SEMANTIC WEB SERVICES: A RESTFUL APPROACH

Otávio Freitas Ferreira Filho, Maria Alice Grigas Varella Ferreira
University of São Paulo, Polytechnic School
São Paulo, Brazil

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. Ontologies. Semantic Web Services. Web Services. Web 2.0. Internet.

1. INTRODUCTION

Notably, the Web has been the ultimate information source for more than a decade, continuously transforming and speeding up content consumption and production processes. Throughout its evolution, the Web has experienced three phases, defined mainly by different mindsets, and not by technological improvements.

The first phase focused on content consumption. Information was typically made available by professional information providers such as companies advertising their products and services, organizations and news services (Kolbitsch; Maurer, 2006). Textual, static pages were offered by this limited group, which retained the necessary knowledge to publish electronic content.

After the wide adoption of the Web around the world, content publishing techniques became popular, and easy-to-use tools enabled common users to publish their own content. This new and also current phase – built upon interaction, collaboration and communication among users – was named Web 2.0. Applications developed within this context are those continually updated and that get better the more people use them, consuming and remixing data from multiple sources. These applications provide their own data and services in a way that allows remixing by others, creating network effects through an architecture of participation (O'Reilly, 2007).

As described, Web 2.0 applications typically offer a set of services responsible for making user-generated content available to third-party applications, which, in turn, are able to access and reuse the fetched data in order to reach completely different goals. There are countless definitions of Web services, but they are usually overlapped when it comes to protocols and languages, as presented by the following list:

- A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL[1]). Other systems interact with the Web service in a manner prescribed by its description using SOAP[2]-messages, typically conveyed using HTTP[3] with an XML[4] serialization in conjunction with other Web-related standards (Haas; Brown, 2004).

- Web services are software components that are developed using specific technologies from three primary technology categories: an XML-based description format (for example, WSDL), an application messaging protocol (for example, SOAP), and a collection or transport protocol (for example, HTTP) (Adams et al., 2002).
- A Web service is a software component described via WSDL and capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP (Broberg, 2002).

The intensive content production driven by Web 2.0 applications in conjunction with the interoperable communication powered by Web services may be considered the main reasons for this new wave on the Web exponential growth. A fairly recent paper written by Gulli and Signorini (2005), measured the Web around 11.5 billion pages indexed by search engines at that time. Considering this scenario, it is clear that stricter and more accurate search criteria must be implemented. Keyword-matching filters are inaccurate, and their limitations have been noted for several years. According to Pretschner and Gauch (1999), search terms are ambiguous, their meaning depends on the context and, more importantly, on the meaning a user assigns to them.

Semantic annotations are meant to overcome these limitations. They form the driving force of the imminent third phase of the Web, called Semantic Web and introduced by Berners-Lee, Handler, and Lassila (2001). The authors describe this phase not as a separate Web but as an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

Many other authors have already covered the need for connections between Web 2.0 and the Semantic Web. Ankolekar et al. (2008) suggest that Web 2.0 applications will keep focusing on community and usability, while drawing on Semantic Web infrastructure to facilitate mashup-like information sharing. Similarly, Bojars et al. (2008) also propose the use of Semantic Web elements to enable Web 2.0 data reuse. Gruber (2008) introduces collective knowledge systems that unlock the collective intelligence of the Social Web with knowledge representation and reasoning. Finally, Hendler and Golbeck (2008) aim at extracting value from the combination between linked semantics and linked social networks.

Unlike Web 2.0, which has been already experienced by millions of users around the globe, the Semantic Web still faces barriers, including technological ones. However, there is clearly a vibrant academic effort bringing both areas together, which will potentially speed up the adoption of the new paradigm.

Obviously, Web services are also affected by the semantic approach. As described previously in this section, Web services are commonly defined according to a specific group of protocols and languages. The definitions presented are quite similar because they actually describe services that belong to the same category, known as RPC (Remote Procedure Call), in which SOAP is the main protocol. This category has been deeply explored by researches and industry professionals, so all the initial attempts at making feasible the development of semantic Web services considered only RPC services, especially those bound to SOAP. Nevertheless, the majority of Web 2.0 applications provide services classified as REST (Representational State Transfer) instead, in which HTTP is the main protocol.

In regards to this gap in researches about semantic Web services, this paper proposes a specific combination of existing protocols and languages in order to enable the development of semantic Web services according to the architectural style REST. This combination will allow the ever-growing amount of Web 2.0 applications to migrate their interfaces to the semantic world.

The initial contribution is related to a mindset shift, since the paper introduces the RESTful approach in a field predominantly taken by SOAP services. It also supports WADL (Web Application Description Language) as standard language to syntactically describe RESTful semantic Web services. Additionally, it supports OWL-S (Ontology Web Language for Services) as the base ontology to semantically describe the services. Finally, the most expressive contribution is the definition of a new ontology called `RESTfulGrounding`, which is an OWL-S extension that delegates services syntactic descriptions to WADL

[1] Web Service Description Language
[2] Simple Object Access Protocol
[3] Hypertext Transfer Protocol
[4] Extensible Markup Language

documents. As opposed to this new scheme, the default OWL-S `Grounding` ontology relies on WSDL documents.

Briefly, this paper was organized as follows: RESTful Web Services are detailed in section 2, Semantic Web Services in section 3, and the new ontology in section 4. Conclusions and limitations are presented in section 5.

2. RESTFUL WEB SERVICES

Web services enable interoperable and platform-agnostic communication amongst applications. They represent one of the most promising approaches to distributed software reuse, increasing the value of existing software assets. Through Web services, any software application on the Web has the potential to reach any other application. If applications exchange messages in ways compliant to Web service standards, they will be able to communicate independently of operating systems, programming languages, processors, and internal protocols (Breitman; Casanova; Truszkowski, 2007).

REST is an architectural style for distributed hypermedia systems. It provides a set of architectural constraints that, when applied as a whole, emphasize scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems (Fielding, 2000). The criteria defined by REST are not bound to any specific protocol.

This section focuses on depicting the concepts introduced by Richardson and Ruby (2007), who coined the term RESTful Web Services to classify the services compliant to all REST criteria. Additionally, these authors put REST and HTTP together, bringing a technological aspect to those abstract criteria.

Shortly after its definition, HTTP was revised and defined by W3C (World Wide Web Consortium) as the default protocol for message transferring over the Web. It means that all systems designed in accordance with the RESTful paradigm have a potential audience composed of all devices connected to the Web, virtually an infinite number. RESTful Web services become entirely portable by taking advantage of the Web as their ready-to-use infrastructure for distribution and access, with no extra dependency in terms of hardware and software.

The RESTful approach is basically composed of five concepts (resource, representation, uniform identifier, unified interface, and execution scope), and three principles (addressability, statelessness, and connectedness). As any Web service, RESTful services receive a request that defines the procedure to be executed, and then return a response containing the execution result. However, two main differences can be observed on RESTful requests. Firstly, the way the action to be executed is defined. Secondly, the way the execution scope is defined.

The next subsections detail both points (respectively through the concepts Unified Interface and Execution Scope) and also all the other concepts and principles that form RESTful services.

2.1 Concepts

RESTful Web services must fulfill all the following concepts.

2.1.1 Resource

A resource is a relevant abstraction within the domain covered by a given service. The service designer may choose any domain objects, from concrete to abstract ones. It is important to bear in mind that a single resource may be composed of a collection of objects. Moreover, some services deal with more than one resource type, but this does not affect its RESTful quality in any way.

2.1.2 Representation

Web services manipulate resource representations, and not the resources themselves, because the latter are just abstractions. Theoretically, a representation is any useful information about the state of a resource (Richardson; Ruby, 2007). Technically, a representation is a resource serialization in a given format, such as XML, XHTML (Extensible Hypertext Markup Language), JSON (JavaScript Object Notation), RDF

(Resource Description Framework), and so forth. XML-based languages are particularly important within the semantic context because they allow for extensible data annotation.

2.1.3 Uniform Identifier

Each resource is linked to at least one URI (Uniform Resource Identifier) that acts simultaneously as its name and locator. On the one hand, an object with no URI cannot be considered a RESTful resource. On the other hand, an object may be identified by a set of different URIs. Identifiers ought to be descriptive, as well as respectful of predefined constraints that include structure, hierarchy, and notation pattern.

2.1.4 Unified Interface

According to the RESTful paradigm, the action to be executed is defined by the HTTP method. This protocol offers five principal methods, namely GET, HEAD, POST, PUT and DELETE. All methods are applied to resources. More specifically, an HTTP method is executed against a given URI. Despite its simplicity, this characteristic is extremely powerful, since it defines a unified interface to all possible services. If a consumer application knows the resources offered by a given service, it will automatically know how to retrieve, create, update, and delete these resources.

2.1.5 Execution Scope

Regarding the execution scope definition, the RESTful paradigm advocates using the HTTP request URI. Therefore, the URI does not contain the service path only, but also any other parameters to uniquely identify the resource to be affected.

2.2 Principles

Additionally, RESTful Web services must be compliant to the principles described as follows.

2.2.1 Addressability

A Web service is classified as addressable if it exposes the interesting aspects of its data set through resources, each with its own unique URI (Richardson; Ruby, 2007). Considering that a single resource may support various representations (as one per serialization format), it might be helpful to actually assign a different URI to each representation. As opposed to this principle, non-addressable services typically present only one URI, which names and addresses the service as a whole, and not the objects to be exposed to consumers. The latter case is performed by RPC services, and keeps the consumers from referring to any particular resource directly, which is undesirable.

2.2.2 Statelessness

As described by Richardson and Ruby (2007), the RESTful approach defines two types of state. Firstly, there is resource state, which is information about the resource. Secondly, there is application state, which is information about the path the client has taken through the application. A RESTful service is stateless if no application state is stored by the server. It means that the client is responsible for sending all information required to a successful method execution. Requests are supposed to happen in absolute isolation.

2.2.3 Connectedness

Resources should point to each other, guiding the client to other application states. A connection is established whenever a resource representation provides an attribute value that is actually the URI of another resource. Fielding (2000) defines this sort of hypermedia scheme as the engine of application state. High connectedness levels empower resource discovering. Indeed, as presented by Richardson and Ruby (2007), in a service that is not connected, the client must use predefined rules to construct every URI it wants to visit.

2.3 Syntactic Description

Unfortunately, there is not a truly well-established standard for syntactically describing RESTful services yet. As depicted previously, academic efforts have covered only SOAP services to a great degree. Nevertheless, Hadley (2009), a senior researcher at Sun Microsystems, proposed an open, XML-based language called WADL (Web Application Description Language) that has gained remarkable momentum in recent time. Regarding possible use cases, the author highlights the support for developing resource modeling tools, automated code generation, and for configuring both client and server through a portable format. Moreover, a WADL parser may power a general-purpose wrapper for any RESTful service.

WADL is composed of several XML elements, and some of them are directly linked to RESTful concepts, such as `Resource`, `Method`, `Request`, `Response`, and `Representation`. Finally, it is important to note that WADL syntactically describes RESTful services, just as WSDL describes SOAP services.

3. SEMANTIC WEB SERVICES

Web service technologies bring a dynamic aspect to overall Web usage. However, the current understanding about Web services – RPC or RESTful – fails to capture enough semantic data. Therefore, semantic services deal with such limitation by augmenting the service description with a semantic layer in order to achieve automated discovery, composition, monitoring, and execution, which are all highly desirable processes (Antoniou; van Harmelen, 2008). The semantic layer in question is usually defined by OWL-S (Martin et al., 2004), a language that provides a primary ontology (`Service`[5]) and three subordinated ones (`Profile`[6], `Process`[7] and `Grounding`[8]).

3.1 The Service Ontology

This ontology acts as the connection point among `Profile`, `Process` and `Grounding`. More specifically, it defines four main classes: `Service`, `ServiceProfile`, `ServiceModel`, and `ServiceGrounding`. The latter three are actually abstract entities that are respectively extended by the classes `Profile`, `Process` and `Grounding`. Each class is named after the ontology it belongs to.

The concrete realizations of `ServiceProfile` should describe the functionality offered by the service in question, whereas `ServiceModel` realizations describe the execution process. The `ServiceGrounding` class presents the base all child classes should extend from in order to describe the access to the service, mainly when it comes to protocols in use. Finally, `Service` provides a simple way of gathering the other entities, building the full service description.

3.2 The Profile Ontology

This ontology specifies the functionality offered by the service, the semantic type of the inputs and outputs, the details of the service provider, and several service parameters, such as quality rating or geographic radius (Antoniou; van Harmelen, 2008). This information is consumed essentially by semantic software agents seeking services. Textual description and contact details form a subset of information that targets human consumption instead.

`Profile`, an extension of `ServiceProfile`, is the main class within this ontology. It describes the functionality through properties such as `hasParameter`, `hasPrecondition`, `hasInput`, and `hasOutput`. It is important to note that a service profile may publish only part of the functionality it provides, which may keep software agents from discovering the service.

3.3 The Process Ontology

This ontology is responsible for defining how a given service is executed, that is, the steps that form the execution flow. A semantic software agent may use this information in order to decide whether the service

[5] http://www.daml.org/services/owl-s/1.2/Service.owl
[6] http://www.daml.org/services/owl-s/1.2/Profile.owl
[7] http://www.daml.org/services/owl-s/1.2/Process.owl
[8] http://www.daml.org/services/owl-s/1.2/Grounding.owl

really satisfies the requirements set by the calling object. When analyzed together, the classes `Profile` and `Process` present all the necessary information for the decision making process run by agents.

One of the most important entities within this ontology is the class `AtomicProcess`. An atomic process is an action to be executed in a single step, at least from the consumer's point of view, so no nested processes take place during the execution.

3.4 The Grounding Ontology

This ontology details the access to a given service, mainly regarding protocols, message formats and transport, resource serialization and addressability. While `Profile` and `Process` describe the service in an abstract way, the `Grounding` ontology aims at a concrete service description, including technical aspects.

Semantic software agents make use of this ontology to prepare the request to be sent to the service. At this moment, all compatibility checking between requester and provider has been performed already, and the agent is ready to communicate directly to the service. The two major classes within this ontology are `Grounding` and `AtomicProcessGrounding`. These entities handle the technical conduction of atomic processes.

4. RESTFUL SEMANTIC WEB SERVICES

This new classification covers semantic Web services designed according to the RESTful paradigm, a quite unique combination that has not drawn much attention from academic initiatives yet, and therefore represents the most significant contribution of this paper. The main goal is to promote the inclusion of RESTful Web services powering the Web 2.0 into the semantic world, potentially speeding up the adoption of the Semantic Web as a whole.

In order to start describing this classification formally, the following points present the principles that will guide the development of RESTful Semantic Web Services. Obviously, the principles that define RESTful services, as well as those defining semantic services have been taken into account. The combination between these groups was planned carefully so that no protocol previously defined will be affected or modified at all. The proposed principles include:

- Follow the addressability principle by assigning a unique URI to each resource being managed;
- Follow the statelessness principle by accepting only request messages carrying all necessary data for a successful method execution;
- Follow the connectedness principle by returning resource representations that link to each other;
- Handle HTTP messaging, and also make use of the unified interface provided by this protocol in order to let consumers manipulate resource states. The minimum interface supports the methods GET, HEAD, POST, PUT, and DELETE;
- Expose its syntactic description to support automated access, a process run by software agents;
- Expose its semantic description to support service searching, selection, and composition. These processes are also run by software agents on the Semantic Web.

The very first decision regarding the protocols being adopted refers to syntactic description. This research supports the WADL language as a suitable option to solve this problem; its XML-based syntax facilitates the integration with other Semantic Web patterns. According to Richardson and Ruby (2007) – the authors who coined the term RESTful Web Services – WADL is really "the most simple and elegant solution" to solving the syntactic description problem.

The second decision is related to semantic description. In this case, OWL-S will be adopted, an upper ontology for services. OWL-S is an OWL extension, which, in turn, is the de-facto language for authoring ontologies, standardized by W3C (World Wide Web Consortium) in 2004.

Figure 1 presents the structure to be used when fully describing RESTful Semantic Web Services.

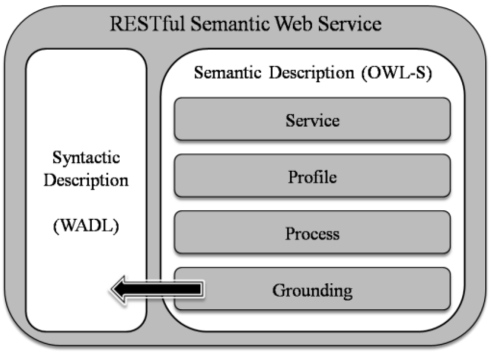

Figure 1. Full description structure of RESTful Semantic Web Services

The proposed scheme does not extend three OWL-S ontologies, namely `Service`, `Profile` and `Process`. This premise reinforces the notion of abstract and concrete elements within the service description. As described earlier, the `Grounding` ontology is the only one that deals with mapping semantic services to their syntactic, concrete description, based on specific technologies. Considering the protocols adopted by this research, such mapping will be labeled OWL-S/WADL Grounding. The ontology `RESTfulGrounding` proposed is responsible for this mapping and will be presented in section 4.1.

4.1 The RESTfulGrounding Ontology

OWL-S/WADL Grounding extends an abstract layer composed of two OWL-S classes, namely `Grounding` and `AtomicProcessGrounding`. Figure 2 is a UML (Unified Modeling Language) class diagram that presents this extension. The packages that represent the ontologies `Service`, `Profile` and `Process` have been suppressed so that the diagram could clearly show the extension designed from the abstract layer.

As seen in the diagram, the classes `WadlGrounding` and `WadlAtomicProcessGrounding` do not belong to the OWL-S `Grounding` ontology, but would potentially do. The presented architecture proposes grouping the new classes in a dedicated ontology, named `RESTfulGrounding`. This approach prevents any modification on the OWL-S specification, which will probably facilitate the adoption of this new ontology.

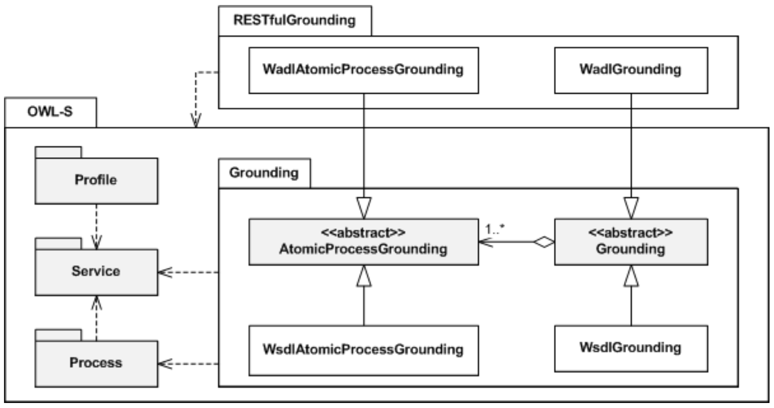

Figure 2. `RESTfulGrounding` as an OWL-S extension for OWL-S/WADL Grounding

In order to formalize the proposal, the `RESTfulGrounding` ontology must be presented in OWL format, which is the major notation standard for Web ontologies. Firstly, Figure 3 presents `WadlGrounding`, an entity that extends the abstract class `Grounding`, as defined by line 02. The code fragment also defines a restriction on lines 04-09, in which the `hasAtomicProcessGrounding` property is constrained to values from `WadlAtomicProcessGrounding`. Since there’s no cardinality involved, this restriction defines a collection, and not only a single object.

The prefix `&grounding` on lines 02 and 06 refers to classes defined by the OWL-S `Grounding` ontology. The header of the document that contains this code fragment must define the referenced ontology's fully-qualified URI in order to let semantic software agents discover all the required class definitions.

```
01 <owl:Class rdf:ID="WadlGrounding">
02   <rdfs:subClassOf rdf:resource="&grounding;#Grounding" />
03   <rdfs:subClassOf>
04     <owl:Restriction>
05       <owl:onProperty
06         rdf:resource="&grounding;#hasAtomicProcessGrounding" />
07       <owl:allValuesFrom
08         rdf:resource="#WadlAtomicProcessGrounding"/>
09     </owl:Restriction>
10   </rdfs:subClassOf>
11 </owl:Class>
```

Figure 3. OWL definition of class `WadlGrounding`

Figure 4 presents the definition of the second most important entity within the `RESTfulGrounding` ontology, the `WadlAtomicProcessGrounding` class. This entity is responsible for mapping any given OWL-S atomic process to a WADL pair composed of the resource affected by the request and the HTTP method to be used. The atomic process is referred to by a property called `owlProcess`, which belongs to the class `AtomicProcessGrounding`. This is the class extended by `WadlAtomicProcessGrounding`, as specified on line 02, Figure 4. Any atomic process is an instance of `AtomicProcess`, a key class within the OWL-S `Process` ontology.

Now, regarding the WADL pair, the resource is an instance of `Resource`, whereas the HTTP method is an instance of `Method`. Obviously, both classes are defined by WADL. According to the cardinality defined within the restriction on lines 04-08 (Figure 4), only one pair resource/method is allowed per atomic process.

```
01 <owl:Class rdf:ID="WadlAtomicProcessGrounding">
02   <rdfs:subClassOf rdf:resource="&grounding;#AtomicProcessGrounding"/>
03   <rdfs:subClassOf>
04     <owl:Restriction>
05       <owl:onProperty rdf:resource="#wadlResourceMethod" />
06       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
07       </owl:cardinality>
08     </owl:Restriction>
09   </rdfs:subClassOf>
10 </owl:Class>
11
12 <owl:ObjectProperty rdf:ID="wadlResourceMethod">
13   <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
14   <rdfs:range rdf:resource="#WadlResourceMethodRef"/>
15 </owl:ObjectProperty>
```

Figure 4. OWL definition of class `WadlAtomicProcessGrounding`

The property `wadlResourceMethod` presented on lines 12-15 connects the atomic process to the resource/method pair. This property accepts instances of `WadlResourceMethodRef`, as seen on line 14 (Figure 4). If it was possible to refer to a resource/method pair through a single URI, the property `wadlResourceMethod` would be typed to `anyURI`, a general-purpose data-type defined by XSD (XML Schema Definition). However, it is not supported by WADL as of version 20090202, which is the most recent version so far. Therefore, this limitation demands the creation of a new class to group the pair elements.

This is exactly the role of class `WadlResourceMethodRef`; the definition of which is presented by Figure 5, lines 01 and 02. This class has two properties: `resource` (lines 04-07) and `method` (lines 09-12). Both are typed to `anyURI`, and respectively refer to a resource and a HTTP method in a given WADL document. These properties are equally constrained by the same cardinality, in which only one instance is allowed (lines 14-22 and 24-32).

```
<owl:Class rdf:ID="WadlResourceMethodRef">
</owl:Class>

<owl:DatatypeProperty rdf:ID="resource">
  <rdfs:domain rdf:resource="#WadlResourceMethodRef"/>
  <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="method">
  <rdfs:domain rdf:resource="#WadlResourceMethodRef"/>
  <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#WadlResourceMethodRef">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#resource"/>
      <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#WadlResourceMethodRef">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#method"/>
      <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 5. OWL definition of class `WadlResourceMethodRef`

The class `WadlAtomicProcessGrounding` has other four properties that deserve some attention in this long exercise of defining the `RESTfulGrounding` ontology. The property `wadlResourceMethod`, the only one presented until this moment, is defined through an OWL structure called `ObjectProperty`. Well, other two properties are defined through the same structure, namely `wadlRequestParam` (Figure 6 and 7) and `wadlResponseParam` (Figures 8 and 9).

```
<owl:ObjectProperty rdf:ID="wadlRequestParam">
  <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
  <rdfs:range rdf:resource="#WadlRequestParamMap"/>
</owl:ObjectProperty>
```

Figure 6. OWL definition of property `wadlRequestParam`

A `WadlAtomicProcessGrounding` instance should have an instance of the `wadlRequestParam` property to each one of the HTTP request parameters. This property provides a mapping made by the class `WadlRequestParamMap`, which in turn relates an OWL-S input parameter to a WADL request parameter.

This mapping entity, defined according to Figure 7 (lines 01-04), extends exactly two classes: `WadlMessageParamMap` (lines 06-22) and `InputMessageMap` (defined by the OWL-S `Grounding` ontology). This multiple inheritance is presented on lines 02 and 03. Unlike the majority of programming languages, the ontology notation language OWL allows for multiple inheritance. As a matter of fact, this ability is held by RDFS (Resource Description Framework Schema), which is actually the base of OWL.

The `WadlRequestParamMap` class also defines a property called `wadlMessageParam` (lines 24-27), which points to the request parameter (WADL `Param` class) set within the mapping. Now, the OWL-S input parameter is set by `MessageMap` (the base class of `WadlMessageParamMap`) through the `owlsParameter` property, as defined within the OWL-S `Grounding` ontology itself.

According to the restriction on lines 09-13, only one instance of `wadlMessageParam` is allowed per mapping. The same cardinality is set to the property `owlsParameter`, as seen on lines 16-20.

```
01 <owl:Class rdf:ID="WadlRequestParamMap">
02   <rdfs:subClassOf rdf:resource="#WadlMessageParamMap"/>
03   <rdfs:subClassOf rdf:resource="&grounding;#InputMessageMap"/>
04 </owl:Class>
05
06 <owl:Class rdf:ID="WadlMessageParamMap">
07   <rdfs:subClassOf rdf:resource="&grounding;#MessageMap"/>
08   <rdfs:subClassOf>
09     <owl:Restriction>
10       <owl:onProperty rdf:resource="#wadlMessageParam"/>
11       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
12       </owl:cardinality>
13     </owl:Restriction>
14   </rdfs:subClassOf>
15   <rdfs:subClassOf>
16     <owl:Restriction>
17       <owl:onProperty rdf:resource="&grounding;#owlsParameter"/>
18       <owl:cardinality rdf:datatype="&xsd;#nonNegativeInteger">1
19       </owl:cardinality>
20     </owl:Restriction>
21   </rdfs:subClassOf>
22 </owl:Class>
23
24 <owl:DatatypeProperty rdf:ID="wadlMessageParam">
25   <rdfs:domain rdf:resource="#WadlMessageParamMap"/>
26   <rdfs:range rdf:resource="&xsd;#anyURI"/>
27 </owl:DatatypeProperty>
```

Figure 7. OWL definition of class `WadlRequestParamMap`

The third property held by `WadlAtomicProcessGrounding` – named `wadlResponseParam` – aims at mapping the HTTP response message parameters. The OWL definition of this property can be found in Figure 8. Similarly to `wadlRequestParam`, the property `wadlResponseParam` is assisted by a mapping class, in this case called `WadlResponseParamMap` (Figure 9).

The two mapping entities (`WadlRequestParamMap` and `WadlResponseParamMap`) extend the abstract class `WadlMessageParamMap`. Consequently, both rely on the property `wadlMessageParam` to point to a WADL message parameter. Specifically in HTTP response messages, the parameter set can be either a representation (WADL `Representation` class) or a HTTP header (WADL `Param` class).

```
<owl:ObjectProperty rdf:ID="wadlResponseParam">
  <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
  <rdfs:range rdf:resource="#WadlResponseParamMap"/>
</owl:ObjectProperty>
```

Figure 8. OWL definition of property `wadlResponseParam`

```
<owl:Class rdf:ID="WadlResponseParamMap">
  <rdfs:subClassOf rdf:resource="#WadlMessageParamMap"/>
  <rdfs:subClassOf rdf:resource="&grounding;#OutputMessageMap"/>
</owl:Class>
```

Figure 9. OWL definition of class `WadlResponseParamMap`

Finally, the last two properties of the class `WadlAtomicProcessGrounding`, out of five, are: `wadlVersion` (Figure 10) and `wadlDocument` (Figure 11). The former only defines which WADL version has been considered by the service, whereas the latter simply defines the WADL document in use. Unlike the others, these two properties are defined through an OWL structure called `DatatypeProperty`, since their values are actually URI addresses, instead of being class instances.

```
<owl:DatatypeProperty rdf:ID="wadlVersion">
  <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
  <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>
```

Figure 10. OWL definition of property `wadlVersion`

```
<owl:DatatypeProperty rdf:ID="wadlDocument">
  <rdfs:domain rdf:resource="#WadlAtomicProcessGrounding"/>
  <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>
```

Figure 11. OWL definition of property `wadlDocument`

All classes and properties presented here formally define the current state of the proposed ontology: `RESTfulGrounding`[9]. An example[10] applied this ontology to Yahoo's News Search Web Service, a popular service among Web 2.0 applications, and designed according to REST principles.

5. CONCLUSION

OWL-S/WADL Grounding, achieved by `RESTfulGrounding`, was clearly inspired by OWL-S/WSDL Grounding, proposed by Martin et al. (2004) when specifying the OWL-S pattern itself. Despite supporting architectural styles completely different from each other, these two approaches are somehow similar, which endorses the extendibility aspect of the abstract layer defined by the OWL-S `Grounding` ontology.

Indeed, new extensions are motivated by the authors, who clearly state that their "intent is not to prescribe the only possible grounding approach to be used with all services, but rather to provide a general, canonical and broadly applicable approach that will be useful for the great majority of cases". Now, the `RESTfulGrounding` ontology will allow the RESTful Web services to join the imminent third phase of the Web. The vibrant Web 2.0 community may now support the Semantic Web by migrating services according to the proposed ontology.

Since this new ontology is based on well-defined protocols, and does not introduce any modification to them, `RESTfulGrounding` can be considered entirely feasible. It could readily be applied to any RESTful service powering a Web 2.0 application.

Two limitations are observed, though. Firstly, only services syntactically described by WADL documents can be semantically described by the new ontology. Despite the ever-growing WADL adoption throughout the Web, several services are certainly not described by any formal protocol whatsoever. Nevertheless, influential companies such as Yahoo are already making use of WADL, which reinforces the proposed model's feasibility. Secondly, `RESTfulGrounding` currently supports only 1:1 direct mapping between OWL-S and WADL parameters.

Further researches may explore more complex mapping schemes. Additionally, they could focus on applying the new ontology to various case studies, proposing adjustments, if needed.

ACKNOWLEDGMENT

The authors thank all the researches who provided the foundation which the proposed ontology has been built upon, mainly Roy Thomas Fielding, Leonard Richardson, Sam Ruby, Marc Hadley, and Tim Berners-Lee. Special acknowledgments to the DAML team for the excellent work on defining the OWL-S standard as well as related ontologies, and to Stephen Thomas, Jacques van Niekerk, and Italo Vega.

[9] http://www.fullsemanticweb.com/ontology/RESTfulGrounding/v1.0/RESTfulGrounding.owl
[10] http://www.fullsemanticweb.com/blog/ontologies/restfulgrounding/

REFERENCES

Adams, H. et al. (2002). *Best practices for Web services: Part 1, back to the basics*. Retrieved February 14, 2009, from https://www.ibm.com/developerworks/webservices/library/ws-best1/.

Ankolekar, A. et al., 2008. The two cultures: Mashing up Web 2.0 and the Semantic Web. *Journal of Web Semantics*, Vol. 6, No. 1, pp. 70-75.

Antoniou, G.; van Harmelen, F., 2008. *A Semantic Web Primer*. The MIH Press, Cambridge, USA.

Berners-Lee, T.; Hendler, J.; Lassila, O., 2001. The Semantic Web. *Scientific American*, Vol. 284, No. 5, pp. 28-37.

Bojars, U. et al., 2008. Using the Semantic Web for linking and reusing data across Web 2.0 communities. *Journal of Web Semantics*, Vol. 6, No. 1, pp. 21-28.

Breitman, K.; Casanova, M. A.; Truszkowski, W., 2007. *Semantic Web: Concepts, Technologies and Applications*. Springer, London, UK.

Broberg, J. (2002). *Glossary for the OASIS Web Service Interactive Applications (WSIA/WSRP)*. Retrieved February 15, 2009, from http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Unpublished doctoral thesis, University of California, Irvine.

Gruber, T., 2008. Collective knowledge systems: Where the Social Web meets the Semantic Web. *Journal of Web Semantics*, Vol. 6, No. 1, pp. 4-13.

Gulli, A.; Signorini, A., 2005. The Indexable Web is More than 11.5 Billion Pages. *Proceedings of the Fourteenth International World Wide Web Conference*. Chiba, Japan, pp. 902-903.

Haas, H.; Brown, A. (2004). *Web Services Glossary, W3C Working Group Note*. Retrieved February 14, 2009, from http://www.w3.org/TR/ws-gloss/.

Hadley, M. (2009). *Web Application Description Language (WADL)*. Retrieved February 25, 2009, from Web Application Description Language: https://wadl.dev.java.net/wadl20090202.pdf.

Hendler, J.; Golbeck, J., 2008. Metcalfe's Law, Web 2.0, and the Semantic Web. *Journal of Web Semantics*, Vol. 6, No. 1, pp. 14-20.

Kolbitsch, J.; Maurer, H., 2006. The Transformation of the Web: How Emerging Communities Shape the Information We Consume. *Journal of Universal Computer Science*, Vol. 12, No. 2, pp. 187-213.

Martin, D. et al., (2004). *OWL-S: Semantic Markup for Web Services*. Retrieved February 15, 2009, from http://www.w3.org/Submission/OWL-S/.

O'Reilly, T., 2007. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, No. 65, pp. 17-37.

Pretschner, A.; Gauch, S., 1999. Ontology Based Personalized Search. *Proceedings of the Eleventh IEEE International Conference on Tools with Artificial Intelligence*. Chicago, USA, pp. 391-398.

Richardson, L.; Ruby, S., 2007. *RESTful Web Services*. O'Reilly Media, Sebastopol, USA.