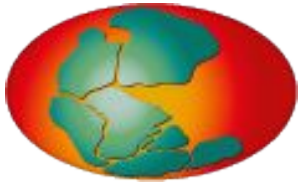
Design patterns for sensor data and metadata

About



Markus Stocker

Research Associate

PANGAEA, MARUM, University of Bremen

@envinf
http://orcid.org/0000-0001-5492-3212

marum
Center for Marine
Environmental Sciences

Universität Bremen

AWI

Overview

- Motivation
- Semantic technologies
- Design patterns
- Take away

Motivation

- Complex infrastructure
- Large data volumes
- Heterogeneity
- Support data reuse
- Automation
- Smart applications

Semantic technologies

History

- First the Web, circa 1990
- Publish content, structure and formatting
- Need to include semantics clear early on
- Even basic entity recognition is hard
- Notion of the Semantic Web was developed
- First semantic technologies circa 1999
- Considerable development since

Principles

- Identify resources
  - Originally, Web resources
    - Such as Web pages, displayed images, ...
  - In practice, also
    - Physical objects, e.g. people, devices
    - Conceptual entities, e.g. LI-7700

Principles

- Describe resources
  - For instance, annotate the LI-7700 product Web page with metadata stating that the page is *about sensors of type* LI-7700 and sensors of such type are *products manufactured by* LI-COR

Principles

- Describe for machines
  - Machine-readable descriptions
  - Machine-interpretable descriptions

Principles

- Machine-readable descriptions
  - Requires standard metadata model
  - One or more serialization formats
  - Development of software

Principles

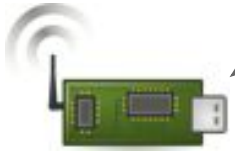
- Machine-interpretable descriptions
  - Use terms of vocabularies, not just tags
  - Tags lack machine interpretable semantics
  - Use *terms* of vocabularies (ontologies)
  - Vocabularies define term semantics
  - Semantics are machine interpretable

Resource identification

- Identify resources by URI
- In practice generally HTTP URI
- Globally identified resources
- Disambiguated resources
- Possibly dereferenceable identifiers

Resource identification

http://example.org/devices/myThermometer



http://licor.com/devices/LI-7700

The LI-7700 was designed to make high quality measurements in extreme environments.

Source: https://www.licor.com/env/products/gas_analysis/LI-7700/

http://www.mta.info/lirr/locomotives/LI-7700

The locomotive LI-7700 of the Long Island Rail Road Company

Source: http://www.rrpicturearchives.net/locoPicture.aspx?id=133778

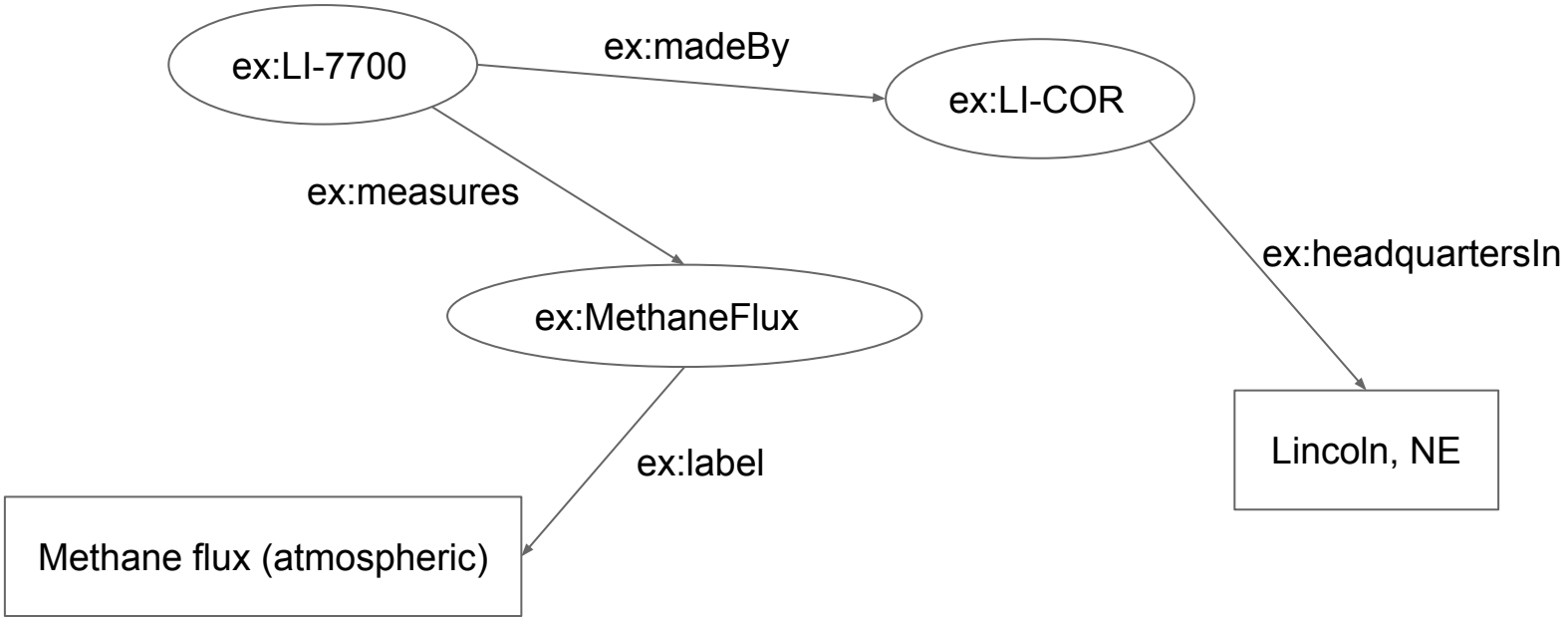
Machine readability

- Metadata model
- Serialization formats
- Query language
- Software

Metadata model

- Resource Description Framework (RDF)
- At its core is the *statement*
  - Property relating two resources
  - Subject, predicate, object triple-structure
- Examples
  - ex:LI-7700, ex:madeBy, “LI-COR”^^xsd:string
  - ex:LI-7700, ex:madeBy, ex:LI-COR

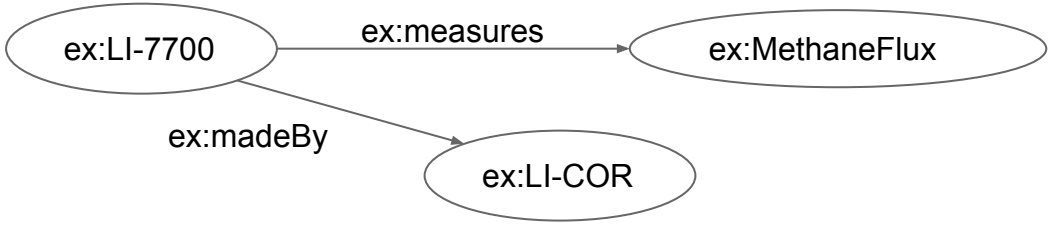
Metadata model



Serialization formats

- **RDF/XML** XML document
- **Turtle** Human friendly
- **N-Triples** Easy to parse
- **JSON-LD** Based on JSON

RDF/XML




```
<rdf:Description rdf:about=”http://example.org#LI-7700”>
    <ex:measures rdf:resource=”http://example.org#MethaneFlux”/>
    <ex:madeBy rdf:resource=”ex:LI-COR”/>
</rdf:Description>
```

Turtle

```
@prefix ex: <http://example.org#> .

ex:LI-7700 ex:measures ex:MethaneFlux ;
    ex:madeBy ex:LI-COR .
```

N-Triples

<http://example.org#LI-7700> <http://example.org#measures> <http://example.org#MethaneFlux> .
<http://example.org#LI-7700> <http://example.org#madeBy> <http://example.org#LI-COR> .

JSON-LD

```
{
    "@context": {
        "measures": "http://example.org#measures",
        "madeBy":"http://example.org#madeBy"
    },
    "@id": "http://example.org#LI-7700",
    "measures":{ "@id":"http://example.org#MethaneFlux" },
    "madeBy":{ "@id":"http://example.org#LI-COR" }
}
```

Query language

- SPARQL - Query language for RDF
- At its core is the *triple pattern*
- Structured as the statement, but ...
- Subject, property, object may be variable
- Example
  - ?sensor ex:measures ex:MethaneFlux

Query language

```
PREFIX ex: <http://example.org#>

SELECT ?sensor ?manufacturer
WHERE {
    ?sensor ex:measures [ ex:label “Methane flux (atmospheric)” ] .
    ?sensor ex:madeBy ?manufacturer .
}
```

Software

- Frameworks
  - Apache Jena, Sesame
  - I/O, processing
- Databases
  - Stardog, Blazegraph, AllegroGraph, Virtuoso
- SPARQL endpoints
- Visualization, browsers

Machine interpretability

- How to describe what RDF data is about?
- What are **thisLI-7700**, **thatLI-7700**?
- Resources, yes, but can we tell more?
- Can we group resources into classes?
- Can we describe classes to machines?

Machine interpretability

- LI-7700 is in fact a class: it is a type
- The class of LI-COR CH4 gas analyzers
- Concrete gas analyzers are instances
- LI-7700 is a term of a vocabulary

Machine interpretability

- Vocabularies describe meaning of terms
- Specifically,
  - Classes (aka concepts)
  - Relations (aka properties)
- Vocabularies are aka ontologies
- Required is a formal ontology language

Ontology languages

- RDF Schema
- Web Ontology Language

RDF Schema

- Language to create basic ontologies
- Enables definition of
  - Classes
  - Sub-class relations between classes
  - Sub-property relations between properties
- Ontologies are RDF documents

RDF Schema

Class: LI-7700
    SubClassOf: SensingDevice

Individual: thisLI-7700
    Types: LI-7700

Web Ontology Language

- Building of expressive ontologies
- Complex class descriptions
  - Intersections, unions, complements
  - Property restrictions
  - Equivalence, disjointness
- Property descriptions
  - Object and datatype properties
  - Inverse, transitive, equivalent

Web Ontology Language

Class: AtmGasFlux
    SubClassOf: Property

Class: LI-7700
    SubClassOf: SensingDevice
        that measures only AtmGasFlux

Individual: MethaneFlux
    Types: AtmGasFlux

Individual: thisLI-7700
    Types: LI-7700
    Facts: measures MethaneFlux

Take away

- Semantic technologies for metadata about resources
- Metadata for machines (primarily)
- Terms used in metadata formally defined in vocabularies (ontologies)

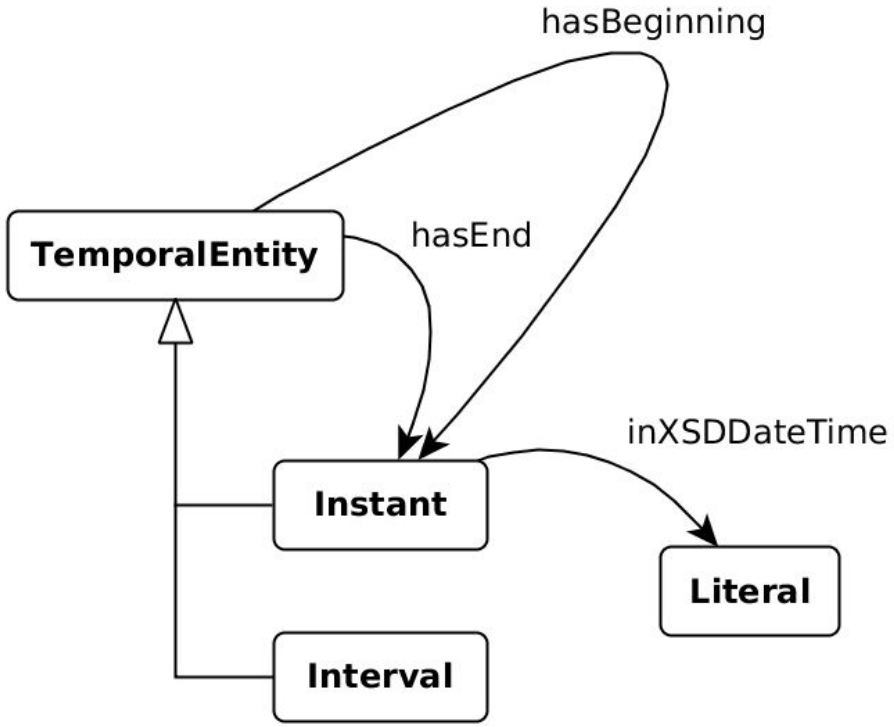
Design patterns

Reusable successful solutions to a recurrent modeling problem

Source: http://ontologydesignpatterns.org/


Time
Datasets
Observations
Quality
Sensors
Quantities
Space
Units
Provenance

Time

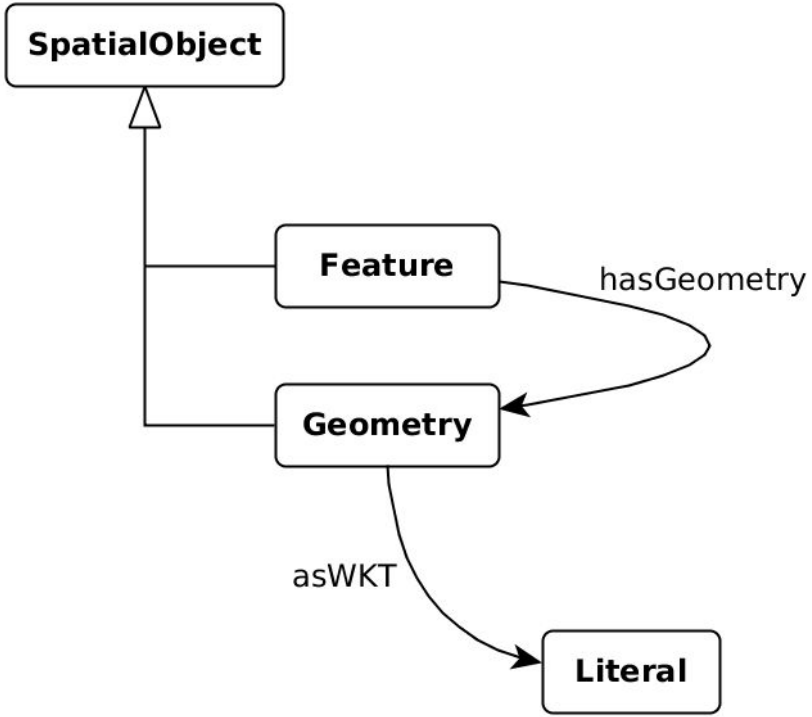


Source: https://www.w3.org/TR/2016/WD-owl-time-20160712/

Time

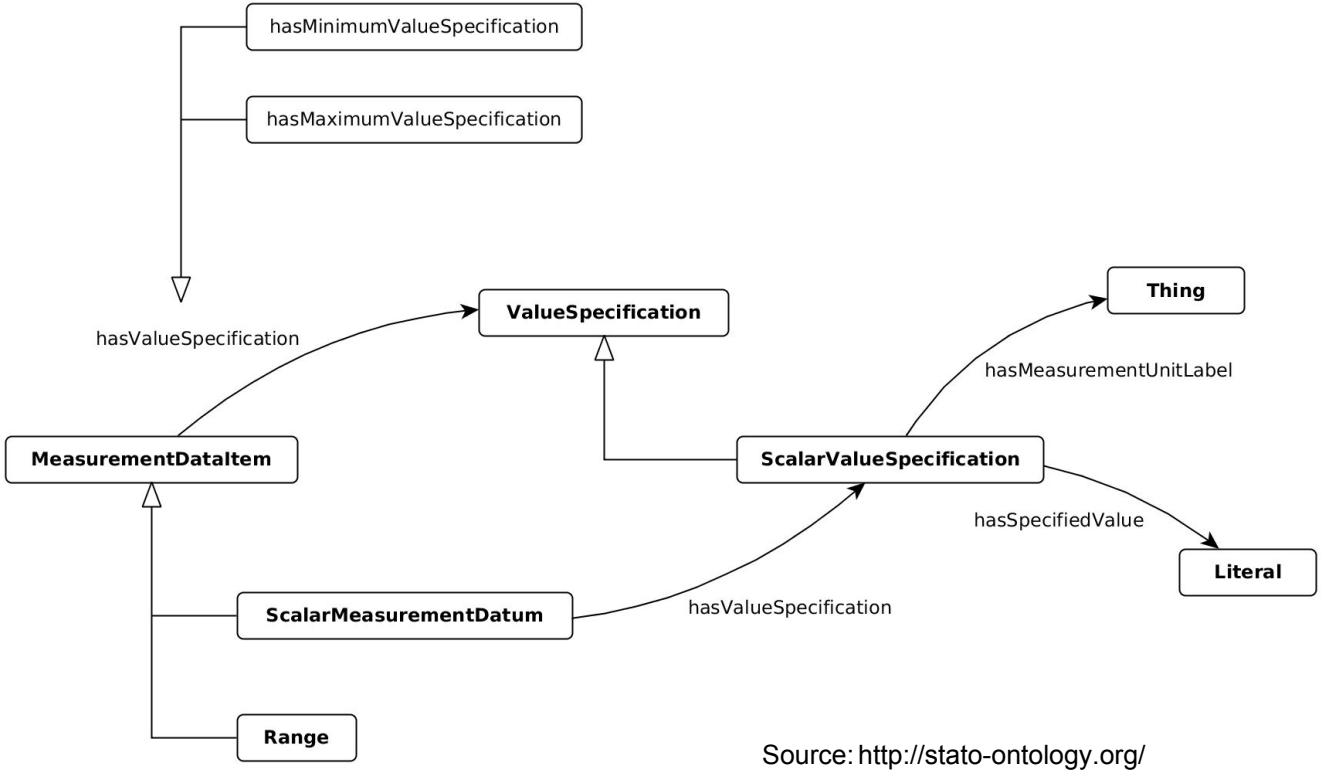
```
<> rdf:type time:Interval ;
 time:hasBeginning [
  rdf:type time:Instant ;
  time:inXSDDateTime "2016-07-20T00:00:00Z"^^xsd:dateTime
 ] ;
 time:hasEnd [
  rdf:type time:Instant ;
  time:inXSDDateTime "2016-07-21T00:00:00Z"^^xsd:dateTime
 ] .
```

Space



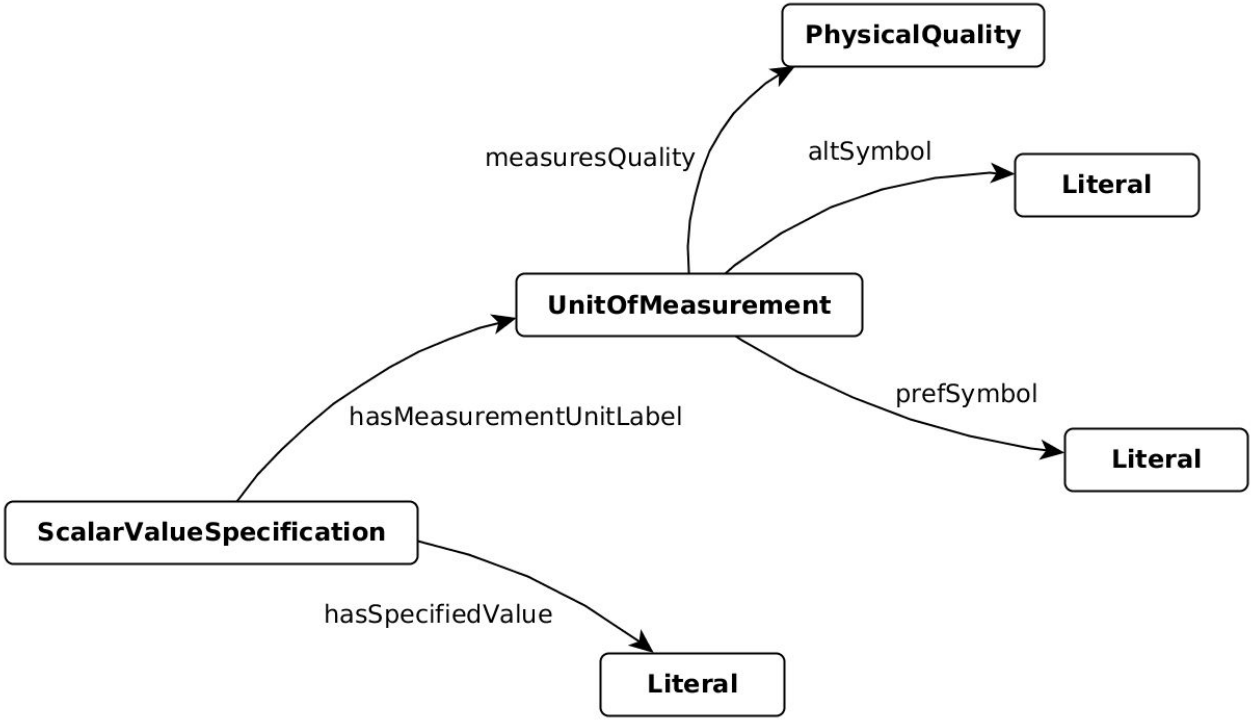
Source: http://www.opengeospatial.org/standards/geosparql

Quantities



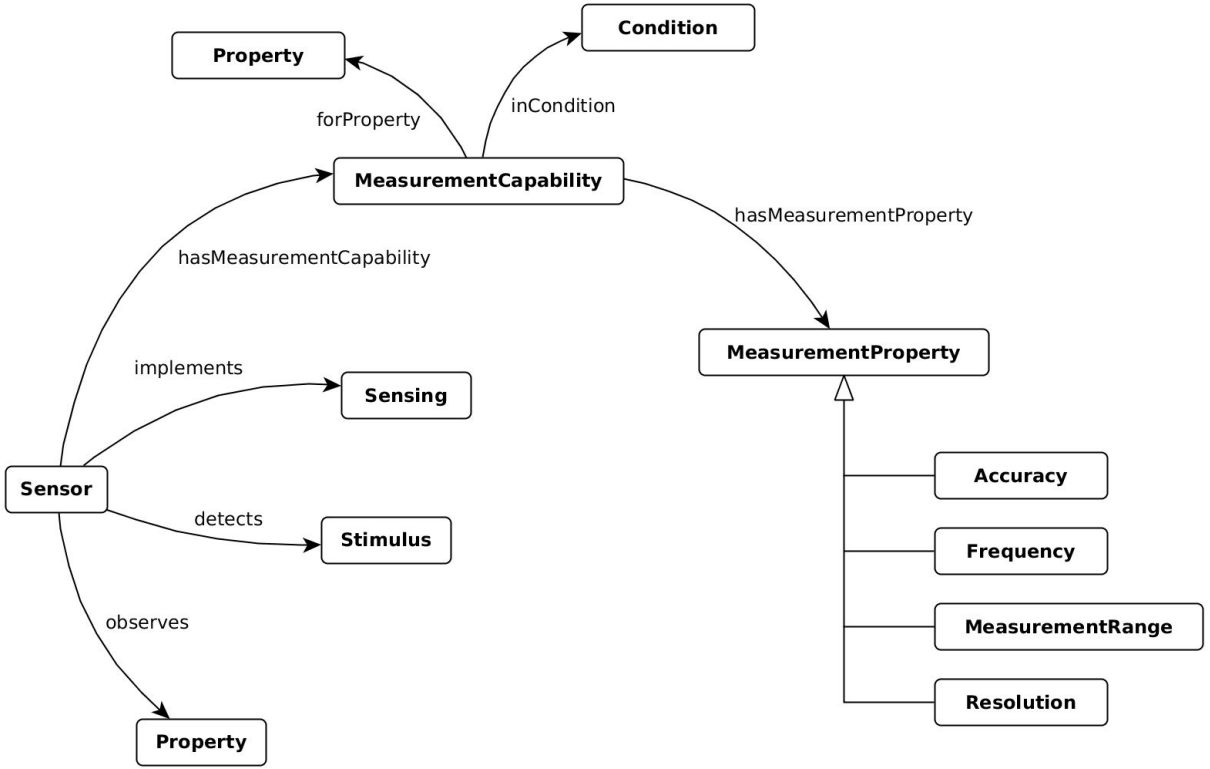
Source: http://stato-ontology.org/
https://github.com/information-artifact-ontology/IAO/

Units



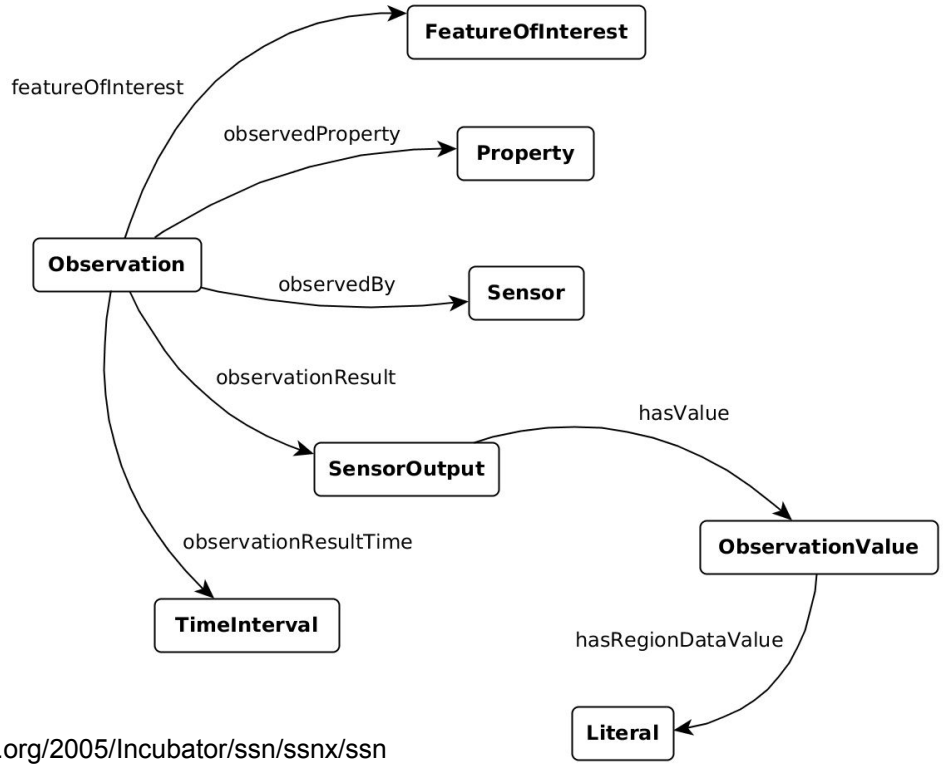
Source: http://idi.fundacionctic.org/muo/muo-vocab.html

Sensors



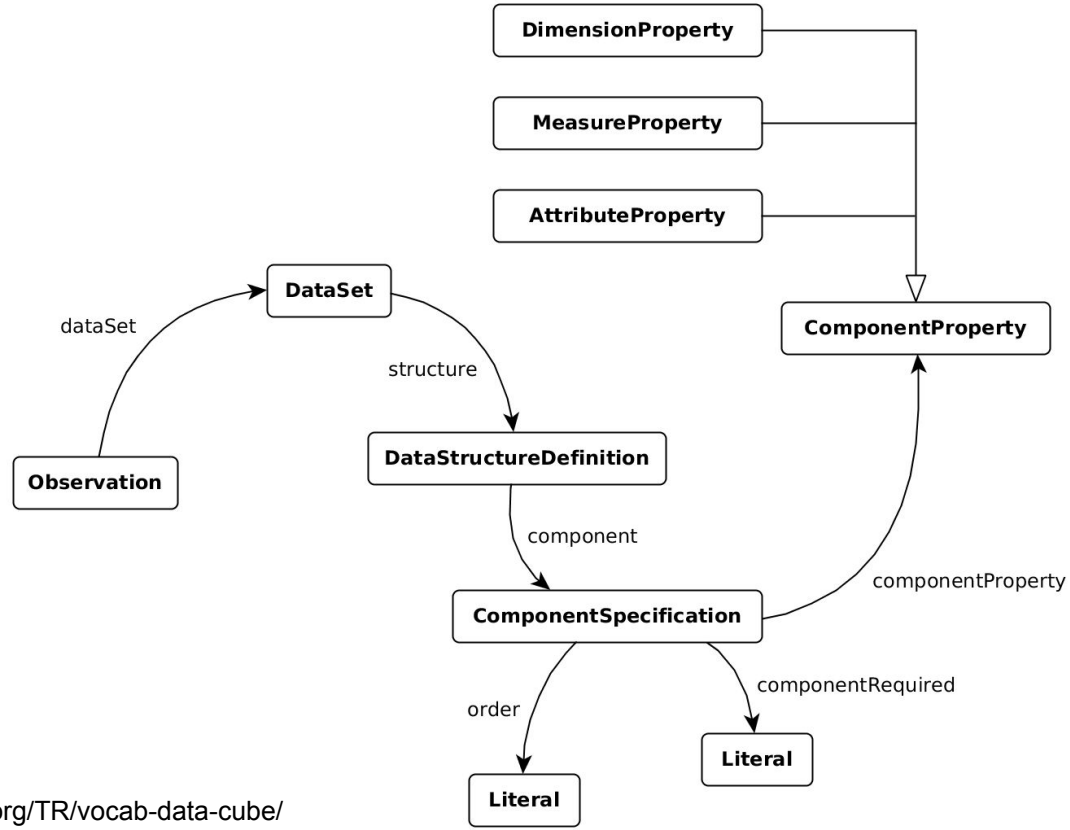
Source: https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

Sensor observations



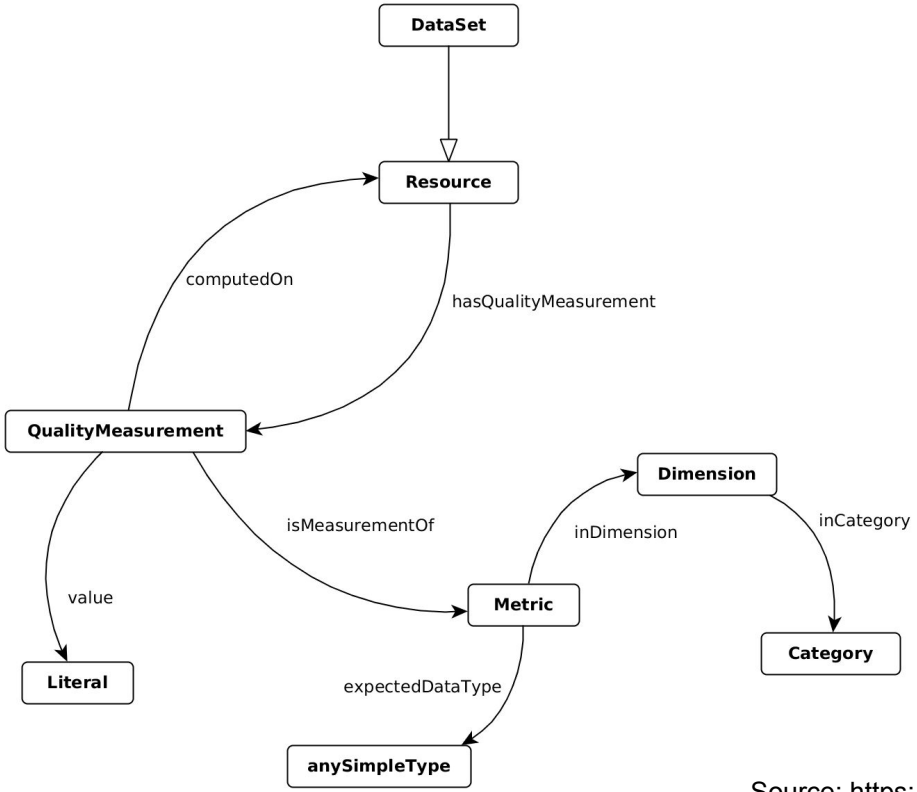
Source: https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

Datasets



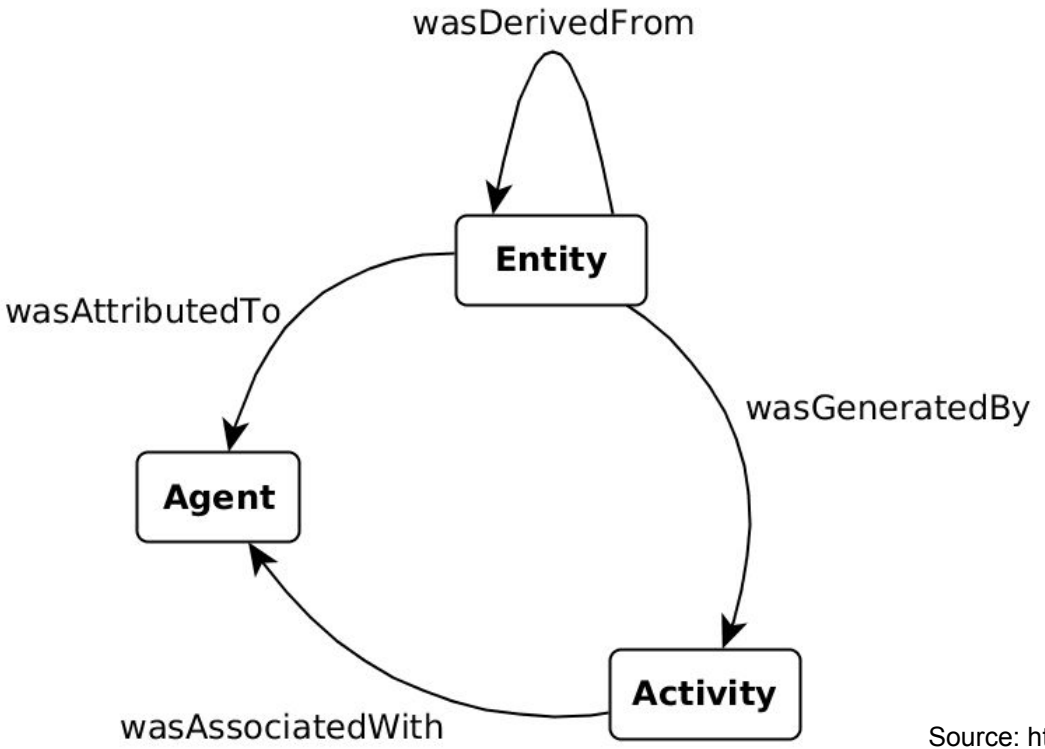
Source: https://www.w3.org/TR/vocab-data-cube/

Quality



Source: https://www.w3.org/TR/vocab-dqv/

Provenance



Source: https://www.w3.org/TR/prov-o/

Take away

- Look for and reuse design patterns
- Use existing vocabularies for inspiration
- Build community to develop new ones
- In practice there are challenges/problems
- Community consensus can be difficult but is crucial
- Metadata and/or data (observations?)
- Steep learning curve with semantic technologies
- Design pattern specs also in XML, JSON, ...