

# Linked Product Data

---

Describing Products in a Linked Data World

Anna Wagner

Mathias Bonduel (presenter)

# Agenda

1. Linked Product Data in the Context of LBD
2. The Building Product Ontology (BPO) - A Core Ontology
3. Property Modelling - 3 Levels of Complexity
4. OMG, FOG and GOM: Geometry as Property
5. Domain Taxonomies
6. Putting everything together

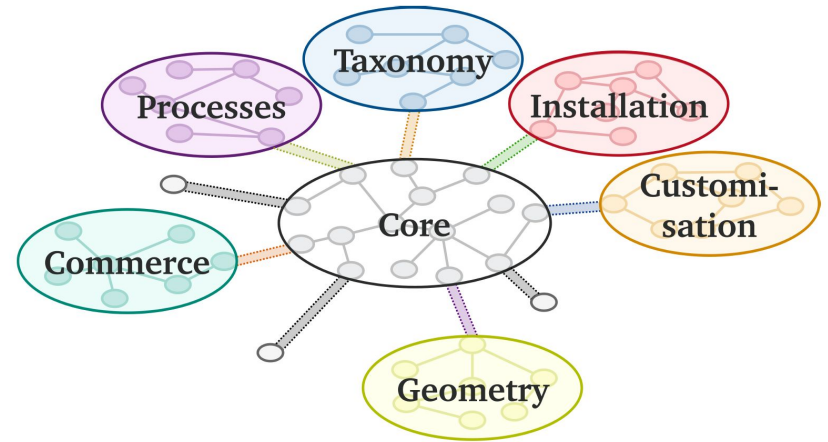


# Linked Product Data in the Context of LBD

- System-based ontologies are focusing on their individual domains, overlapping components, such as (multi-functional) products may belong to multiple systems
  - BOT: building topology
  - SEAS: smart energy aware components and systems
  - TSO/FSO: HVAC systems
  - ... and many more
- A building product (component) ontology can enable consistent and unified querying across components of all connected domains

# Linked Product Data in the Context of LBD

- Linked product data may serve as connecting core
- A metadata schema (core) can enhance unified querying
- Dedicated connector ontologies support unified querying in disregard of the applied ontology (or other schema) of the connected domains
- The configuration of all connected domains may be customized for each project



# Building Product Ontology (BPO) - A Core Ontology Approach

- **BPO**: a core ontology (meta model) for uniform querying over heterogeneous building product data
- Focus on product composition, properties and interconnections
- Flexibility to model parametric and multi-functional products
- Aligned to common vocabularies
  - Schema.org
  - GoodRelations
  - SEAS
- Classification of entities is not part of BPO
  - Domain taxonomy required

## BPO: Building Product Ontology

Release 04.11.2019

**This version:**

<http://w3id.org/bpo/1-2/>

**Previous version:**

<http://w3id.org/bpo/1-1/>

**Latest version:**

<https://w3id.org/bpo>

**Revision:**

1.2

**Authors:**

[Anna Wagner \(iib, TU Darmstadt\)](#)

[Laura Kristina Moeller \(iib, TU Darmstadt\)](#)

[Christian Leifgen \(iib, TU Darmstadt\)](#)

[Christian Eller \(iib, TU Darmstadt\)](#)

**Contributors:**

[Johannes Eisenlohr \(Fraunhofer ISE\)](#)

[Tim Rist \(Fraunhofer ISE\)](#)

[Gesa Benndorf \(Fraunhofer ISE\)](#)

[Wendelin Sprenger \(Ed. Zueblin AG\)](#)

[Christoph Maurer \(Fraunhofer ISE\)](#)

[Tillmann E. Kuhn \(Fraunhofer ISE\)](#)

**Download serialization:**

Format [JSON LD](#)

Format [RDF/XML](#)

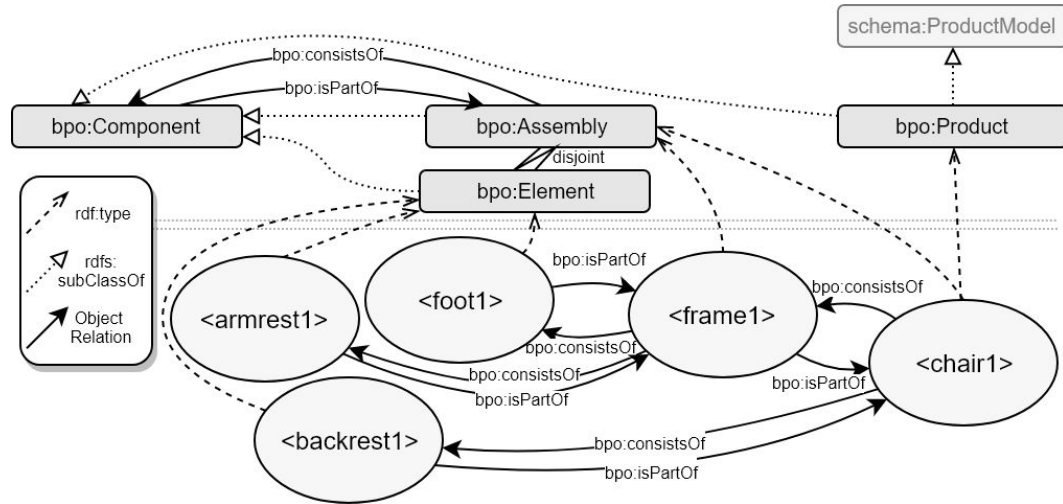
Format [N Triples](#)

Format [TTL](#)

**License:**

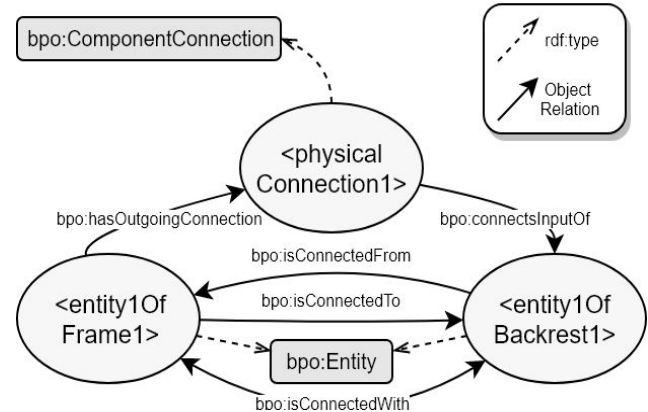
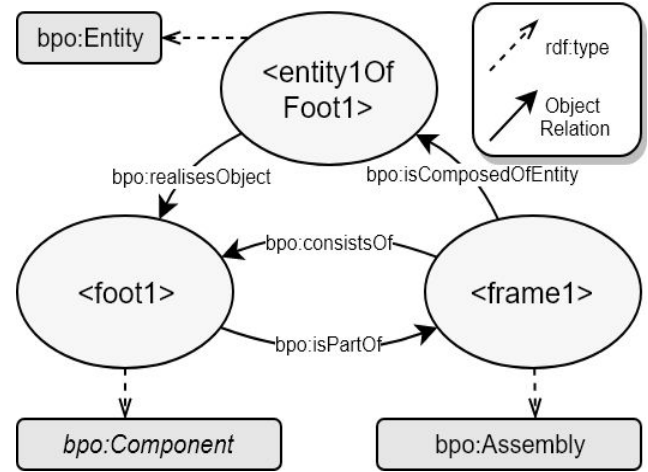
# BPO: Product Composition

- `Component`: Super-class that represents all objects that are part of the product composition
- `Assembly`: Components that consist of other Components
- `Element`: Components that cannot - or will not - be decomposed any further
- `Product`: Component that can be bought
- Transitive decomposition properties (`consists of` / `is part of`) for uniform querying in disregard of chosen modelling approach



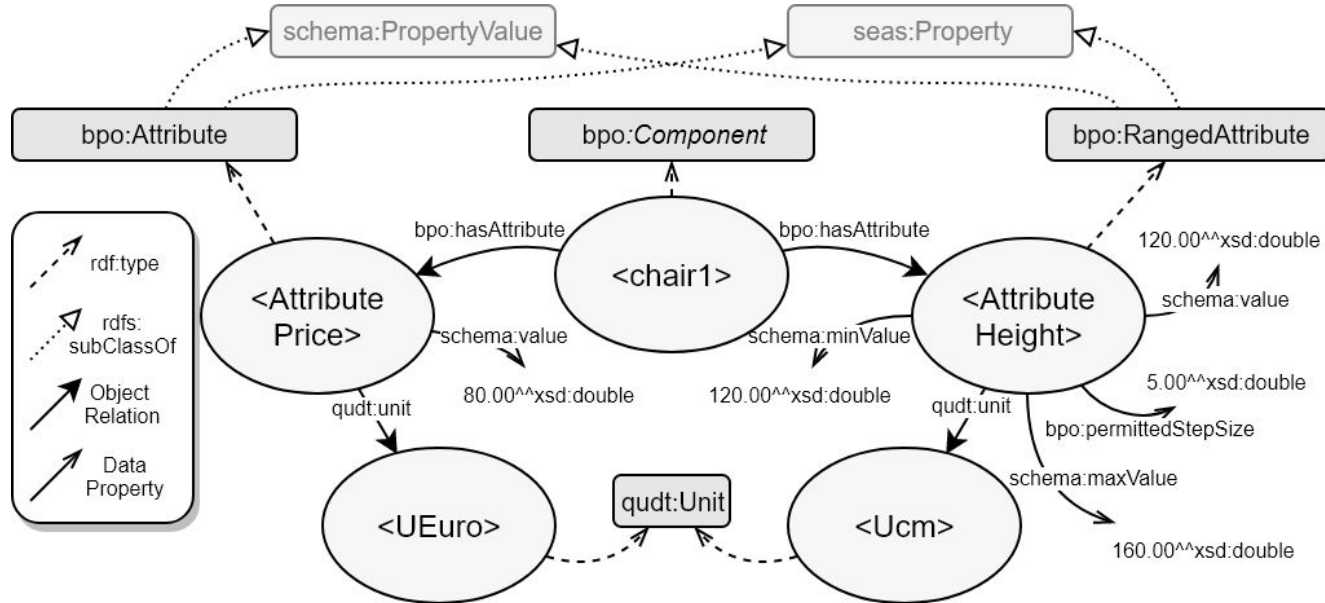
# BPO: Entities and Interconnections

- **Entities:** represent a component instance that may hold varying properties to their realised component (e.g. colour or placement)
- **Component Connection (is connected to/from):** represent a connection between to entities that may or may not be directed
- Chain axioms for uniform querying in disregard of chosen modelling approach
  - On entity usage
  - On direct and objectified component connections



# BPO: Attributes

- `Attribute`: represent product properties with a precise value
- `Ranged Attribute`: represent product properties with a value range
- Value range may be defined with min and max values and with or without step sizes
- Application of QUDT as well-known vocabulary for units







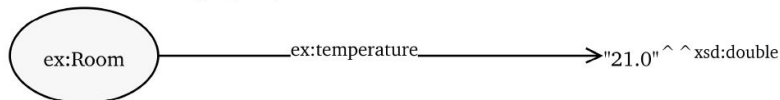
## BPO: Conclusion

- Minimalistic “core” or top-level ontology for building product data
- Additional domain knowledge and schemas need to be added
- No domain taxonomy included in BPO
- Extensions may be required for individual use cases

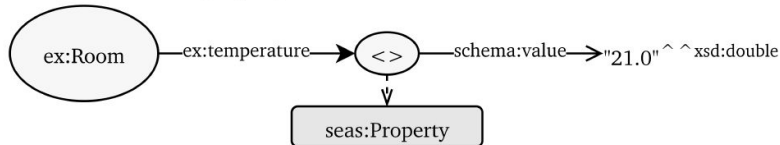
# Property Modelling - 3 Levels of Complexity

- Different use cases require different complexity in modelling properties
- Combinations of approaches are possible, but might complicate uniform querying
- In the LBD community, 3 levels are commonly agreed upon:

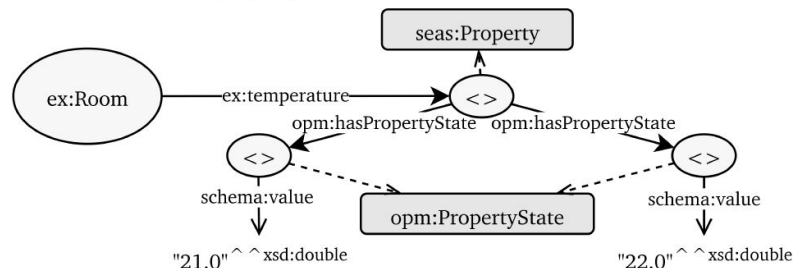
## Level 1 - Direct link to property value



## Level 2 - Intermediate 'property' node for metadata



## Level 3 - Intermediate 'property state' node for version control

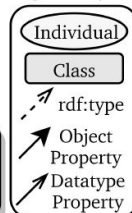


1. Dedicated datatype properties
2. Objectified properties with Property node
3. Objectified value with Property State node

### Legend - Prefixes

**ex** : <https://example.org/> (imaginary example URI)    **schema** : <http://schema.org/>  
**opm** : <https://w3id.org/opm#>    **seas** : <https://w3id.org/seas/>  
**rdf** : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>    **xsd** : <http://www.w3.org/2001/XMLSchema#>

### Legend - Objects



# Level 1: Datatype Properties

## Pro

- Datatype properties from domain taxonomies can be re-used directly
- Smaller graph and high querying performance due to less nodes that need to be navigated

## Con

- Additional information on the properties cannot be added (e.g. units)
  - Units may be added via a [Custom Datatypes](#) approach and string embedded units
- Retrieving *only* all properties is more complex (if possible), unless the applied taxonomy extends a generic `has property` relation
- Properties cannot be referenced, e.g. for parametric descriptions

# Level 2: Property Nodes and SEAS

- [SEAS](#) as modular ontology family that cover various aspects of smart energy aware systems
- [Feature of Interest ontology](#) introduces the concept of objectified property nodes with a generic `hasProperty` relation (extending W3C's [SSN](#))
- Classification of properties may be realized via property classes or dedicated object properties (not part of SEAS)

OVERVIEW

[Introduction](#)

CORE PATTERNS

[Features and Properties](#)


[Property evaluation](#)

[Systems and their connections](#)

[Procedure executions](#)

PROJECT DOCUMENTATION

[Project Information](#)

ENHANCED BY 

## The SEAS Feature of Interest ontology.

- **This version:** v1.0 - <https://w3id.org/seas/FeatureOfInterestOntology-1.0>
- **Latest published version:** <https://w3id.org/seas/FeatureOfInterestOntology>
- **Creators:**
  - <http://www.maxime-lefrancois.info/me#>
- **Issued:** 2016-07-01
- **Modified:** 2016-09-21
- **Other visualizations:** [VOWL](#), [Turtle](#), [RDF/XML](#).

This ontology defines feature of interest and their properties, as an extension of the core classes of the SSN ontology (<https://www.w3.org/ns/ssn/>).

A feature of interest is an abstraction of a real world phenomena (thing, person, event, etc). A feature of interest is then defined in terms of its properties, which are qualifiable, quantifiable, observable or operable qualities of the feature of interest.

Figure below provides an overview of the concepts in this ontology:

```
graph TD
    FOI[FeatureOfInterest] -- "1. isPropertyOf" --> P[Property]
    FOI -- "hasProperty" --> P
    car1["<car/1>"] -.-> FOI
    car1 -- "hasProperty" --> speed["<car/1/speed>"]
    speed -.-> P
    speed -- "<speed> 1" --> P
```

# Level 2: Property Nodes and SEAS

## Pro

- Additional information can be added to the properties
  - E.g. author, unit, creation/modification date
- Properties can be referenced
- Level 1 can be inferred by a reasoner

## Con

- (Comparatively) Lower querying performance, as more nodes need to be covered
  - May be mitigated by using dedicated object properties for given properties
- Datatype properties of domain taxonomies cannot be re-used directly

# Level 3: Ontology for Property Management (OPM)

- OPM: Introduce a property state as intermediate node between a `seas:Property` node and its value
- Can be used for evolving properties
- Includes concepts for calculating property values and defining the values' origin or base calculation

**W3C Community Group**  
Draft Report

**UNOFFICIAL DRAFT**

## Ontology for Property Management

Draft Community Group Report 22 November 2018

**W3C**

**Latest editor's draft:**  
<https://w3c-lbd-cg.github.io/opm/>

**Implementation report:**  
<https://w3c-lbd-cg.github.io/opm/>

**Editors:**  
[Mads Holten Rasmussen](#) (Niras | Technical University of Denmark)  
[Maxime Lefrançois](#) (Ecole Nationale Supérieure des Mines de Saint-Étienne)

**Contributors (ordered alphabetically):**  
Mathias Bonduel, KU Leuven

Copyright © 2018 the Contributors to the Ontology for Property Management Specification, published by the [Linked Building Data Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable summary is available.

---

### Abstract

The Ontology for Property Management (OPM) is an ontology for describing temporal properties that are subject to changes as the building design evolves.

The namespace for OPM terms is <http://www.w3id.org/opm#>

The suggested prefix for the OPM namespace is `opm`

The Turtle version of the OPM ontology is available at <http://www.w3id.org/opm/opm.ttl>

### Status of This Document

This specification was published by the [Linked Building Data Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn more about [W3C Community and Business Groups](#).

### General Information

1.	Introduction
2.	Origins of OPM
3.	Axiomatization
3.1	Overview of Classes and Properties
3.2	Property States
3.2.1	Overview and examples
3.2.2	Specification
3.2.2.1	<code>opm:PropertyState</code>
3.2.2.2	<code>opm:CurrentPropertyState</code>
3.2.2.3	<code>opm:OutdatedPropertyState</code>
3.2.2.4	<code>opm:Deleted</code>
3.2.2.5	<code>opm:hasPropertyState</code>
3.3	Reliability
3.3.1	Overview and examples
3.3.2	Specification
3.3.2.1	<code>opm:Assumed</code>
3.3.2.2	<code>opm:Confirmed</code>
3.3.2.3	<code>opm:Derived</code>
3.3.2.4	<code>opm:Required</code>
3.3.2.5	<code>opm:documentation</code>
3.4	Calculations
3.4.1	Overview and examples
3.4.2	Specification
3.4.2.1	<code>opm:Calculation</code>
3.4.2.2	<code>opm:argumentPaths</code>
3.4.2.3	<code>opm:inferredProperty</code>
3.4.2.4	<code>opm:expression</code>
3.4.2.5	<code>opm:toRestriction</code>

# Level 3: Ontology for Property Management (OPM)

## Pro

- Additional information can be added to the property values
  - E.g. author, creation/modification date
- Property values can be referenced
- Level 2 and 1 can be inferred by a reasoner

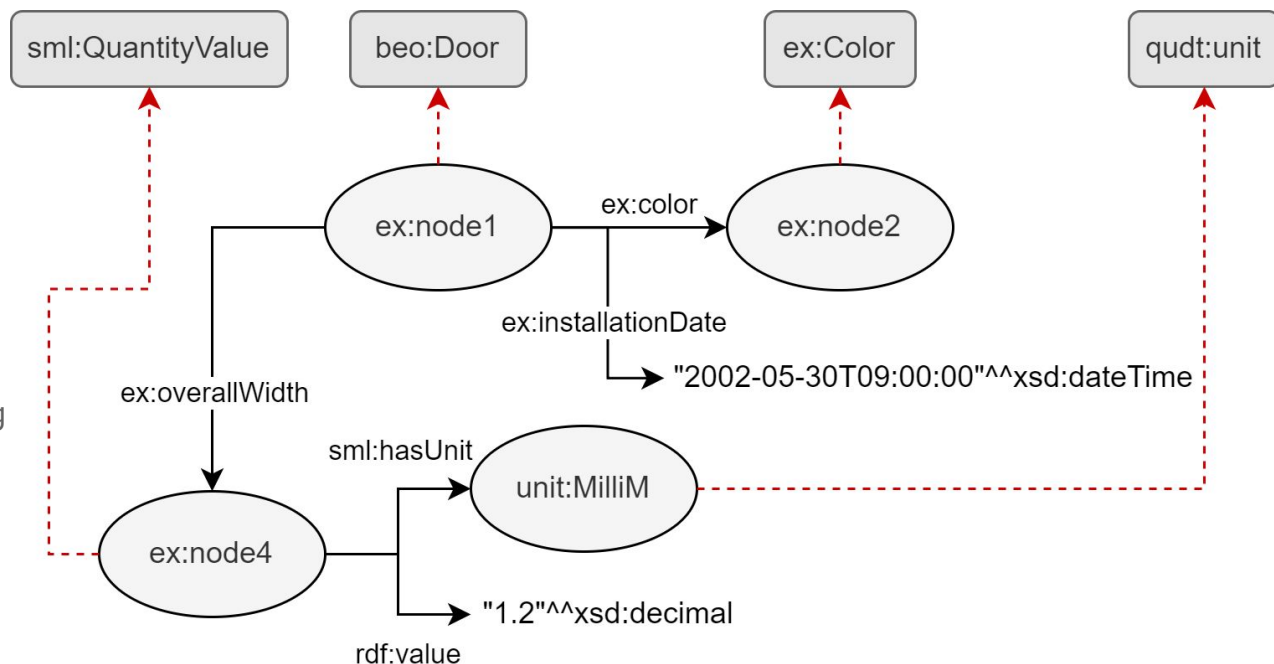
## Con

- (Comparatively) Lowest querying performance, as more nodes need to be covered
  - May be mitigated by searching for current property state and using dedicated object properties
- Datatype properties of domain taxonomies cannot be re-used directly



# Properties modelling according Semantic Modeling and Linking (SML, EN 17632-1:2022)

- Meant for data exchange first
- Agreements on when to use which level, i.e. a mix between:
  - Level 1 for qualitative properties
  - Level 2 for quantitative properties > units using QUDT



# Property Modelling - Conclusion

- Complexity may vary depending on the observed project
- Reasoning can always infer lower complexity levels
- Advantages and disadvantages need to be weighed thoroughly before deciding on a complexity level
  - High expressivity vs. high performance

# OMG, FOG and GOM: Geometry as Properties

## OMG - Ontology for Managing Geometry

- Defines relations between building objects and their geometry description(s) in 3 complexity levels
- Allows multiple geometry descriptions per object
- Provides means to describe dependencies (geometry - geometry | geometry - property)
- Versioning and grouping of geometries
- Linking to RDF-based geometry or non-RDF geometry (embedded string, encoded in case of binary format, or a reference to an external geometry file)

## FOG - File Format Ontology for Geometry

- Extension of OMG
- taxonomy of specific relations for connecting geometries per geometry file format (incl. supporting content like rendering materials, images)
- Link to parts of bigger geometry descriptions > taxonomy of geometry identifiers

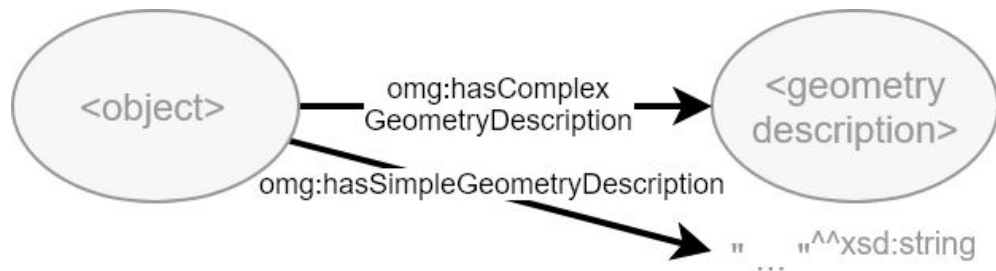
## GOM - Geometry Metadata Ontology

- Adds further metadata to geometry descriptions, such as classifications, coordinate systems, transformations, etc.

# OMG and FOG - Level 1

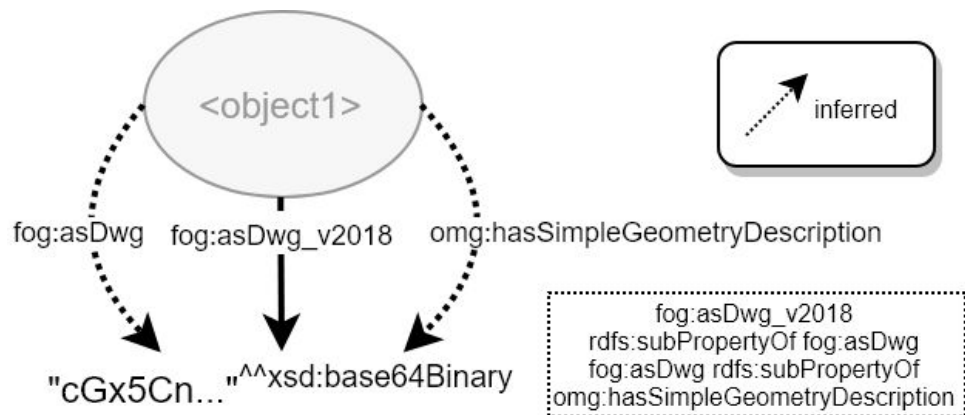
## OMG:

- directly connects geometry to any object
- both simple (non-RDF) and complex (RDF-based) geometries



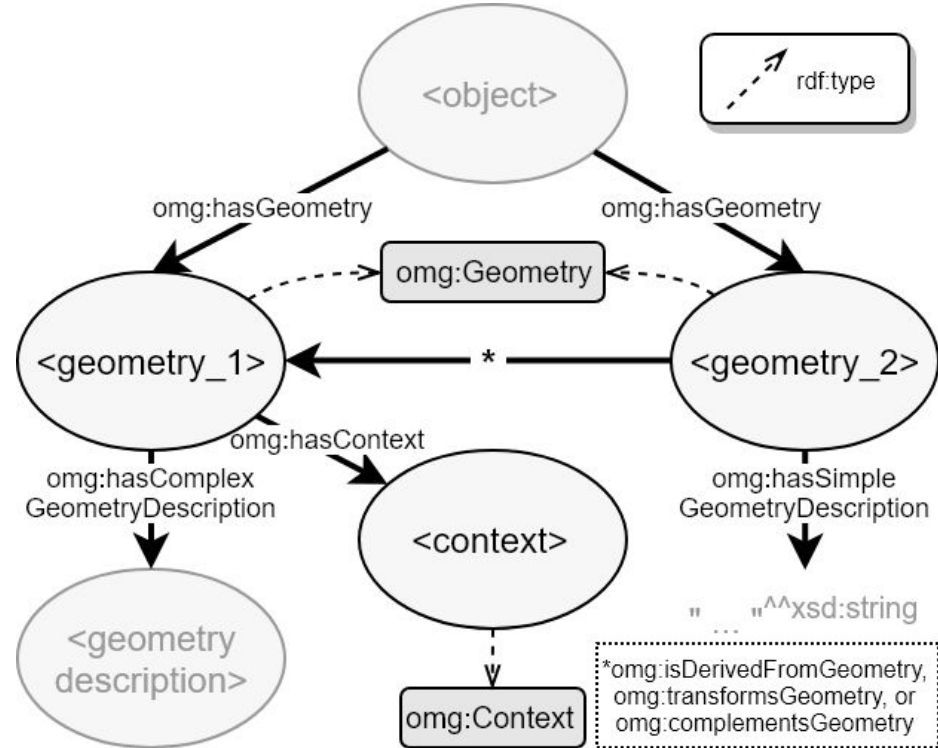
## FOG:

- Adds a taxonomy to those relations that indicates the observed geometry format and serialisation



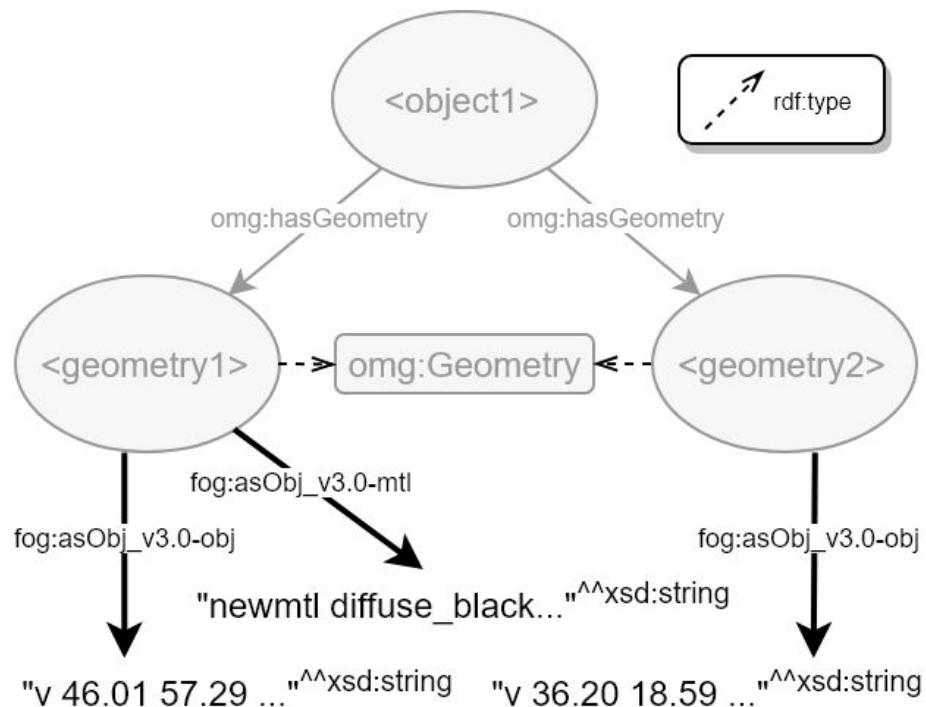
# OMG - Level 2

- intermediate 'Geometry' node for metadata, grouping, relating to other geometries/properties, etc.
- relations between geometry nodes:
  - derivation: keeping data synced by modeling dependencies
  - transformation: avoiding unnecessary data redundancies
  - complementation: adding details to already defined geometries
- 'Context' for grouping
  - simplifies extraction of geometries by context (e.g. planning phase or role)



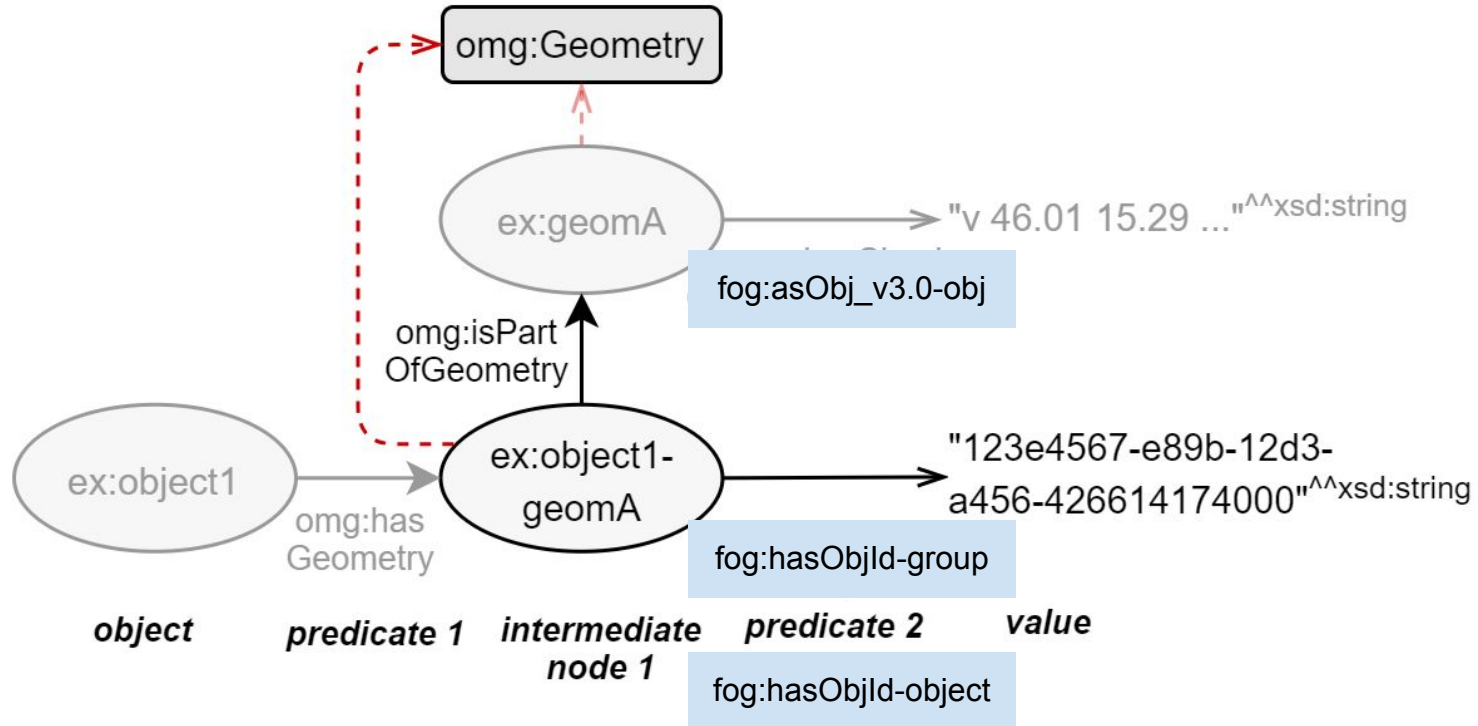
# FOG - Level 2

- Multi-file geometry descriptions are possible
  - e.g. OBJ with MTL
- Identifier of geometry objects in a larger geometry description can be added via FOG



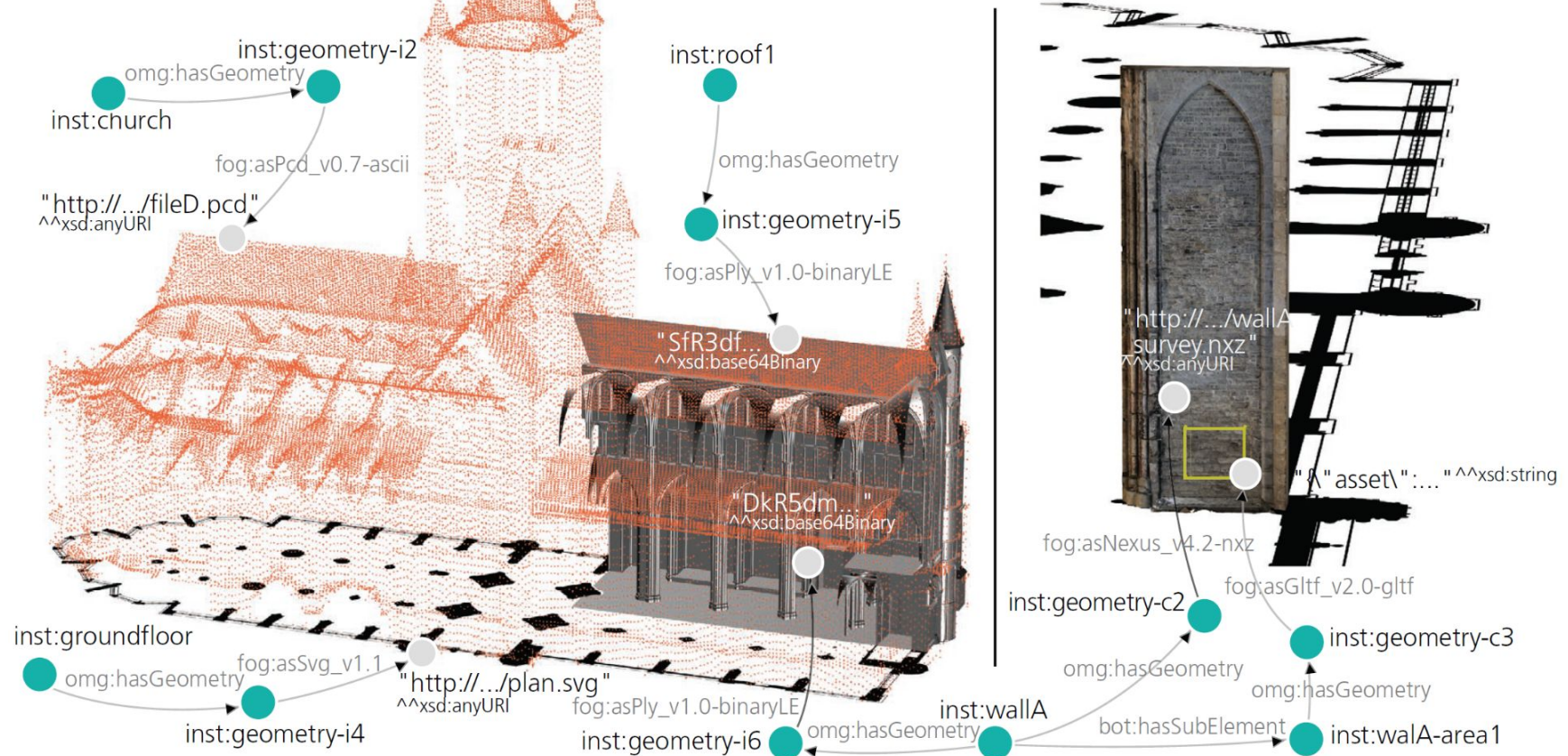
# FOG - Level 2

- Identifier of geometry objects in a larger geometry description can be added via FOG



[1] M. Bonduel, A. Wagner, P. Pauwels, M. Vergauwen, and R. Klein. "Including widespread geometry schemas into Linked Data-based BIM applied to built heritage". In: Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction 172.1 (2019), pp. 34–51. doi: 10.1680/jsmic.19.00014.

# OMG/FOG - Level 2 example





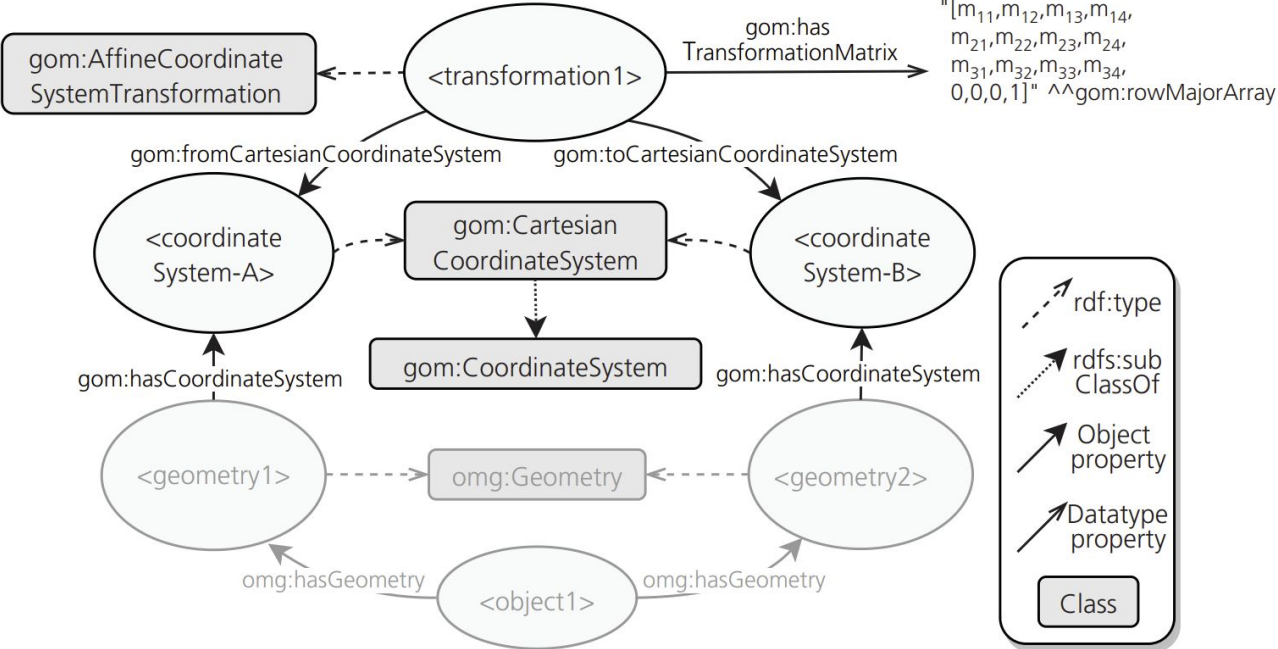
# OMG/FOG - Level 2 example

[1] M. Bonduel, A. Wagner, P. Pauwels, M. Vergauwen, and R. Klein. "Including widespread geometry schemas into Linked Data-based BIM applied to built heritage". In: Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction 172.1 (2019), pp. 34–51. doi: 10.1680/jsmic.19.00014.

- Visual created with three.js and Comunica.js for querying the RDF data
- Online sample: <https://mathib.github.io/fog-demo-app/>

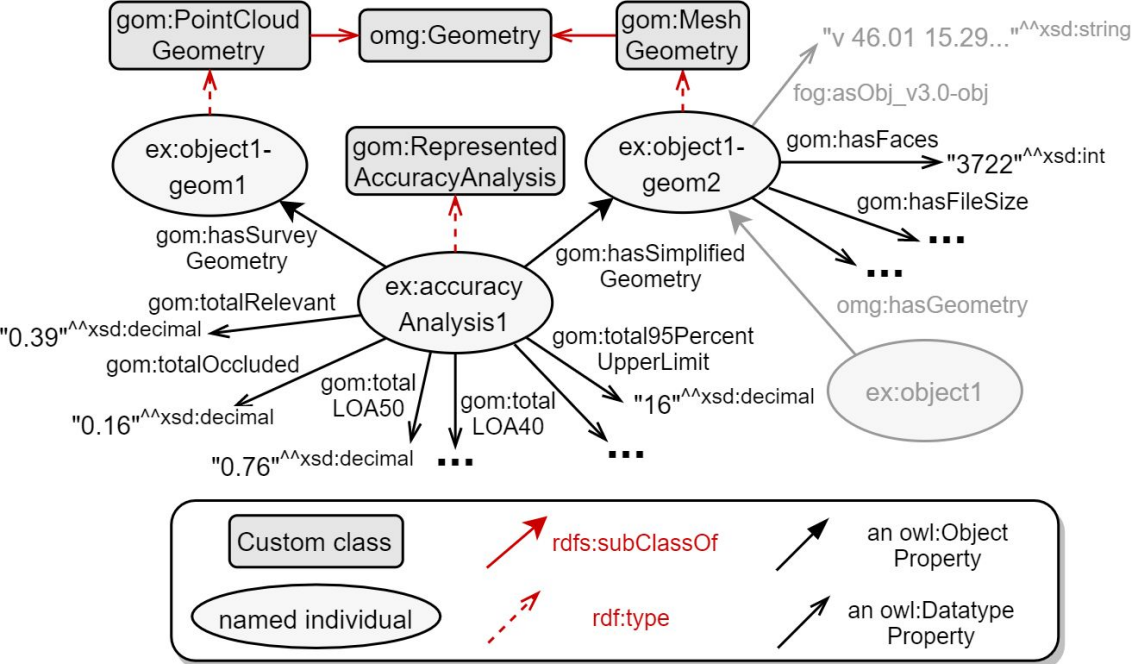
# GOM - Level 2

- Coordinate systems and their transformations can be added to the objectified geometry node



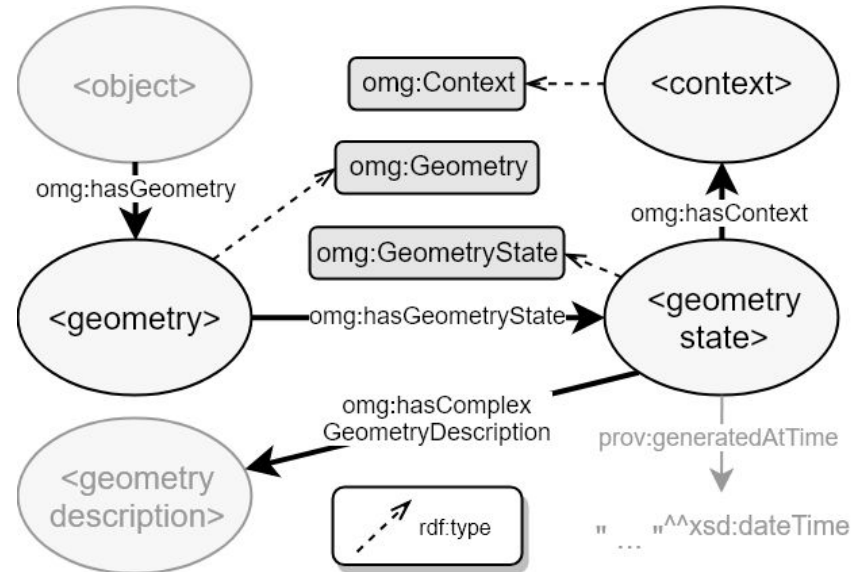
# GOM - Level 2

- Other metadata that can be added:
  - Length unit of the coordinate system
  - Geometry type (e.g. point cloud, plan geometry, etc.)
  - Geometric properties (e.g. surface areas, volumes, etc.)
  - File metadata (e.g. file size, used authoring tooling, etc.)



# OMG - Level 3

- intermediate 'GeometryState' node to add more metadata to the geometry
  - similar to `opm:PropertyState`
  - a timestamp can be added
  - `omg:CurrentGeometryState`
  - enables versioning of geometry
  - can also be related to a context
- Dependent geometry nodes and derived geometry state nodes
  - relation between geometries to define their actual derivation
  - can be used to query for outdated geometry descriptions



# OMG, FOG and GOM Conclusion

- Dedicated connector ontologies for geometry descriptions enable unified querying in disregard of the applied geometry formats
- Linking to **any** RDF-based and non RDF-based geometries:
  - included in the graph (geometry as a graph, or embedded in a string)
  - referenced external file
- Geometry is considered rather as property than as core feature
- Entities may have multiple geometry descriptions
  - For different purposes (e.g. bounding box, levels of details, etc.)
  - In different formats and schemas, as well as linking methods
- Dependencies between geometry descriptions allow modelling of conversion processes or transformations

# Domain Taxonomies

## Construction industry:

- [FreeClassOWL](#): [GoodRelations](#)-compliant definition of terms related to the construction industry, oriented to e-commerce; also related to Google's [schema.org](#)
- Product and property taxonomies: definition of building element types and properties (e.g. [BEO/MEP](#) derived from IFC)
- [Semantic bSDD](#): Linked Data interface to the buildingSMART Data Dictionary (WIP)

## Other domains:

- Units (e.g. Ontology-based Specification of [Quantities, Units, Dimensions and Types \[QUDT\]](#), [Custom Datatypes \[CDT\]](#), [Ontology of unity of Measure \[OM\]](#), etc.)
- Materials, construction-related domains, damages, classification of spatial elements, etc.

# Putting Everything Together

- BPO as metadata schema (core)
- Property modelling complexity chosen based on requirements
- OMG/FOG/GOM as dedicated connector ontologies
- Taxonomies to add domain classifications
- Alignment to domain ontologies to describe products in their systems

