAP Message Security 1.1 (WS-Security). 2006. Disponível em: <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>. Acesso em: 15 fev. 2009.

O'REILLY, T. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications & Strategies, n. 65, p. 17-37, 2007.

PRETSCHNER, A.; GAUCH, S. Ontology Based Personalized Search. In: IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 11., 1999, Chicago.

RICHARDSON, L.; RUBY, S. RESTful Web Services. Sebastopol: O'Reilly, 2007. 419 p.

W3C: World Wide Web Consortium. Latest Layercake Diagram. 2007. Disponível em: <http://www.w3.org/2007/03/layerCake-small.png>. Acesso em: 30 nov. 2009.

APÊNDICE A – WEB 2.0

Enquanto a Web Semântica concede mais poder aos computadores, permitindo que estes possam compreender a informação disponível na Internet, a Web 2.0 privilegia o poder aos usuários (ISOTANI, 2009). É exatamente por esse motivo que a segunda fase na evolução da Web é também conhecida como Web Social, um ambiente no qual a geração do conteúdo já não é mais exclusividade dos serviços noticiários ou das grandes corporações. Assim, a Web tornou-se mais democrática, permitindo aos próprios usuários não apenas publicar, mas também avaliar e classificar o conteúdo publicado.

Tal descentralização gerou certa resistência por parte daqueles que detinham o conhecimento sobre as técnicas de publicação de conteúdo eletrônico. A crítica era geralmente associada à qualidade da informação publicada, ou seja, questionava-se a credibilidade da fonte da informação. Entretanto, como observado por Isotani et al. (2009), tanto a criação quanto a avaliação do conteúdo por pessoas de uma mesma comunidade criam mecanismos sociais que impedem, ou pelo menos dificultam, que informações de baixa qualidade tenham grande impacto dentro da comunidade.

Esse efeito que emana da interação entre membros de uma comunidade é também conhecido como Inteligência Coletiva (*Collective Intelligence*). Segundo Heylighen (1999), a idéia básica é que um grupo de indivíduos pode ser inteligente de uma forma que nenhum de seus membros é. O autor relata que comportamentos complexos e inteligentes emergem da sinergia criada por interações simples entre os indivíduos que seguem regras simples.

Dessa forma, a própria comunidade seria capaz de classificar, avaliar e filtrar o conteúdo. Anteriormente, tais atividades estavam sujeitas aos critérios e valores de uma entidade central.

Diferentes termos surgiram em pesquisas acadêmicas para descrever tal efeito. Gruber (2008), por exemplo, utiliza termo Inteligência Acumulada (*Collected Intelligence*) para evidenciar a geração de conhecimento novo a partir da soma das partes que formam o grupo.

Aplicações da Web 2.0 incluem redes sociais, plataformas para compartilhamento de vídeos, músicas, fotos e links, bases de conhecimento não-estruturado, serviços para classificação de notícias, entre outras. Alguns exemplos são apresentados no Quadro 5.

Tipo	Aplicação	Endereço
Rede Social	Facebook	http://www.facebook.com/
	Orkut	http://www.orkut.com/
Compartilhamento de Vídeos	You Tube	http://www.youtube.com/
	Vimeo	http://www.vimeo.com/
Compartilhamento de Músicas	Blip.fm	http://www.blip.fm/
	Last.fm	http://www.last.fm/
Compartilhamento de Fotos	Flickr	http://www.flickr.com/
	Picasa Web	http://www.picasaweb.com/
Compartilhamento de Links	Delicious	http://www.delicious.com/
	Stumble Upon	http://www.stumbleupon.com/
Base de Conhecimento	Wikipedia	http://www.wikipedia.org/
	Ubuntu Team Wiki	http://wiki.ubuntu.com/
Classificação de Notícias	Reddit	http://www.reddit.com/
	Digg	http://www.digg.com/
Atualização de Status	Twitter	http://www.twitter.com/
	Jaiku	http://www.jaiku.com/

Quadro 5 – Exemplos de aplicações Web 2.0

A massiva adoção de tais ferramentas pode ser facilmente percebida através da análise dos dados estatísticos[28] da maior rede social do planeta, neste momento (Dezembro de 2009): Facebook.

- 300 milhões de usuários ativos
- 045 milhões de atualizações de status por dia
- 002 bilhões de fotos compartilhadas por mês
- 014 milhões de vídeos compartilhados por mês
- 003 milhões de eventos criados por mês
- 130 amigos em média para cada usuário
- 070 idiomas disponíveis

Outro fator importante para a notável penetração das aplicações Web 2.0 nas mais diversas sociedades ao redor do mundo é a disponibilização de serviços Web. Como visto ao longo desse trabalho, os serviços permitem a extração e reutilização do conteúdo publicado pelos usuários. Assim, ao aumentar o número de canais de distribuição, uma aplicação Web 2.0 alcança uma audiência muito maior, composta pelos usuários das demais aplicações que reusam o conteúdo extraído através do serviço Web.

Os serviços ainda permitem que o conteúdo seja consumido em outros dispositivos, como os telefones celulares, por exemplo. Conseqüentemente, os usuários têm acesso ao conteúdo a qualquer momento, em qualquer lugar, a partir de qualquer dispositivo.

Um exemplo notável do potencial dos serviços na Web 2.0 é uma plataforma de atualização de status chamada Twitter. Nela, usuários podem utilizar 140 caracteres para dizer o que estão fazendo naquele exato momento. É uma idéia simples, mas que se tornou uma febre; são dezenas de milhões de usuários registrados.

Ao disponibilizar serviços Web, o Twitter permitiu que diversos desenvolvedores ao redor do mundo criassem aplicações com propósitos completamente diferentes da idéia original. São aplicações multimídia, financeiras, estatísticas, de automação, de viagens, de negócios, de produtividade, de busca, etc.

[28] http://www.facebook.com/press/info.php?statistics

Um catálogo chamado One Forty[29], criado recentemente, já lista mais de 2100 aplicações, todas desenvolvidas sobre os serviços oferecidos pelo Twitter.

Os desenvolvedores das aplicações Web 2.0 tipicamente projetam seus serviços de acordo com o estilo arquitetural REST, incluindo os engenheiros responsáveis pelo Facebook e pelo Twitter. Diferentemente do padrão SOAP, a abordagem RESTful não impõe qualquer protocolo além do próprio HTTP, o que reduz as barreiras de adoção.

Ao delegarem a tarefa de criação de conteúdo aos próprios usuários, aplicações da Web Social só fazem sentido quando sua utilização é massiva. Portanto, quanto menores as barreiras de adoção, maior a probabilidade de uma aplicação alcançar o sucesso na atual fase da Web.

Infelizmente, a grande maioria do conteúdo publicado na Web 2.0 não é descrito semanticamente por qualquer padrão formal. Portanto, a Web Semântica terá que assumir o desafio de trazer estrutura aos dados já publicados, assim como àqueles que ainda serão criados. Espera-se que o ambiente semântico aumente a precisão das buscas e traga mais automação para diversas atividades realizadas diariamente por todos os usuários da Web.

Entretanto, para que os agentes da Web Semântica possam utilizar os inúmeros serviços Web 2.0 como fontes de dados em processos semânticos, essas interfaces deverão ser descritas formalmente. Descrições semânticas e sintáticas permitirão, respectivamente, o descobrimento e a execução automáticos dos serviços.

A ontologia proposta por este trabalho permite a descrição semântica de serviços RESTful. Como visto, trata-se de uma especialização do padrão OWL-S que utiliza a linguagem WADL para a descrição sintática.

É importante notar que a descrição semântica de serviços RPC/SOAP continuará sendo possível. O padrão OWL-S permite a coexistência de mapeamentos ao definir uma camada abstrata em sua arquitetura.

[29] http://oneforty.com/