Dynamic Linked Data via Linked Open Services

Barry Norton[1], Reto Krummenacher[2], Adrian Marte[2] and Dieter Fensel[2]

[1] AIFB, Karlsruhe Institute of Technology, Germany
[2] Semantic Technology Institute, University of Innsbruck, Austria
firstname.lastname@{kit.edu | sti2.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 allows for 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 (LOD) 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 such technologies. This is by no means a completely novel aim. Indeed, the seminal reference on the vision of the Semantic Web [1] 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 achieve these aims. At the same time SWS approaches have all built primarily upon the 'WS-*' stack, grounded in SOAP and WSDL, seeking primarily to prove their worth to the enterprise.

In the mean time, the services actually achieving widespread use on the Web (both 'Web 2.0' and semantic 'Web 3.0') have largely turned away from SOAP, towards REST, and increasingly abandoned 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 are neglecting semantics, not just SWS-like service descriptions, but also the messages they communicate (even where these concern Linked Data resources).

We introduce as a running example, and will describe in more detail later, the services offered over GeoNames data, an important component of the Linked Data cloud.

As shown in Figure 1,[1] while many services are provided, 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.

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

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

This position 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-driven interactions (rather than retrieval only). 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 sketch, in Section 5, our ideas for composing such services. In Section 6 we discuss related efforts, and then draw conclusions in Section 7.

2 Motivation for Linked Open Services

An initial trigger for our position is that the datasets that initiatives like 'Linking Open Data' have managed to openly expose on the Web seem to be *semantically* available only for passive retrieval. Otherwise the semantics are hidden away behind (somewhat) RESTful APIs 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. The appearance of Semantic Web pipes and other mash-up tools and techniques strengthens our interest in finding more powerful, but still simple, solutions that allow the smooth integration of service interactions with Linked Data sources.

While XML is still arguably the predominant format for data transfer over the Internet, it is often simplified as JSON.[2] As mentioned previously, this is oddly also the

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

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

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.

It might seem natural that the years of work on Semantic Web services would provide such a possibility, however, we need to 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 SA-WSDL: WSMO-Lite [10], MicroWSMO [5] and SA-REST[3] 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 the syntactic output message, associated with to service execution. For instance, in describing a weather service, 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 much more about semantics for Web services, than about services for the Semantic Web. Our proposal, on the other hand, aims at capturing the *knowledge* derived from a service invocation; not just in the sense of "20 degrees Celsius is the current temperature *in Vienna*",[4] 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 consumers. This will, moreover, have the effect that services can thereby more easily be integrated in service compositions that generalise 'mash-ups' to accommodate side effects, while still remaining data focussed.

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

[4] 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.

3 Principles

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

The taken approach was much inspired by the Linked Data movement that was set in motion with a number of principles, which we reproduce here:[5]

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 [7], but to propose a list of further service-specific principles to be followed for openly exposing services over Linked Data.

1. **Describe services' input and output as SPARQL graph patterns.**
2. **Communicate RDF by RESTful content negotiation.**
3. **The output should make explicit its relation with the input.**

Associated with the last principle is an optional fourth:

4 Make the lifting/mapping openly available as 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[6]. 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 of services as manifested in the fourth principle (cf. example in Section 4). In fact, we go further and, as sketched in our original LOS paper [3], propose SPARQL as baseline for open specifications of LOS composition (Section 5):

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

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

[6] With the exception of the work of Sbodio and Moulin[8], discussed below.

4 Example

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 community. 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.[7] In order to expose non-RDF Web services according to these principles, 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,[8] 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.

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 output pattern which is specified by means of a SPARQL graph pattern (Table 1). In particular, the service returns the latest observations from the nearest weather station, according to reverse geocoding.

[7] http://www.linkedopenservices.org

[8] http://www.geonames.org/export/JSON-webservices.html#weather

While the original service offered by GeoNames accepts untyped latitude and longitude coordinates, our corresponding LOS service accepts a geographic points according to the WGS84 ontology (consumption pattern in Table 1). After a successful HTTP POST request with this 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, and returns the weather observations as one RDF graph in the body of the HTTP response.

LOS services, according to the principles, should be implemented as Web resources that 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 body and in the present case the coordinates are applied to construct the query string for the GeoNames Weather service at hand.[9] After invocation, a SPARQL CONSTRUCT query is used to encode the knowledge contribution of the service, according to the production pattern of the service (Table 1), contrasted with the original JSON output in Table 2.

Table 2. Automatic JSON2RDF Lifting

GeoNames output JSON:

```
{"weatherObservation":
  {"clouds":"broken clouds","windDirection":180,
   "ICAO":"EDDT","lng":13.32,"temperature":"9","windSpeed":"07",
   "stationName":"Berlin-Tegel","lat":52.57,... }}
```

RDF output:

```
_:node16804 met:weatherObservation [
  weather:hasStationID airport:EDDT;
  met:stationName "Berlin-Tegel";
  wgs84:lat "52.57"^^xsd:double; wgs84:long "13.32"^^xsd:double;
  weather:hasVisibilityEvent metar:BrokenCloud;
  weather:hasWindEvent [weather:windDirection "180"^^xsd:short],
                       [weather:windSpeedKnots "07"^^xsd:short];
  weather:hasTemperatureEvent
                  [a weather:CurrentTemperature;
                   weather:celsiusTemperature "9"^^xsd:short];
   ... ] .
```

[9] An example URL as given by GeoNames would be `http://ws.geonames.org/findNearByWeatherJSON?lat=52.6&lng=13.3`.

5 Linked Open Services and their Composition

So far we have presented LOS as an alternative approached to overcome the claimed limitations of existing SWS approaches with respect to the loss of implicit knowledge associated with service invocation, and the use of non-standard and unfamiliar languages for the description of service contributions and interactions. Having linked RDF data produced by means of service interaction, however, allows us to go a step further and to propose process models based on LOS principles.

As the required input of a service is defined by the description of the expected knowledge state prior to invocation (i.e., the consumption pattern), we have a means for providing data-centric preconditions. In principle, any service for which the consumption pattern can be satisfied can be invoked, and its execution adds a knowledge contribution — that might be required by some subsequent service — to the service composition.

To this end, we propose to compose services by means of semantic spaces (a semantic-minded evolution of tuplespaces into a blackboard-style communication and coordination platform for the realization of collaborative problem-solving activities [2][6]) that collect knowledge and effect the control and data flow. The fundamental principle on which computation over semantic spaces is based is that participating agents or services do not communicate with each other directly, but instead collaboration is guided by the coordinated access to and the (concurrent) modification of a shared set of RDF statements.

In the simplest case, all the required and produced RDF data is shared in a process-owned space (termed process space) against which SPARQL queries are executed. Generalizing this setting, without altering any of the core concepts, the input data could also be derived by more complex CONSTRUCT queries over public data sources, or by means of Semantic Web Pipes [4] that aggregate data from various sources on the Web or the LOD that explicitly enhance the knowledge state maintained in the process space. As a direct consequence of the described implicit invocation of services within a process, there is no need to ship data around between Linked Open Services. In fact, any service selects precisely the data from the space that is required, and as such, the input to one service is a priori independent of the output of the predecessors. In other words, the execution of a process does not really require the specification of any data flow; control flow, however, can be enforced as shortly outlined at the end of Section 3.

At a more fine-grained level, activities, which are stages of a process, subscribe their consumption patterns at the process space, and consume the required RDF when available. The execution of a service then contributes to the overall process knowledge maintained in the space. In other words, before carrying out the lowering, to derive the input message for the invocation, the declared ‘consume’ CONSTRUCT (as of the consumption pattern) is carried out over the process space, and any required external Linked Data sources, populating a smaller activity space that stores all the activity related data (e.g., input, output and links between the two). After the invocation the returned message, lifted if necessary, is added first to the local activity space, and the ‘produce’ CONSTRUCT (as of the production pattern) is then carried out over the activity’s space in order to derive the RDF graph that is injected into the process space. In this way, implicit knowledge about the links between the input and output, as well as

provenance and context related to the authority (i.e., the service) can be included in the activity's contribution to the process' knowledge base.

6 Related Work

The most directly related initiatives have taken the names 'Linked Services' [7] and 'Linked Data Services' (LIDS) [9]. 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 ersatz SPARQL CONSTRUCT query, with a service 'endpoint' specified in the FROM clause. We note that this is fundamentally incompatible with true RESTful services, 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 too. 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,[10] and the work with slideshare.net by Paul Groth at the Vrije Universiteit in Amsterdam.[11] 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

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

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.

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.[12] RDFizer offers various translators from arbitrary data objects into RDF; e.g., jpeg, BibTEX and Javadoc. Virtuoso Sponger provides proprietary means for transforming non-RDF Web sources, including services, into 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 for encoding the desired "according to [service provider] at [timestamp]" aspect of the implicit knowledge in service interaction.

7 Conclusion

In this position paper we have outlined and motivated the principles guiding the creation of Linked Open Services and their composition. We have illustrated how they should be created such that RDF is available, via HTTP content negotiation, for direct communication. Moreover, we propose to use SPARQL graph patterns to describe the required input, and the expected output, capturing the full knowledge contribution of service execution. A LOS 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.

For the wrapping of non-native Linked Open Services, as for the GeoNames example, we have already implemented 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 in the effort.

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.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 35–43 (2001)

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

2. 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 (Nov 2004)
3. Krummenacher, R., Norton, B., Marte, A.: Towards Linked Open Services. In: 3rd Future Internet Symposium (September 2010)
4. Le-Phuoc, D., Polleres, A., Morbidoni, C., Hauswirth, M., Tummarello, G.: Rapid Prototyping of Semantic Mash-Ups through Semantic Web Pipes. In: 18th World Wide Web Conference. pp. 581–590 (April 2009)
5. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services . In: OTM Workshops. pp. 917–926 (November 2009)
6. 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)
7. Pedrinaci, C., Domingue, J., Krummenacher, R.: Services and the Web of Data: An Unexploited Symbiosis. In: AAAI Spring Symposium (March 2010)
8. 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)
9. 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)
10. 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)