Consuming Dynamic Linked Data

Barry Norton[1] and Reto Krummenacher[2]

[1] AIFB, Karlsruhe Institute of Technology, Germany
[2] Semantic Technology Institute, University of Innsbruck, Austria
firstname.lastname@{kit.edu | sti-innsbruck.at}

Abstract. The establishment of Linked Data principles has ushered in a remarkable new era of widespread semantics-based interoperation. In particular, the use of RDF, SPARQL and HTTP allow a high degree of generic tool support and lightweight composition of data sources. This rosy picture, however, mainly applies to static data. When it comes to dynamics - both on-the-fly computation and the causation of side effects - the current norm is to drop the use of RDF in favour of JSON and XML. By introducing updates, the core SPARQL standard starts to address dynamics, but only in a very limited manner. Similarly while the use of HTTP is preserved, REST principles are also often abandoned. Linked Open Services establish principles for semantics-based interoperation and interlinkage in a dynamic environment, building on both Linked Data and REST principles and the use of HTTP, RDF and SPARQL.

1 Introduction

Linked Data is identified with the application of a set of principles, and related best practice, regarding the use of RDF, SPARQL and HTTP in the publication of data. Linked Open Data is the result of the application of these principles in combination with the push to opening up public sector and other data.

On the most basic level Linked Open Services (LOS) concern the open exposure of *functionalities* on the Web using semantic technologies. This is by no means a novel aim. Indeed, the seminal reference on the vision of the Semantic Web [3] included a prediction of agents that would coordinate Web-based functionalities. For some years the Semantic Web Services (SWS) movement has put itself forward as the ostensible means to bring together and achieve these aims. At the same time SWS approaches, such as OWL-S [10], WSMO [6], and SA-WSDL [1], have all built primarily upon the 'WS-*' stack, grounded in SOAP and WSDL, and sought to prove their practical worth to enterprise services in closed environments.

The interfaces presented to users by SWS environments — such as the OWL-S (formerly DAML-S) Virtual Machine [12], WSMX [7] and IRS-III [4] — have been of two forms: either services are composed ('orchestration'-style) within an environment where (primarily) SOAP-based messages are 'lifted' to a semantics-based representation and available *within the system* at this level, or a SOAP-based interface is made available for *consuming* services, where messages are lowered again to XML.

[1] http://www.w3.org/TR/sawsdl/, with which we include preceding work, especially WSDL-S, and subsequent work such as WSMO-Lite.

There are two choices, then, for building semantic applications making use of SWS: either the SWS environments become the entire platform, i.e., a commitment that all development has to take place within the SWS environment; or the services are consumed externally using XML as per any other SOAP service (however much more helpful they may be in being discoverable based on their semantic description). Neither of these options has proven very desirable or succesful in the real world, unlike Linked Data.

At the same time, the services actually achieving widespread use on the Web (both the 'Web 2.0' version, with its user-inclusive nature, social-networking emphasis and, most importantly openness to 'mash'ability via open APIs, and the Semantic extensions of this, which has been called 'Web 3.0', where resources are described in RDF and data is 'linked', in this form, and queryable via SPARQL endpoints) have largely turned away from SOAP, towards REST, and increasingly abandon even the underlying XML. While the resource-oriented view of the Web has found *static* description in RDF profitable (witness the billions of statements of Linked Data now available on the Web), *services*, which provide *computation* over these resources, is both incompatible with SWS, eschewing SOAP-based communication in favour of RESTful APIs, and hand-crafted, are neglecting semantics-based representations themselves, not just in terms of SWS-like service descriptions, but in terms of the messages they communicate (even though these concern Linked Data resources).

We introduce as a running example, and will describe in more detail later, the services offered over the GeoNames[2] data, an important component of the Linked Data Cloud. As shown in Figure 1[3], while many services are now provided (36 are listed in total — we have omitted 'get', since this is merely resource retrieval, not computation, and 'rssToGeo', since this offers neither XML, JSON or RDF, being merely a format converter), all but one are available for communication in JSON (annotated 'J'), and only one is available for communication in RDF ('R'). XML (annotated 'X') is somewhere in between, losing favour to JSON.

astergdem	J X –
children	J X –
cities	J X –
countryCode	J X –
countryInfo	J X –
countrySubdivision	J X –
earthquakes	J – –
extendedFindNearby	– X –
findNearby	J X –
findNearbyPlaceName	J X –
findNearbyPostalCodes	J X –
findNearbyStreets	J X –
findNearbyStreetsOSM	J X –
findNearByWeather	J X –
findNearbyWikipedia	J X –
findNearestAddress	J X –
findNearestIntersection	J X –
findNearestIntersectionOSM	J X –
gtopo30	J X –
hierarchy	J X –
neighbourhood	J X –
neighbours	J X –
ocean	J X –
postalCodeCountryInfo	J X –
postalCodeLookup	J X –
postalCodeSearch	J X –
search	J X R
siblings	J X –
srtm3	J X –
timezone	J X –
weather	J – –
weatherIcao	J – –
wikipediaBoundingBox	J X –
wikipediaSearch	J X –

Fig. 1. The current services offered over the geonames Linked Data set

[2] `http://www.geonames.org`

[3] Reproduced from `http://www.geonames.org/export/ws-overview.html`

The paper is structured as follows. In Section 2 we look further into the issues introduced here and at existing efforts to restore the role of semantic representations in computation (rather than retrieval) driven interactions. In Section 3 we outline our proposed principles to unite and reinforce these efforts. In Section 4 we present our own illustrative efforts in applying these principles, and then draw conclusions in Section 6.

2 Motivation for Linked Open Services

An initial trigger for the work presented in this paper is the fact that many of the datasets that the Linking Open Data, and related, initiative(s) have managed to openly expose on the Web seem to be *semantically* available only passively via (retrieval-based) SPARQL endpoints. Otherwise the semantics are hidden away behind custom (REST) services that obscure the RDF data. At the same time the tremendous success of Linked (Open) Data is (rightly) proclaimed as 'boot-strapping the Semantic Web' and the variety of services that emerged on top of this inter-linked data cloud is steadily increasing[4]. The appearance of Semantic Web pipes and other mash-up tools and techniques emphasized our interest in finding more advanced but still simple solutions that would allow to smoothly integrate service interactions with the available Linked Data sources.

While XML is still arguably the predominant format for data transfer over the Internet, these days, and especially in the user-oriented applications in which semantic technologies are finding success, it is often simplified as JSON or YAML.[5] As mentioned previously, this is oddly also the case for some services that are associated with RDF-described resources, such as for example the GeoNames services, which are per default assumed (via their intimate association with the GeoNames data) to be themselves part of the Linked Open Data cloud. In order to invoke such services in a semantic application, it is currently necessary to transform from RDF to the expected data format of the service implementation, and to map the output of the service back into RDF. We, instead, seek service principles that naturally integrate service invocation with the largely growing Linked Data cloud.

While it might seem natural that the years of work on Semantic Web Services would provide such a possibility, we would point out that the combination of semantic technology and Web services in form of Semantic Web services has until now been largely oriented towards extension of the WS-* stack with ontology- and rule-based descriptions. Even the most contemporary 'lightweight' SWS approaches associated with SAWSDL: WSMO-Lite [17], microWSMO [9] and SA-REST[6] still focus on linking services, operations and messages of the syntactical service descriptions to some concepts or values in an external ontology in order to add semantics. To this end, the semantic descriptions still require lifting and lowering to map to the execution world of the service, and what is more critical in our opinion, is the fact that the service annotations do not offer any information about the *implicit* knowledge, not explicitly represented in

[4] Although critical of the 'missing' semantics in service interaction, we would wish to clearly state that without the work done by the Linked Data initiative we would not have had a playground nor a starting point for our work.

[5] See `http://www.json.org/` and `http://www.yaml.de/en/`

[6] `www.w3.org/Submission/SA-REST/`

the syntactic output message, associated with to service execution. For instance, in describing a weather service, with with one might communicate at the syntactic level the input 'Vienna' and obtain the output '20', *within* a SWS we might *classify* the input as a city, automatically derive ('lower' to) the syntactic message for service interaction and subsequently derive a semantic representation of the output. This, however, is merely a classification, i.e., represents "20 degrees celsius is a temperature".

In this sense, mainstream Semantic Web services research is still much more about semantics for Web services, than about services for the Semantic Web. LOS, on the other hand, aim to capture the *knowledge* derived from a service invocation; not just, in fact, "20 degrees celsius is the current temperature *in Vienna*" (the most important part of which is missed by lifting in SWS since it is not represented in the output message, but implicit in its link to the input message), but also "according to [this service provider], based on an interaction at [this time]".

The LOS aim, furthermore, is to capitalise on the Linked Open Data cloud and to make service description more easily and directly appealing to the growing Linked Data community, and in this way, to make services accessible to the LOD specialists. The same time, services become Linked Data citizens by having their input and output descriptions show how each service is a consumer and producer of RDF data, and how a service invocation contributes to the knowledge of its consumer. This will, moreover, have the effect that services can thereby be more easily be integrated into mash-ups. We believe that this is much closer to the original vision of agents on the Semantic Web. Indeed only by such an accessible integration of service functionalities can the Web of Data realise its promises. This requires more than purely data-oriented mash-ups and must involve on-the-fly computation and the possibility to trigger real-world side effects in back-end systems.

For all of these reasons we seek a model for services, which we term Linked Open Services, in which i) services become RDF 'prosumers', ii) all communication is conducted at the semantics level, and iii) the descriptions encode how a service contributes to the knowledge base of its execution environment.[7] Linked Open Services will provide an approach to the envisaged dynamics of the Web of Data, which cannot be reduced to SPARQL only.

We recall that the WWW was originally developed with the goal to be a collaborative space in which people could collectively design, discuss, share and manage things [2]. The objective of the WWW was and still is to establish an infrastructure that facilitates the writing of powerful collaborative applications – already back in 1998 Tim Berners-Lee argued that machines should be able to participate and help [1]. Providing service the means to collaboratively solve problems in a shared information spaces is then argued to provide a contribution in this direction [5, 11]. To this end, we intend with our work to be a possible step towards Read-Write Linked Data, beyond the simple updates proposed in newer SPARQL work, towards being an enabler of *Read-Compute* Linked Data.

[7] A localized process execution environment based on a shared memory infrastructure was presented in [8].

3 LOS! Principles

The Linked Data movement was set in motion with a number of principles[8], which we reproduce here:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs. so that they can discover more things.

The aim of LOS is not just to apply these principles to services, an approach already proposed in [13], but to propose a list of further service-specific principles to be followed in openly exposing services over Linked Data.

1. **Describe services as *LOD prosumers* with input and output descriptions as *SPARQL graph patterns*.**
2. ***Communicate RDF* by *RESTful content negotiation*.**
3. **Communicate and describe the *knowledge contribution* resulting from service interaction, including *implicit knowledge* relating input, output and service provider.**

Associated with the last principle is an optional fourth:

4 **When wrapping non-LOS services, extend the (lifted, if non-RDF) message to make explicit the implicit knowledge, and to use Linked Data vocabularies, using *SPARQL CONSTRUCT* queries.**

The first principle is perhaps the most significant in terms of challenging the state of practice in both existing services for Linked Data, and in the established wisdom of Semantic Web Services[9]. We believe that classification of input and output messages, an approach due to OWL-S and more-or-less followed in the OWL-S community since, is insufficient for Principle 3 to be met. We choose SPARQL precisely because of its familiarity and amenability to the Linked Data community. At the same time it neatly captures information currently lost in non-semantic lifting and lowering languages, for instance the XSLT and XSPARQL proposed in the SAWSDL specification, about what precisely is communicated.

Our use of SPARQL, however, goes beyond static description as can already be seen in the fourth principle, which will be exemplified in Section 4. In fact we go further and, as sketched in our original LOS paper [8], from which we have derived three principles for LOS composition:

1. Decide control flow conditions based on SPARQL ASK queries
2. Base iteration on SPARQL SELECT queries
3. Define dataflow/mediation based on SPARQL CONSTRUCT queries

[8] `http://www.w3.org/DesignIssues/LinkedData.html`

[9] With the exception of the work of Sbodio and Moulin[15], discussed below.

4 Realisation and Examples

In this section we provide an example of our proposed approach, which are part of an effort to exposing LOS online at large scale, akin to the aims of the Linking Open Data initiative. An initial evaluation of the approach for the current paper is hence provided, but more importantly the discussion of these services is on-going and open at the LOS community site[10].

As stated in the motivations section, XML, JSON or YAML are the predominant syntaxes for the exchange of information on the Web, even, and that is very unfortunate, for a significant number of services on top of the Linked Open Data Cloud; such as for example most of the GeoNames services.[11]

In order to expose such non-RDF Web services as LOS, it is necessary to transform from RDF to the expected data format of the service implementation, and to map the output of the service back into RDF. In the concrete case of considered GeoNames Weather services,[12] which we use as real-life example, the invocation is triggered by an HTTP GET request, and the result data is serialized as either a JSON Array or an XML document.

According to the LOS principles, a Linked Open Service should describe the knowledge contribution resulting from the service interaction, including implicit knowledge relating input, output and service provider. In the present case of a Web service that does not already communicate RDF, the LOS implementation must not only specify what graph patterns the service consumes and produces, respectively, but must take care of how the input RDF is 'lowered' to the expected data format of the service, respectively how the output of the service is 'lifted' back to RDF. In contrast to traditional Semantic Web service frameworks, we do not assume that a general-purpose grounding component is taking care of the transformation, but that it is part of the LOS implementation to ensure the desired mapping between the non-RDF service interactions and the knowledge consumed and produced. This explicit inclusion into the service implementation has furthermore the advantage that the link between the syntactical and the semantic world is provided by the owner of the service; i.e. the one that knows best what the knowledge value of the service should be.

Although we claim in this paper that truly interactive services that provide a basis for collaboration on the Web should per se communicate linked data, we realise that there is still a lot of value hidden behind existing services that we want to dynamically bind to the Linked Data cloud. In order to show how existing services could be wrapped to exhibit LOS characteristics, and to depict how such services can contribute to the knowledge body of an LOD consumer, we present one of our LOS GeoNames Weather Services in the continuation of this section.

The weather services are available at `http://www.linkedopenservices.org/services/geonames/weather/`, where the LOS-specific input and output descriptions are given as exemplified in Table 1. Note that the prefix–namespace bindings are omitted, but can be found on the mentioned service description page.

[10] `http://www.linkedopenservices.org`
[11] `http://www.geonames.org/export/ws-overview.html`
[12] `http://www.geonames.org/export/JSON-webservices.html#weather`

Table 1. LOS GeoNames Weather Consumption and Production Patterns

Input (Consumption Pattern):

```
[a wgs84:Point; wgs84:lat ?lat; wgs84:long ?long]
```

Output (Production Pattern):

```
[ metar:weatherObservation [
  weather:hasStationID ?icao ;
  metar:stationName ?station ;
  geonames:inCountry ?country ;
  wgs84:lat ?lat ; wgs84:long ?lng ; wgs84:alt ?alt ;
  metar:datetime ?dateTime ;
  metar:observation ?observation ;
  weather:hasVisibilityEvent ?clouds ;
  weather:hasWindEvent [weather:windDirection ?windDirection],
                       [weather:windSpeedKnots ?windSpeed] ;
  weather:hasTemperatureEvent
    [a weather:CurrentTemperature ;
     weather:celsiusTemperature ?temperature],
    [a weather:CurrentDewPoint ;
     weather:celsiusTemperature ?dewPoint],
    [weather:humidityPercent ?humidity] ;
  weather:hasWeatherEvent ?condition ;
  weather:hasPressureEvent [metar:hectoPascal ?pressure ] ] ]
```

The given LOS GeoNames Weather service (getNearByWeather) augments a callers knowledge base with linked data according to the production pattern in Table 1. The service returns the latest observations from the nearest weather station, according to reverse geocoding. While the original offered by GeoNames accepts an untyped lattitude and longitude, our corresponding LOS service accepts a geographic points according to the WGS84 ontology. After a successful HTTP POST request with the required RDF in the body, the LOS implementation lowers the input RDF to the expected data format of the weather service, transforms the resulting JSON into RDF by two distinct lifting steps, and returns the the set of weather stations and observations as one RDF graph in the body of the HTTP response.

The three steps of lowering, lifting and graph construction are depicted in more detail in the rest of the section.

LOS Lowering and Service Invocation

Our LOS services are implemented as Web resources, and we thus fully rely on HTTP as the access protocol to the resources and on communication of RDF by RESTful content negotiation. An HTTP GET call to a service resource allows for accessing the service description that includes all the necessary information about how to communicate with the service; e.g. consumption and production patterns, ontologies and data sets used

to construct the RDF graphs. As stated above, service invocations are triggered by an HTTP POST for which the syntax of the exchanged RDF is negotiated; e.g., text/n3 or application/rdf+xml.

The input knowledge is extracted from the HTTP message and in the present case the coordinates of the two corner points of the bounding box are applied to construct the query string for the GeoNames Weather service.[13]

Generic JSON2RDF Transformation

In order to transform the JSON message which is returned from GeoNames into a knowledge contributor, we make use of a generic json2rdf Java library.[14] The transformation process is exemplified in Table 2. All the JSON attribute keys are automatically lifted to predicates of a temporary json2rdf-internal schema; i.e., we take a simple RDFication step. The values are either kept as literals, either as strings or numeric values depending on their JSON type – the size and type of the numbers determines the numeric type, or automatically transformed to URIs by means of templates that make the json2rdf library configurable. Templates relate attributes to URL patterns that are constructed by means of the JSON value; as long as this is doable in a generic why. In Table 2 the key–value pair "ICAO": "CYOW" is mapped onto the triple [json2rdf:ICAO airports:CYOW] by the template ICAO ⇒ airports:*icao*. Note that for ease of readability, we are always using prefixes instead of full URIs.

Table 2. Automatic JSON2RDF Lifting

GeoNames output JSON:

```
{"weatherObservation":
  { "clouds": "few clouds", "windDirection": 20,
    "ICAO": "CYOW", "lng": 41.7,
    "temperature": "35", ... }}
```

Generic RDF after lifting:

```
[ json2rdf:weatherObservation [
    json2rdf:clouds "few clouds" ;
    json2rdf:windDirection "20"^^xsd:short ;
    json2rdf:ICAO airports:CYOW ;
    json2rdf:lng "41.7"^^xsd:double ;
    json2rdf:temperature "35" ;
    ... ]]
```

[13] An example URL as given by GeoNames would be `http://ws.geonames.org/findNearByWeatherJSON?lat=43&lng=-2`.

[14] The library is currently undergoing a clean up and is expected to be publicly available latest by the conference.

While templates offer a first possibility to link automatically lifted JSON messages to the Linked Data cloud, the temporary json2rdf predicates do not allow to gain any knowledge from the RDF. For this last transformation step from RDF to RDF we apply standard SPARQL CONSTRUCT queries, as explained in the next paragraph.

Specific SPARQL Lifting

In the minimal case the last step consists of applying one SPARQL CONSTRUCT query to the automatically created RDF model. The query has a head equal to the output pattern of the LOS, and is hence an explicit specification of how to transform the internal json2rdf data into an RDF graph that provides the promised knowledge contribution of the service. For the particular case of our LOS GeoNames Weather service we introduced an additional intermediate step that lifts the string values of, for example, json2rdf:clouds to a URL of the METAR ontology (metar:FewClouds). [15]

Although this last step requires some SPARQL engineering, it still provides a quite straightforward approach for wrapping non-RDF Web services as Linked Open Services. Similarly to the use of json2rdf in our example, we could have used any XML to RDF transformation framework to construct a generic RDF representation of an XML document. In a last step, SPARQL CONSTRUCT would be used again to construct the knowledge contribution of the service. For the given example an extract of the returned RDF is shown in Table 3. The graph illustrates how the internal representation is further lifted to be integrated into the Linked Data cloud.

Note finally that, in general, the returned data may include further triples than the graph patterns explicitly declare, and furthermore that the pattersn may include explicit flexibility with OPTIONAL clauses (as well as UNIONs and FILTERs).

Table 3. RDF output of the LOS GeoNames Weather service

```
_:node16804 metar:weatherObservation _:node16805 .
_:node16805 weather:hasStationID airports:CYOW ;
            metar:stationName "Ottawa Int'L. Ont." ;
            wgs84:long "-75.6666666666667"^^xsd:double ;
            weather:hasVisibilityEvent metar:FewClouds ;
            weather:hasWindEvent _:node16806 , _:node16811 .
_:node16806 weather:windDirection "20"^^xsd:short .
_:node16811 weather:windSpeedKnots "14"^^xsd:short .
_:node16820 weather:CurrentTemperature ;
            weather:celsiusTemperature "35"^^xsd:short ...
```

[15] The ontology at `http://www.linkedopenservice.org/ns/METAR` is an extension to the DAML weather ontology at `http://www.csd.abdn.ac.uk/research/AgentCities/WeatherAgent/weather-ont.daml` and offers specific instances and internationalised labels to better support the GeoNames weather services. The instance metar:FewClouds is, for example, of type daml-weather:FewCloudsLayer.

5 Related Work

The most directly related initiatives have taken the names 'Linked Services [14] and 'Linked Data Services' (LIDS) [16]. The core notions of Linked Services, within which LOS will fit, concern the exposure of service *descriptions* using Linked Data principles. While the preference is expressed that services should communicate RDF, no particular means to achieve this are given and in existing descriptions the entrenched SWS approach of merely classifying inputs and outputs is followed, although more recent work has adapted a partonomy vocabulary to specify more details of what is communicated. LIDS, on the other hand, are very close to LOS in their specification, using SPARQL graph patterns to define input and output. The primary difference at the level of description is that the patterns are combined, in LIDS, into an erstaz SPARQL CONSTRUCT query, with a service 'endpoint' specified in the FROM clause. We note that this is fundamentally incompatible with true RESTful, where there is no single endpoint, but each resource is identified and directly operated on via HTTP methods.

LIDS are also currently concerned only with retrieval; in REST terms, like Linked Data and SPARQL until update bindings are recommended, all operations are based on HTTP GET. LOS is particularly concerned with accommodating side-effecting computation. The other primary difference between LOS, Linked Services and LIDS is in composition. General Linked Services do not yet specify any means at all to achieve composition. LIDS are oriented towards simple automated composition based on the user submission of a (syntactically restricted) SELECT query, which is met by combining data from known LIDS.

Differences aside, however, the existence of these fledgling approaches is heartening for the combination of services and Linked Data and we would note foremost that Linked Open Services do not mean to compete, but to agree shared principles and achieve convergence. We note in particular that LIDS meet Principles **1** and **2** and could easily be created in accordance with Principle **3**. Similarly all principles are compatible with the general aims of Linked Services, simply being more concrete.

Two interesting projects that more recently emerged with similar baseline motivations are the Clerezza project,[16] and the work with slideshare.net by Paul Groth at the Vrije Universiteit in Amsterdam.[17] These efforts, having emerged independently of our ideas, emphasize the trend towards more RDF-minded services and interfaces on top of Linked Data, and clearly support our cause.

The Apache Incubator project Clerezza works on developing components for building RESTFul Semantic Web applications and services. The Clerezza argument, which we second, is that many Web application frameworks simply try to inject non-Web design patterns into the Web environment to profit from the Web-as-the-Platform, instead of developing native Web-based solutions. To this end, Clerezza allows to develop applications that integrate perfectly in the Semantic Web providing all accessible resources in machine understandable formats without imposing additional burdens on the developer. More technically spoken, there is an API to access RDF Graphs with a JAX-RS implementation for which type handlers bind JAX-RS resources to RDF types.

[16] `http://incubator.apache.org/clerezza/`
[17] `http://linkeddata.few.vu.nl/slideshare/`

There are also generic tools that do venture in service 'territory' from the Linked (Open) Data community. The D2R Server, for example, allows for publishing relational databases on the Semantic Web. While D2R is about producing RDF from static legacy data sources Projects like RDFizer and Virtuoso Sponger take this further towards the world of services.[18] RDFizer offers various translators from arbitrary data objects into RDF; e.g., jpeg → RDF, BibTEX → RDF, or Javadoc → RDF. The Virtuoso Sponger provides a generic wrapper for transforming non-RDF Web sources such as (X)HTML pages, binary files and Web services into RDF. The third aspect is most interesting for us, as the Sponger wrapps services such as Google, Del.icio.us, or Flickr and offers the service resources as RDF. We note that especially in this area, although still based mainly on retrieval and transformation rather than general computation, there is significant work to be reused and aligned with in LOS on *provenance* as Linked Data; i.e. vocabularies that allow us exactly to encode the "according to [service provider] at [timestamp]" aspect of the implicit knowledge in service interaction.

6 Conclusion

In this paper we have motivated and introduced the principles guiding the creation of Linked Open Services. We have illustrated how their creation can be achieved both 'from scratch' and by wrapping existing services. In both cases RDF is available, via HTTP content negotation, for direct communication and SPARQL graph patterns are used to describe the required input, and the expected output, capturing the full knowledge contribution of service execution. A LOS wrapper specifies not only what graph patterns a service consumes and produces, respectively, but also how the input RDF is 'lowered' to the expected data format of the service, respectively how the output of the service is 'lifted' back to RDF by means of a SPARQL CONSTRUCT with a head equal to the output pattern.

In the case of wrapping services we have already produced a library, JSON2RDF, to aid the process. In the longer term our aims are to build on JAX-RS to offer support, either over or alongside Clerezza, for LOS production. In both cases, however, the use of these libraries, and the Java language, will be by no means a requirement. It is a fundamental of LOS that the only platform is the Web, and that anyone managing RDF resource descriptions over HTTP should be enabled to take part.

So far we have motivated our work by the emerging need for exposing services as RDF resources on the Web that can be more easily integrated into RDF-minded mashups and the Linked Open Data cloud. In a previous publication, we have promoted our idea of exposing services as RDF prosumers with the goal to enable service compositions in semantic spaces [8]. In this context we have published a set of LOS! Composition Principles that foster the execution of workflows entirely based on SPARQL and the sharing of RDF data. While the details of this work are not subject to this paper, the creation and execution of SPARQL-driven processes is another motivator for our work.

[18] `http://simile.mit.edu/wiki/RDFizers` and `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSponger`.

Acknowledgement: The work is supported by the EU FP7 IP SOA4All, and the e-Infrastructures project SEALS. We thank our colleagues from these projects for their valuable discussions and related work, in particular Adrian Marte (SEALS / STI Innsbruck) Jacek Kopecky and Carlos Pedrinaci (SOA4All / Open University).

References

1. Berners-Lee, T.: Semantic Web Road map. W3C Design Issues (September 1998)
2. Berners-Lee, T.: Read-Write Linked Data. W3C Design Issues (2009/2010)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 35–43 (2001)
4. Domingue, J., Cabral, L., Hakimpour, F.and Sell, D., Motta, E.: IRS III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. In: Proceedings of the Workshop on WSMO Implementations (WIW 2004) (2004)
5. Fensel, D.: Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In: IFIP Int'l Conference on Intelligence in Communication Systems. pp. 43–53 (November 2004)
6. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services. Springer (2006)
7. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: Wsmx - a semantic service-oriented architecture. In: ICWS '05: Proceedings of the IEEE International Conference on Web Services. pp. 321–328. IEEE Computer Society (2005)
8. Krummenacher, R., Norton, B., Marte, A.: Towards Linked Open Services. In: 3rd Future Internet Symposium (September 2010)
9. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services . In: OTM Workshops. pp. 917–926 (November 2009)
10. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic markup for web services. Available at `http://www.daml.org/services/owl-s/1.1/overview/` (November 2004)
11. Nixon, L., Simperl, E., Krummenacher, R., Martin-Recuerda, F.: Tuplespace-based computing for the Semantic Web: A survey of the state of the art. Knowledge Engineering Review 23(1), 181–212 (March 2008)
12. Paolucci, M., Ankolekar, A., Srinivasan, N., Sycara, K.: The DAML-S virtual machine. In: The SemanticWeb - ISWC 2003. LNCS, vol. 2870, pp. 290–305. Springer (2003)
13. Pedrinaci, C., Domingue, J., Krummenacher, R.: Services and the Web of Data: An Unexploited Symbiosis. In: AAAI Spring Symposium (March 2010)
14. Pedrinaci, C., Domingue, J., Krummenacher, R.: Services and the Web of Data: An Unexploited Symbiosis. In: AAAI Spring Symposium - Linked Data Meets Artificial Intelligence (March 2010)
15. Sbodio, M., Moulin, C.: SPARQL as an Expression Language for OWL-S. In: Workshop on OWL-S: Experiences and Directions at 4th European Semantic Web Conference (June 2007)
16. Speiser, S., Harth, A.: Taking the lids off data silos. In: Proceedings of the 6th International Conference on Semantic Systems (iSemantics). ACM International Conference Proceeding Series (2010)
17. Vitvar, T., Kopecky, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: 5th European Semantic Web Conferences. pp. 674–689 (June 2008)