⇔ http://bit.ly/lebo-ipaw-2012

Towards Unified Provenance Granularities

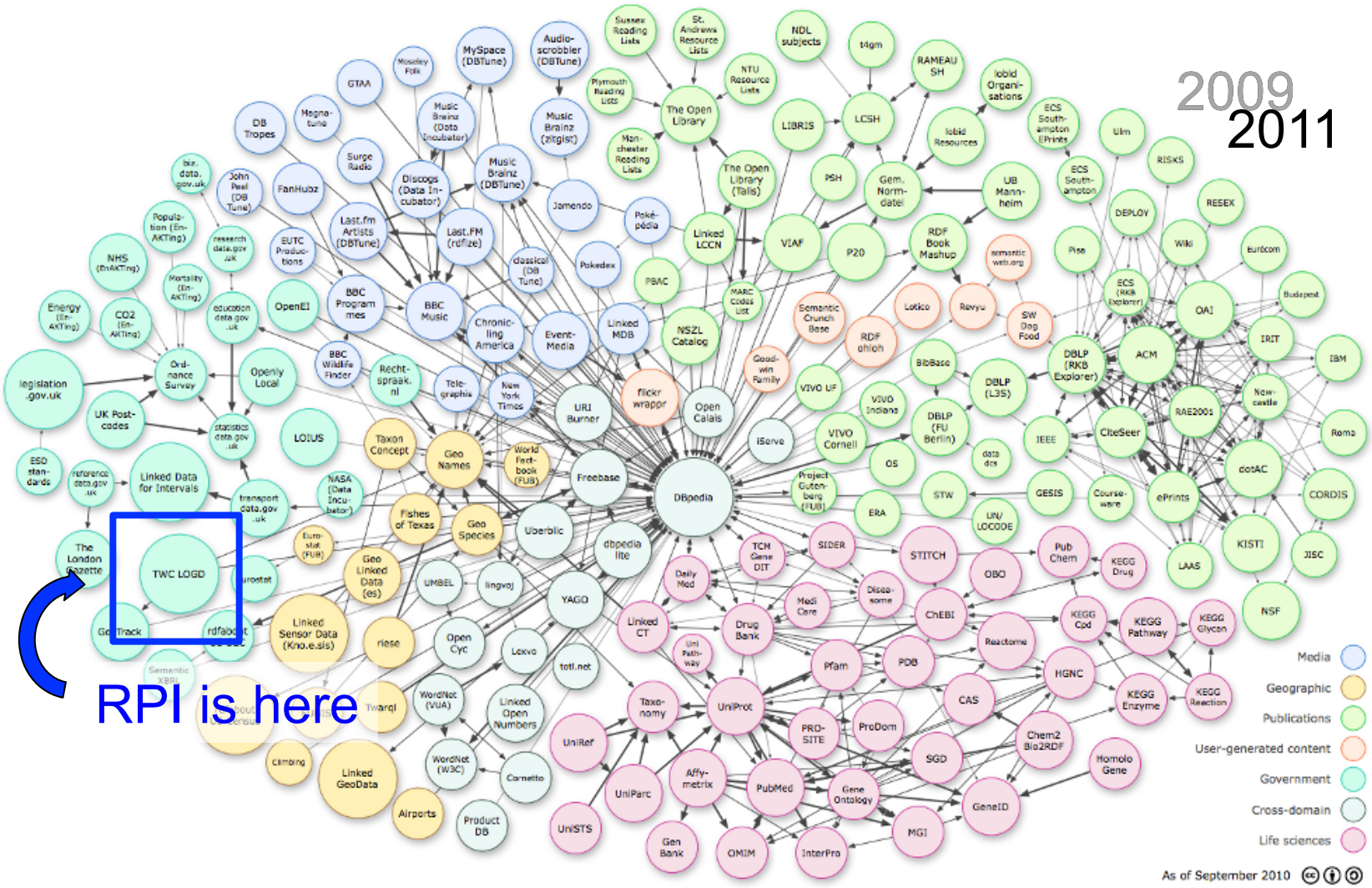
Timothy Lebo, Ping Wang, Alvaro Graves, Deborah McGuinness
Tetherless World Constellation
Rensselaer Polytechnic Institute

Rensselaer

Outline

- The Linked Open Data (LOD) cloud
  - … and RPI's little corner, LOGD.
- The Integrator's Dilemma
  - … and the need for provenance.
- Incongruent provenance granularity
  - … and the need for *the right* provenance.
  - (SemantAqua water quality portal)
- Our approach, applied
- Future work

Linked Open Data

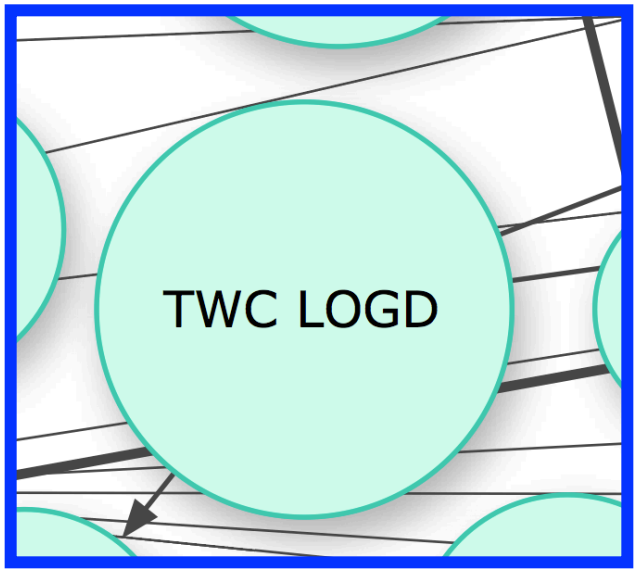


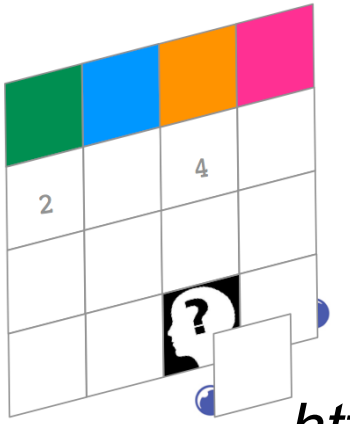
RPI is here

LOGD: Linking Open Government Data

http://logd.tw.rpi.edu

- 2,154 datasets from
- 61 source organizations =
- 10 billion triples.
- 710,000 datasets from 183 source organizations to go!





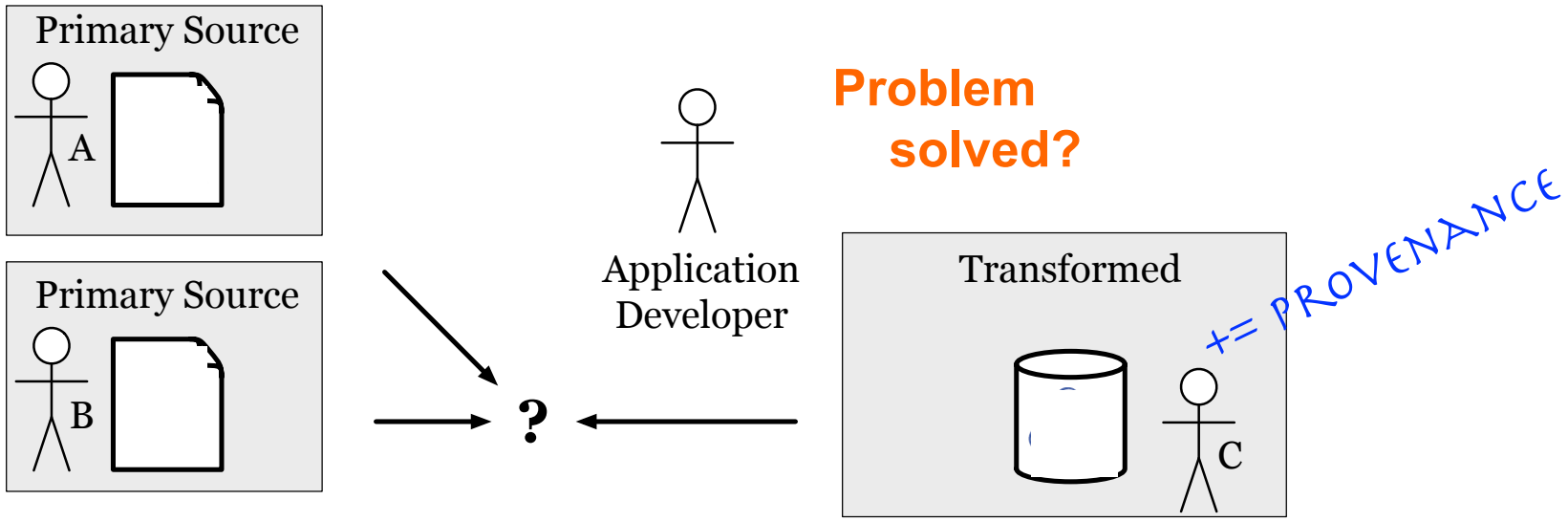
csv2rdf4lod is used for all LOGD data handling:

- Cataloging, Retrieval, Organizing, Conversion, Publishing

github SOCIAL CODING

*https://github.com/**timrdf**/csv2rdf4lod-automation/wiki*

The Integrator's Dilemma



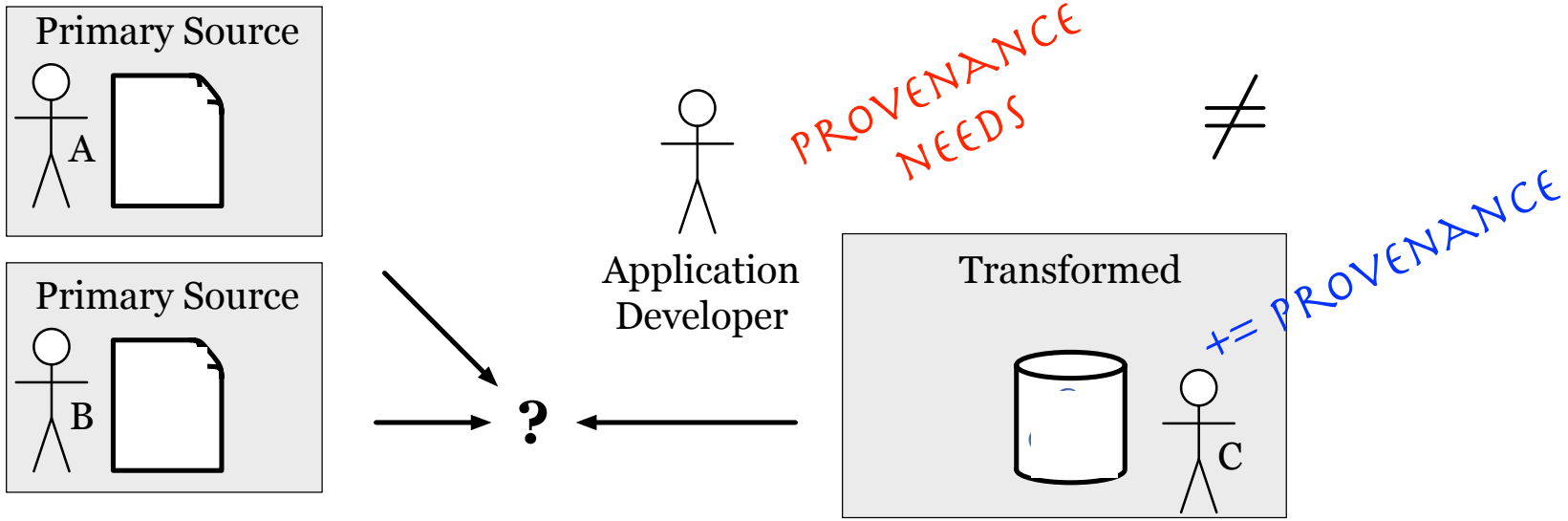
+ "Original"
+ Authoritative
+ Domain Expertise

- "Transformed"
- Non-Authoritative
- No Domain Expertise

- Not uniform structure
- Not linked

+ Uniform structure
+ Linked (RDF)

Incongruent Provenance Granularities



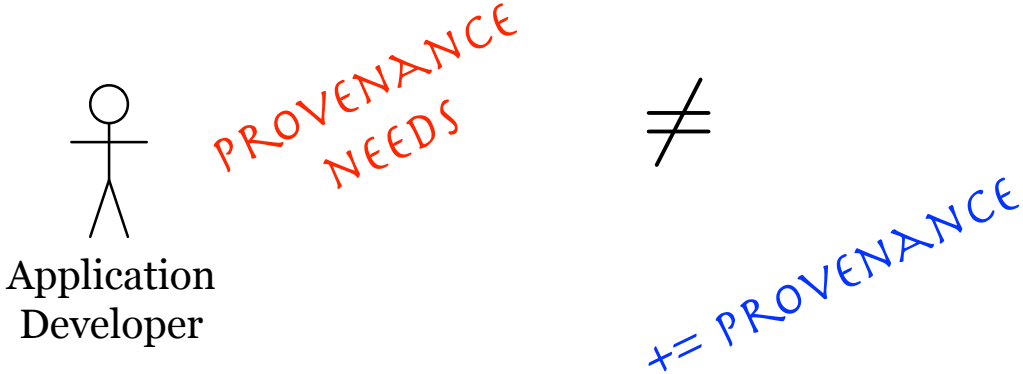
+ "Original"
+ Authoritative
+ Domain Expertise

- "Transformed"
- Non-Authoritative
- No Domain Expertise

- Not uniform structure
- Not linked

+ Uniform structure
+ Linked (RDF)

Incongruent Provenance Granularities



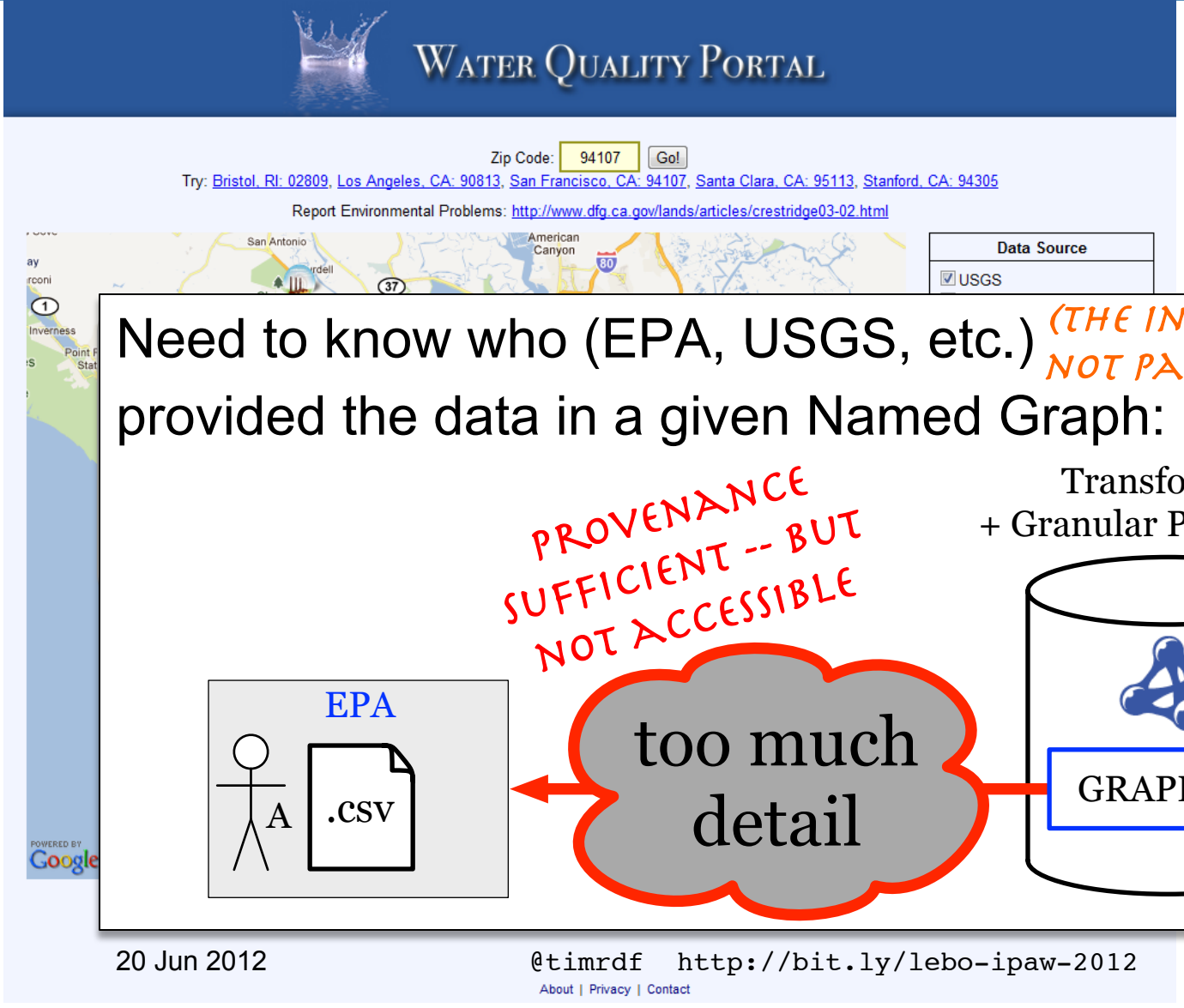
Tim's First Law of Producing Provenance:

> *For any provenance **record**,*
> *there exists a provenance **consumer** that will need it,*
> *but will not like how it is provided.*

∴ Dear provenance publishers,

> *You will never, ever, get it right.*

An example incongruence, with SemantAqua



EPA
USGS

Samples
Regulations

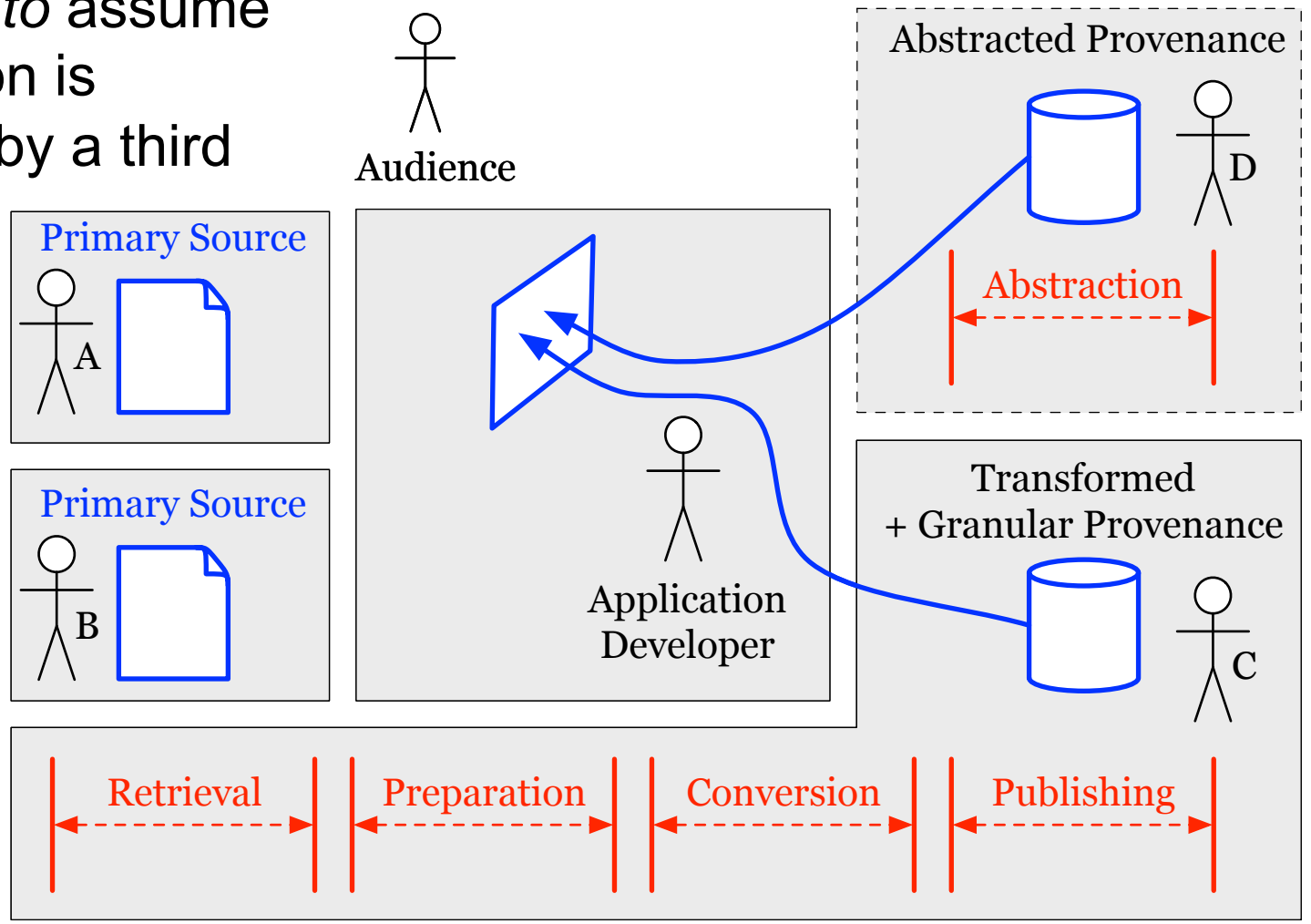
(Too much) provenance captured by csv2rdf4lod

- Retrieval
  - Agent*, URL, URL mod time, retrieval time, HTTP interactions, file checksum
- Preparation
  - Agent*, recipe, command
  - {mod time, checksum} x {input file, output file}
- Conversion
  - Agent*, start time, SW checksum and version, input filepath and URL, transform parameters, output filepath and URL
- Publishing
  - Agent*, source file URL <Retrieval>
  - destination named graph

* := Agent as person and user account

Approach: Overview

We *need to* assume abstraction is provided by a third party.
(T1LOPP)



Approach: 5 Steps

1. ***Define*** the *type of entity* whose provenance is required.
   e.g. a SPARQL endpoint's named graph
2. ***Define*** the *type of provenance* required.
   e.g. Who is attributed to the named graph's source?
3. Implement and ***deploy*** the independent service.
   e.g. <in: a named graph, out: its attribution>
4. [optionally] ***Find*** the service based on 1 and 2.

5. ***Accumulate**** results for the entities of interest.

* : with provenance!

Provenance Abstractions using the SADI Semantic Web Services Framework

```
rapper -g -o turtle http://sadiframework.org/examples/hello
```

HTTP GET service description

```
...
mygrid:inputParameter [
    mygrid:objectType
       <http://sadiframework.org/examples/hello.owl#NamedIndividual>;
    ... ];
mygrid:outputParameter [
   mygrid:objectType
    <http://sadiframework.org/examples/hello.owl#GreetedIndividual>;
   ... ];
...
```

SADI

Find. Integrate. Analyze.

sadiframework.org

```
...                                    # send.ttl
<http://purl.org/twc/instances/TimLebo>
  a hello:NamedIndividual;
  foaf:name "Tim" .
```

Steps:
1 input
2 output
3 deploy
4 discover

```
curl -d @send.ttl.rdf http://sadiframework.org/examples/hello
```

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:hello="http://sadiframework.org/examples/hello.owl#">

  <hello:GreetedIndividual rdf:about="http://purl.org/twc/instances/TimLebo">
    <hello:greeting rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Hello, Tim!</hello:greeting>
  </hello:GreetedIndividual>
</rdf:RDF>
```

HTTP POST for abstraction

Step 5: Abstract it!

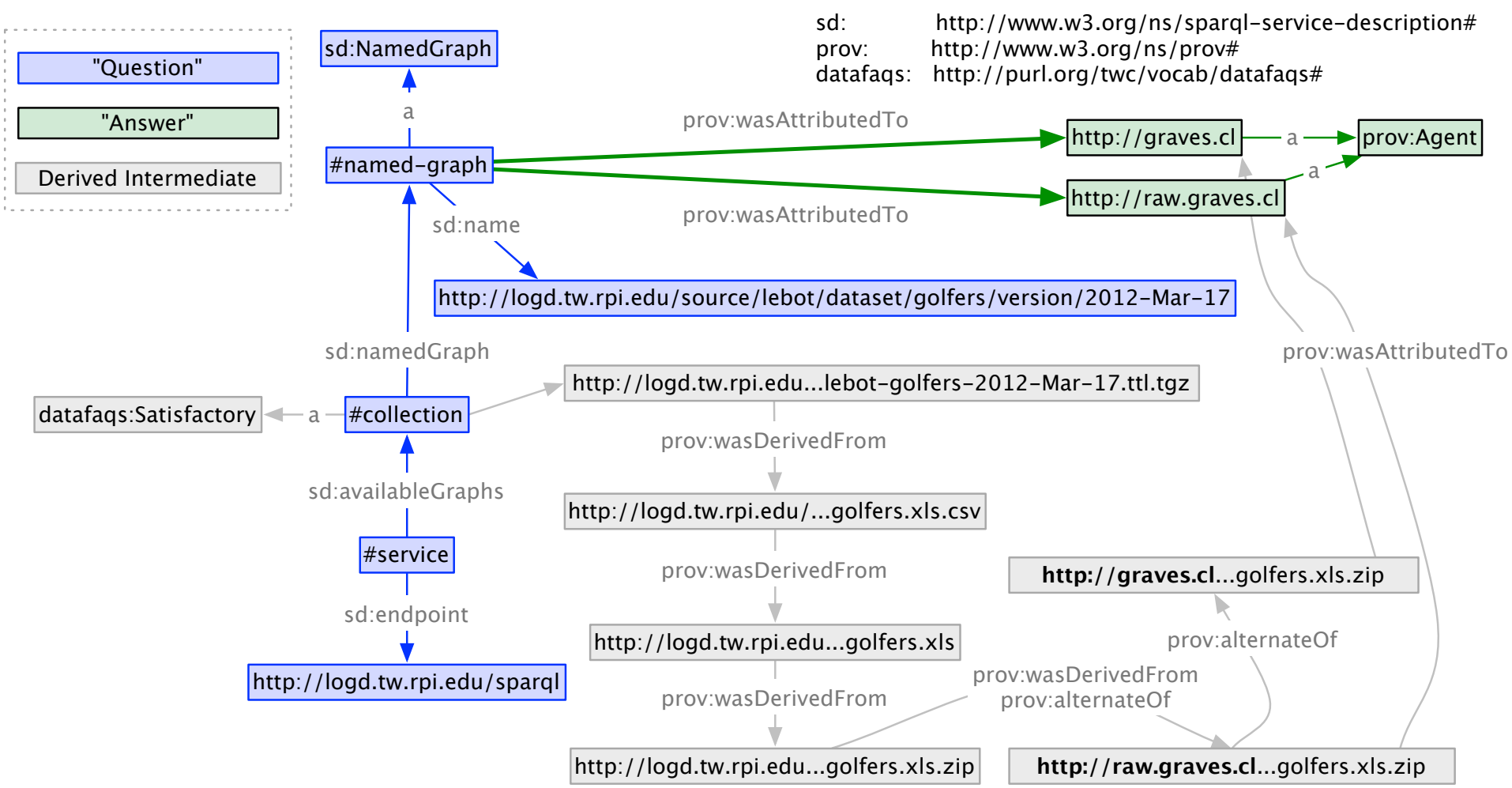
```
@prefix sd:        <http://www.w3.org/ns/sparql-service-description#> .

:service
   a sd:Service;
   sd:endpoint <http://logd.tw.rpi.edu/sparql>;
   sd:availableGraphs :collection;
.

:collection
   a sd:GraphCollection, dcat:Dataset;
   sd:namedGraph :named-graph;
.

:named-graph
   a sd:NamedGraph;
   sd:name   <http://logd.tw.rpi.edu/source/lebot/dataset/golfers>;
   sd:graph :graph;
.

:graph a void:Dataset .
```

Getting the abstraction is one HTTP POST

curl -H "Content-type: text/turtle" -d @golfers.ttl

http://aquarius.tw.rpi.edu/projects/datafaqs/services/sadi/faqt/provenance/named-graph-derivation

Step 5: The abstraction we need



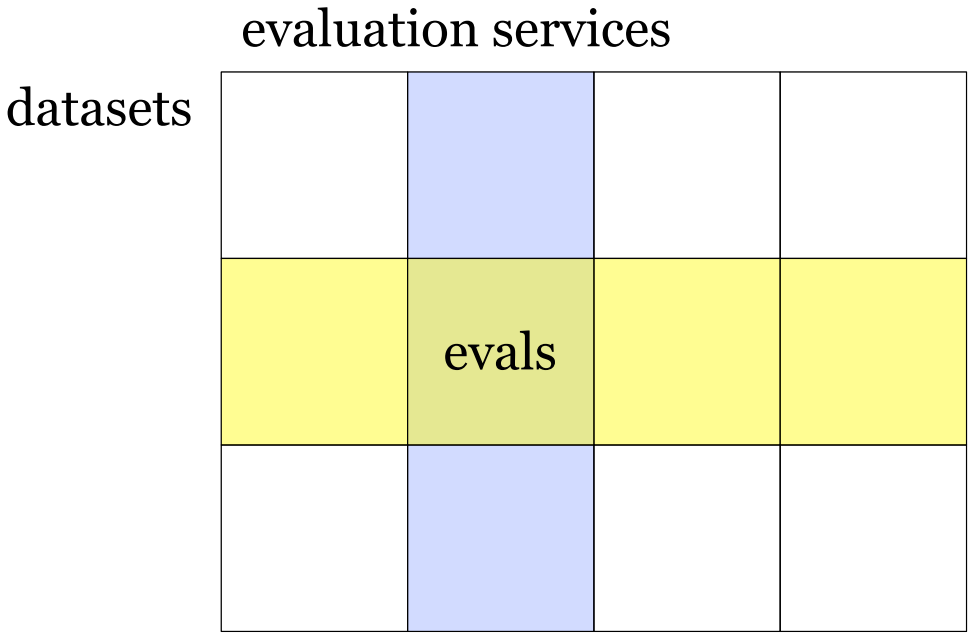
Approach: 5 Steps

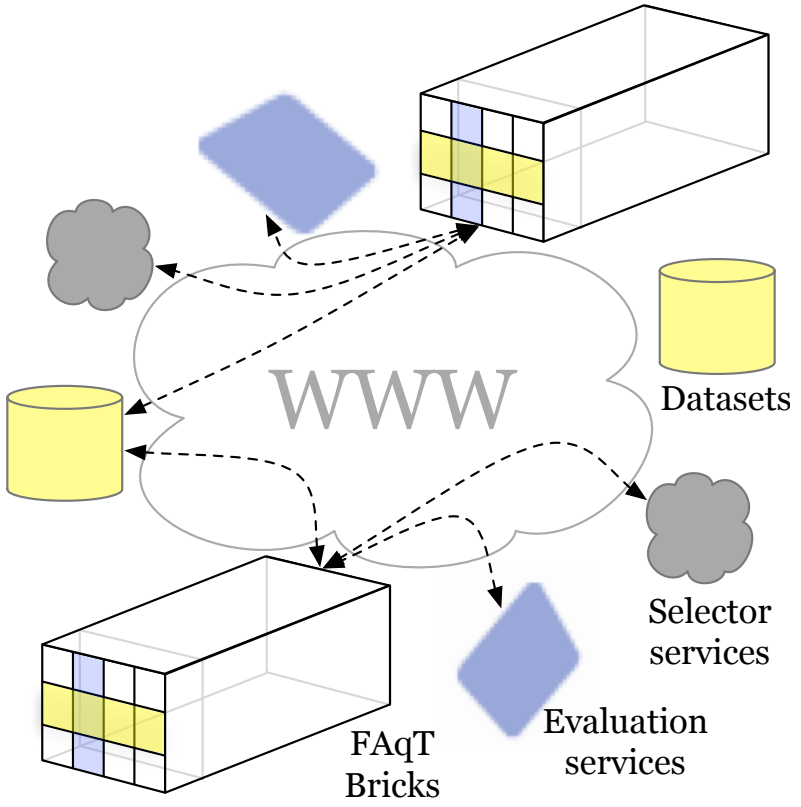
1. ***Define*** the *type of entity* whose provenance is required.
   e.g. a SPARQL endpoint's named graph
2. ***Define*** the *type of provenance* required.
   e.g. Who is attributed to the named graph's source?
3. Implement and ***deploy*** the independent service.
   e.g. <in: a named graph, out: its attribution>
4. [optionally] ***Find*** the service based on 1 and 2.

5. ***Accumulate**** results for the entities of interest.

* : with provenance!

Accumulating provenance abstractions

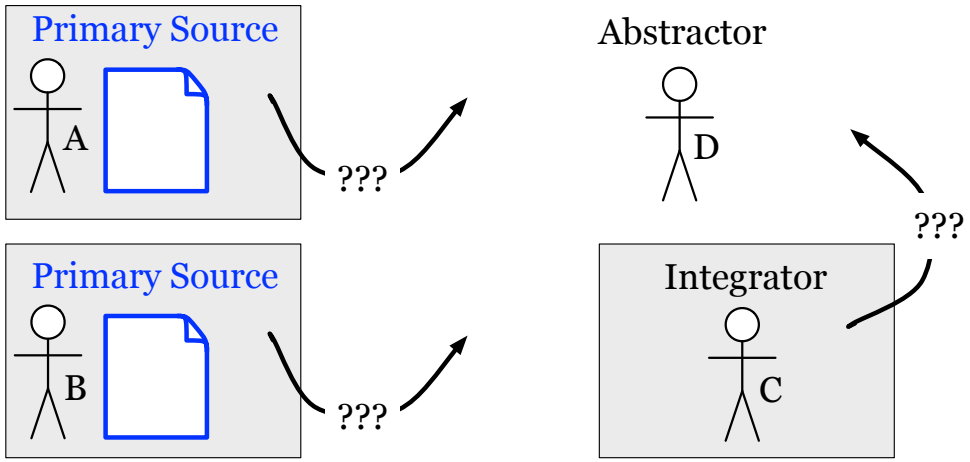




*https://github.com/**timrdf**/DataFAQs/wiki*

Future Work

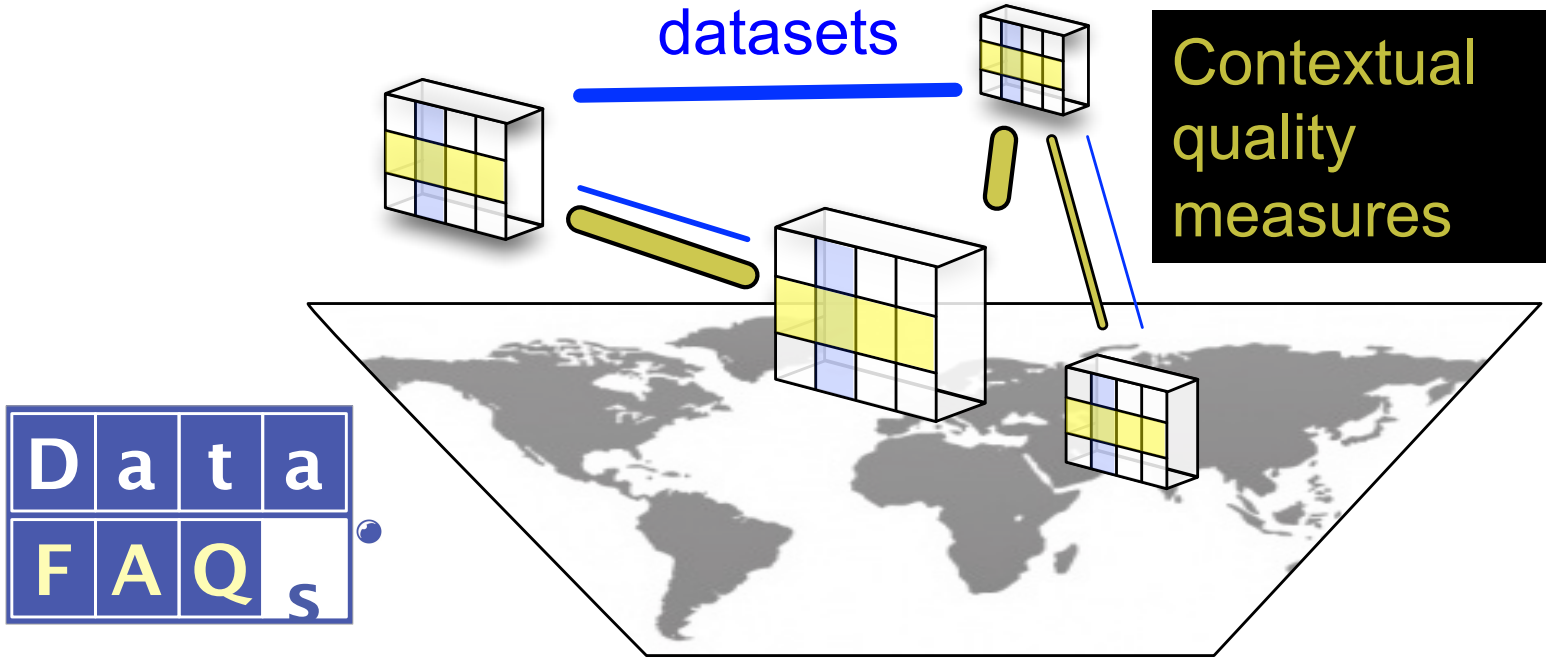
- Provenance ping-back



- Explore uses of provenance of abstract provenance (by abstracting it?)

Future Work

- Apply more generally to semantic web computation
- Automated, uncoordinated, asynchronous feedback to provenance [data] publishers



Conclusion

- The Integrator's Dilemma
- Incongruent provenance granularity

Tim's First Law of Producing Provenance:

> *For any provenance **record**,*
> *there exists a provenance **consumer** that will need it,*
> *but will not like how it is provided.*

- Applying abstraction services as a third party
- Using DataFAQs and provenance ping backs

Thanks!

- Ping Wang
- Alvaro Graves
- Jim McCusker
- Dominic DiFranzo
- Yu Chen
- Josh Shinavier

Deborah McGuinness