SSSC 2011
Linked Data and Services

Barry Norton [1], Denny Vrandecic [2]
[1] Ontotext AD, BG/UK
[2] AIFB, Karlsruhe Institute of Technology, DE

Berkeley, US; August 8, 2011
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
  - Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
HTTP Overview

HTTP, by which all documents on the WWW are served, is a client server protocol. Every interaction based on:

- Request
  - Method
    - GET
    - PUT
    - POST
    - PATCH
    - DELETE (+ OPTIONS, HEADER, TRACE, CONNECT)
  - URL
  - Header
  - [Optional] Body (with POST, PUT, PATCH)

- Response
  - Response code (integer)
  - Header
  - [Optional] Body
HTTP GET Example

Retrieval example:

```bash
GET /web/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Accept-Language:en
```

Can negotiate on (human) language, but also...

```
HTTP/1.0 200 OK
Date: Sun, 07 Nov 2010 01:00:00 GMT
Content-Type: text/html

<html>
<head> ...
<body>
```
HTTP Conneg Example

Content negotiation (coneg) example:

GET /id/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Accept: text/html

HTTP/1.0 302 Moved Temporarily
Date: Sun, 07 Nov 2010 00:30:00 GMT
Location: http://www.aifb.kit.edu/web/Barry_Norton

GET /id/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Accept: application/rdf+xml

HTTP/1.0 302 Moved Temporarily
Date: Sun, 07 Nov 2010 00:45:00 GMT
Location: http://www.aifb.kit.edu/portal/index.php?
title=Spezial:Exportiere_RDF/Barry_Norton
HTTP PUT/PATCH Examples

PUT/PATCH example:

PUT /web/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Content-Type: text/html

<html> ...

(new resource or complete update)

HTTP/1.0 200 OK (or 201 CREATED)
Date: Sun, 07 Nov 2010 00:10:00 GMT

PATCH /web/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Content-Type: text/html

Change...

(partial update)

HTTP/1.0 200 OK
Date: Sun, 07 Nov 2010 00:10:00 GMT
HTTP POST Examples

POST-compute example:

POST /web/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Content-Type: something

(input -> computation -> output)

POST-append example:

POST /web/Barry_Norton HTTP/1.1
Host: www.aifb.kit.edu
Content-Type: text/html

(new related resource)

HTTP/1.0 200 OK
Date: Sun, 07 Nov 2010 00:10:00 GMT
Content-Type: something

Result....

HTTP/1.0 201 CREATED
Date: Sun, 07 Nov 2010 00:10:00 GMT
Location:http://www.aifb.kit.edu/...
Representational State Transfer

HTTP, maintained by IETF not W3C, is just one (primary) implementation of an architectural style called:
REST = REpresentational State Transfer

REST Principles
1. Application state and functionality is divided into resources
2. Every resource is uniquely addressable
3. All resources share a uniform interface:
   a) A constrained set of well-defined operations
   b) A constrained set of content types
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
  - Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
The defining principles of Linked Data are few and simply stated:

**Linked Data Principles**
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.

These are the latest version of their statement, contained in a W3C Note ‘Linked Data – Design Issues’ by Tim Berners-Lee

[http://www.w3.org/DesignIssues/LinkedData.html](http://www.w3.org/DesignIssues/LinkedData.html)
A foundational issue in Linked Data was the distinction of URIs for real-world objects versus (e.g., RDF) documents that might describe them.

One solution is to include a Cool URI with a hash, as follows:

http://www.w3.org/People/Berners-Lee/card#a Foaf:Person.

Note that Web browsers already crop hash URIs in this way.
Linked Data Principle 2

Linked Data Principles

...  
2. Use HTTP URIs so that people can look up those names...

• HTTP allows a second way to distinguish real-world objects from documents, e.g. In DBPedia:

GET
Accept: application/rdf+xml

http://dbpedia.org/data/Vienna.xml

http://dbpedia.org/resource/Vienna
 a dbpedia-owl:PopulatedPlace.

GET
Accept: text/html

http://dbpedia.org/page/Vienna

• Principles say HTTP 303 and Location header should be used
Linked Data Principle 3

Linked Data Principles

... 3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

- While RDF/XML should be the default for look-up
  - RDFa annotations in HTML are now also standard
  - it is increasingly encouraged to also offer Turtle:

  ```
  GET  
  Accept: text/n3
  ```

  ```
  http://dbpedia.org/resource/Vienna
   a dbpedia-owl:PopulatedPlace.
  ```

  ```
  http://dbpedia.org/data/Vienna.n3
  ```

- A dump of the whole dataset, and a SPARQL endpoint for queries are also encouraged (see Publishing)
Linked Data Principle 4

Linked Data Principles

4. Include links to other URIs, so that they can discover more things.

- There are several ways to reuse URIs:
  - At schema level –
    - direct reuse of class/property
    - (RDFS) sub-class/-property
    - (OWL) equivalent class/property
    - SKOS broad match

- At instance level –
  - direct reuse
  - (RDFS) seeAlso
  - (OWL) sameAs

There are a number of standard schemas, considered next for publishing, and datasets, considered in the Cloud, that should always be considered.
What to Return for a URI

• **The (immediate) description:** All triples that have the resource's URI as the subject.
• **Backlinks:** All triples that have the resource's URI as the object. This is redundant, but it allows bi-directional traversal.
• **Related descriptions:** Anything about related resources that may be of interest in typical usage scenarios; use prudence.
• **Metadata:** Any metadata such as the author and licensing information.
• **Syntax:** At least RDF descriptions as RDF/XML which is the only official syntax for RDF.
  • As RDF/XML is not very human-readable, the data could additionally be provided in other formats; e.g., for MIME-type application/x-turtle.

Note that text/n3 and text/rdf+n3 are currently better-supported though a registered Internet Media type will be established
Publication Means

There are a number of popular means to publish Linked Data:

- Host RDF/XML as static files behind Web server
- Include RDFa in HTML generated from existing content management system
- Publish direct from triplestore
- Expose relational database via translation
Publishing Static RDF

- The easiest way to create some simple Linked Data content is to host a hand-edited file on a Web server
- For example this is often used to identify a person by hosting a FOAF file (see later)
- There are consequences with respect to best practice:
  - the Web server should be configured to respect (Accept) requests according to the proper Internet Media types
  - it is messy (based on duplication) to offer alternative serialisations
  - SPARQL-based queries have to load whole graph so these should be small
- This approach is only sensible at small scale
Publishing Dynamic RDFa

• Not covered in depth in this module, RDFa is W3C Recommendation for including RDF as annotations to HTML
• It is argued that this can be the minimally invasive way to augment existing Web systems:
  • Web servers need minimal reconfiguration (should recognise application/xhtml+xml media type)
  • Some generic content management systems (CMS), e.g. Drupal, and some eCommerce solutions, already include RDFa support
  • It is argued e.g., for GoodRelations (see later) that annotation of text features can be optional and ‘RDFa blocks’ included into HTML
• Arguably, though, RDFa is least readable to humans, though both RDF/XML and Turtle can easily be obtained from a ‘distiller’

http://www.w3.org/TR/rdfa-syntax/
Publishing Direct from Triple Store

• The most expedient way to follow principles and best practice for large datasets is to store data in RDF form in a triple store
• This has a number of advantages:
  • Most triplestores allow HTTP negotiation of serialisation
  • Allows direct processing of SPARQL queries, and therefore provision of a SPARQL endpoint, over data
  • Implicitly supports per resource deferencing via DESCRIBE queries
• Good solution for hosting new datasets, however in many cases there already exists an infrastructure which cannot be so easily replaced.
Exposing Relational Database via Translation

- Many large datasets are managed using relational DBMSs
- D2R is a popular solution for providing, via translation:
  - Dereferencing of resources
  - SPARQL processing, and
  - complete RDF dumps
- Translation rules expressed in D2RQ, a Turtle encoding of:
  - Mapping between major relational tables and RDFS classes
  - ‘Bridges’ between columns and RDF properties, including
  - Conditions and (programmatic) translations
- The W3C has recently chartered a working group to work on a standard for such mappings

http://www.w3.org/2009/08/rdb2rdf-charter
A vocabulary of growing importance in publishing datasets is VoID, the Vocabulary of Interlinked Datasets, defining in RDFS:

- Access metadata:
  - Example resources via resolvable URIs and URI patterns – `void:exampleResource`
  - SPARQL endpoints – `void:sparqlEndpoint`
  - Dumps – `void:dataDump`
- Used vocabularies – `void:vocabulary`
- Statistics – `void:triples`, `void:distinctObjects`
- Interlinkage – `void:LinkSet`, `void:target`

VoID also guides the use of existing vocabularies for
- Licensing, via `dc:license`
- Contact information, via `foaf:Person`

http://www.w3.org/TR/void/
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
- Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
The consumption of Linked Data follows two strategies that are comparable to current Web use, with additional advantages, and one further:

- **Linked Data Browsing** – Linked Data best practice and inter-linkage are exploited to offer an effective human guided traversal of the Web of Data
- **Linked Data Search** – semantics are exploited to improve on existing Web search over the Web of Data
- **Linked Data Mash-ups** – datasets are brought together to create new applications
Linked Data Browsing

Traditional Web Browsers:
• render pre-existing documents
• display these according to their fixed HTML representation
• navigate according to pre-existing hyperlinks (in document).

Linked Data Browsers
• render resources according both to pre-existing HTML representations, but also interpretations of RDF terms:
  • rdfs:label implies a label for something, which may have internationalisation via language tagging
  • rdfs:comment implies further information that may be optionally displayed, e.g., as a pop up
  • foaf:depiction implies availability of a picture, etc.
• navigate also according to related datasets, even if not nominated by original publisher of resource of focus
Tabulator is an extension to the Firefox browser with views for tables, maps, calendars, timelines, etc. and mechanism for extension to other views

http://www.w3.org/2005/ajar/tab
Tabulator is an XHTML server that aggregates Linked Data about nominated resources.

It displays this using coloured ‘marbles’ to illustrate where each item came from and to allow navigation.

http://marbles.sourceforge.net/
Linked Data Search

Traditional Web search, covered in previous module:
• primarily keyword-based
• crawls documents via hyperlinks
• stores each as ‘bag of words’
• uses (ambiguous) hyperlinks only to judge popularity
• presents results mostly as a set of links

Semantic / Linked Data Search
• crawls given RDF-described inter-linkage structure
• integrates information about unambiguous resources
• interprets query generally, using inference to find matches
• presents results intelligently and in coherent fashion
Sig.ma allows keyword-based search and presents aggregated results, via Sindice, in style of Linked Data Browsing, as well as list of sources.

http://sig.ma/
Google increasingly uses semantic annotations to improve on traditional search results, for instance in finding recipes & showing pictures, ingredients, timing & reviews.
Linked Data Mash-up

Building an application aggregating Linked Data balances two strategies:
• Utilise a triple store that has application data already available via –
  • **Crawling** ahead of time, using tools such as LDSpider
  • **Loading** of pertinent datastores from dumps
• On-the-fly aggregation, via –
  • **Run-time dereferencing** of RDF resource descriptions
  • **Federation** between SPARQL endpoints

Significant effort is still needed to achieve reusable visualisation of Linked Data; many current mash-ups are forced to transform data to non Semantic Web formats.
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
  - Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
Real-time updates to a large (ferocious) audience

RDF-based communication efficiently realised using memcached
Motivating Example Datasets

The GeoNames geographical database is available for download free of charge under a creative commons attribution license. It contains over 10 million geographical names and consists of 7.5 million unique features.

The MusicBrainz music metadata database is available for download under a public domain (CC no rights reserved) and CC Attribution-NonCommercial-ShareAlike 2.0 license. It contains over 10 million track descriptions.
Motivating Example Linked Data

GeoNames categorises geo features in 9 feature classes sub-categorized into 645 feature codes. An SKOS taxonomy reflects this structure and OWL is used to describe features; URIs can be resolved to RDF and a dump is available. Links are made to DBPedia and other LOD sets.

The MusicBrainz NGS schema has been mapped into the Music Ontology. HTML pages describing MusicBrainz entities are now annotated in RDFa. A dump will soon be made available. Links are made to DBPedia and LOD sets, such as BBC Music Reviews, use MB IDs.
Motivating Example APIs

GeoNames offers a number of reverse geolocation (and containment)-based retrievals – the feature is not directly identified, but a set of qualifying resources are computed from a point (circle/box).

MusicBrainz incorporates PUIDs and uses MusicIP’s MusicDNS audio fingerprinting technologies. When a PUID is submitted they are fuzzily matched to a set of similar tracks.

In both cases computation is needed, before retrieval, to locate the resource to be included. This needs to happen near the data, not be remotely pushed to it.
"weatherObservation":
{"clouds":"broken clouds",
"weatherCondition":"drizzle",
"observation":"LESO 251300Z 03007KT 340V040 CAVOK 23/15 Q1010",
"windDirection":30,
"ICAO":"LESO", ...
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
  - Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
LOS Principles

REST Principles
1. Application state and functionality is divided into resources
2. Every resource is uniquely addressable
3. All resources share a uniform interface:
   a) A constrained set of well-defined operations
   b) A constrained set of content types

Linked Data Principles
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.

Linked Open Service Principles
1. Describe services as LOD prosumers with input and output descriptions as SPARQL graph patterns
2. Communicate RDF by RESTful content negotiation
3. The output should make explicit its relation with the input
Describe services’ input and output as SPARQL graph patterns
Los Weather Service


Output: `[met:weatherObservation [ weather:hasStationID ?icao geonames:inCountry ?country; ...
weather:hasWindEvent [weather:windDirection ?windDirection], [weather:windSpeed ?windSpeed]`
2

Communicate RDF by RESTful content negotiation
LOS Principle 2

GET
Accept: text/html
303 REDIRECT /page

GET
Accept: application/rdf+xml
(or text/n3)
303 REDIRECT /data

GET /weather
Accept: application/rdf+xml
(or text/n3)
200 <rdf:Description>
The **output** should make **explicit** its **relation** with the **input**.
Input: \(?p \text{ a wgs84:Point; wgs84:lat } \text{ ?lat; wgs84:long } \text{ ?long.}\)

Output: [met:weatherObservation [weather:hasStationID ?icao ...

...]]

Linked Open Services Principles

*When wrapping non-LOS services:*

Make the **lifting/mapping open as SPARQL CONSTRUCT queries**
Outline

- Background
  - Web Technologies
- Linked Data
  - Linked Data Principles
  - Data Publishing
  - Data Consumption
- Linked Services
  - Linked Services Motivation
  - Linked (Open) Service Principles
  - Linked Service Implementation
**LOS Wrapping Overview**

**JSON**
```
{"weatherObservation":
  {
  "clouds":"broken clouds",
  "weatherCondition":"drizzle",
  "observation":"EDDT 031520Z 25013KT 9999 ...",
  "countryCode":"DE",
  "windDirection":30, ...
  }
}
```

**RDF**
```
[weather:hasVisibilityEvent metar:BrokenClouds ;
  weather:hasWeatherEvent metar:Drizzle ;
  metar:observation "LESO 251..." ;
  weather:hasWindEvent
    [weather:windDirection "30"^^xsd:integer]
]
```

**RDF(S)**
- WeatherObservation
  - WeatherEvent
    - VisibilityEvent
    - WindEvent
      - metar:BrokenClouds
        rdf:type weather:BrokenCloudLayer
        rdf:value "broken clouds"@en;
        rdf:value "разбити облаци"@bg.
LOS Wrapping Overview

JSON
{"weatherObservation":
{"clouds":"broken clouds",
"weatherCondition":"drizzle",
"observation":"EDDT 031520Z 25013KT 9999 …",
"countryCode":"DE",
"windDirection":30, ...}

RDF
[weather:hasVisibilityEvent metar:BrokenClouds;
weather:hasWeatherEvent metar:Drizzle;
metar:observation "LESO 251…";
weather:hasWindEvent [weather:windDirection "30"^^xsd:integer]]

CONSTRUCT {
  ?bkn weather:hasVisibilityEvent met:BrokenClouds .
  ?dz weather:hasWeatherEvent met:Drizzle .
  ...}
WHERE {
  {OPTIONAL {?bkn temp:clouds "broken clouds"}} UNION ...
}

CONSTRUCT {
  ?f weather:hasVisibilityEvent ?clouds ;
  weather:hasWindEvent [weather:windDirection ?windDirection] ;
  ...
}
WHERE {
  temp:windDirection ?windDirection ;
  OPTIONAL {_:c weather:hasVisibilityEvent ?clouds }
  ...
}
LOS Wrapping Overview

**JSON**

```json
{"weatherObservation":
  {"clouds":"broken clouds",
   "weatherCondition":"drizzle",
   "observation":"EDDT 031520Z 25013KT 9999 ...",
   "countryCode":"DE",
   "windDirection":30, ...
  }
}
```

**RDF(S)**

```
RDF(S)

WeatherObservation

VisibilityEvent

WeatherEvent

WindEvent

metar:BrokenClouds

rdf:type weather:BrokenCloudLayer

rdf:value "broken clouds"@en;

rdf:value "разбити облаци"@bg.
```

**RDF**

```
RDF

[ weather:hasVisibilityEvent metar:BrokenClouds ;
  weather:hasWeatherEvent metar:Drizzle ;
  metar:observation “LESO 251...” ;
  weather:hasWindEvent
    [weather:windDirection “30”^^xsd:integer]
]
```
_:a1 met:weatherObservation [  
  weather:hasStationID <http://www.linkedopenservices.org/services/geo/SpatialResources/point/ICAO/EDDT> ;  
  met:stationName "Berlin Tegel" ;  
  geonames:inCountry "DE" ;  
  wgs84:lat "52.566666"^^xsd:double ;  
  wgs84:long "13.316667"^^xsd:double ;  
  wgs84:alt "37.0"^^xsd:double ;  
  met:datetime "2010-10-27 08:20:00" ;  
  met:observation "EDDT 270820Z 18007KT CAVOK 06/02 Q1025 NOSIG" ;  
  weather:hasVisibilityEvent metar:BrokenCloud ;  
  weather:hasWindEvent [weather:windDirection "180"^^xsd:short] ,  
  [weather:windSpeedKnots "07"^^xsd:short] ;  
  weather:hasTemperatureEvent  
  [a weather:CurrentTemperature ; weather:celsiusTemperature "6"^^xsd:short] ,  
  [a weather:CurrentDewPoint ; weather:celsiusTemperature "2"^^xsd:short] ,  
  [weather:humidityPercent "75"^^xsd:short ;  
Geonames Airports

WGS84 geospatial, but only lat/long (not alt)

‘alternateName’ for both labels and identification schemes (ICAO, IATA)

Only ~100 resources
Some ICAO, IATA (with specific properties) but noisy

Some WGS84, but not consistently

~3500 verifiable airports (see next)
NCAR METAR Station List

~8500 airports and other METAR reporting stations
LOS METAR Data

http://www.linkedopenservices.org/data/METARStations.rdf.xml

Uniform WGS84 lat/long/alt

ICAO and IATA-based URIs plus skos:notation to originals

owl:sameAs
- between ICAO/IATA
- to Geonames (108)
- to DBPedia (1159)
# Geo-spatial query syntax

The special syntax used to query geo-spatial data makes use of SPARQL’s RDF Collections syntax. This syntax uses round brackets as a shorthand for the statements connecting a list of values using `rdf:first` and `rdf:rest` predicates with terminating `rdf:nil`. Statement patterns that use one of the special geo-spatial predicates supported by OWLIM-SE are treated differently by the query engine. The following special syntax is supported when evaluating SPARQL queries (the descriptions all use the namespace `cgeo`: <http://www.ontotext.com/ontowm/geo#>):

<table>
<thead>
<tr>
<th>Construct</th>
<th>Nearby (lat long distance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>?point omgeo:nearby(?lat ?long ?distance)</td>
</tr>
<tr>
<td>Description</td>
<td>This statement pattern will evaluate to true if the following constraints hold:</td>
</tr>
<tr>
<td></td>
<td>• ?point geo:lat ?plat .</td>
</tr>
<tr>
<td></td>
<td>• ?point geo:long ?plong .</td>
</tr>
<tr>
<td></td>
<td>• Shortest great circle distance from (?plat, ?plong) to (?plat, ?long) &lt;= ?distance</td>
</tr>
</tbody>
</table>

Such a construction will use the geo-spatial indices to find bindings for `?point` that lie within the defined circle. Constants are allowed for any of `?lat ?long ?distance`, where latitude and longitude are specified in decimal degrees and distance is specified in either kilometres (‘km’ suffix) or miles (‘mi’ suffix). If the units are not specified, then ‘km’ is assumed.

<table>
<thead>
<tr>
<th>Restrictions</th>
<th>Latitude is limited to the range -90 (South) to +90 (North)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Longitude is limited to the range -180 (West) to +180 (East)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
<th>Find the names of airports that are within 50 miles of Seoul:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>PREFIX geo-pos: <a href="http://www.w3.org/2003/01/geo/wgs84_pos#">http://www.w3.org/2003/01/geo/wgs84_pos#</a></td>
</tr>
<tr>
<td></td>
<td>PREFIX geo-ont: <a href="http://www.geonames.org/ontology#">http://www.geonames.org/ontology#</a></td>
</tr>
<tr>
<td></td>
<td>PREFIX omgeo: <a href="http://www.ontotext.com/ontowm/geo#">http://www.ontotext.com/ontowm/geo#</a></td>
</tr>
<tr>
<td></td>
<td>SELECT distinct ?airport WHERE {</td>
</tr>
<tr>
<td></td>
<td>?base geo-ont:name &quot;Seoul&quot; .</td>
</tr>
<tr>
<td></td>
<td>?base geo-pos:lat ?latBase .</td>
</tr>
<tr>
<td></td>
<td>?link omgeo:nearby(?latBase ?longBase &quot;50mi&quot;) .</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
</tbody>
</table>
Related Activities

- Apache Incubator Clerezza
  Components (bundles) for building RESTful Semantic Web applications and services leveraging Jax-RS

- Linked Services / iServe
  A platform for publishing and browsing Semantic Web services as linked data, based on OWLIM

- Linked Data Services (LIDS)
  An interface spec for data-services, accompanied by a machine-interpretable description of inputs and outputs.

- Iqbal, Sdobio & Moulin
  Pre and post-conditions, as well as user goals in SPARQL
Related Activities

- **ReLL** = Resource Linking Language, Wilde (Berkeley)
- **DEIMOS**, Ambite et al. (USC)
- **iS** = iServe, Pedrinaci et al. (OU)
- **ISM** = Iqbal, Sbodio & Moulin (Galway)
- **LIDS** = Linked Data Services, Speiser & Harth (AIFB)
- **LOS** = Linked Open Services (Innsbruck -> AIFB)

**Activities**
- RDF Communication & Interlinking
- SPARQL Desc.
- Side Effects