

Linked Data Meets Services and Processes:

Linked Open Services

Barry Norton, Reto Krümmenacher

SemData@ESWC, May 30, 2010

KARLSRUHE SERVICE RESEARCH INSTITUTE (KSRI)
INSTITUTE OF APPLIED INFORMATICS AND FORMAL DESCRIPTION METHODS (AIFB)

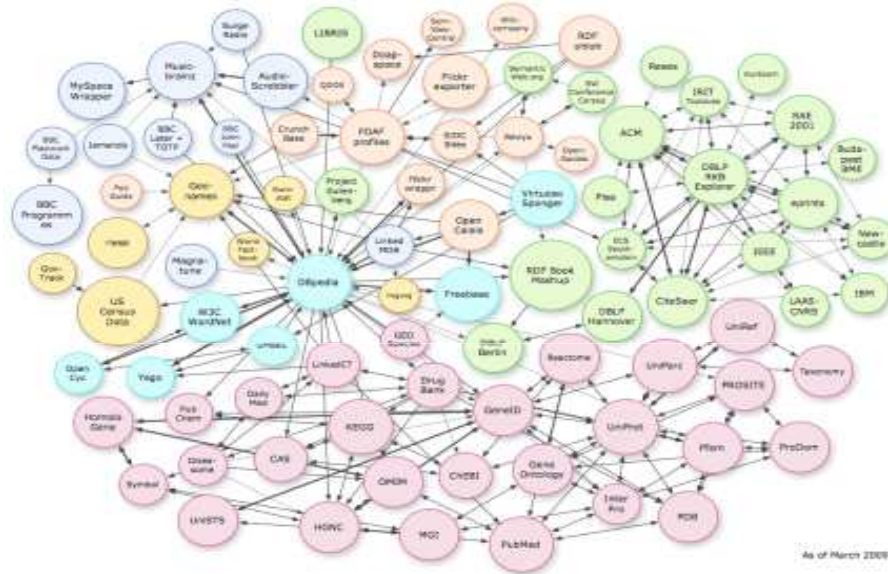
SEMANTIC TECHNOLOGIES INSTITUTE (STI)



Agenda

- State of the art in combination of Linked Open Data and services
 - Services over the LOD Cloud
 - (SWS) Service descriptions in the LOD Cloud
- Why not just SWS?
- Linked Open Services
- Outlook

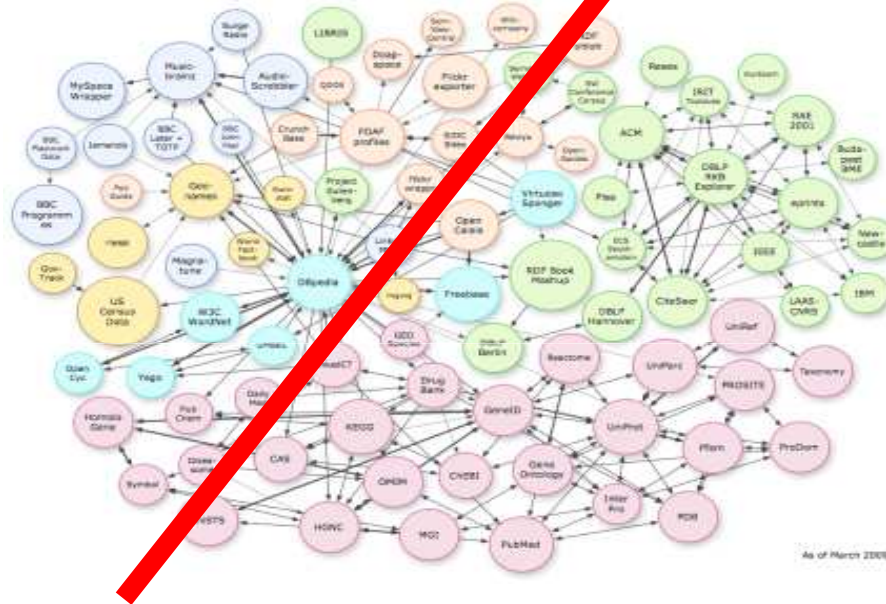
State of the Art – GeoNames.org



GeoNames WeBservices overview

WebService	XML	JSON	RDF	CSV	TXT	RSS	KML
1 asterqdem	XML	JSON			TXT		
2 children	XML	JSON					
3 cities	XML	JSON					
4 countryCode	XML	JSON			TXT		
5 countryInfo	XML	JSON		CSV			
6 countrySubdivision	XML	JSON					
7 earthquakes		JSON					
8 extendedFindNearby	XML						
9 findNearby	XML	JSON					
10 findNearbyPlaceName	XML	JSON					
11 findNearbyPostalCodes	XML	JSON					
12 findNearbyStreets US-only	XML	JSON					
13 findNearbyStreetsOSM	XML	JSON					
14 findNearByWeather Note-1	XML	JSON					
15 findNearbyWikipedia	XML	JSON				RSS	
16 findNearestAddress US-only	XML	JSON					
17 findNearestIntersection US-only	XML	JSON					
18 findNearestIntersectionOSM	XML	JSON					
19 get	XML	JSON					
20 qtopo30	XML	JSON			TXT		
21 hierarchy	XML	JSON					
22 neighbourhood US-only	XML	JSON					
23 neighbours	XML	JSON					
24 ocean	XML	JSON					
25 postalCodeCountryInfo	XML	JSON					
26 postalCodeLookup		JSON					
27 postalCodeSearch	XML	JSON					
28 rssToGeo						RSS	KML
29 search	XML	JSON	RDF				
30 siblings	XML	JSON					
31 srtm3	XML	JSON			TXT		
32 timezone Note-1	XML	JSON					
33 weather		JSON					
34 weatherIcao		JSON					
35 wikipediaBoundingBox	XML	JSON					
36 wikipediaSearch	XML	JSON					
Total	31	34	1	1	4	2	1

State of the Art – GeoNames.org Services



WebService	XML	JSON	RDF	CSV	TXT	RSS
1 asterqdem	XML	JSON			TXT	
2 children	XML	JSON				
3 cities	XML	JSON				
4 countryCode	XML	JSON			TXT	
5 countryInfo	XML	JSON		CSV		
6 countrySubdivision	XML	JSON				
7 earthquakes		JSON				
8 extendedFindNearby	XML					
9 findNearby	XML	JSON				
10 findNearbyPlaceName	XML	JSON				
11 findNearbyPostalCodes	XML	JSON				
12 findNearbyStreets US-only	XML	JSON				
13 findNearbyStreetsOSM	XML	JSON				
14 findNearByWeather Note-1	XML	JSON				
15 findNearbyWikipedia	XML	JSON				RSS
16 findNearestAddress US-only	XML	JSON				
17 findNearestIntersection US-only	XML	JSON				
18 findNearestIntersectionOSM	XML	JSON				
19 get	XML	JSON				
20 qtopo30	XML	JSON			TXT	
21 hierarchy	XML	JSON				
22 neighbourhood US-only	XML	JSON				
23 neighbours	XML	JSON				
24 ocean	XML	JSON				
25 postalCodeCountryInfo	XML	JSON				
26 postalCodeLookup		JSON				
27 postalCodeSearch	XML	JSON				
28 rssToGeo			RDF			RSS KML
29 search	XML	JSON				
30 siblings	XML	JSON				
31 srtm3	XML	JSON			TXT	
32 timezone Note-1	XML	JSON				
33 weather		JSON				
34 weatherIcao		JSON				
35 wikipediaBoundingBox	XML	JSON				
36 wikipediaSearch	XML	JSON				
Total	31	34	1	1	4	2 1

State of the Art – GeoNames.org Weather Service

Weather

Weather data is provided in the METAR (METeorological Aerodrome Report) format. Translations for weather conditions are available in English (default), German, Spanish, French, Italian, Dutch, Finnish, Arabic, Portuguese, Hebrew, Polish, Russian and Bulgarian. Add the parameter 'lang=' with the language code for weather conditions in the desired language.

Help with the [translation of METAR weather conditions](#) is welcome.

Units:
Elevation = meter
Wind speed = Knots
Temperature = Celsius

State of the Art – GeoNames.org Weather Service

Weather

Weather data is provided in the METAR (METeorological Aerodrome Report) format. Translations for weather conditions are available in English (default), German, Spanish, French, Italian, Dutch, Finnish, Arabic, Portuguese, Hebrew, Polish, Russian and Bulgarian. Add the parameter 'lang=' with the language code for weather conditions in the desired language.

Help with the [translation of METAR weather conditions](#) is welcome.

Units:
Elevation = meter
Wind speed = Knots
Temperature = Celsius

```
{"weatherObservation":  
  {"clouds":"broken clouds",  
   "weatherCondition":"drizzle",  
   "observation":"LESO 251300Z 03007KT  
                 340V040 CAVOK 23/15 Q1010",  
   "windDirection":30,
```

State of the Art – GeoNames.org Weather Service

Roman has translated the Metar Weather conditions into Bulgarian. The Metar web services now support Bulgarian (bg) as language parameter. E
<http://ws.geonames.org/weatherIcaoJSON?ICAO=LSZH&lang=bg>

the same condition in English :

<http://ws.geonames.org/weatherIcaoJSON?ICAO=LSZH>

clouds.null= n/a
 clouds.SKC = чисто небе
 clouds.CLR = чисто небе
 clouds.FEW = няколко облака
 clouds.SCT = разкъсана облачност
 clouds.BKN = разбити облаци
 clouds.OVC = плътна облачност
 clouds.VV = вертикална видимост
 condition.null=n/a
 condition.DZ=слаб дъжд
 condition.RA=дъжд
 condition.SN=сняг
 condition.SG=снежни зърна
 condition.IC=ледени кристали
 condition.PL=ледени парчета
 condition.GR=град
 condition.GS=дребен град/снежни парчета
 condition.UP=неизвестен валеж
 condition.BR=лека мъгла
 condition.FG=гъста мъгла
 condition.FU=мъгла
 condition.VA=вълканична пепел
 condition.SA=пясък
 condition.HZ=дим

Weather

Weather data is provided in the METAR (METeorological Aerodrome Report) format. Translations for weather conditions are available in English (default), German, Spanish, French, Italian, Dutch, Finnish, Arabic, Portuguese, Hebrew, Polish, Russian and Bulgarian. Add the parameter 'lang=' with the language code for weather conditions in the desired language.

Help with the [translation of METAR weather conditions](#) is welcome.

Units:
 Elevation = meter
 Wind speed = Knots
 Temperature = Celsius

```
{
  "weatherObservation":
  {
    "clouds": "broken clouds",
    "weatherCondition": "drizzle",
    "observation": "LESO 251300Z 03007KT
      340V040 CAVOK 23/15 Q1010",
    "windDirection": 30,
  }
}
```

State of the Art – Combination of LOD & Services

- Last SemData Workshop presented ‘Linked Services’, which are the exposure of service descriptions as LOD
- Service model based on ‘Minimal Service Model’, which is “SAWSDL in RDF”:
 - ‘De-XMLised’ (WSDL) RPC model in RDF(S)
 - Ontology/vocabulary classification of inputs/outputs
 - Pointer to ‘lifting and lowering schemas’
 - turn XML-based messages into instances of these classes

Why not just SWS?

JSON

```

{"weatherObservation":
  {"clouds":"broken clouds",
   "weatherCondition":"drizzle",
   "observation":"LESO 251300Z 03007KT
                 340V040 CAVOK 23/15 Q1010",
   "windDirection":30,

```

XSPARQL

RDF

```

[ rdf:value "30"^^xsd:int;
  # lifting

```

```

rdf:type :WindReport

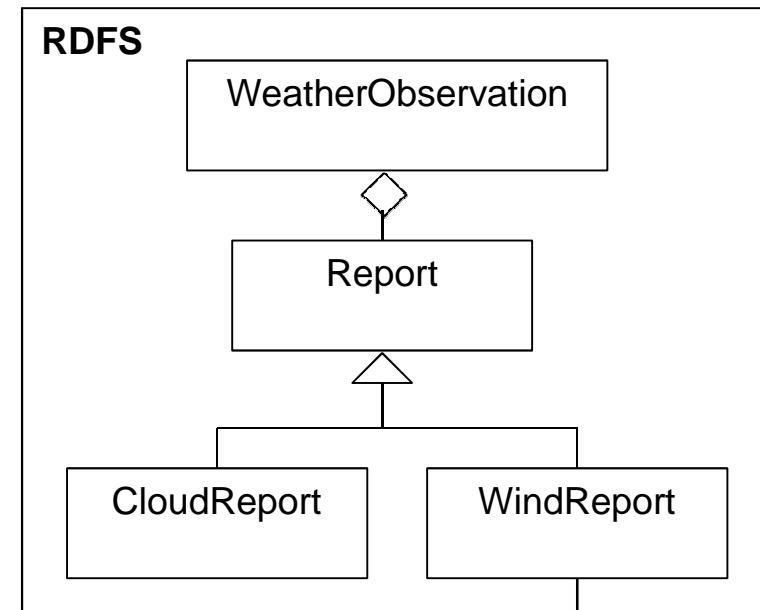
```

```

#classification]

```

RDFS



Why not just SWS?

JSON

```

{"weatherObservation":
  {"clouds":"broken clouds",
   "weatherCondition":"drizzle",
   "observation":"LESO 251300Z 03007KT
                 340V040 CAVOK 23/15 Q1010",
   "windDirection":30,

```

XSPARQL

RDF

[rdf:value ???

lifting

rdf:type :WindReport

← #classification]

RDFS

WeatherObservation

Report

CloudReport

WindReport

Services as LOD

JSON

```

{"weatherObservation":
  {"clouds": "broken clouds",
   "weatherCondition": "drizzle",
   "observation": "LESO 251300Z 03007KT
                  340V040 CAVOK 23/15 Q1010",
   "windDirection": 30,

```

XSPARQL

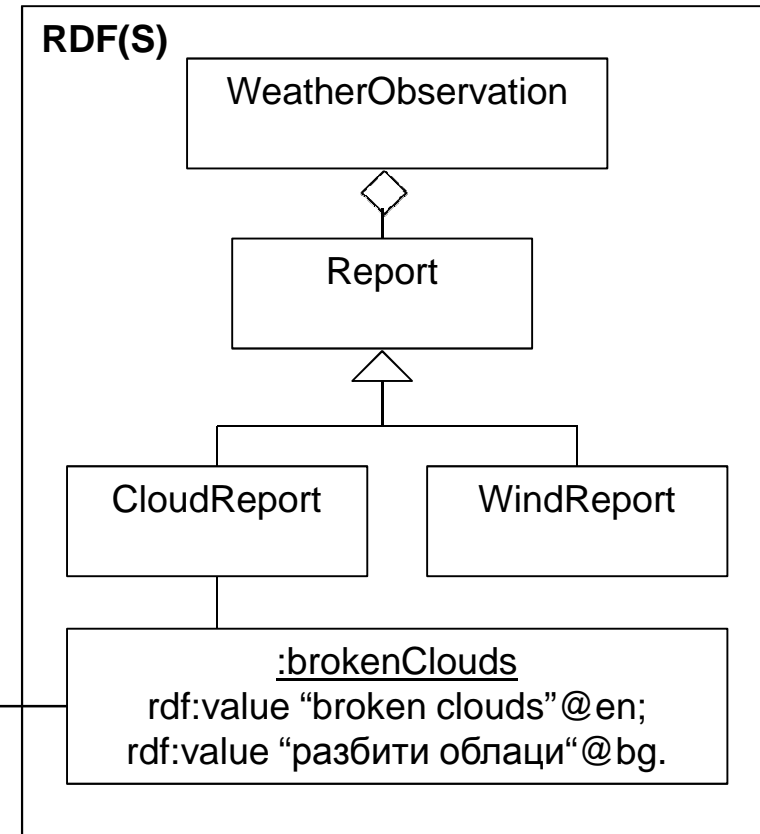
RDF

```

[ rdf:value :brokenClouds ← # lifting
  rdf:type :WindReport      #classification]

```

RDF(S)



Services as LOD

JSON

```

{"weatherObservation":
  {"clouds":"broken clouds",
   "weatherCondition":"drizzle",
   "observation":"LESO 251300Z 03007KT
                 340V040 CAVOK 23/15 Q1010",
   "windDirection":30,

```

XSPARQL

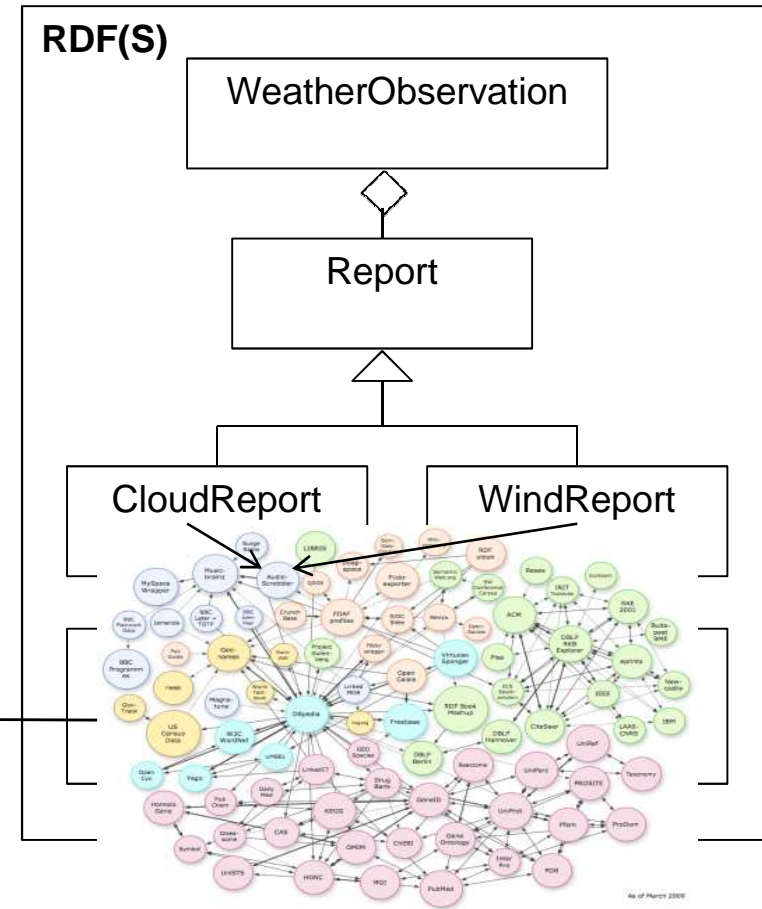
RDF

```

[ rdf:value "30"^^xsd:int;
  # lifting
  <http://www.w3.org/2007/ont/unit/UnitName> ←
  # implicit knowledge
  rdf:type :WindReport
  #classification]

```

RDF(S)



Services as LOD

JSON

```

{"weatherObservation":
  {"clouds":"broken clouds",
   "weatherCondition":"drizzle",
   "observation":"LESO 251300Z 03007KT
                 340V040 CAVOK 23/15 Q1010",
   "windDirection":30,

```

XSPARQL



Where?



RDF

```

[ rdf:value "30"^^xsd:int;
                                     # lifting
<http://www.w3.org/2007/ont/unit/UnitName> ...
                                     # implicit knowledge
rdf:type :WindReport
                                     #classification]

```

Services as LOD

JSON

```
{
  "weatherObservation": {
    "clouds": "broken clouds",
    "weatherCondition": "drizzle",
    "observation": "LESO 251300Z 03007KT
      340V040 CAVOK 23/15 Q1010",
    "windDirection": 30,
  }
}
```

XSPARQL

Where?
Says who?

RDF

```
[ rdf:value "30"^^xsd:int;
  # lifting
  <http://www.w3.org/2007/ont/unit/UnitName> ...
  # implicit knowledge
  rdf:type :WindReport
  #classification]
```

Services as LOD

JSON

```
{
  "weatherObservation": {
    "clouds": "broken clouds",
    "weatherCondition": "drizzle",
    "observation": "LESO 251300Z 03007KT
                    340V040 CAVOK 23/15 Q1010",
    "windDirection": 30,
  }
}
```

XSPARQL

RDF

```
[ rdf:value "30"^^xsd:int;
  # lifting
  <http://www.w3.org/2007/ont/unit/UnitName> ...
  # implicit knowledge
  rdf:type :WindReport
  #classification]
```

Implicit relationship of
input and output

Where?
Says who?

Services as LOD

JSON

```
{
  "weatherObservation": {
    "clouds": "broken clouds",
    "weatherCondition": "drizzle",
    "observation": "LESO 251300Z 03007KT
                    340V040 CAVOK 23/15 Q1010",
    "windDirection": 30,
  }
}
```

XSPARQL

RDF

```
[ rdf:value "30"^^xsd:int;
  # lifting
  <http://www.w3.org/2007/ont/unit/UnitName> ...
  # implicit knowledge
  rdf:type :WindReport
  #classification]
```

Implicit relationship of
input and output

Implicit in interaction with
particular service

Where?
Says who?

Services as LOD

JSON

```
{
  "weatherObservation": {
    "clouds": "broken clouds",
    "weatherCondition": "drizzle",
    "observation": "LESO 251300Z 03007KT
      340V040 CAVOK 23/15 Q1010",
    "windDirection": 30,
  }
}
```

XSPARQL

RDF

```
[ rdf:value "30"^^xsd:int;
  # lifting
  <http://www.w3.org/2007/ont/unit/UnitName> ...
  # implicit knowledge
  rdf:type :WindReport
  #classification]
```

Implicit relationship of
input and output

Implicit in interaction with
particular service

Where?

Says who?

Simply lifting I/O
does not capture
knowledge contribution
of service execution

Linked Open Services (Principles/Manifesto)

- Describe *and expose* services as **LOD prosumers**
 - Describe *inputs and* output as **SPARQL graph patterns**
 - Expose RESTfully with **negotiable RDF**
- Encode implicit knowledge in **knowledge contribution**
 - Encode using SPARQL **CONSTRUCTs**
- Builds **LOD-friendly processes**:
 - Conditions – SPARQL **ASKs**
 - Iteration – SPARQL **SELECTs**

LOS! Example

```
POST /examples/weather/CAO
Host: www.linkedopenservices.org
Content-Type: application/rdf+xml
```

```
<rdf:RDF ...>
  <geonames:City about="http://www.geonames.org/.../Vienna">
...
</rdf:RDF>
```

```
@prefix geonamesCities:<...>

[geonamesCities:vienna
 :weatherCondition
  [:cloudReport :brokenClouds;
 :windReport [rdf:value "20"^^xsd:int ; unit:kph]]
(+ reification for provenance)
```



LOS!

“разбити облаци”@bg.

Outlook

- Linked Open Services Tutorial @ ISWC
- LinkedOpenServices.org/examples
 - Descriptions of *real services*
- LinkedOpenServices.org/ns
 - Service and process models
- LinkedOpenServices.org/blog
 - RSS feed of developments
- LinkedOpenServices.org/wiki
 - *Open* development

