Integrating Data and Services through Functional Semantic Service Descriptions

Ruben Verborgh[1], Thomas Steiner[2], Davy Van Deursen[1], Sam Coppens[1], Erik Mannens[1], Rik Van de Walle[1], and Joaquim Gabarró Vallés[2]

[1] Ghent University – IBBT, ELIS – Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
{ruben.verborgh, davy.vandeursen, sam.coppens, erik.mannens, rik.vandewalle}@ugent.be
[2] Universitat Politécnica de Catalunya – Department LSI
08034 Barcelona, Spain
{tsteiner, gabarro}@lsi.upc.edu

1 Semantics and Hyperlinks: Key to an Integrated Web of Services

Without any exaggeration, the Linked Data movement has significantly changed the Semantic Web world. Meanwhile, intelligent services—the other pillar of the initial Semantic Web vision—have not undergone a similar revolution. Although several important steps were taken and significant milestones were reached, we are far from our envisioned destination.

What makes the Web so difficult for machines? So far, we have only seen successful clients for *specific* purposes, mostly tailored to the API of a *certain* site or service. This contrasts with human behavior: we surf the Web for several *different* purposes on a *variety* of websites. The discrepancy originates in two related aspects: semantics and hyperlinks. The Resource Description Framework (RDF) and the Linked Data effort help to overcome the problem of *data* semantics by providing machine-interpretable data with linked concepts. On the other hand, *services* tend not to provide semantics or links, neither to internal pages nor to external sites. As a consequence, in order to consume a Web service, automated agents currently must have an implementation of the API and know how to construct URIs. In contrast, humans do not need a manual to browse a website, because they can interpret text and follow links.

Summarizing, if we want automated agents to consume the Web, services should provide:

– **functional semantics** — what can an agent do with the service?
– **meaningful hyperlinks** — in what directions can an agent proceed?

The above two aspects form a necessary condition to integrate services into the semantic and interlinked world of Linked Data. As we will argue in Section 2, today's service descriptions are unable to fulfill this premise. Therefore, we discuss the lightweight semantic service description method RESTdesc [4] in Section 3, enabling a linked Web of Services through the linked Web of Data. We continue by indicating how RESTdesc could be an ideal counterpart for the integration of data and services.

2 Shortcomings of Existing Technologies

The main problem with existing service description technologies is their focus on technical aspects (input and output parameter types, exceptions. . .) and invocation modalities. They do not or insufficiently capture the defining characteristics of a service: *what* does it do, and *how* does it relate/link to other data and services? For example, it is insufficient to describe a book author lookup service as *"a service with a book as input and a person as output"*. Does this person represent a reader of the book, the editor, or maybe even the subject? And how does this book relate to a library service? A human has to determine this in advance and hardwire the service into applications.

The Web Service Description Language (WSDL) is one of the most widely known service description paradigms. However, WSDL is mostly concerned with technical descriptions and lacks the mechanisms to capture the functional semantics of services. A more recent proposal, the Web Application Description Language (WADL), also neglects this functional aspect, disabling runtime service discovery. A first step towards semantics was offered by Semantic Annotations for WSDL (SAWSDL). However, SAWSDL only gives a semantic expression to the input and output parameters of services, rather than connecting them in a functional way.

A well-known semantic format is OWL for Services (OWL-S). While OWL-S still focuses on input and output parameters, it allows to describe functional relations using different expression languages (SWRL, DRS, KIF, others are possible). However, these expressions are unintegrated and form a separate layer. Unfortunately, this layer is ignored unless interpreted explicitly by a supported toolset, much like unsupporting browsers ignore JavaScript in HTML pages.

The Web Service Modeling Ontology (WSMO) is an alternative to OWL-S. Although they share the same goals, substantial differences between both approaches exist. The most relevant difference for our purpose is the logical expressivity. While OWL-S provides the aforementioned different expression languages, a drawback from the interoperability viewpoint, WSMO employs a single family of layered logic languages. However, when expressed in RDF syntax, WSMO expressions become similarly unintegrated and hence not self-descriptive. The same holds for WSMO-Lite, which extends SAWSDL with conditions and effects.

3 RESTdesc—Bringing the Benefits of Linked Data to Services

3.1 The Semantic Web Embraces RESTful Services

In this section, we build the bridge from RESTful Web services to semantically well-defined and -described Web Services. In short, RESTful services are about proper use of HTTP by representing information in a resource-oriented way. Since RDF also centers on resources, an interesting parallel between both emerges: the uniquely addressable resources in RESTful services can be seen as RDF resources and vice versa. Insights into their similarities (and current differences) have already resonated within the REST community [1]. All of the above implies that the current lack of links between services could be solved with tools from the RDF and Linked Data world.

Like Linked Data, we believe services should also be self-describing. We also argued that current description methods do not capture a service's essential characteristics. Therefore, RESTdesc [2,4] offers a functionality-centered and self-describing alternative. Rather than using a meta-vocabulary to describe parameters and modalities, it directly explains the HTTP request required to achieve a certain goal, which can be expressed using a vocabulary of choice.

As an example, consider a music fingerprinting Web service, which analyses an uploaded audio fragment and returns song title, artist, and album. Using RESTdesc, this service might be described as in Listing 1. We immediately notice the absence of traditional parameter descriptions, and the focus on resources, in this case a music fragment. This does, however, not mean that the variables are not well-defined. On the contrary, their definition is implied by the example `music` ontology. For instance, with the help of a reasoner, clients can determine that the response represents a song and the `artist` variable a musical artist, because of the domain and range of the `performedBy` property.

Moreover, the description precisely explains the functionality of the service as a *relationship* between the fragment and the song: the former is a `fragmentOf` the latter. Reasoners can use the Notation3 rule in Listing 1 to infer that, if a client needs the title of a fragment uploaded at `/uploads/radio_capture`, it should perform a `GET` request of `/uploads/radio_capture/song`.

3.2 Hyperlinks Make Services Cross Boundaries

Complementary to the service description in Listing 1, the fingerprinting service can supply a `fragmentOf` link that indicates the functional relationship between the fragment and the song. The service itself can

```
@prefix music: <http://example.org/music#>.
@prefix http: <http://www.w3.org/2006/http#>.
@prefix tmpl: <http://purl.org/restdesc/http-template#>.

{ ?fragment a music:Fragment. }
=>
{
  _:request http:methodName "GET";
            tmpl:requestURI (?fragment "/song");
            http:resp [ tmpl:represents ?song ].
  ?fragment music:fragmentOf ?song.
  ?song music:title _:title;
        music:performedBy _:artist;
        music:album _:album.
}.
```

Listing 1. RESTdesc description of a music fingerprinting service. The **`fragmentOf`** property plays the role of a hyperlink that semantically and functionally connects a fragment to its associated song.

provide this either by an HTML link, RDFa, or an HTTP `Link` header. For example, the fragment `/uploads/radio_capture` may have a link to the recognized song in an HTTP header, defining the relationship with the `fragmentOf` property:

```
Link: </uploads/radio_capture/song; title="Recognized song";↩
      rel="related http://example.org/music#fragmentOf"
```

The easy transition from Linked Data to Web service and back is possible thanks to the resource-oriented nature of the REST paradigm. This duality makes all of the previously separate benefits available to both worlds, as we will examine in the next section. Clients could take this further and look up more knowledge, effectively relying on Linked Data to support their decisions, thanks to the well-defined semantics of the relations [3].

Interestingly, these links can cross server boundaries. Agents are no longer limited to a specific server and implementation, but can follow a `fragmentOf` link to another server and receive song metadata, without prior knowledge about this server's structure. For instance, after the music fingerprinting service, an entirely different independent service could be used to retrieve, the album cover for the recognized song. It is up to service providers to supply links and to describe what a relation means, but after that, clients can autonomously consume data by following links.

3.3 Powerful Service Matching and Composition

An important feature of functional service descriptions is that they allow for straightforward matching and composition. RESTdesc descriptions benefit from the operational semantics entailed by the underlying Notation3 language. This means that both matching and composition can be solved using one of the available general-purpose Notation3 reasoners. Three interesting cases are possible:

– **functional discovery** — Example: a client has an audio fragment and wants to know what the server can do with it. By reasoning (the precondition will trigger the rule in Listing 1), the server indicates that it can find the song this fragment belongs to.
– **functional matching** — Example: a client has an audio fragment and wants to know what song it belongs to. Since Listing 1 explicitly connects fragments and songs with a `fragmentOf` relationship, it is able to verify the match. Other services with the same input and output, but a different functional connection, will not match.
– **functional composition** — RESTdesc service descriptions can be hooked in complex ways by similar reasoning methods. Example: consider i) an online radio service that generates audio fragments and ii) an online music store. A relevant query could be "buy the song currently on the radio". This is not trivial, since a semantic relationship is required: the current song and the store are connected by the audio fingerprinting service. However, Notation3 reasoners are able to solve this query by propagating the semantics throughout the composition process.

The necessity of an integration between data and services is apparent in the above three cases: while the different services may use separate ontologies, relationships between the services and ontologies can still allow a successful process. Again, this comes down to adequate semantics and relevant hyperlinks.

4 Conclusion

This position paper puts the spotlight on well-known but rarely remedied shortcomings of the current Web of Services: they do neither provide adequate semantics nor relevant links. We have indicated how RESTdesc is able to help solving this problem in an integrative way, and how it is able to fulfill service discovery, matching, and composition.

We therefore believe that RESTdesc could be a W3C standardization starting point for bridging services built using different paradigms, as we identified points where standardization would help integration of services and data. In the end, our aim is to evolve towards a Web of Agents built upon a symbiosis of services and data, as envisioned in the initial Semantic Web outlines.

References

1. Page, K.R., Roure, D.C.D., Martinez, K.: REST and Linked Data: a match made for domain driven development? In: Proceedings of the 2nd International Workshop on RESTful Design. ACM, New York, NY, USA (Mar 2011)
2. RESTdesc website, `http://restdesc.org/`
3. Verborgh, R., Steiner, T., Van Deursen, D., Van de Walle, R., Gabarró Vallés, J.: Efficient Runtime Service Discovery and Consumption with Hyperlinked RESTdesc. In: Proceedings of the 7th International Conference on Next Generation Web Services Practices (Oct 2011)
4. Verborgh, R., Steiner, T., Van Deursen, D., Van de Walle, R., Gabarró Vallés, J.: Description and Interaction of RESTful Services for Automatic Discovery and Execution. In: Proceedings of the FTRA International Workshop on Advanced Future Multimedia Services (submitted)