Target Vocabulary Maps

An Unreasonable Application of OWL

Niklas Lindström

KUNGL. BIBLIOTEKET

Wait…

Before we begin…

How Long Is The Coast Of Britain?

Unit = 200 km,

length = 2400 km

Unit = 50 km,

length = 3400 km

https://en.wikipedia.org/wiki/How_Long_Is_the_Coast_of_Britain%3F_Statistical_Self-Similarity_and_Fractional_Dimension

How Far is Sweden

From the Semantic Web?

Libris XL: the core of a new generation of Libris systems

In June 2018, XL went into production at the National Library Of Sweden, replacing the old MARC21 system with one based on **Linked Data**, and building upon BIBFRAME.

https://libris.kb.se/find.jsonld

Two Perspectives: Logical

Everything is **RDF**.

Interlinked entities with structured (bnode) or literal property values.

The Vocabulary (Ontology) is core:

- Type and Property Hierarchies
- Domains and Ranges
- Restrictions

Two Perspectives: Technical

Everything is **JSON-LD** internally.

Used as “just JSON” in code, storage and for indexing. In all layers alike.

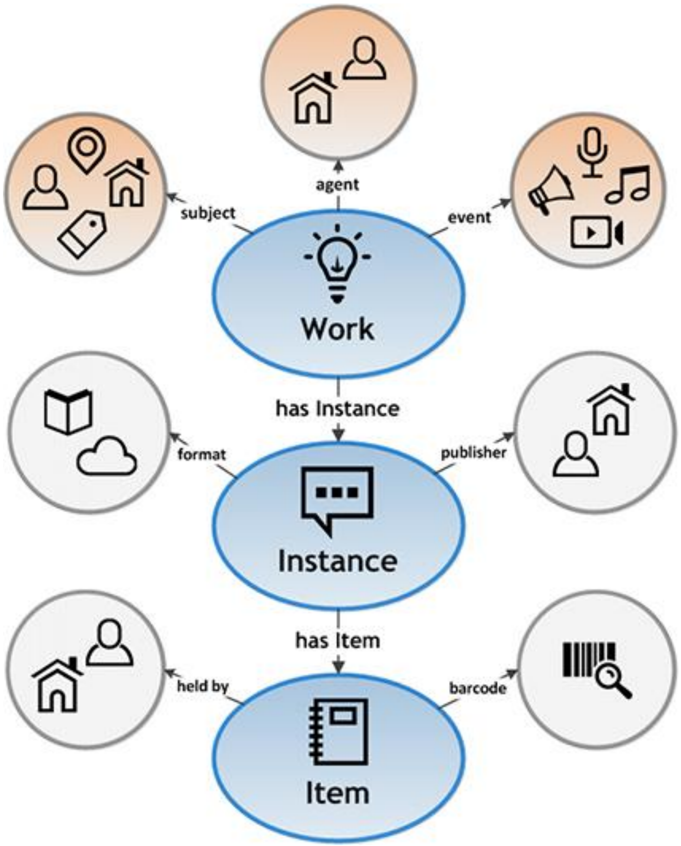
Conforming to the semantics.

All public features (editing, searching, displaying) *comply with the logical perspective*.

BIBFRAME 2.0

The model created by LC to replace MARC21.

We formally decided to **align** our model with BF2 in 2017.



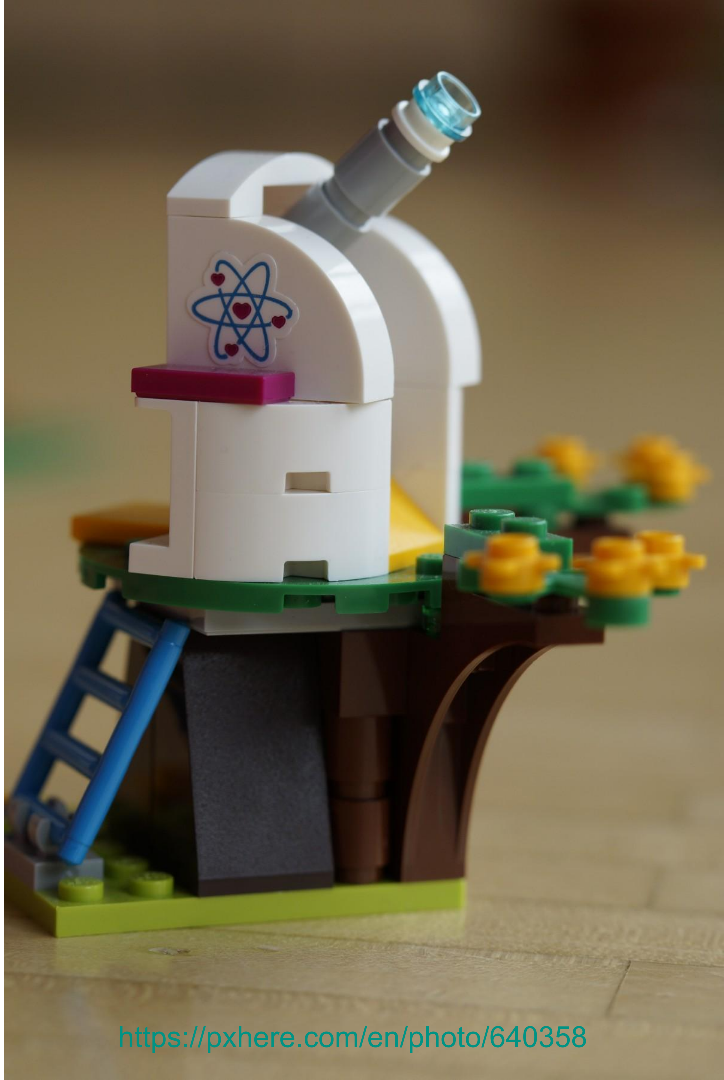
https://www.loc.gov/bibframe/docs/bibframe2-model.html

Not Just* BIBFRAME 2

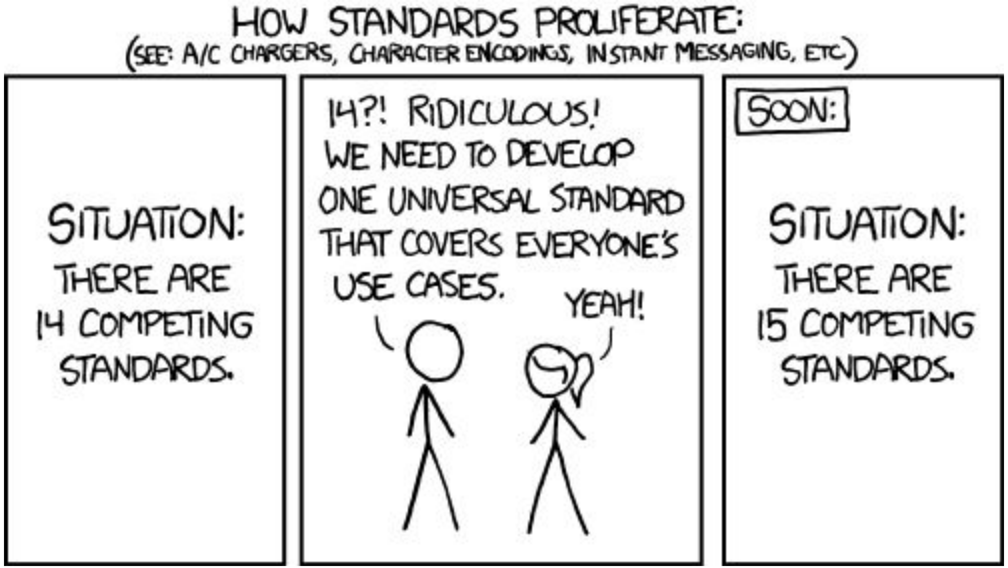
KBV: Our local **Application Vocabulary**, for our specific needs.

It has a core of **BIBFRAME 2** *equivalencies* (+ some of RDA, SKOS/MADS, Schema.org where needed).

* = Not *actually*? This depends on how someone reads/*interprets* our data…



https://pxhere.com/en/photo/640358



https://xkcd.com/927/

First Mapping Attempt: JSON-LD Contexts

Currently deployed.

Generate a JSON-LD context from our internal application vocabulary; **using its RDFS and OWL assertions**.

Maps “our” terms to “public” terms.

Shortcomings:

- Only ONE target (a potpourri of prefixes)!
- Only works on the exact same level of granularity!

https://github.com/libris/lxltools/blob/0.8.1/lxltools/contextmaker.py

https://libris.kb.se/context.jsonld

Various Shapes of Usable

BF2:

```
</instance/a>
    bf:identifiedBy [ a bf:Isbn ;
        rdf:value "12-3456-789-0" ] ;
    bf:provisionActivity [ a bf:Publication ;
            bf:agent </org/a> ;
            bf:date "2017" ] .
```

Schema.org:

```
</instance/a>
    schema:isbn "12-3456-789-0" ;
    schema:publisher </org/a> ;
    schema:datePublished "2017" .
```

From Ridgid & Fragile to Robust & Flexible

Currently

We only accept data conforming to our internal shape.

“Just JSON”. By being JSON-LD, it *is* RDF. We just don’t use all the flexibility yet.

Goal

Use our RDF vocabulary mappings to enable a richer I/O system.

On the vocabulary, data granularity and identity matching levels alike.

What Links Enable

Open World Assumption = There's More To Know

Case: Enriching from Wikidata

Locally: cooperate nationally with libraries, agencies etc. *using* LD.

Globally: LoC, DNB, BnF, ORCID, ISSN...

Requires:

- **Shapes & Mappings**

Geografiskt ämnesord • Hedeby (övergiven stad) • Övergivna städer

FÖREDRAGEN BENÄMNING	Hedeby (övergiven stad)
BREDARE	› Geografiskt ämnesord: Övergivna städer
VARIANT	› Geografiskt ämnesord: Haithabu (Extinct city) › Geografiskt ämnesord: Hedeby
MARC:HASADDEDENTRYGEOGRAPHICNAME	› Marc:AddedEntryGeographicName: {Namnlös}
SAMMA SAK SOM	resource/auth/345651
EXAKT MATCH [WIKIDATA]	https://www.wikidata.org/entity/Q165414
ALTERNATIVNAMN [WIKIDATA]	Гаддеби *(ru)* Слисторп *(ru)* Хэдебю *(ru)* Haithabu *(es)*
BESKRIVNING [WIKIDATA]	◦ ciudad de Dinamarca *(es)* ◦ bedeutende Siedlung dänischer Wikinger bzw. schwedischer Waräger *(de)* ◦ city *(en)* ◦ stad in Duitsland *(nl)* ◦ פיורד בגרמניה *(he)* ◦ ドイツの都市 *(ja)* ◦ qytet në Gjermani *(sq)*
KOORDINATER [WIKIDATA]	Point(9.5652777777778 54.491111111111)
BILD [WIKIDATA]	

Different Data Shapes? ETL or …?

How to do RDF(!) Alignments in Production?

A. Code them as needed? Extract, transform and load, using SPARQL Constructs (or XSLT, or GraphQL, or AWK/Perl/Python…)

   How to specify selection of vocabulary terms? Handling granularities?

B. Declare them! Make Vocabulary Maps, just using RDFS and OWL?

   Then, that can be used by generic code. Though Inference mightn't be the One Solution™!

Meanwhile

On The “Semantic” Web

No Reason In Sight

The myth of OWL inferencing...

Schema.org (some assertions, yes); but consumption by Google? NO!

DCAT-AP? NO!

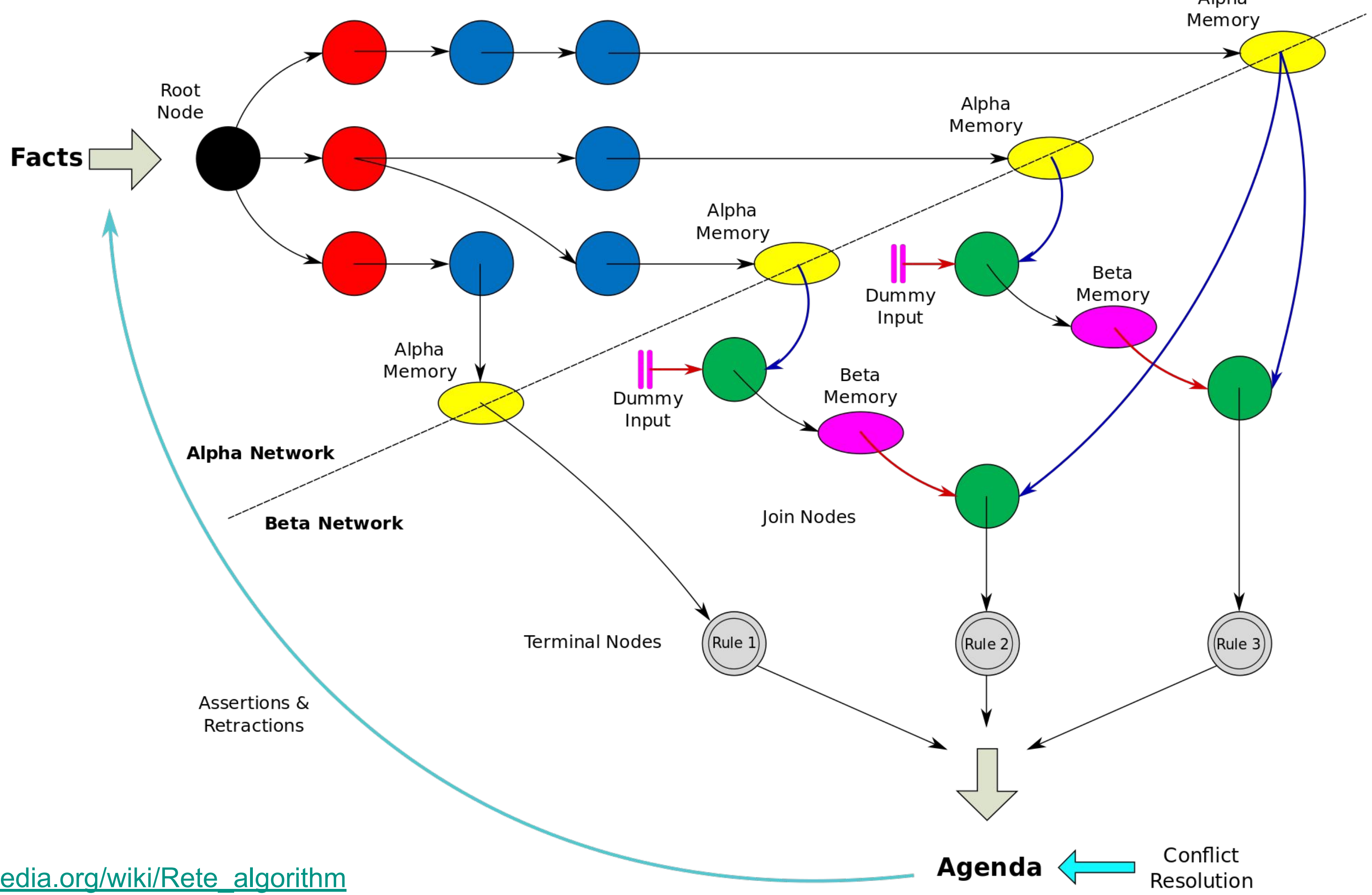
LC:s BIBFRAME 2? NO!

Linked Data regularly means using RDF as a “raw” data format. For better and worse.

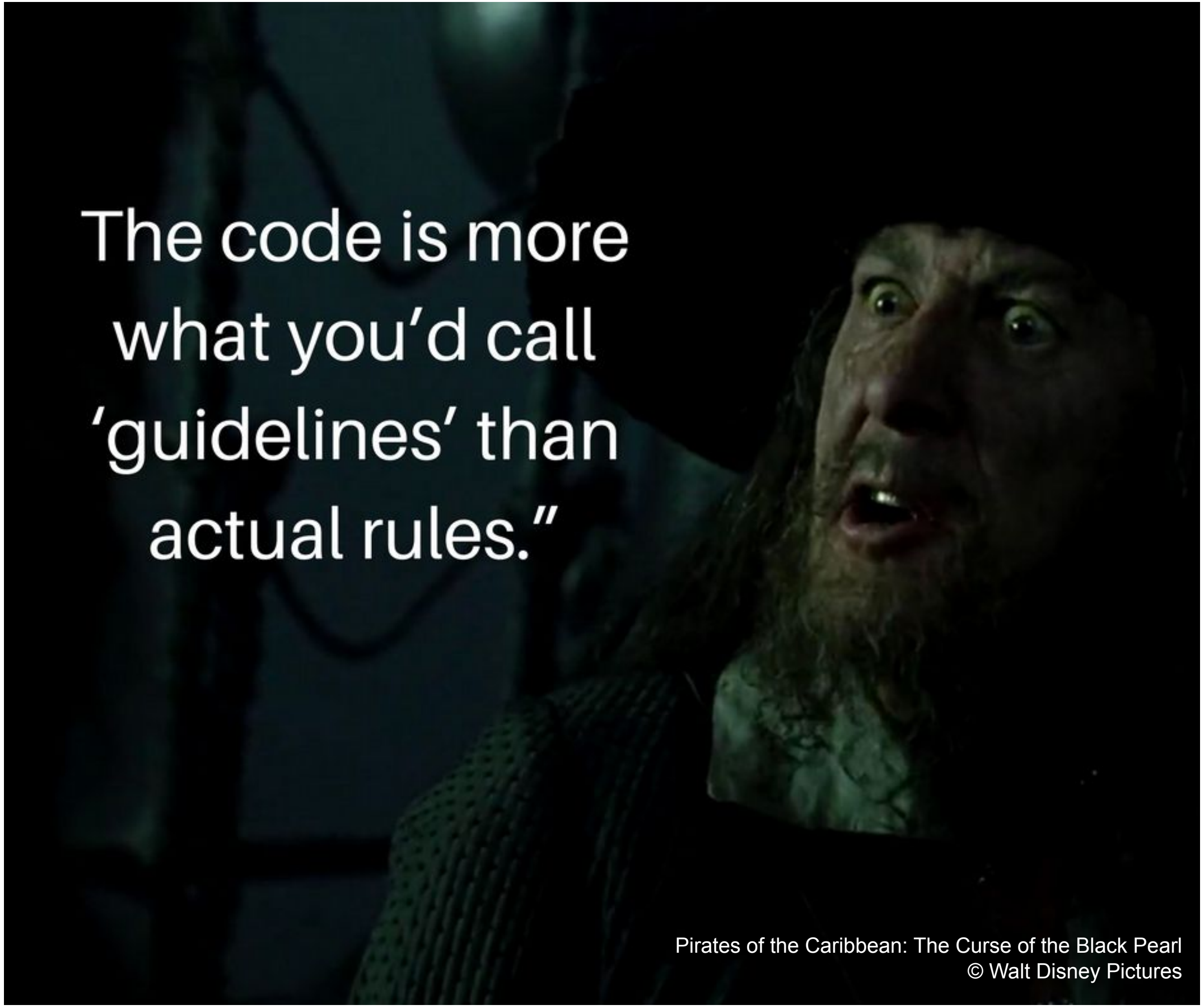
WHY?

Nobody follows the rules?

Rete



https://en.wikipedia.org/wiki/Rete_algorithm



Pirates of the Caribbean: The Curse of the Black Pearl

RDFS & OWL as Mapping Tools

```
rdfs:subPropertyOf
rdfs:subClassOf
owl:equivalentProperty
owl:equivalentClass

owl:propertyChainAxiom

rdfs:domain
rdfs:range

rdf:Statement
```

Target Vocabulary Maps

Proof Of Concept

For every term in the ***target* vocabulary, or selection of terms**, find all possible paths from all known vocabularies to that term, by following the RDFS and OWL mapping assertions.

- Paths include property chain axioms with range-restricted subproperties.
- Statement-like entities can provide direct predicates from e.g. qualified events.

This is stored as a **structured lookup table**, used by a very *simple* algorithm to map from source type or predicate(s) to target type or predicate(s).

Code: https://github.com/niklasl/ldtvm/ (< 300 LOC including curlies)

Mapping Targets

Source

```
prefix : <http://id.loc.gov/ontologies/bibframe/>

</work/a> a :Print ;
    :carrier lccarrier:nc ;
    :instanceOf </abstract/a> ;
    :identifiedBy [ a :Isbn ;
                    rdf:value "12-3456-789-0" ] ;
    :provisionActivity [ a :Publication ;
            :agent </org/a> ;
            :date "2017" ] .

</abstract/a> a :Text ;
    :content :Text ;
    :title [ :mainTitle "A" ] ;
    :contribution [ :agent </person/a> ;
                    :role lcrel:ill ] .
```

Target A

```
prefix : <http://purl.org/dc/terms/>

</work/a> a :BibliographicResource ;
    :format bf:Print ;
    :isFormatOf <f/abstract/a> ;
    :identifier "12-3456-789-0" ;
    :issued "2017" ;
    :publisher <f/org/a> .

</abstract/a> a :BibliographicResource ;
    :contributor </person/a> ;
    :title "A" .
```

Target B

```
prefix : <http://schema.org/>

</work/a> a :Book,
        :Product ;
    :exampleOfWork <f/abstract/a> ;
    :isbn "12-3456-789-0" ;
    :datePublished "2017" ;
    :publisher <f/org/a> .

</abstract/a> a :Book ;
    :illustrator </person/a> ;
    :name "A" .
```

Basic Mappings

Map term X to term Y

Given:

```
<> a foaf:Document ;
   dc:title "A" .
```

Target: [RDFS]

Expect:

```
<> a rdfs:Resource ;
      rdfs:label "A" .
```

Assuming:

```
foaf:Document rdfs:subClassOf rdfs:Resource .

dc:title rdfs:subPropertyOf rdfs:label .
```

Varying Granularities

Structured Values and Shorthand Properties

Given:

```
</instance/a>
    bf:identifiedBy [ a bf:Isbn ;
        rdf:value "12-3456-789-0" ] .
```

Target: Schema.org

Expect:

```
</instance/a>
    schema:isbn "12-3456-789-0" .
```

Assuming:

```
schema:isbn
    owl:propertyChainAxiom (
          [ rdfs:subPropertyOf bf:identifiedBy ;
            rdfs:range bf:Isbn ]
          rdf:value
        ) .
```

Flattening Events

Given:

```
</instance/a>
    bf:provisionActivity [ a bf:Publication ;
            bf:agent </org/a> ;
            bf:date "2017" ] .
```

Target: Schema.org

Expect:

```
</instance/a>
    schema:publisher </org/a> ;
    schema:datePublished "2017" .
```

Assuming:

```
schema:datePublished
    owl:propertyChainAxiom (
      [ rdfs:subPropertyOf bf:provisionActivity ;
        rdfs:range bf:Publication ]
       bf:date
     ) .

schema:publisher
    owl:propertyChainAxiom (
      [ rdfs:subPropertyOf bf:provisionActivity ;
        rdfs:range bf:Publication ]
      bf:agent
    ) .
```

Beyond OWL: Qualified Relation as Reification

Given:

```
</work>
    bf:contribution [ a bf:Contribution ;
            bf:agent </person/a> ;
            bf:role lcrel:aut
        ] .
```

Target: DC Terms

Expect:

```
</work>
    dc:creator </person/a> .
```

Assuming:

```
bf:Contribution
    rdfs:subClassOf rdf:Statement .

bf:contribution rdfs:range bf:Contribution ;
    rdfs:subPropertyOf
            [ owl:inverseOf rdf:subject ] .

bf:role rdfs:domain bf:Contribution ;
    rdfs:subPropertyOf rdf:predicate .

bf:agent rdfs:domain bf:Contribution ;
    rdfs:subPropertyOf rdf:object .

lcrel:aut
    rdfs:subPropertyOf dc:creator .
```

Limitations

What To Leave Out?

Normalization?

Concept Scheme Mappings? Complementary?

Error Correction?

Bad Literals.

String Idiosyncrasies (structure versus presentation):

Forms of Names, microsyntaxes, “authorized access points”.

“Creative” semantics, misinterpretations, **philosophical traps** (platonism, perdurantism, nominalism).

“Punning?” (c.f. Jeni Tennison, 2012)

But isn’t OWL Dangerous?

Open Worlds Collide!

The TVM approach doesn’t assume a fully flat open world. (More like linkable layers of worlds.)

An Application Vocabulary is a data wrapper (for semantic encapsulation).

It supports “follow your nose” from your data! Your *specific notion* of “title” gets a URI. (Cf. Application Profiles.)

(Also, *never* consume owl:sameAs undiluted!)

https://svgsilh.com/image/3254424.html

Next Steps

Data Ingestion & Profile Negotiation

Using Target Vocabulary Maps in production.

Reworking ETL pipelines, accessing linked data described in BF, Schema.org, SKOS, DC etc., using our Application Vocabulary mappings.

Internally:

- Normalize on selected Concept Schemes.
- Cleaning up poor literals (e.g. dates).

Publishing:

- Supporting Profile Negotiation (to provide "just BF", "just DC", "just SKOS" views, and some combinations (APs)).
- Simple Schema.org snippets (for SEO).

Just Raw Data & ETL while
Waiting for the Semantic Web?

Raise the Bar Just a Bit!
Map your Target Vocabularies!