Semantic Web Application Patterns: Pipelines, Versioning and Validation

David Booth, Ph.D. (Consultant)
<david@dbooth.org>

W3C Linked Enterprise Data Patterns Workshop

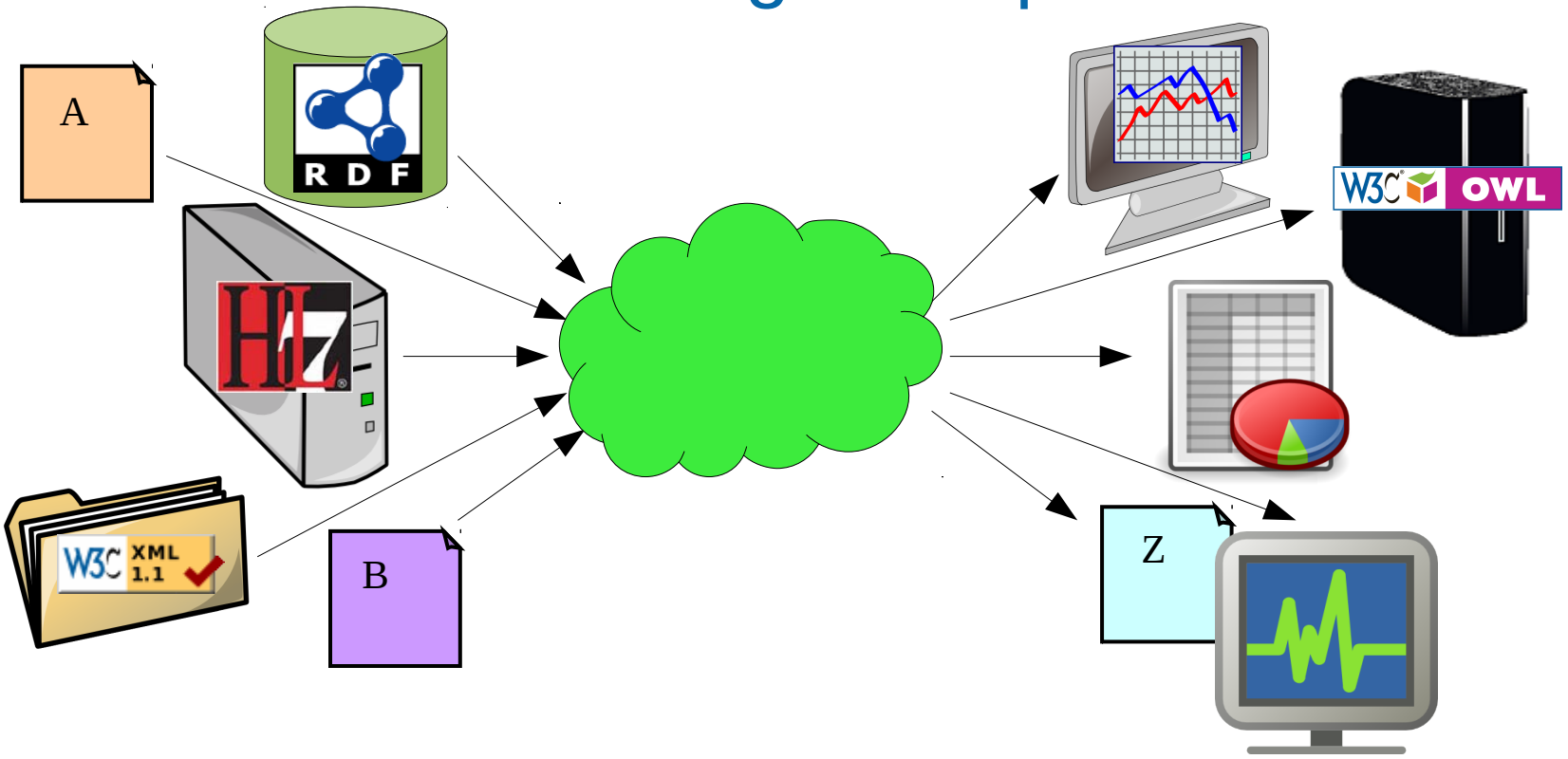
7-Dec-2011

Please download the latest version of these slides:
http://dbooth.org/2011/ledp/

Recent speaker background

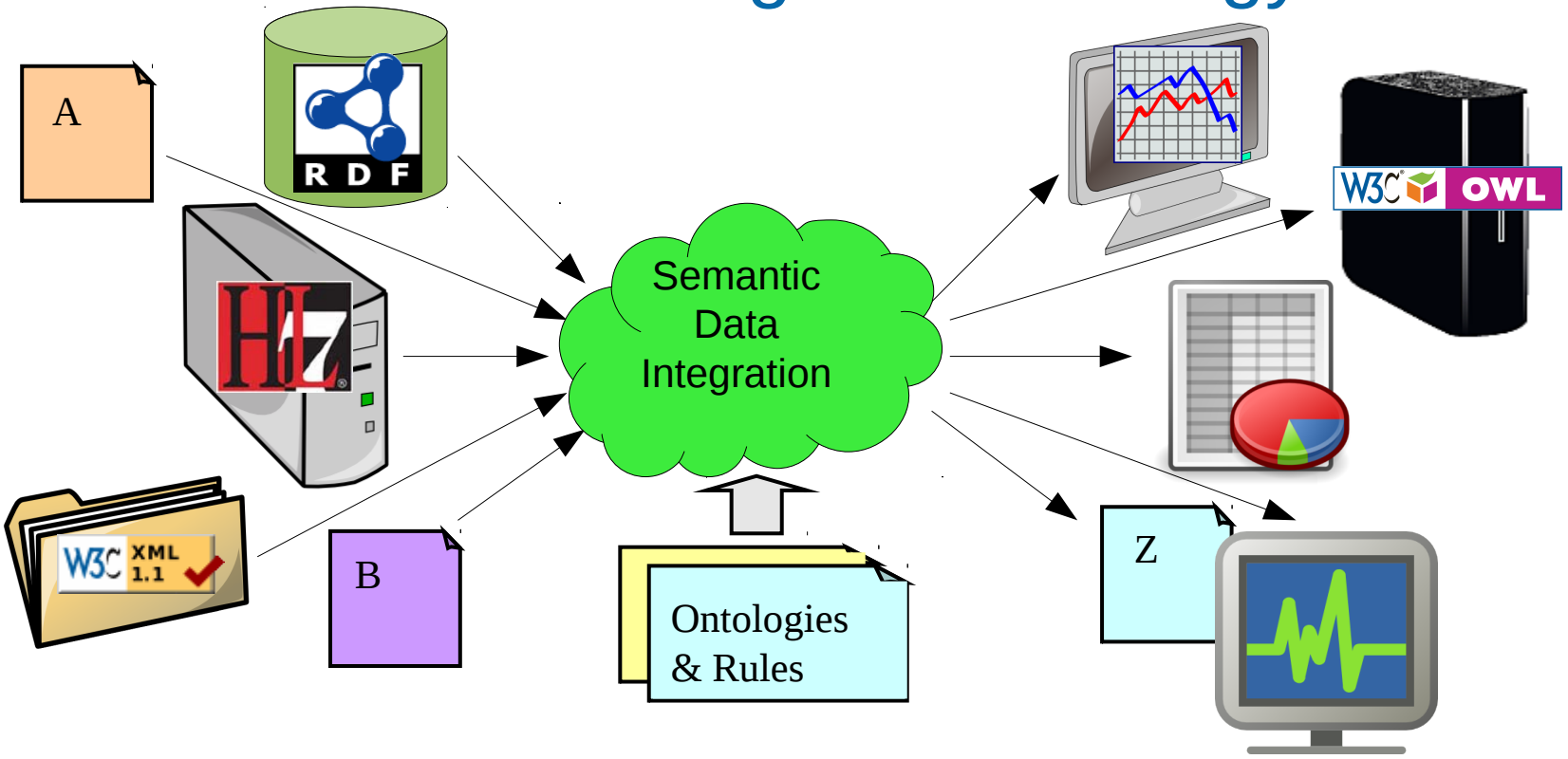
- **Cleveland Clinic**
  - **Using semantic web technology to produce data from patient records for outcomes research and reporting**
- **PanGenX**
  - **Enabling personalized medicine**

Semantic integration problem



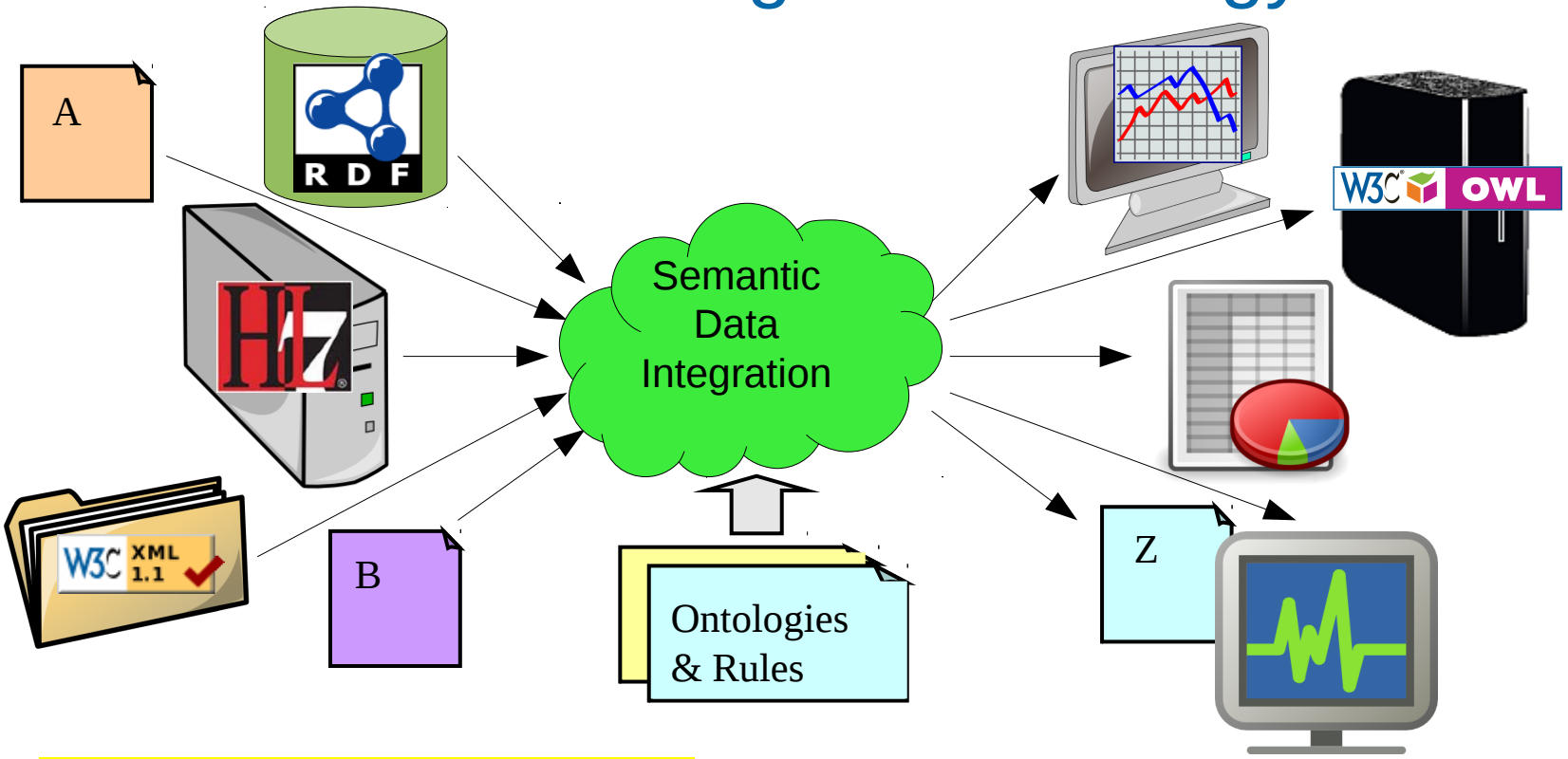
- **Many data sources, many applications**
- **Many technologies and protocols**
- **Each application wants the illusion of a single, unified data source**

Semantic integration strategy



1. **Data production pipeline**
2. **Use RDF in the middle; Convert to/from RDF at the edges**
3. **Use ontologies and rules for semantic transformations**

Semantic integration strategy

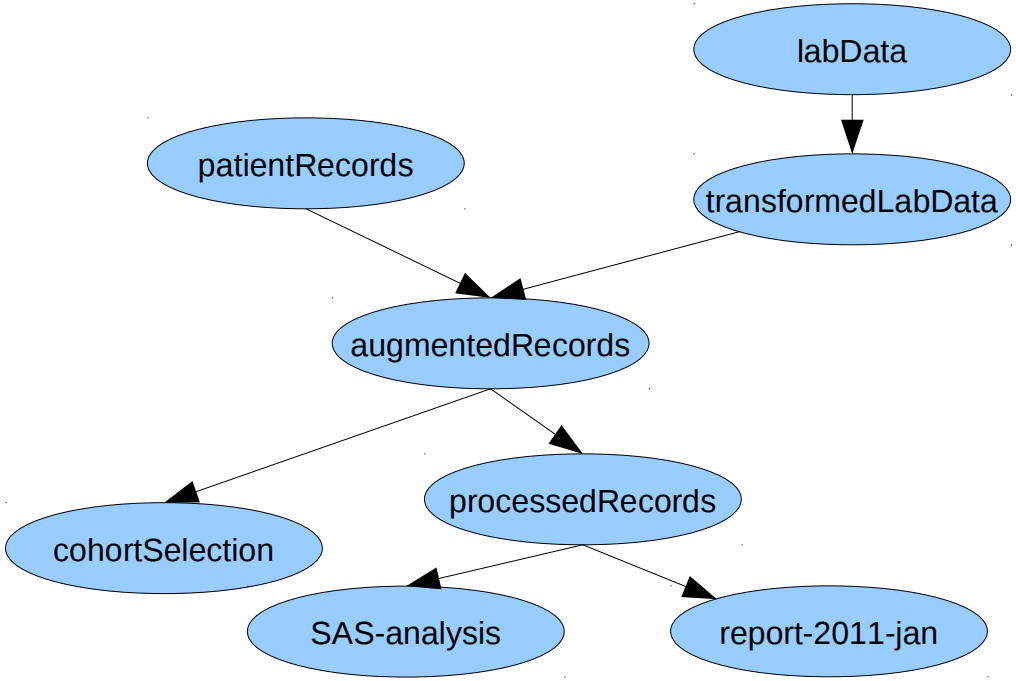


1. Data production pipeline

2. Use RDF in the middle; Convert to/from RDF at the edges

3. Use ontologies and rules for semantic transformations

Simplified(!) monthly report pipeline



- **Multiple data sources – diverse formats / vocabularies**
- **Multiple data production stages**
- **Multiple consuming applications**
  - **Overlapping but differing needs**

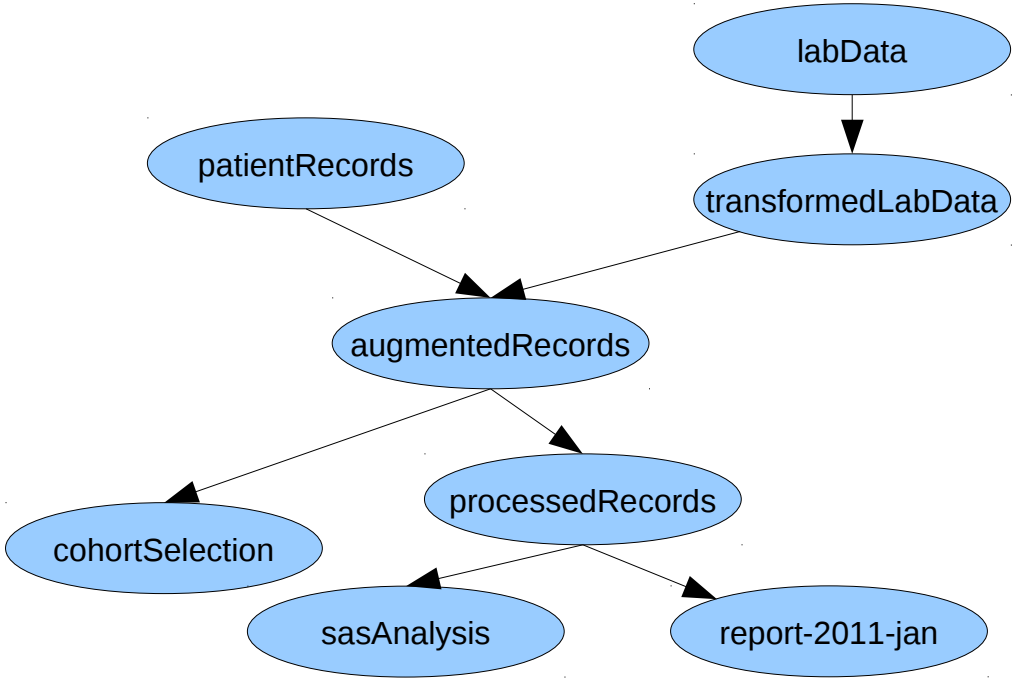
The job is not done after conversion to RDF!

- **Pipeline is still needed *within* RDF**
  - **Transforming between ontologies**
  - **Harmonizing the RDF**
  - **Inferencing**
- **Too inefficient to use one big monolithic graph**
  - **E.g., 200k patient records, 80M triples**
- **Pipeline can operate on named graphs**
  - **Easier to manage**
  - **Facilitates provenance**
  - **More efficient to update**
    - **E.g., each patient record is a graph**

RDF Pipeline framework

- **Open source project “RDF Pipeline”**
  - **http://code.google.com/p/rdf-pipeline/**
  - **Currently in POC**
- **Data production pipeline framework based on wrappers**
- **Pipeline of nodes is described in RDF**
  - **A data dependency graph**
- **Each node implements one processing stage**

Example pipeline . . .



. . . and RDF description

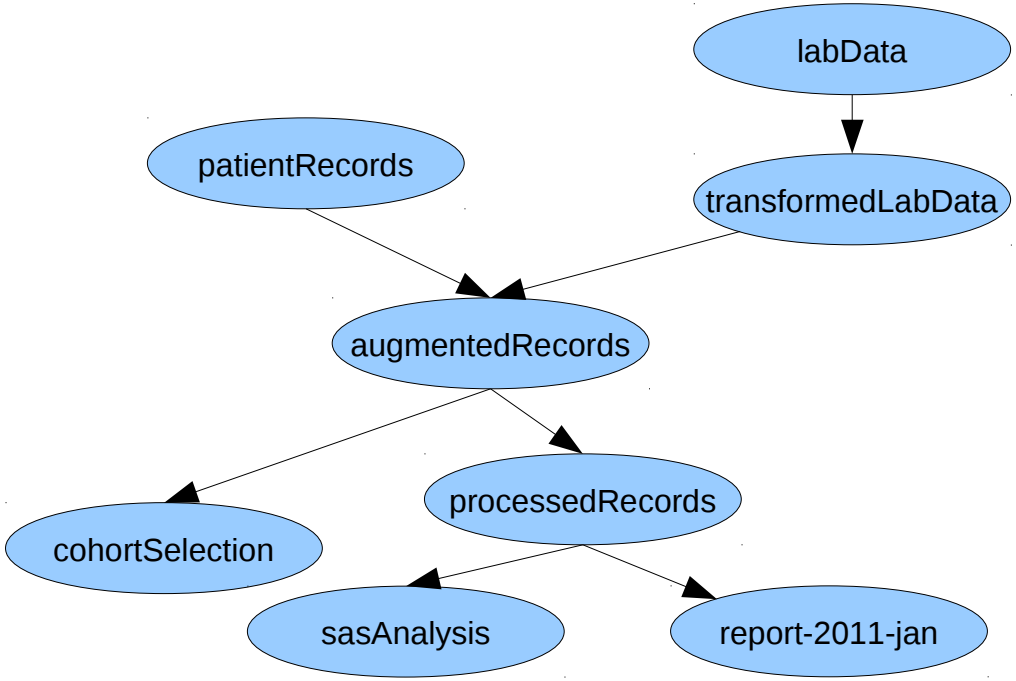
```
1.  @prefix p: <http://purl.org/pipeline/ont#> .
2.  @prefix : <http://localhost/> .
3.  :patientRecords a p:Node .
4.  :labData a p:Node .
5.  :transformedLabData a p:Node ;
6.          p:inputs ( :labData ) .
7.  :augmentedRecords a p:Node ;
8.          p:inputs ( :patientRecords :transformedLabData ) .
9.  :processedRecords a p:Node ;
10.         p:inputs ( :augmentedRecords ) .
11.  :report-2011-jan a p:Node ;
12.         p:inputs ( :processedRecords ) .
13.  :sasAnalysis a p:Node ;
14.         p:inputs ( :processedRecords ) .
15.  :cohortSelection a p:Node ;
16.         p:inputs ( :augmentedRecords ) .
```

How to use the RDF Pipeline framework

1. **Provide an updater for each node**
   - **Any language, any data (assuming a wrapper is available)**
   - **Any kind of processing**
   - **Generates the output of the node from its inputs**
2. **Put your updaters where wrappers can find them**
3. **Describe your pipeline in RDF**
   - **Inputs**
   - **Updaters**

Done!

Updater invocation



- **Data updates automatically propagate through the pipeline**
  - **Think “Make” or “Ant” – dependency graph**
- **Updater is run depending on node's updater policy**
  - **E.g., Lazy, Eager, Periodic, etc.**
- **Wrappers take care of this**

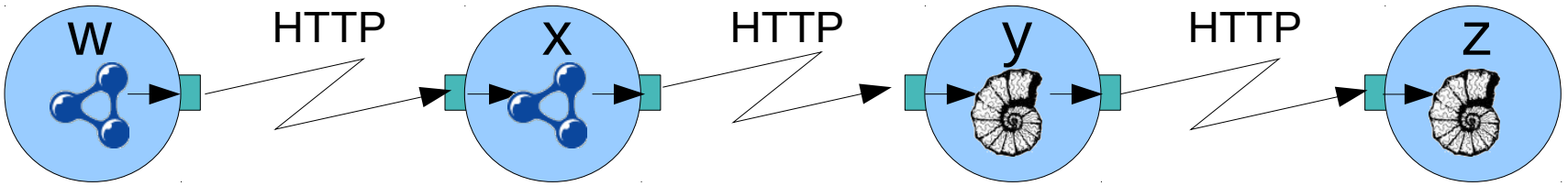
Example wrapper types

- **FileNode:**
  - **Invoked as a shell command**
  - **Inputs/output are files**
- **SparqlGraphNode:**
  - **Invoked as a SPARQL update**
  - **Inputs/output are named graphs**

Logical view - Inter-node communication

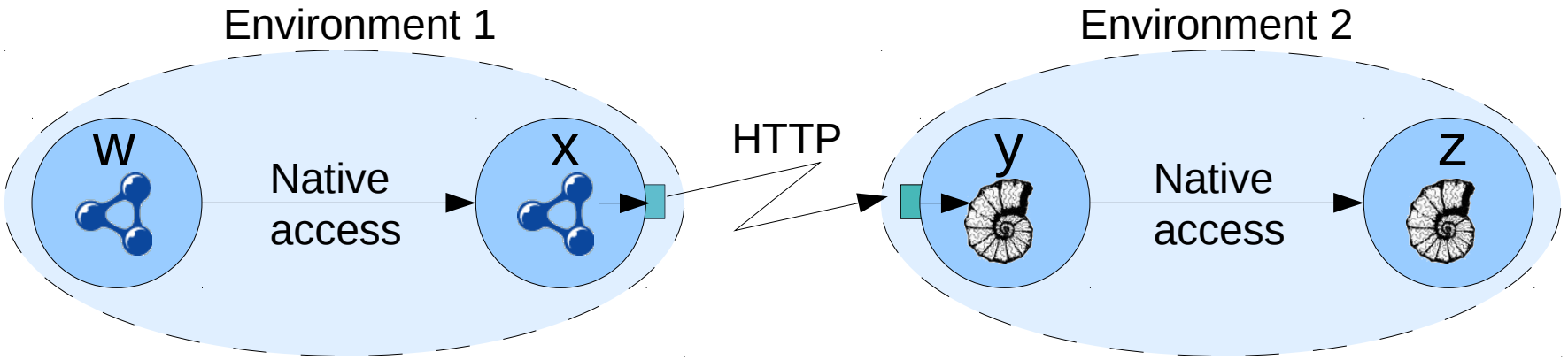
- **Nodes pass data from one to another . . .**

Physical view - Unoptimized



- **Wrappers handle inter-node communication**
- **By default, nodes use HTTP**

Physical view - Optimized

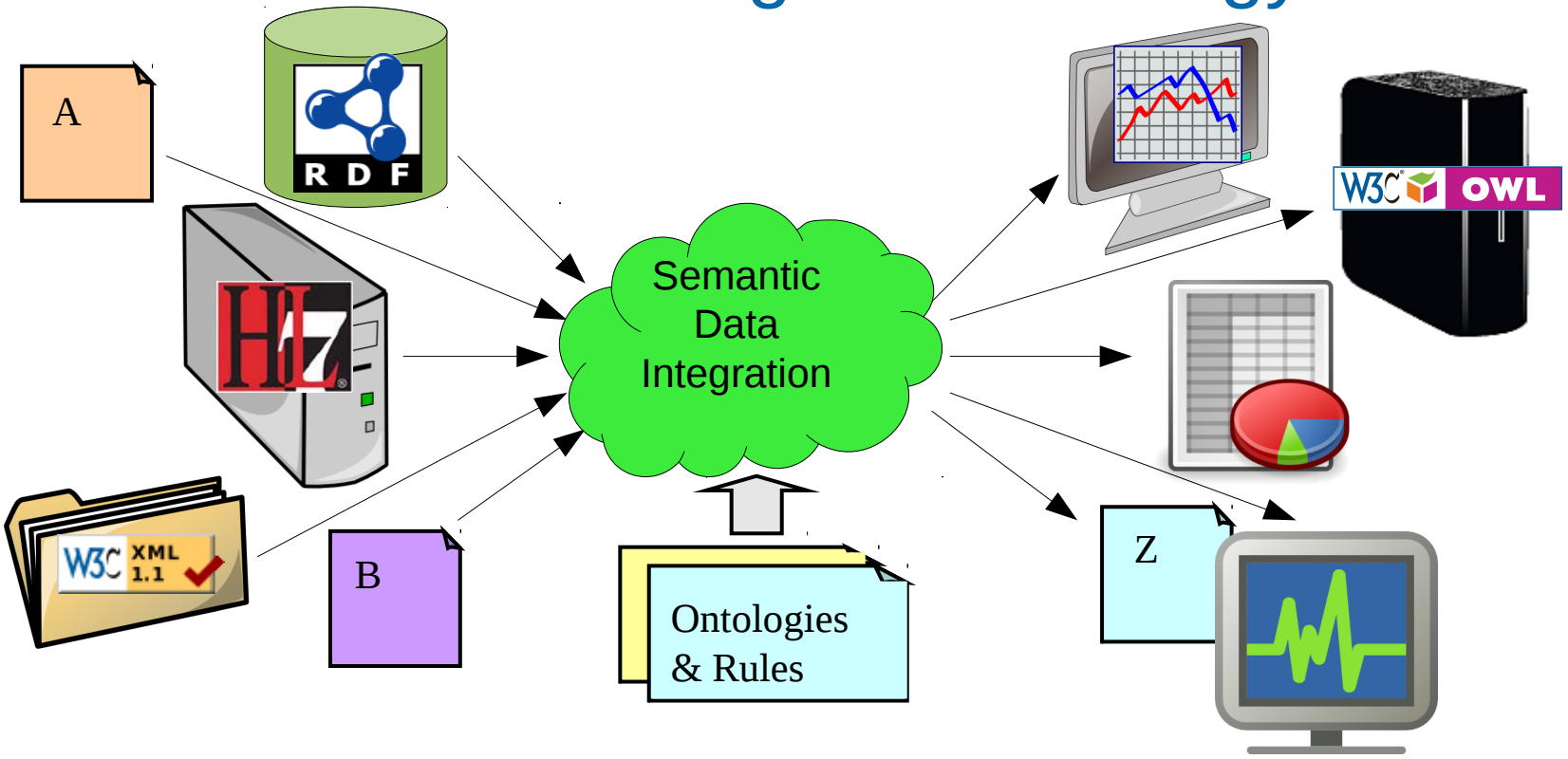


- **Nodes that share an implementation environment communicate directly, using native access, e.g.:**
  - **One SparqlGraphNode to another in the same RDF store**
  - **One FileNode to another on the same server**
- **Very efficient**

Why the RDF Pipeline framework?

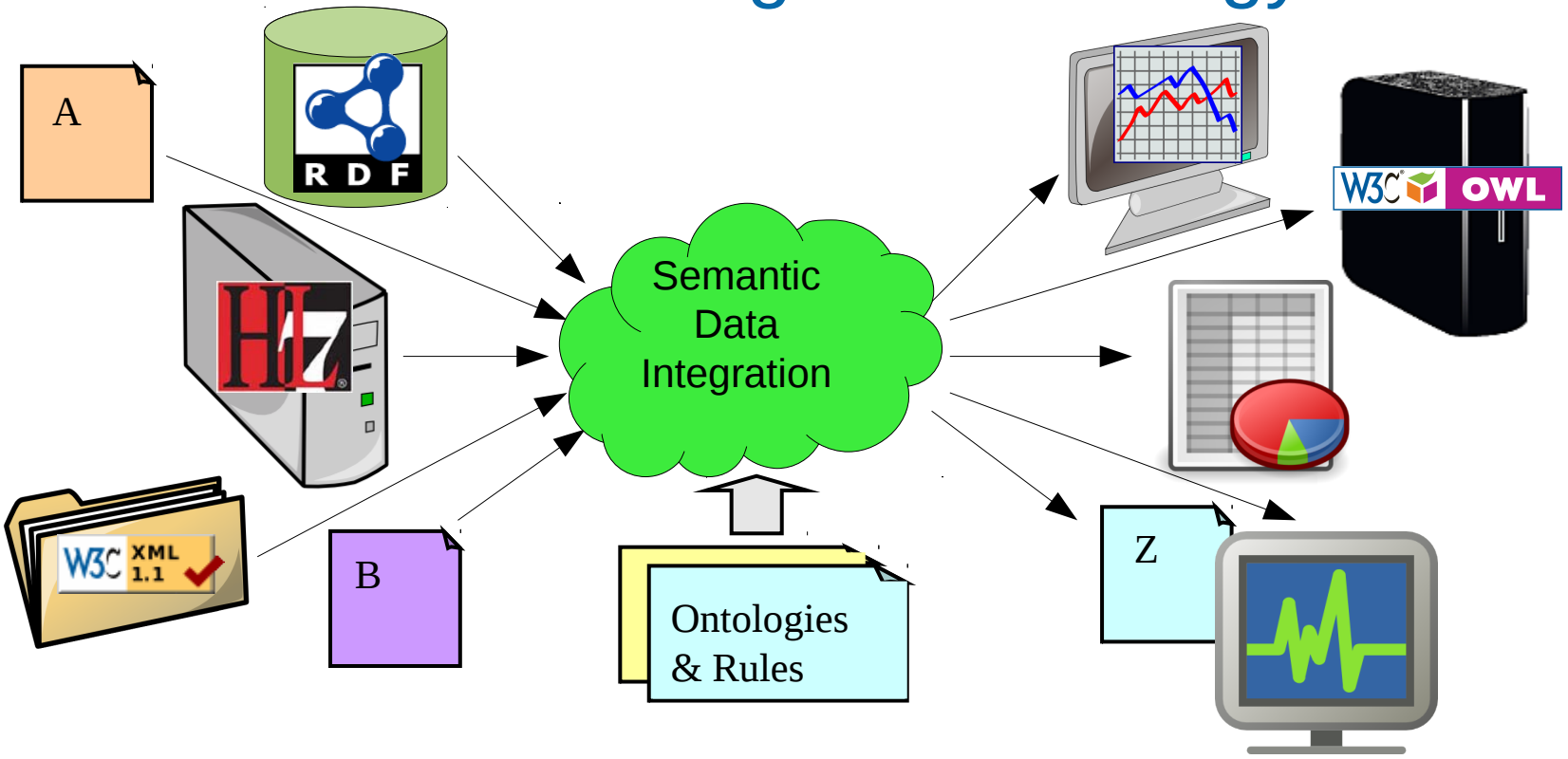
- **Easy to create & maintain**
  - **No API**
  - **Easy to visualize**
  - **Very loosely coupled**
- **Flexible**
  - **Data agnostic**
  - **Programming language agnostic**
- **Efficient**
  - **Decentralized**
  - **Data updates propagate automatically**

Semantic integration strategy



1. **Data production pipeline**
2. **Use RDF in the middle; Convert to/from RDF at the edges**
3. **Use ontologies and rules for semantic transformations**

Semantic integration strategy



1. **Data production pipeline**
2. **Use RDF in the middle; Convert to/from RDF at the edges**
3. **Use ontologies and rules for semantic transformations**

Pattern: SPARQL as a rules language

- **SPARQL can be used as a rules language**
  - **CONSTRUCT or INSERT**
  - **If the WHERE clause is satisfied, new triples are asserted**
- **Not recursive, but still convenient**
- **Simplifies development and maintenance**
  - **Same language as for queries**
- **INSERT is more efficient than CONSTRUCT**
  - **CONSTRUCT involves an extra client round-trip, as results are returned**
  - **INSERT operates directly within the RDF store**

Need for virtual graphs

- **Dynamic combination of named graphs**
- **E.g., if *myVirtualGraph* includes *graph1*, *graph2*, *graph3* then:**

```
SELECT . . .
FROM VIRTUAL myVirtualGraph
WHERE . . .
```

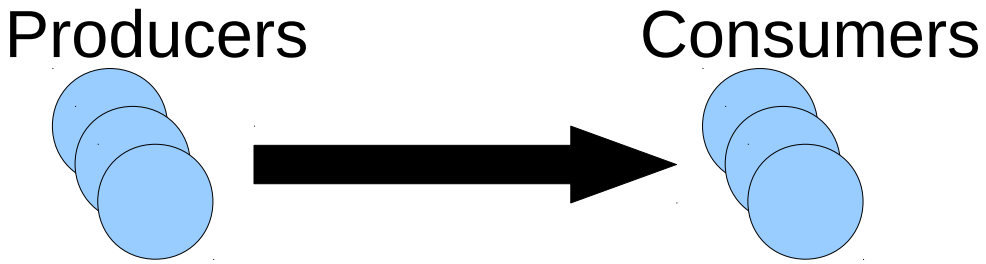
would be equivalent to:

```
SELECT . . .
FROM NAMED graph1
FROM NAMED graph2
FROM NAMED graph3
WHERE . . .
```

URI versioning

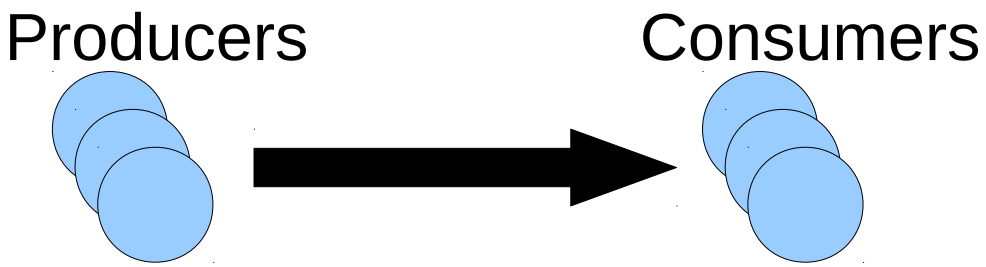
- **The dilemma: Change the URIs? Or change the semantics?**
  - **Changing URIs hurts apps that don't understand the new URIs**
  - **Changing semantics hurts apps that depended on stable semantics**
- **Point 1: Publish your URI versioning policy!**
- **Point 2: In RDF, old and new URIs can coexist peacefully**
  - **Data can use *both* old and new URIs**
  - **I.e., data can be monotonic**

Validation in the open world



- **Two roles: data producer and data consumer**
- **Multiple data producers, multiple consumers**
- **In RDF, extra data should not disturb existing data**
- **How to validate?**

Validation in the open world (cont.)



- **Two kinds of validation needed:**
  - **Model integrity (defined by the producer)**
    - **Does the data contain what the producer promised?**
  - **Suitability for use (defined by the consumer)**
    - **Does the data contain what this consumer expects?**
- **Each producer can supply a validator for data it provides**
- **Each consumer can supply a validator for data it expects**
- **SPARQL ASK can be used as validator language**

Questions?

Wrapper responsibilities

- **Inter-node communication**
  - **HTTP or native**
- **Node invocation**
  - **Per update policy**
- **Caching**
- **Serializing for HTTP transmission**

Wrapper responsibilities

- **Inter-node communication**
  - **HTTP or native**
- **Node invocation**
  - **Per update policy**
- **Caching**
- **Serializing for HTTP transmission**