Integrated Environment for Visual Data-level Mashup Development

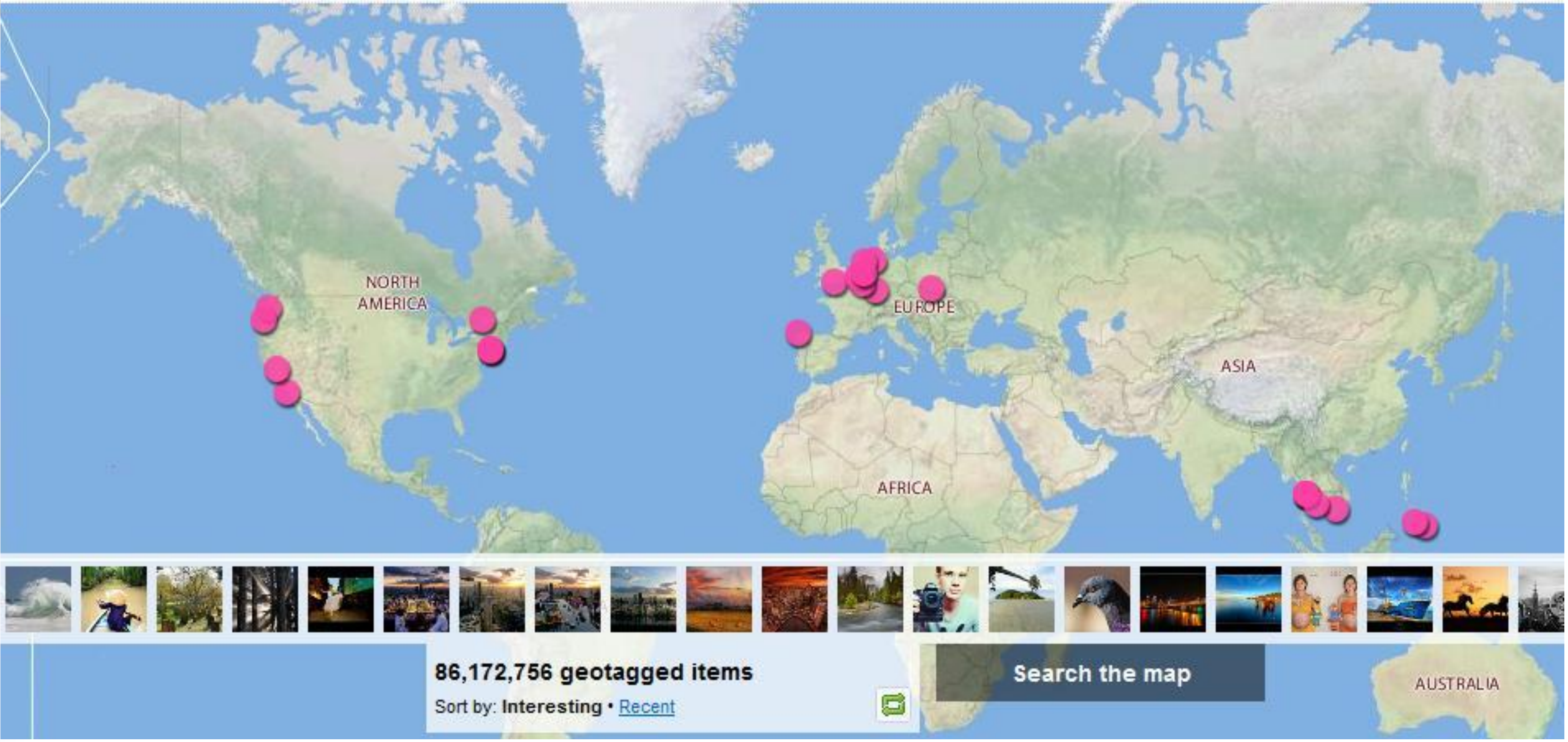
Adam Westerski

Universidad Politécnica de Madrid

Summary

1. Introduction
   - Mashup environments
   - Mashing-up problems
2. *Mashup environments Integration*
   - *Background*
   - *Motivations*
   - *Goals*
   - *Methodology*
3. *Experiments*
   - *Examples*
   - *Problems encountered & Lessons learned*
4. Conclusions

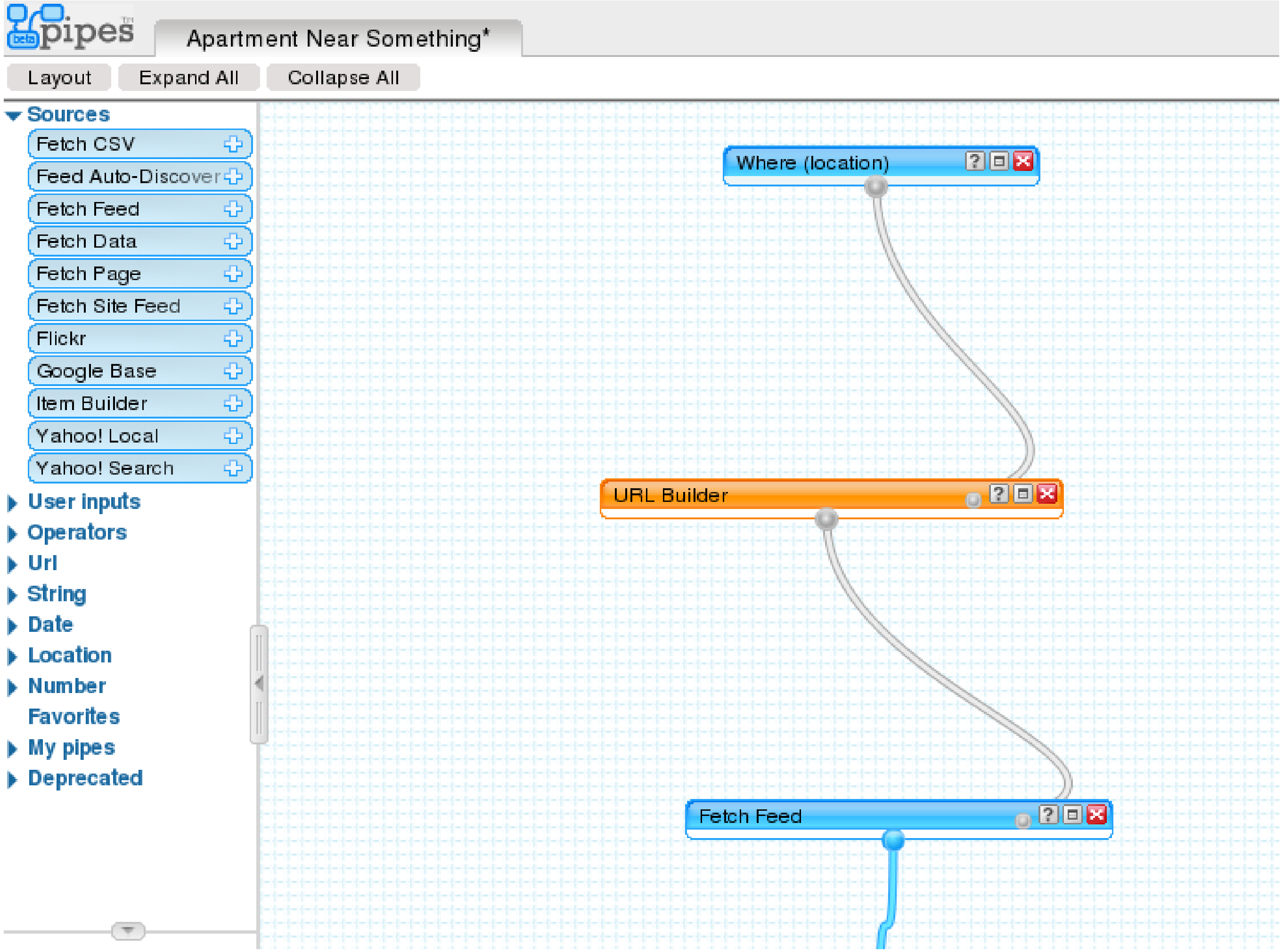
Mashup example



86,172,756 geotagged items

Sort by: Interesting • Recent

Mashup environments



Mashup environments

Mashup problems

- Not complex enough ?
- Too complex ?
- Not reliable enough ?
- Too narrow ?
- ....

- **Is it worth it ?**

Mashup tools integration:

Background

- Work done for EU project Romulus
  - Aid web application development
  - Mashups one of the ways to achieve that

- Lots of tools developed:
- ...
- DERI Pipes (Semantic Web Data Mashups)
- Romulus Mashup Builder (Service Mashups)

- **Question:** what to do with all of that ?

Mashup tools integration: Motivation

- Integrate tools to facilitate better efficiency
- Domain specific data mashup tools lack functionality

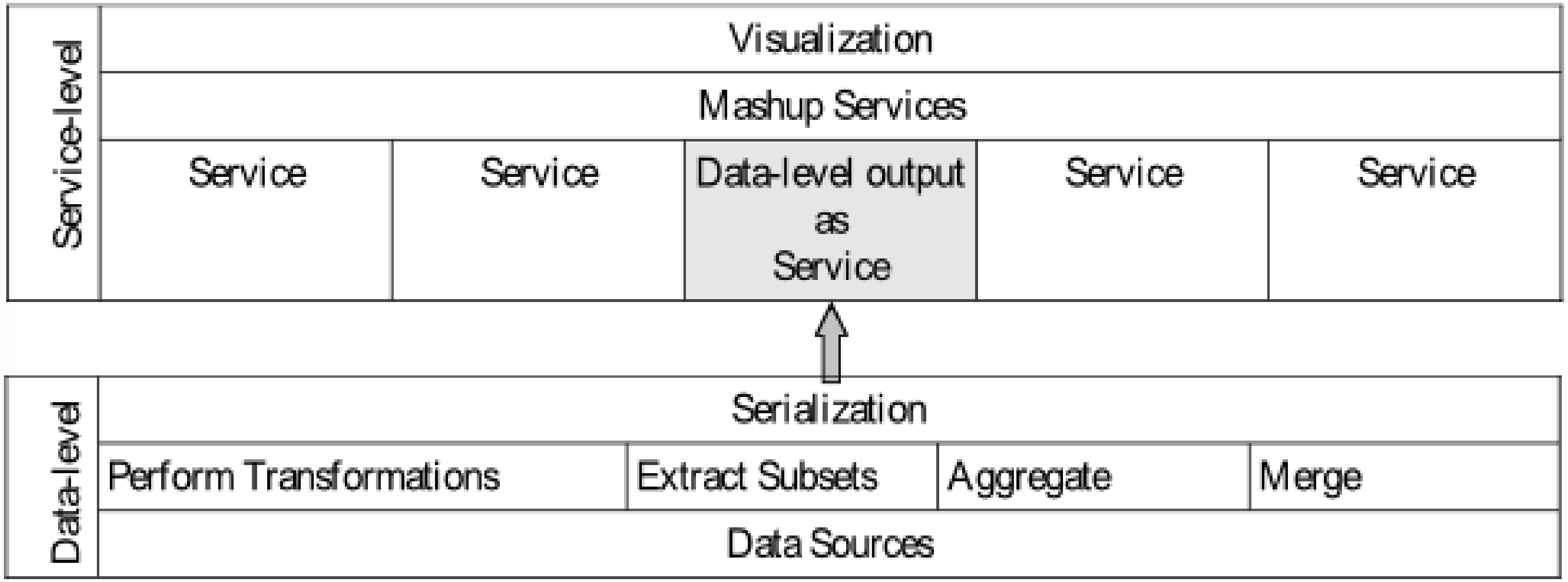
Mashup tools integration:

Goals

1. More effective mashup construction
2. Integrate mashup development as part of software engineering process
3. Drive the research and design for current mashup tools

Mashup tools integration:

Methodology



Service-level

Visualization

Mashup Services

Service

Service

Data-level output as Service

Service

Service

Data-level

Serialization

Perform Transformations

Extract Subsets

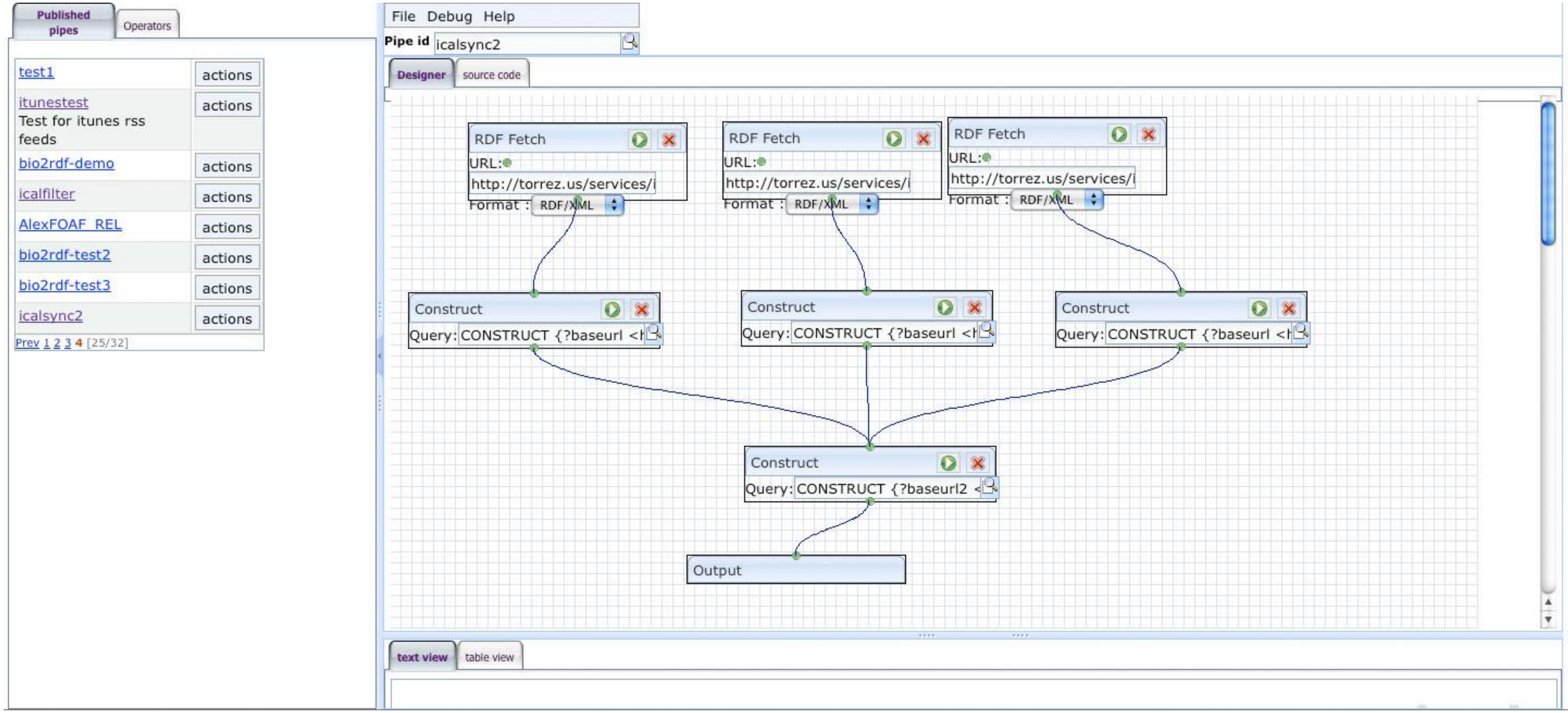
Aggregate

Merge

Data Sources

Experiments:

DERI Pipes



test1	actions
itunestest Test for itunes rss feeds	actions
bio2rdf-demo	actions
icalfilter	actions
AlexFOAF_REL	actions
bio2rdf-test2	actions
bio2rdf-test3	actions
icalsync2	actions

Prev 1 2 3 4 [25/32]

Pipe id icalsync2

RDF Fetch
URL: http://torrez.us/services/i
Format : RDF/XML

RDF Fetch
URL: http://torrez.us/services/i
Format : RDF/XML

RDF Fetch
URL: http://torrez.us/services/i
Format : RDF/XML

Construct
Query: CONSTRUCT {?baseurl <!

Construct
Query: CONSTRUCT {?baseurl <!

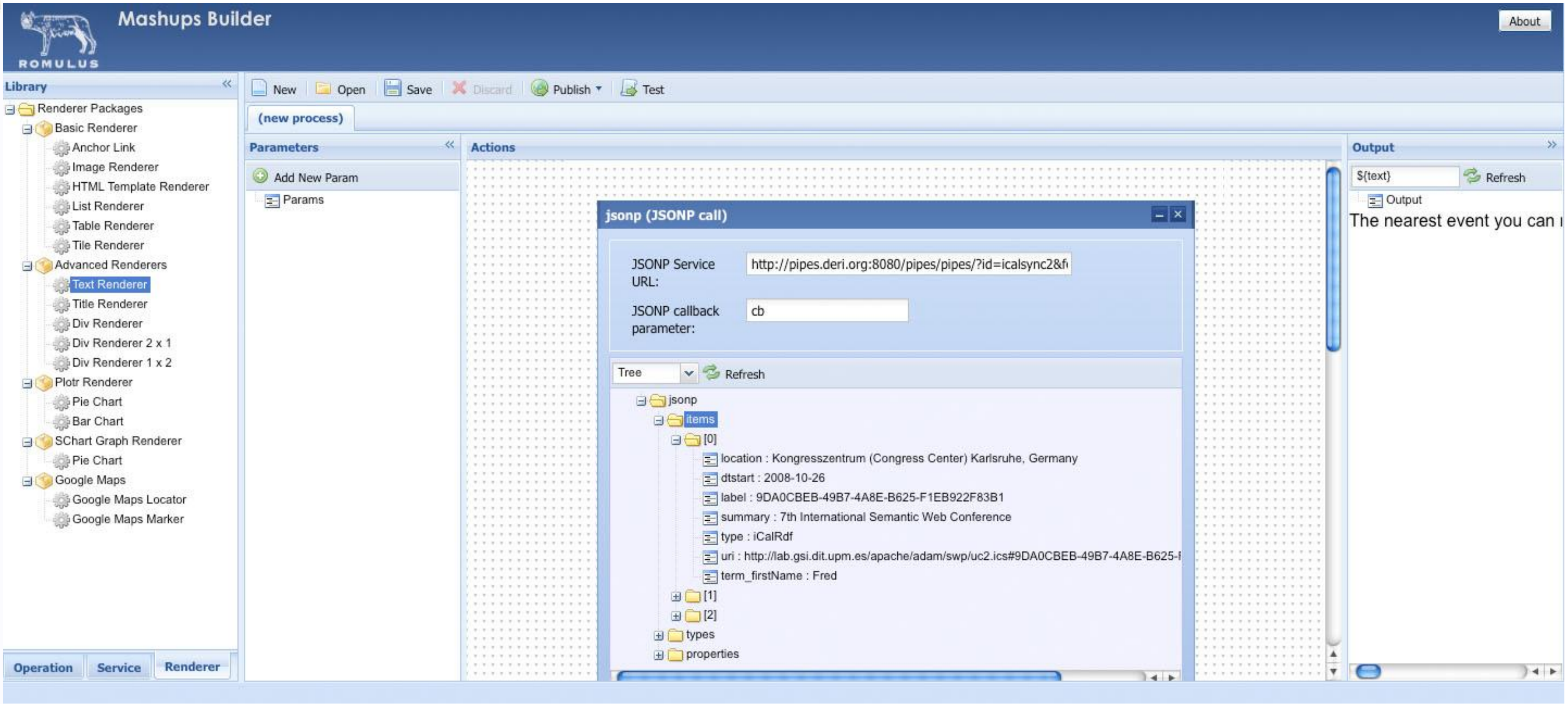
Construct
Query: CONSTRUCT {?baseurl <!

Construct
Query: CONSTRUCT {?baseurl2 <

Output

Experiments:

Romulus Mashup Builder



- Renderer Packages
  - Basic Renderer
    - Anchor Link
    - Image Renderer
    - HTML Template Renderer
    - List Renderer
    - Table Renderer
    - Tile Renderer
  - Advanced Renderers
    - Text Renderer
    - Title Renderer
    - Div Renderer
    - Div Renderer 2 x 1
    - Div Renderer 1 x 2
  - Plotr Renderer
    - Pie Chart
    - Bar Chart
  - SChart Graph Renderer
    - Pie Chart
  - Google Maps
    - Google Maps Locator
    - Google Maps Marker

jsonp (JSONP call)

JSONP Service URL: http://pipes.deri.org:8080/pipes/pipes/?id=icalsync2&f

JSONP callback parameter: cb

Tree | Refresh

- jsonp
  - items
    - [0]
      - location : Kongresszentrum (Congress Center) Karlsruhe, Germany
      - dtstart : 2008-10-26
      - label : 9DA0CBEB-49B7-4A8E-B625-F1EB922F83B1
      - summary : 7th International Semantic Web Conference
      - type : iCalRdf
      - uri : http://lab.gsi.dit.upm.es/apache/adam/swp/uc2.ics#9DA0CBEB-49B7-4A8E-B625-F
      - term_firstName : Fred
    - [1]
    - [2]
  - types
  - properties

Output: ${text} | Refresh

- Output

The nearest event you can

Experiments:

The mashup creation process

Goal: show upcoming event from personal calendar what friends can you meet there

How: analyze iCalendar instances

Experiments:

The mashup creation process

Data-level:

1. Convert iCal to RDF (where needed)

2. Filter out needed data (i.e. name, surname, event name, start date etc.) SPARQL Query

3. Mashup! (detect same events, people participating) SPARQL Constructs

4. Save and publish

Service-level:

1.Extract information from data level

2. Prepare the data before rendering and mashup again!

3. Render data (HTML/widget output)

Experiments:

Problems

- Still complex
- Data level fulfils most of the requirements but it takes most time and effort
- Often when needed something very particular -> **lack of operators again!**

Conclusions

- **Some thoughts**
- Big expensive environments from one vendor or integrate small/dedicated ones ?
- Data-level complexity vs. Service-level complexity
- Mashup output and construction Standardization ?

opemashup ALLIANCE

- **Future work**
- How about software development frameworks for creating mashups?