Chowlk framework

María Poveda Villalón, Ontology Engineering Group
Serge Chávez-Feria, Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

[mpoveda@fi.upm.es]
[serge.chavez.feria@upm.es]

@MariaPovedaV
@SergeCF90

3rd December 2020
Thursday talk

License

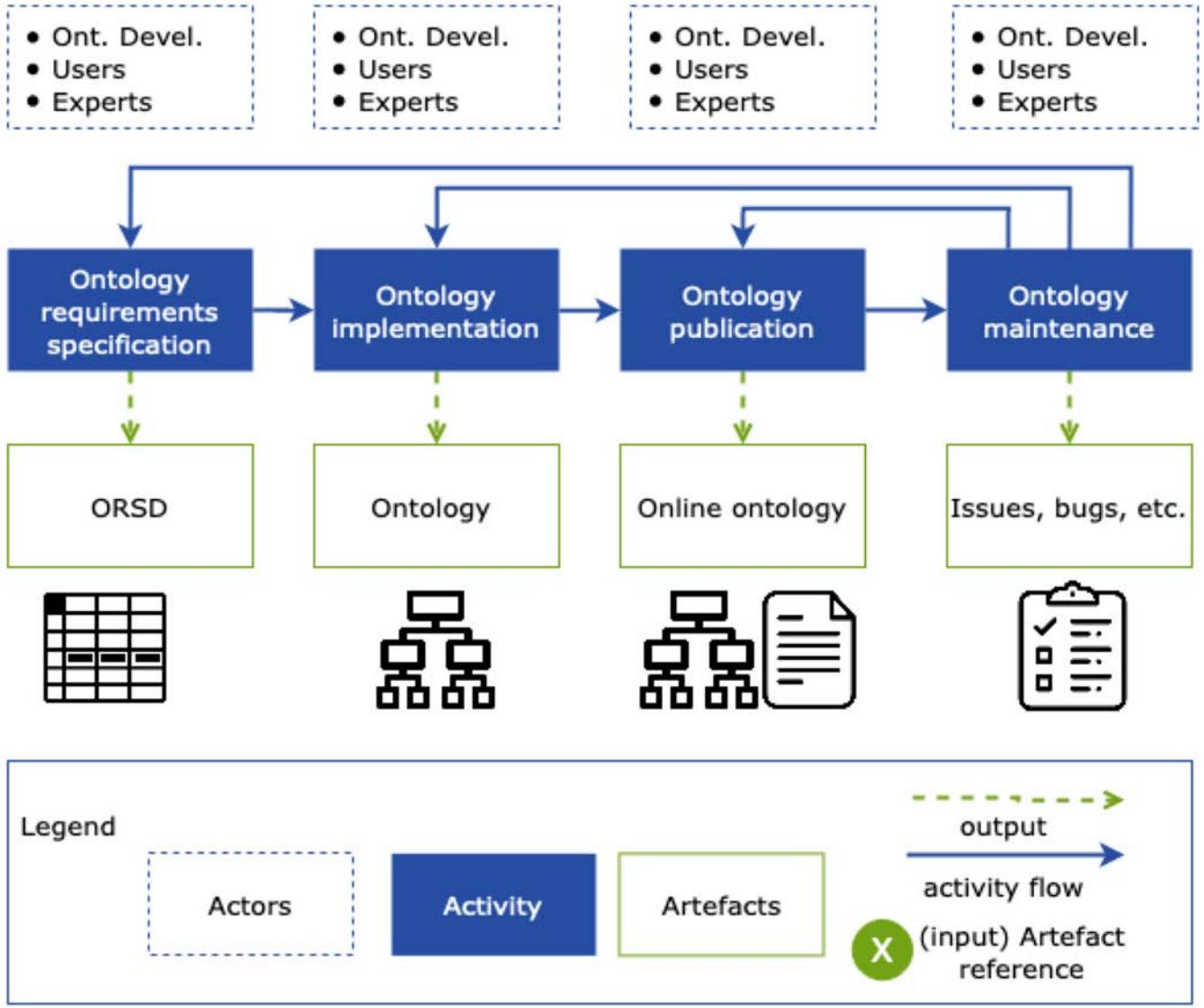
- This work is licensed under Creative Commons Attribution – Non Commercial – Share Alike License
- *You are free:*
  - *to Share — to copy, distribute and transmit the work*
  - *to Remix — to adapt the work*

- Under the following conditions
  - *Attribution — You must attribute the work by inserting*
    - "[source http://www.oeg-upm.net/]" at every reused slide
    - A slide declaring: "This material is partially based on "Chowlk framework" by María Poveda-Villalón and Serge Chávez-Feria"
  - *Non-commercial*
  - *Share-Alike*

No need to say but: Disclaimer

- Work in progress
- We welcome examples, criticisms, questions, suggestions, etc.

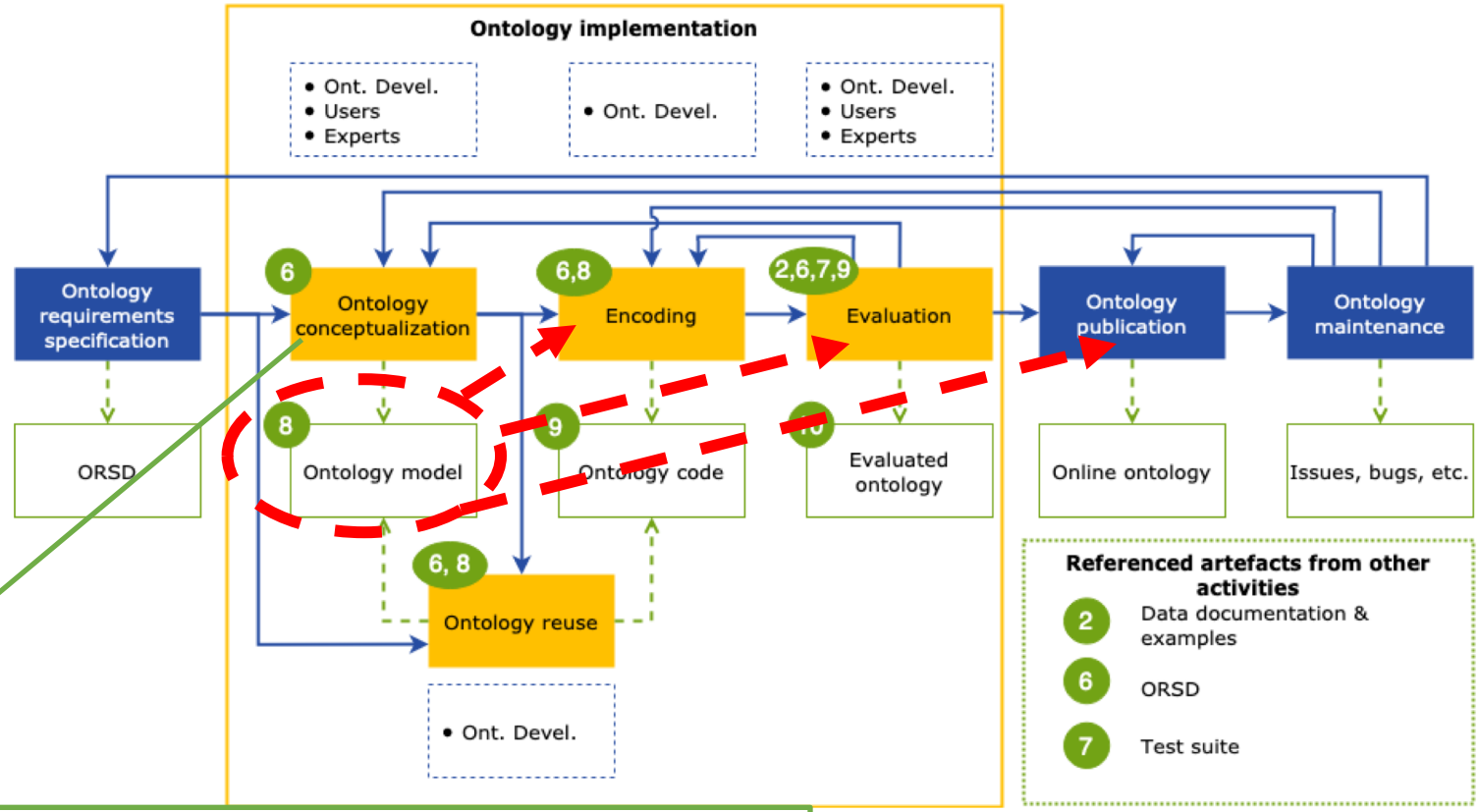
Ontology development process overview



http://lot.linkeddata.es/

Motivation

- Communication with domain experts
- Input for implementation
- Used in evaluation
- Used in publication
- Key for adoption



- **Goal**: build an ontology model from the ontological requirements identified.
  - Could be **graphical** or described in a formal system
- You can use
  - Blackboard
  - Pen & pencil
  - Drawing tools (diagrams.net, visio, yEd…)

- Different notation
- Different content
- Different level of detail

… in the best case scenario

Diagrams heterogeneity

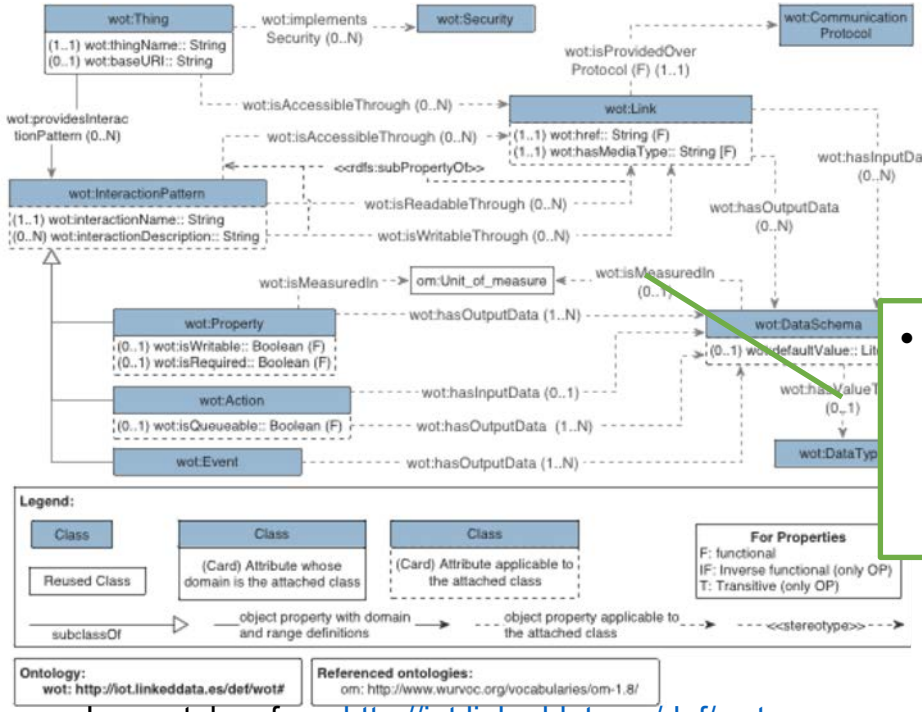


Image taken from http://iot.linkeddata.es/def/wot

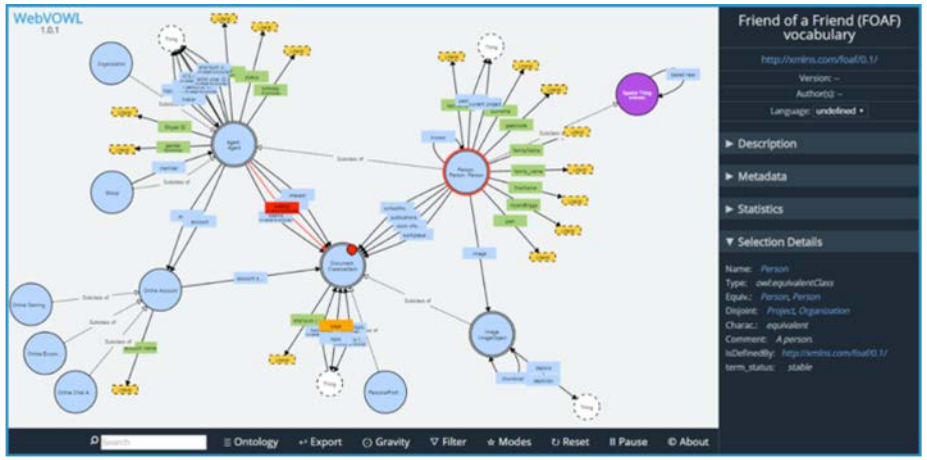


Image taken from http://vowl.visualdataweb.org/webvowl.html

- Based on UML_Ont (Haase, Peter, et al. "D1. 1.2 updated version of the networked ontology model." *NeOn Project Deliverable D* 1 (2009).)

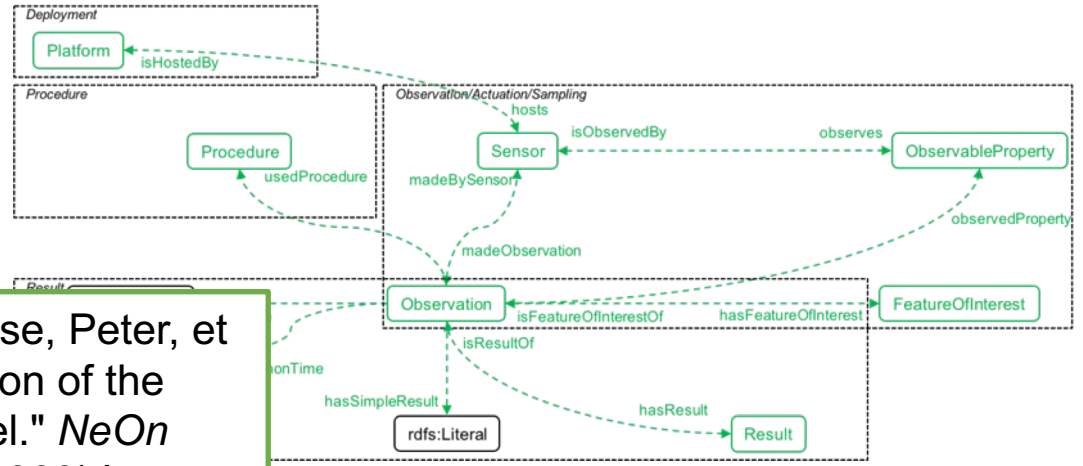


Image taken from https://www.w3.org/TR/vocab-ssn/

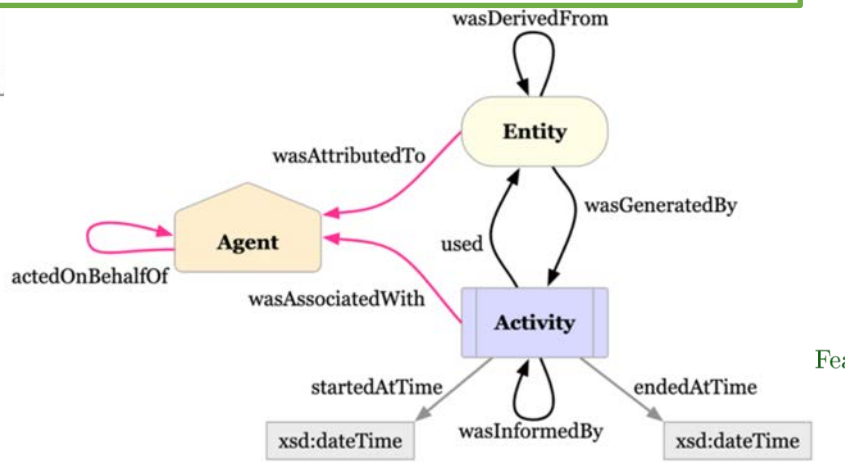


Image taken from https://www.w3.org/TR/2013/REC-prov-o-20130430/

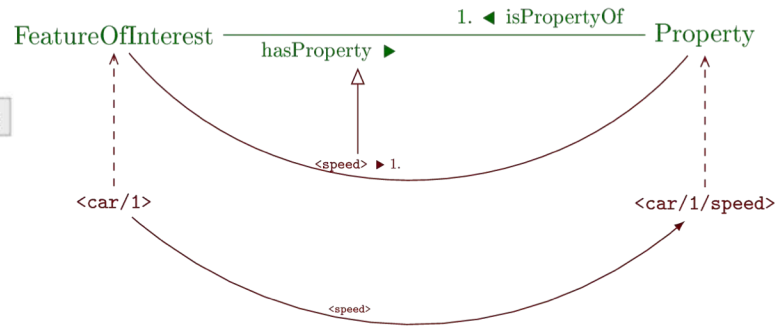
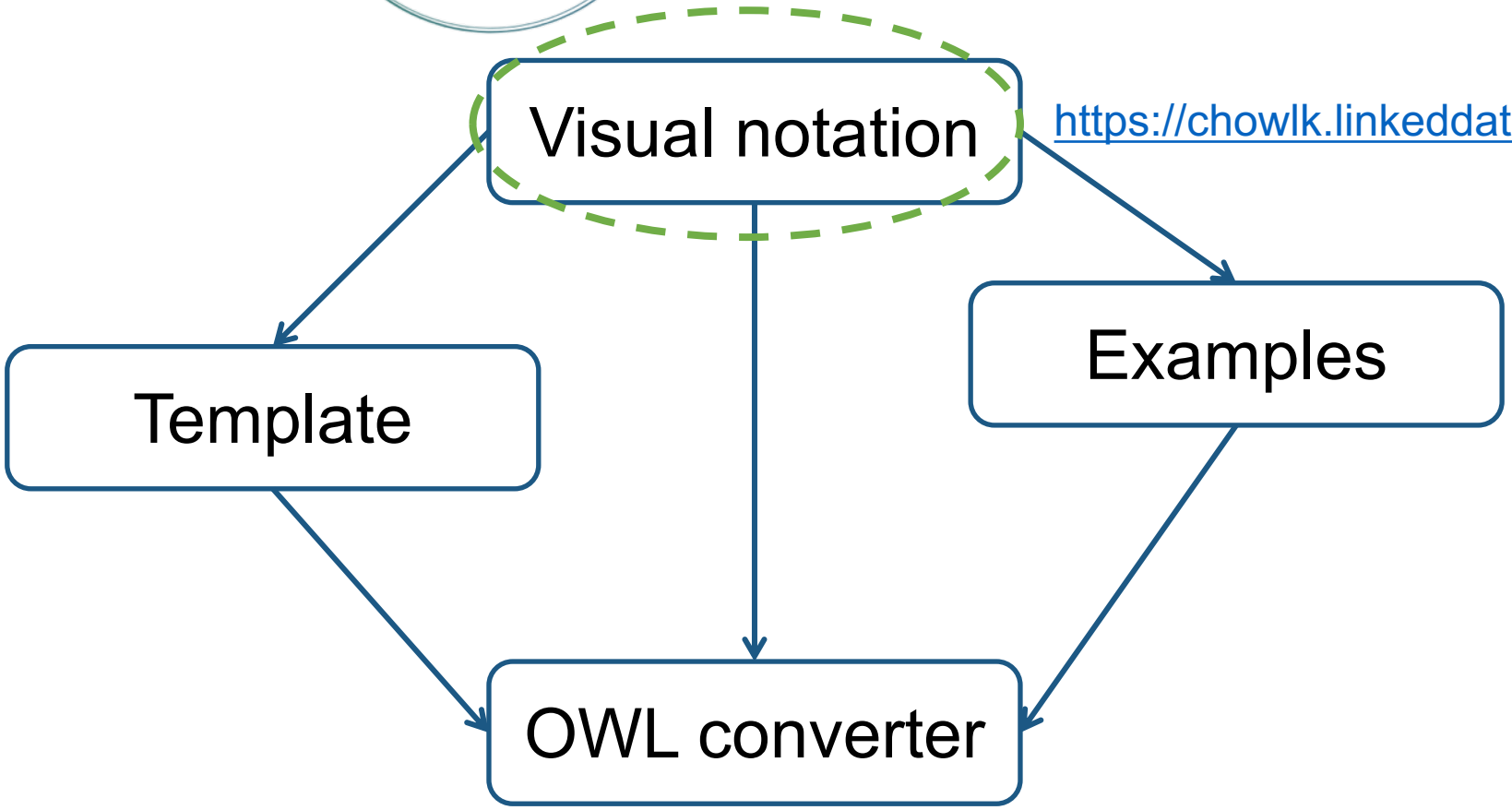


Image taken from https://ci.mines-stetienne.fr/seas/FeatureOfInterestOntology-1.0

https://chowlk.linkeddata.es

Chowlk

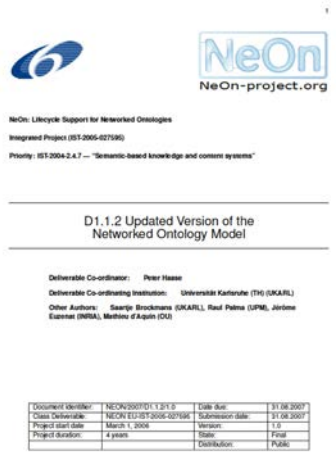
(pronounce as /ˈtʃɔːk/ (chôk))



Chowlk Ontology visual notation

https://chowlk.linkeddata.es/chowlk_spec

- UML-based
- Building blocks
  - OWL entities
    - Classes
    - Object properties
    - Datatype properties
    - Properties Characteristics
    - Relations between properties
    - Constraints
    - Individuals
  - *Metadata*
- Alternatives
  - Compact vs extended versions
  - Notation alternatives
- Other recommendations
  - Colour
  - Modules



UML_Ont (Haase, Peter, et al. "D1. 1.2 updated version of the networked ontology model." *NeOn Project Deliverable D* 1 (2009).

Adaptation for PhD dissertation

Blog posts

Book chapter “Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web” D. Garijo & M. Poveda-Villalón

Chowlk visual notation

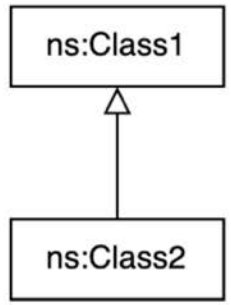
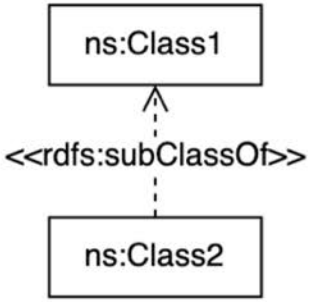
Classes

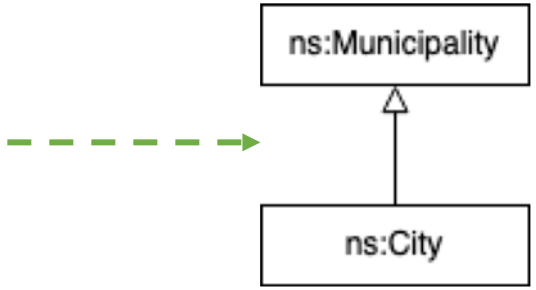
Diagram BLOCK	Description
ns:Class	Definition for a named class.
	Definition of an unnamed class. Usually used to represent axioms.
	Definition of an unnamed class to represent logical combinations between other classes, such as AND or OR operators

ns:Municipality

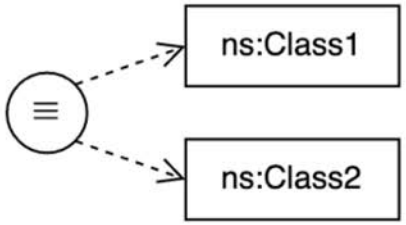
- Used for class constraints and compositions (later)

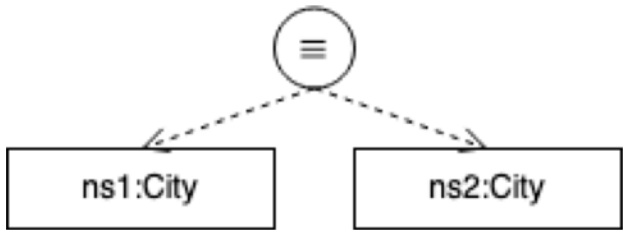
Classes – Subclass of

Diagram BLOCK	Description
1 	Option 1 to express that `ns:Class2` is sub-class of `ns:Class1`.
2 	Option 2 to express that `ns:Class2` is sub-class of `ns:Class1`.

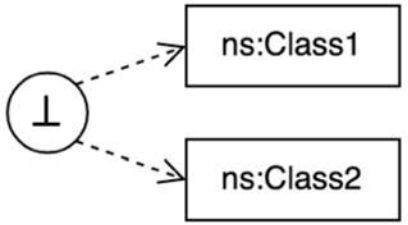


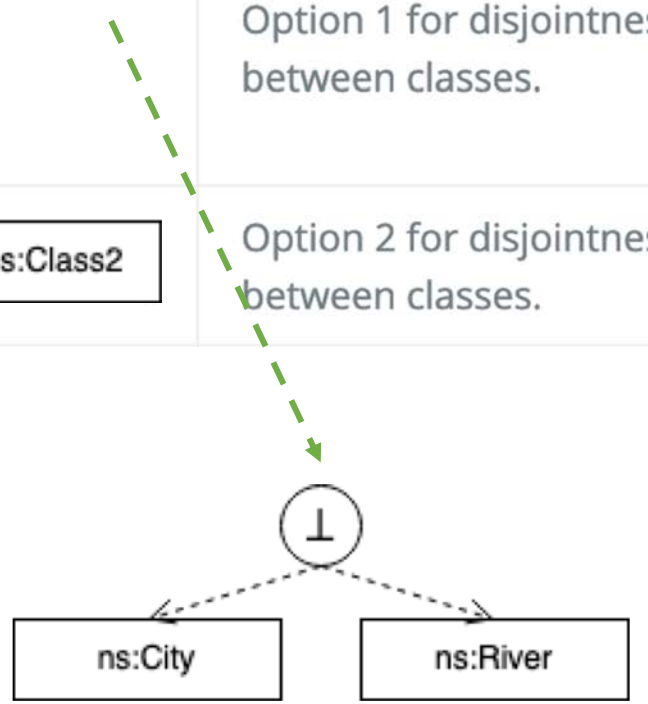
Classes – Equivalent

Diagram BLOCK	Description
1 	Option 1 for equivalent classes.
 2	Option 2 for equivalent classes.

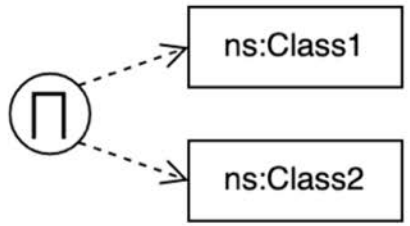
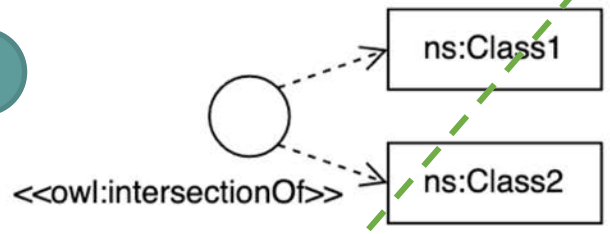


Classes – Disjoint

Diagram BLOCK	Description
1 	Option 1 for disjointness between classes.
 2	Option 2 for disjointness between classes.

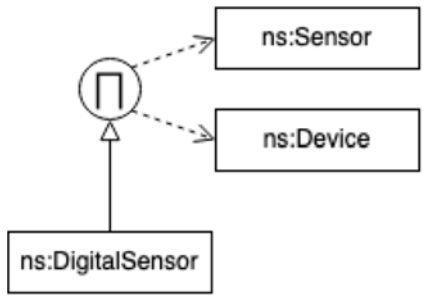


Classes - Intersection

Diagram BLOCK	Description
1 	Option 1 for intersection between classes.
2 	Option 2 for intersection between classes.

Could be also used for class constraints (later)

Subclass of use



Equivalent class of use

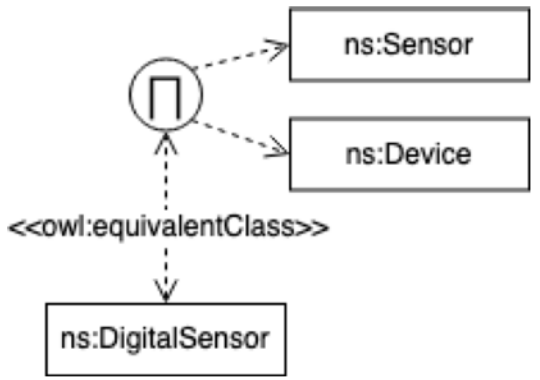
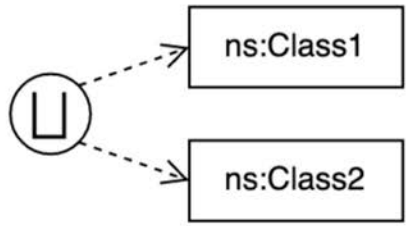
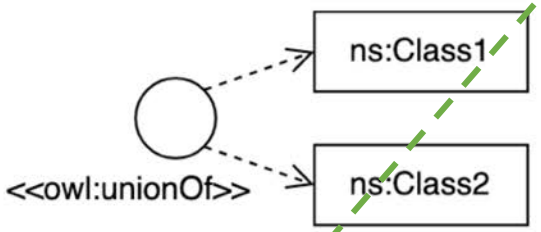
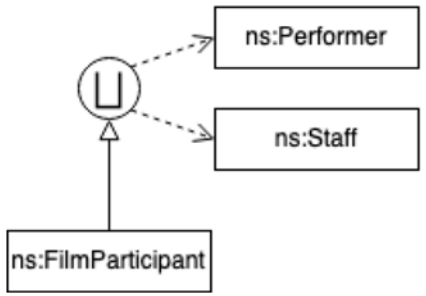


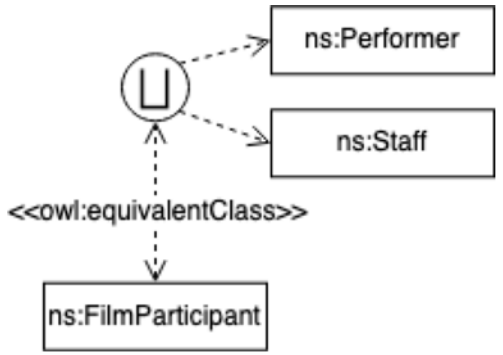
Diagram BLOCK	Description
1 	Option 1 for union of two concepts.
2 	Option 2 for union of two concepts.

Could be also used for class constraints (later)

Subclass of use



Equivalent class of use

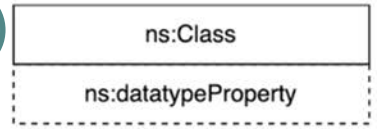


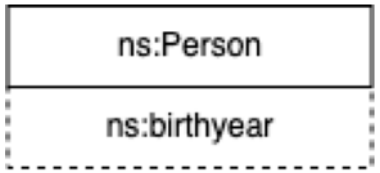
Classes - Complement

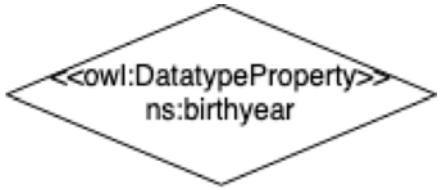
Diagram BLOCK	Description
	Option 1 for complement of two concepts.

Could be also used for class constraints (later)

Properties: object and datatype properties

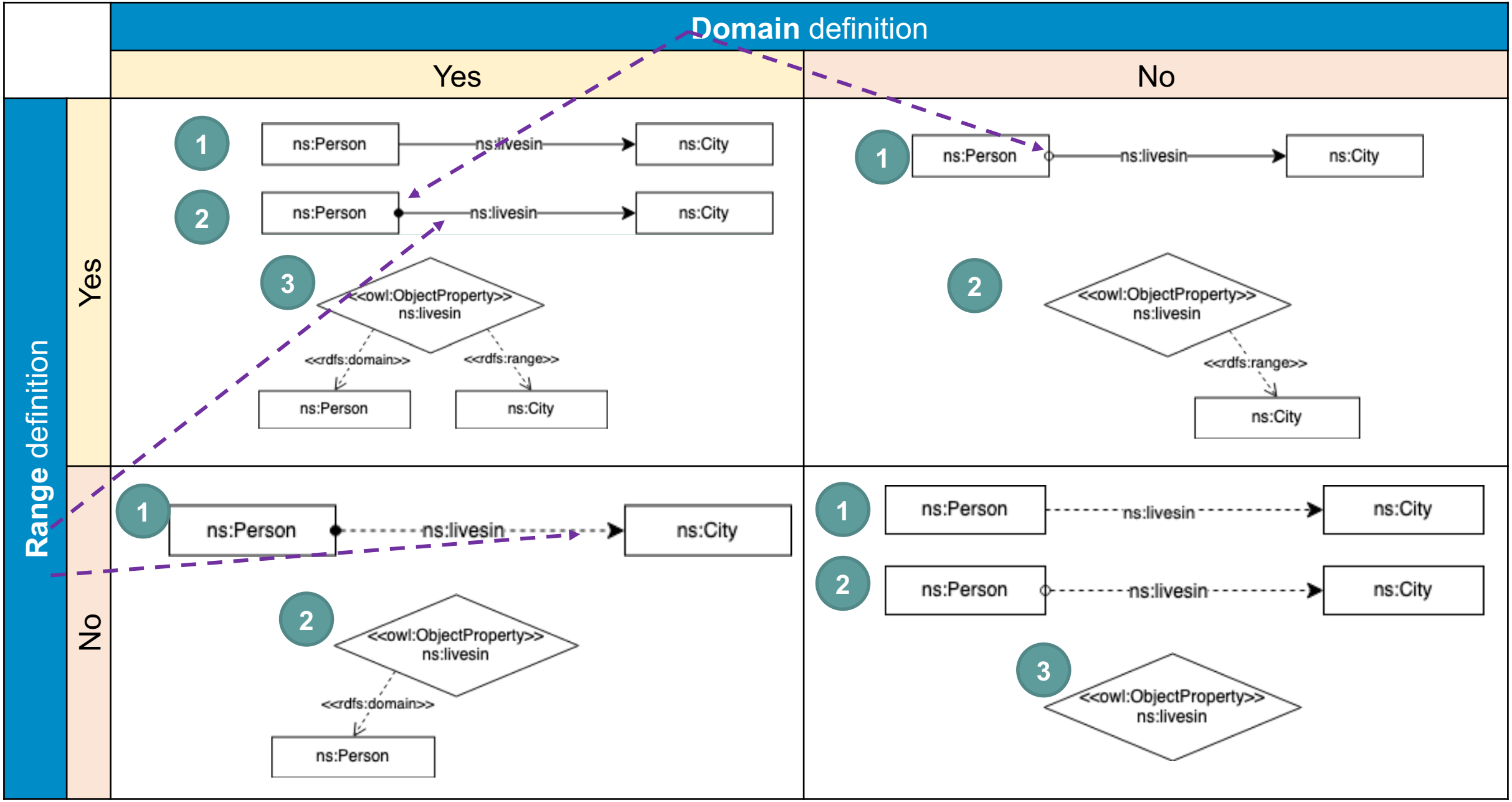
Diagram BLOCK	Description
1 	Standard way to represent object properties. Variations can apply to the type of line or the connections style depending on the range or domain specification. For more details see section 2.10.
2 	Alternative way to represent object properties. Mainly used to represent relations between the properties like `rdfs:subpropertyOf`.
1 	Standard way to represent datatype properties attached to a specific `owl:Class` element. Variations can apply to the type of outer line depending on the domain and range specification. For more details see section 2.11.
2 	Alternative way to represent datatype properties. Mainly used to represent relations between the properties like `rdfs:subpropertyOf`.



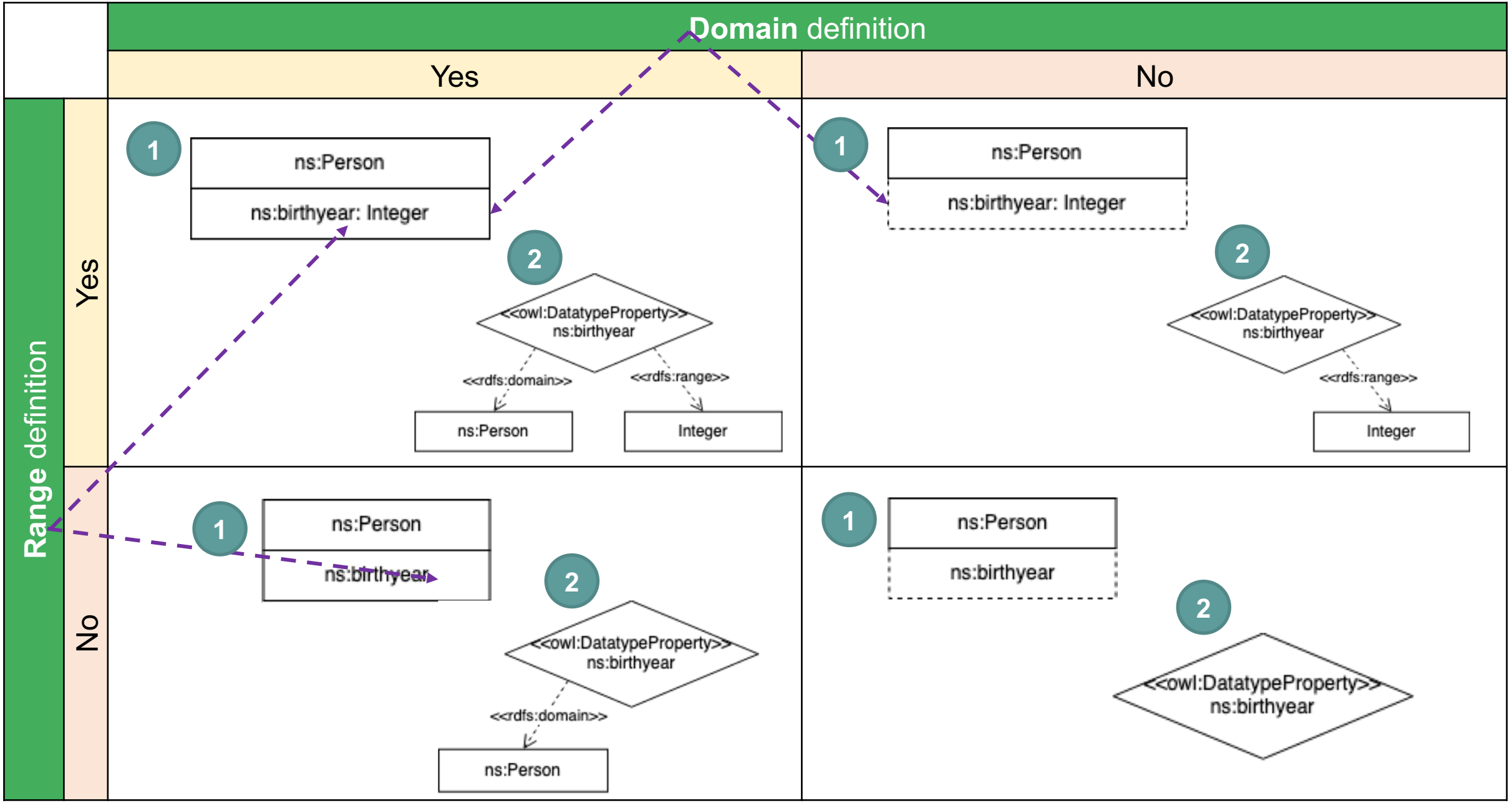


- Used for property hierarchies, inverse, equivalents, complex domain/range…

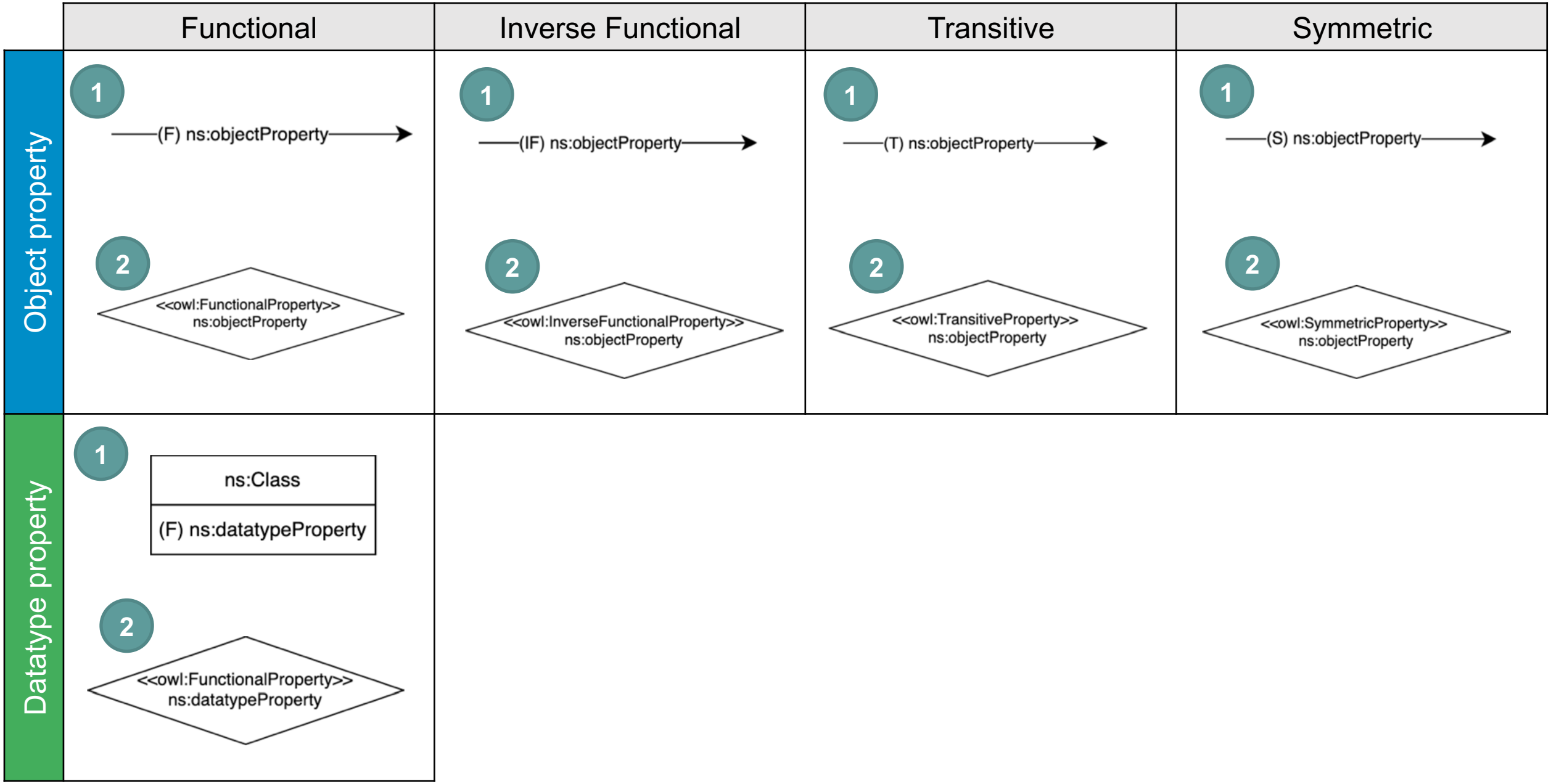
Object properties – domain and range



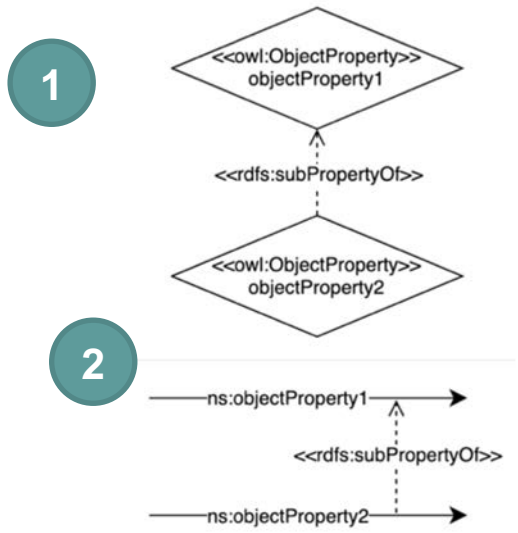
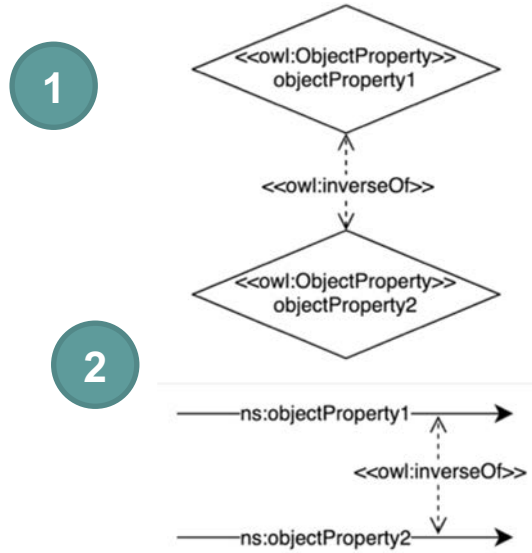
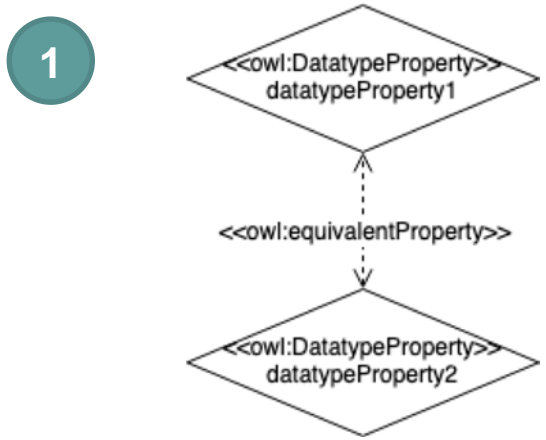
Datatype properties – domain and range



Properties characteristics



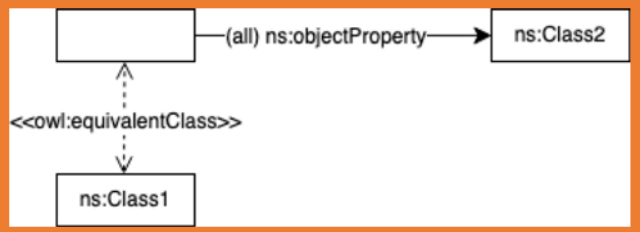
Relations between properties

	Subproperty	Equivalent	Inverse
Object property			
Datatype property			

Constraints

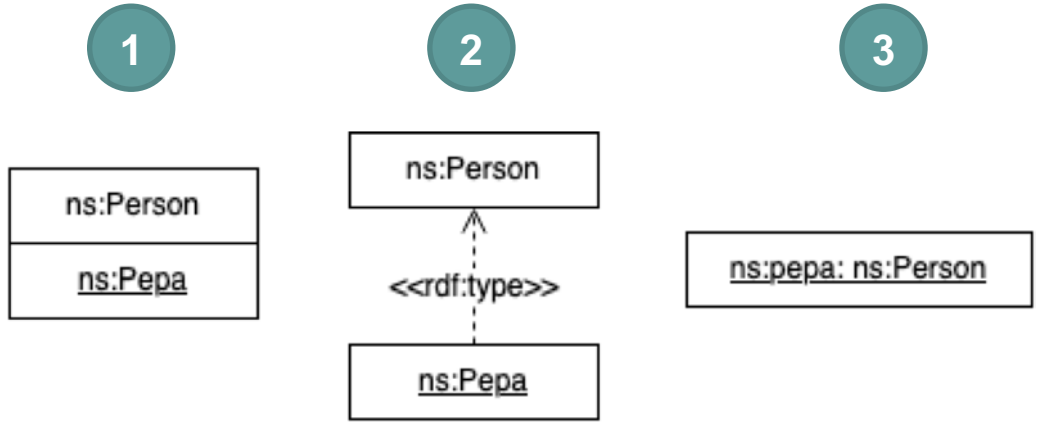
	Universal	Existential	Cardinality
Object property	1 ns:Class1 —(all) ns:objectProperty→ ns:Class2 2 ns:Class1 —(∀) ns:objectProperty→ ns:Class2 3 ns:Class1 —<<owl:allValuesFrom>> ns:objectProperty→ ns:Class2	1 ns:Class1 —(some) ns:objectProperty→ ns:Class2 2 ns:Class1 —(∃) ns:objectProperty→ ns:Class2 3 ns:Class1 —<<owl:someValuesFrom>> ns:objectProperty→ ns:Class2	1 ns:Class1 —ns:objectProperty (N1..N2)→ ns:Class2
Datatype property	1 ns:Class (all) ns:datatypeProperty: datatype 2 ns:Class (∀) ns:datatypeProperty: datatype	1 ns:Class (some) ns:datatypeProperty: datatype 2 ns:Class (∃) ns:datatypeProperty: datatype	1 ns:Class ns:datatypeProperty (N1..N2)

Attention! By default indicates a subclassOf axiom constraint for ”ns:Class1”.
For equivalent class combine with
equivalent class axiom + anonymous class

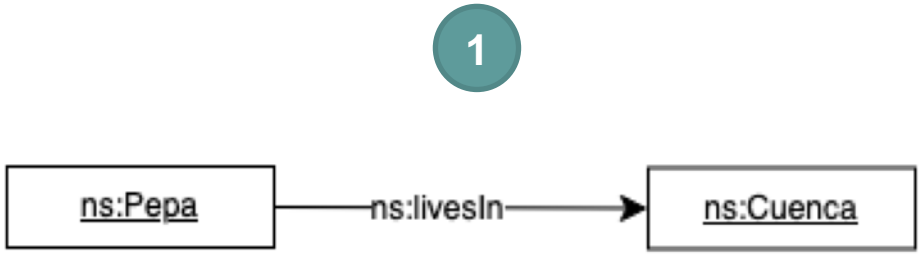


Individual declaration

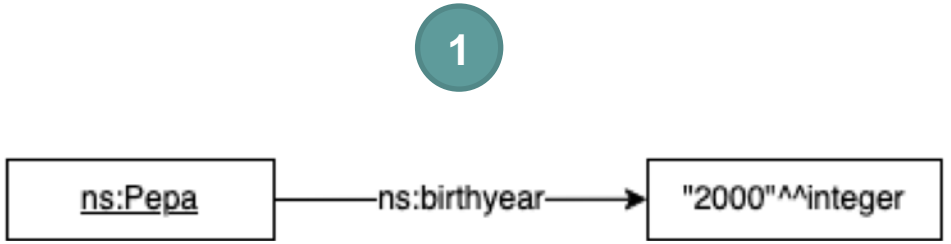
Class membership



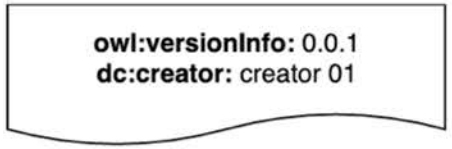
Object property usage



Datatype property usage



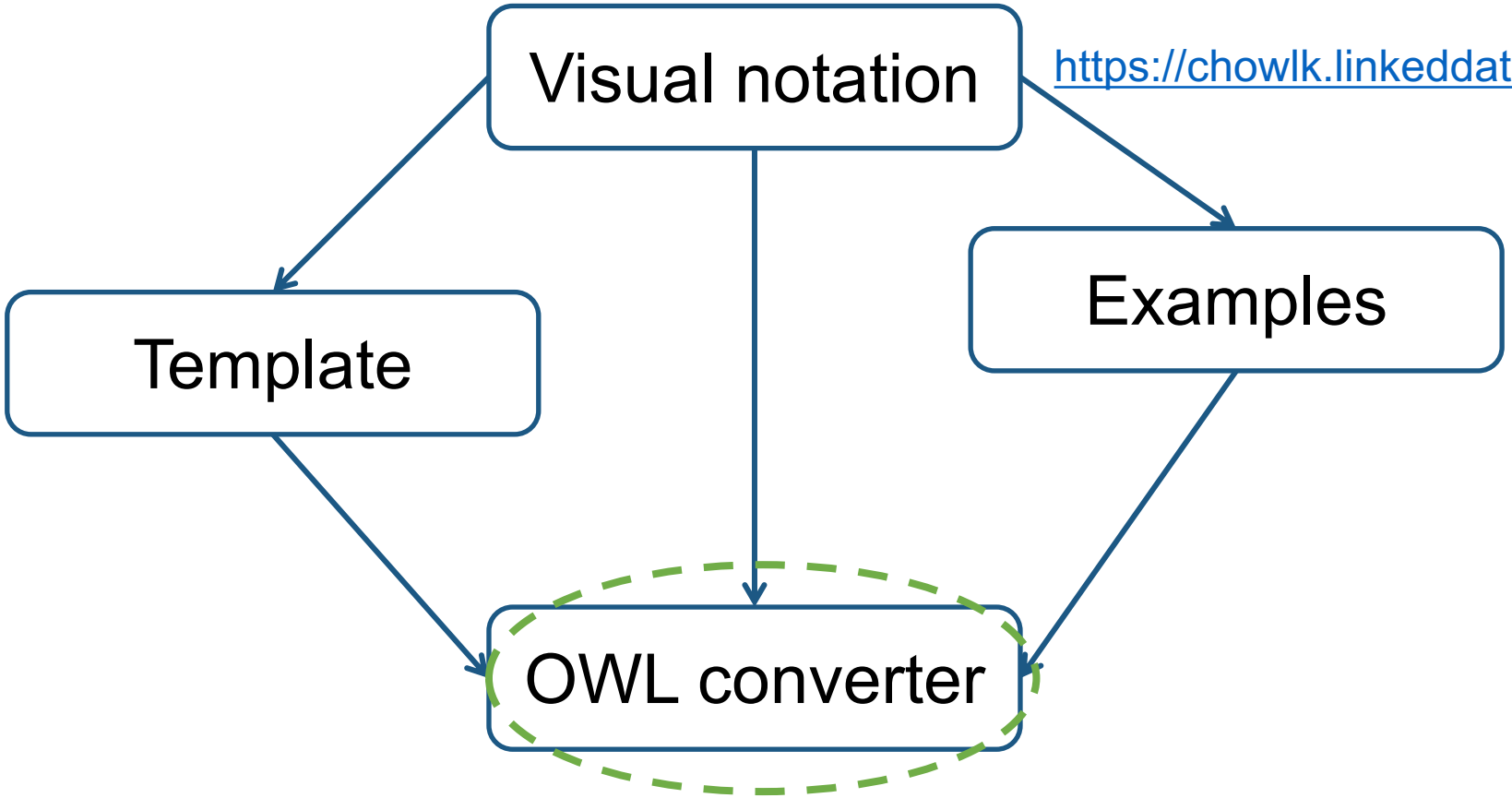
Metadata blocks

	Block to indicate all the namespaces used in the ontology. The first namespace is the URI used for the current ontology. It is obligatory to include all the namespaces being used in order to use the ontology converter service.
	Block to indicate the annotation properties describing the ontology. The annotations in use should include the prefix and the annotation name, as indicated in the figure. If custom annotations are utilized, the namespace block should include the prefixes and namespaces for those annotation properties.

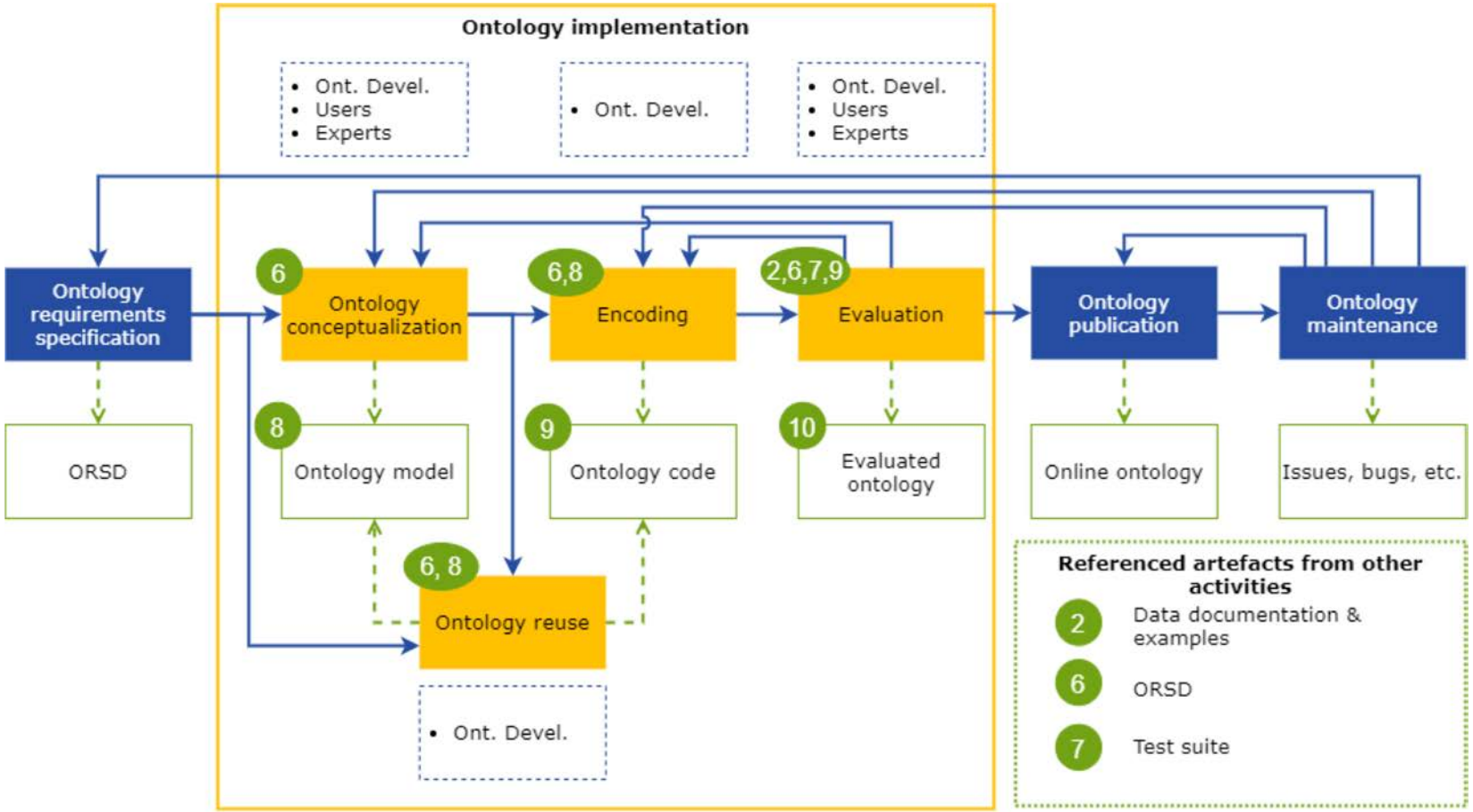
Chowlk

https://chowlk.linkeddata.es

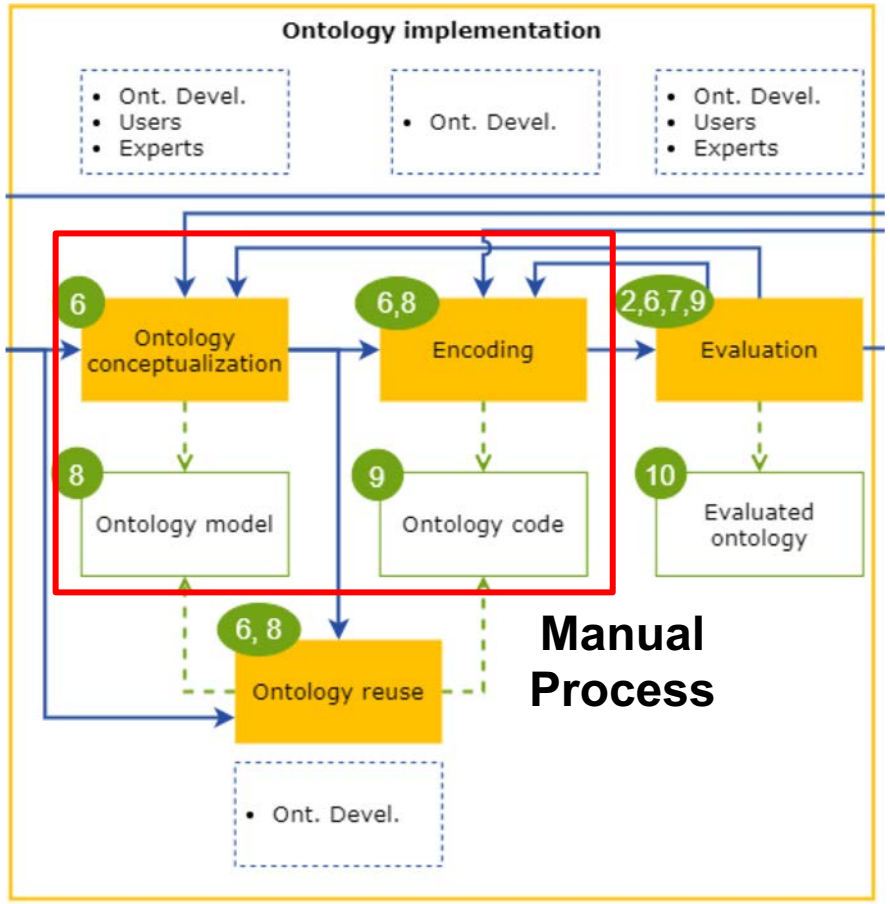
(pronounce as /ˈtʃɔːk/ (chôk))



- The current methodology

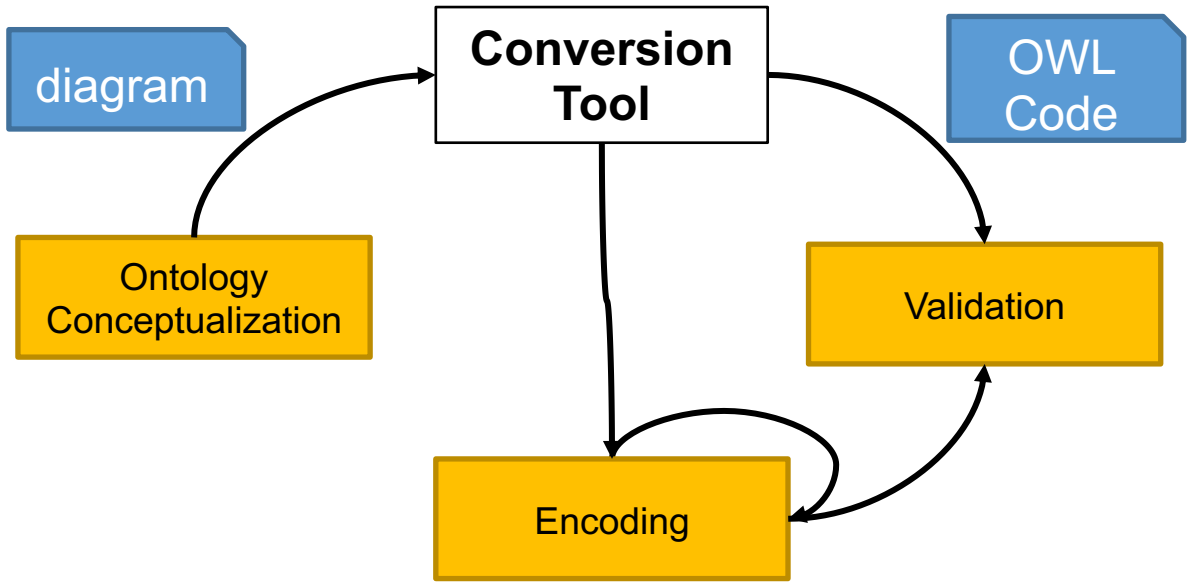


- The current methodology



- It is a repetitive process.
- A direct transformation to OWL could sometimes be done.
- During the implementation we can commit errors.

- So why not to develop an automatic tool to generate the OWL code from the conceptualization?

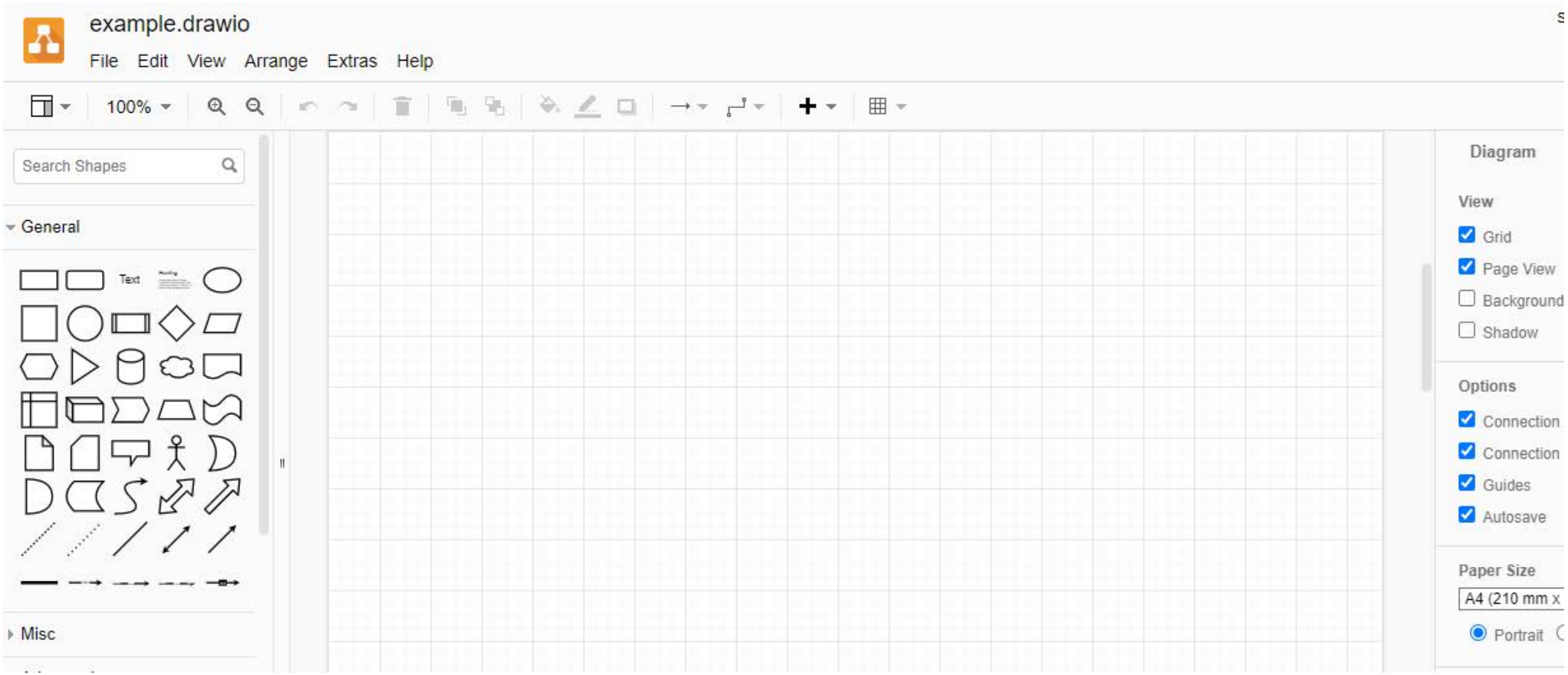


- **But first of all, which software do we use to make our diagram?**

- **Requirements**:
  - Be flexible enough to represent all the elements in the visual notation.
  - Allow us to export the diagram in a structured form to be easily parsed.
  - In the best scenario, should be a web-based open source platform.

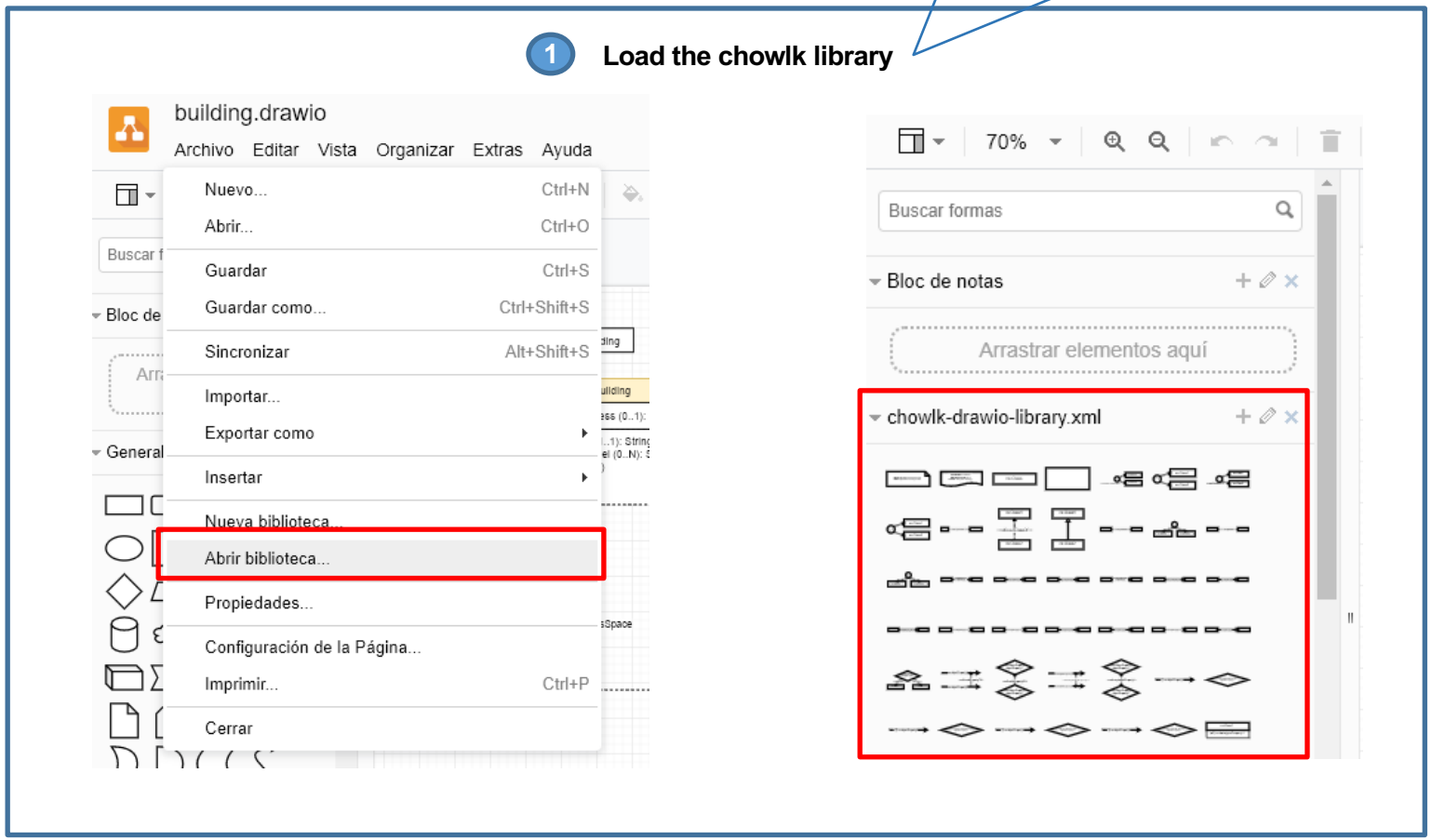
- The winner for us is: diagrams.net

The artist formerly known as draw.io

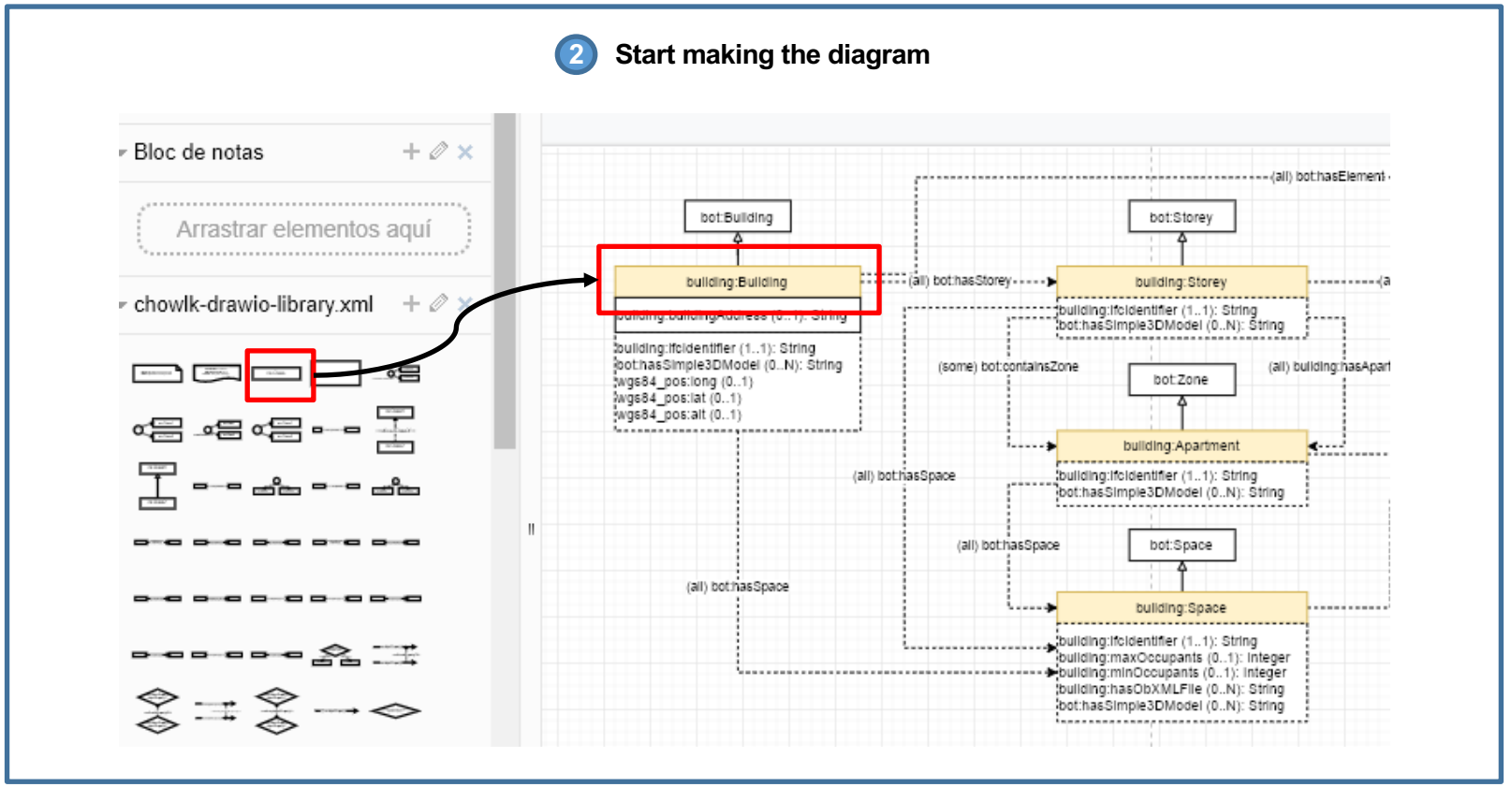


- How to use it to make ontologies?

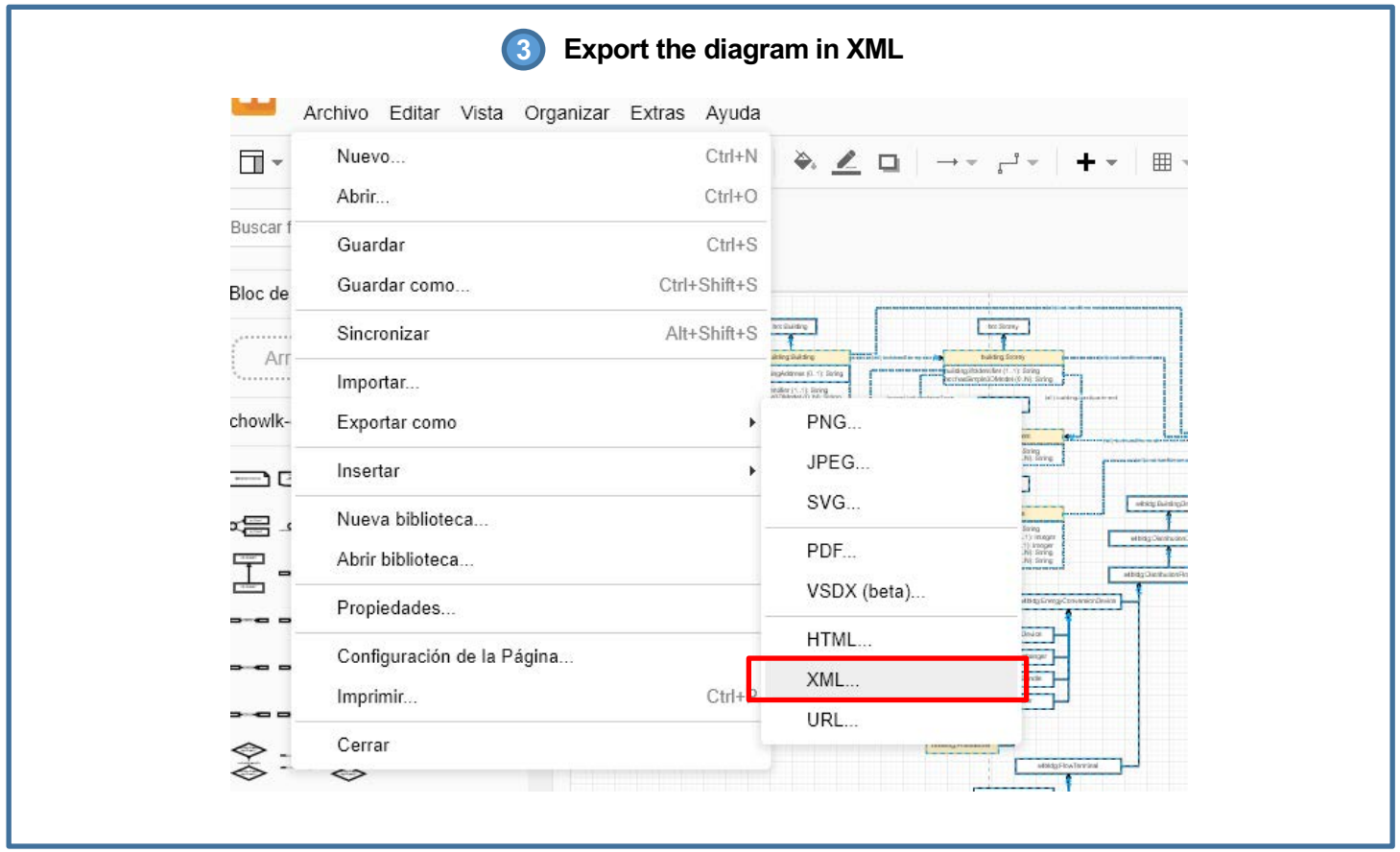
You can find the notation library at https://chowlk.linkeddata.es/resources/chowlk-drawio-library.xml



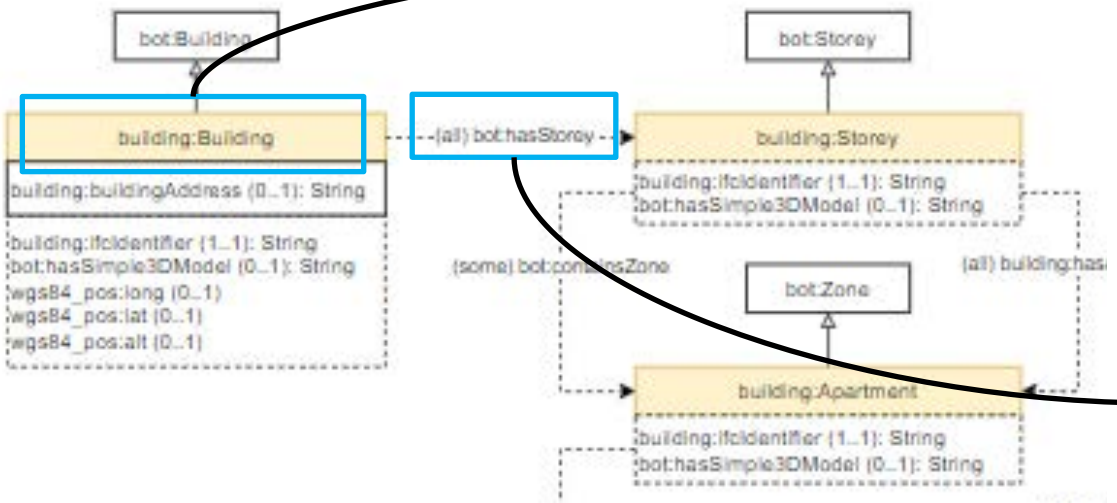
- How to use it to make ontologies?



- How to use it to make ontologies?

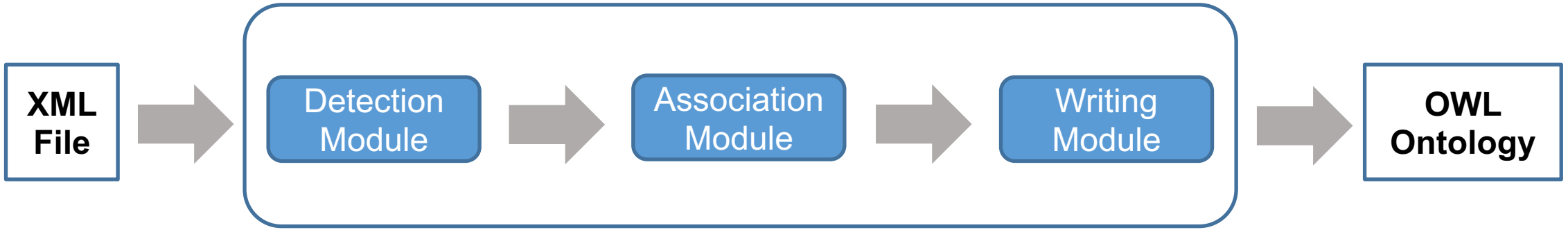


- The XML structure of the diagram

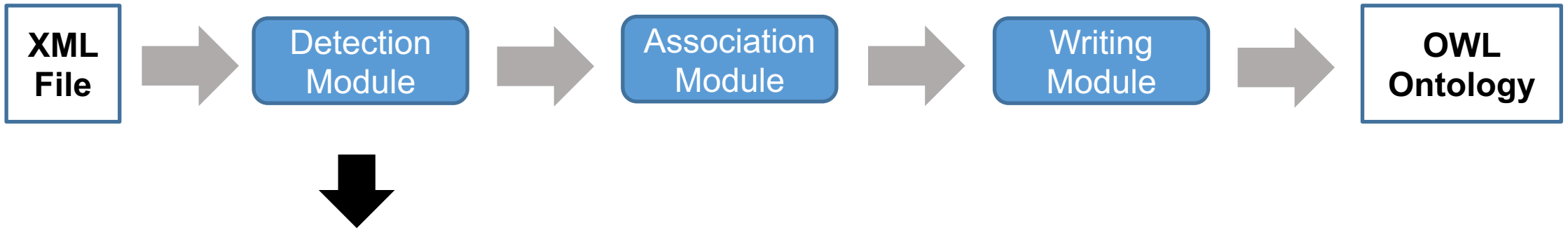



```
<?xml version="1.0" encoding="UTF-8"?>
<mxfile host="Chrome" modified="2020-11-16T14:41:32.661Z" agent="
  <diagram id="hJhljKE25JFFkrsz6_xK">
    <mxGraphModel dx="20674" dy="19061" grid="1" gridSize="1" gui
      <root>
        <mxCell id="0" />
        <mxCell id="1" parent="0" />
        <mxCell id="2" style="edgeStyle=orthogonalEdgeStyle;round
          <mxGeometry relative="1" as="geometry">
            <mxPoint x="-17271" y="-17490" as="targetPoint" />
          </mxGeometry>
        </mxCell>
        <mxCell id="3" value="building:Building" style="rounded=0
          <mxGeometry x="-17377" y="-17470.239999999998" width="2
        </mxCell>
        <mxCell id="4" value="(all) bot:hasStorey" style="edgeSty
          <mxGeometry x="-0.1468" y="2" relative="1" as="geometry
            <Array as="points">
              <mxPoint x="-17470" y="-17457" />
              <mxPoint x="-17470" y="-17403" />
            </Array>
            <mxPoint as="offset" />
          </mxGeometry>
        </mxCell>
```

- **The Architecture**

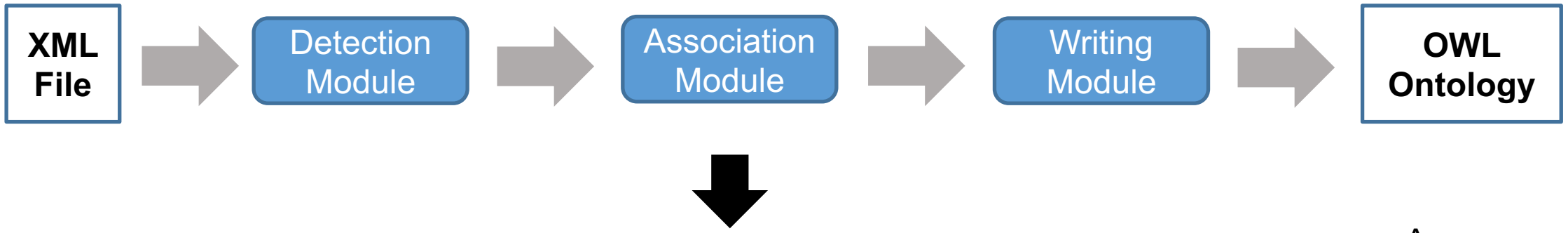


- **The Architecture**

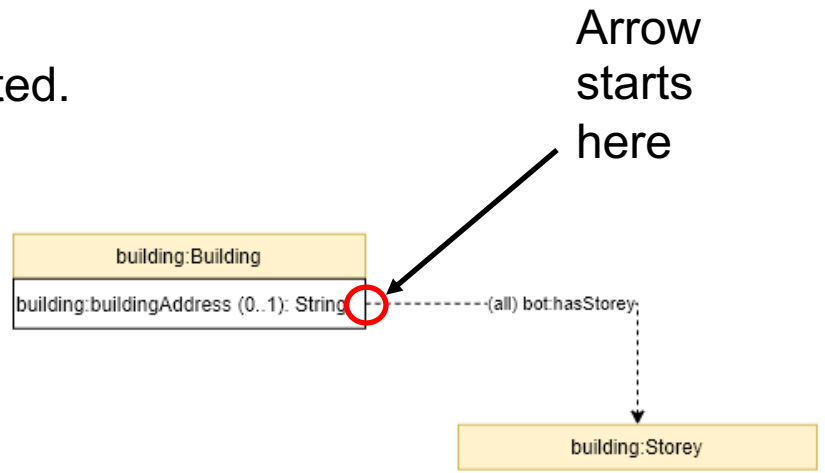


- Checks the syntax of the blocks
- We can adapt this module to support other visualizations.
- If the Chowlk notation is not follow it provides a report.
- Some blocks are identified directly by checking the syntax, others need context.

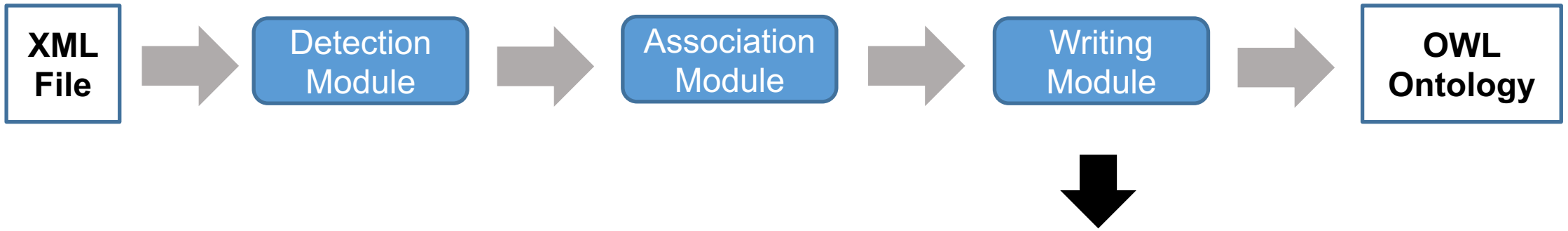
- **The Architecture**



- Association between elements detected.
- Concept + relations + attributes.
- Need to disambiguate certain associations.



- **The Architecture**

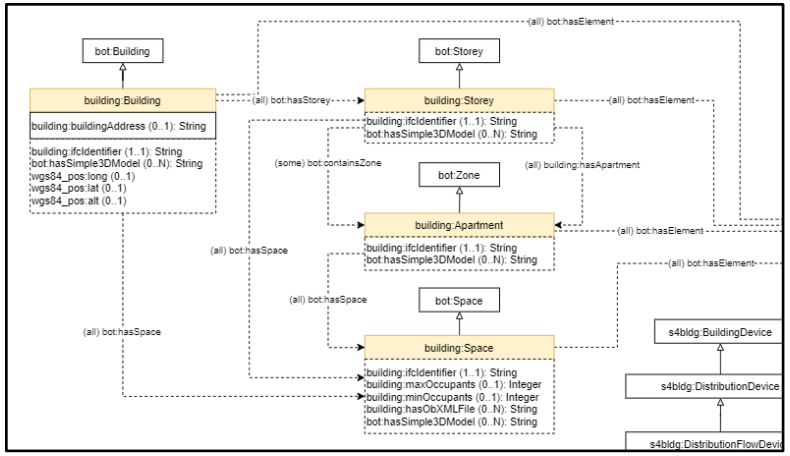


- Write each element in TTL.
- Labels automatically generated from URI.
- Include certain namespaces by default.
- Export it also in RDF/XML.

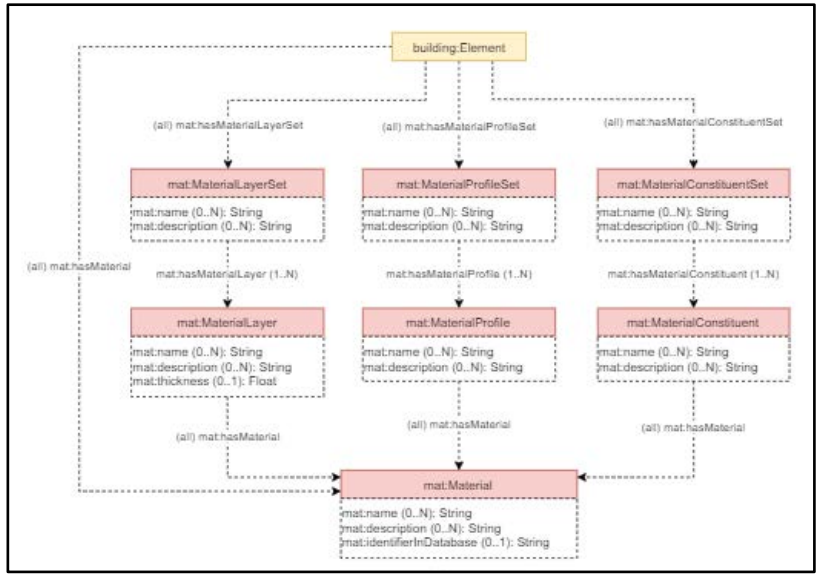
Demo Time

- The BIMERR Ontology Network

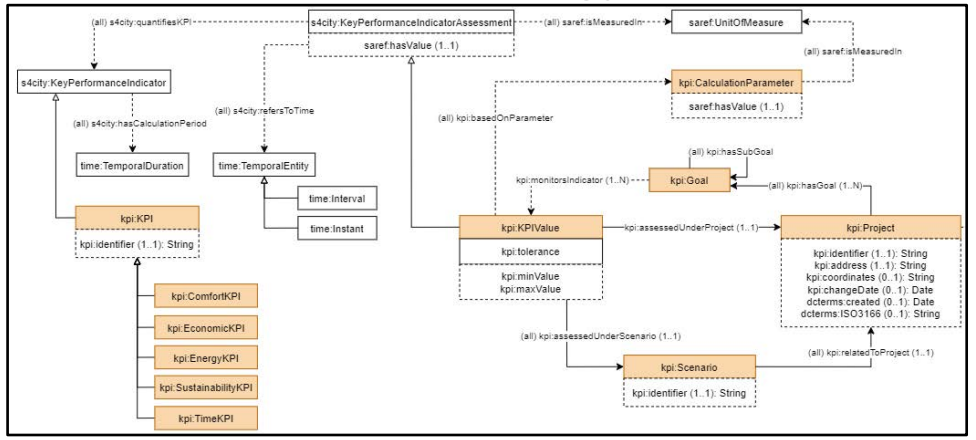
Building Ontology



Materials Ontology



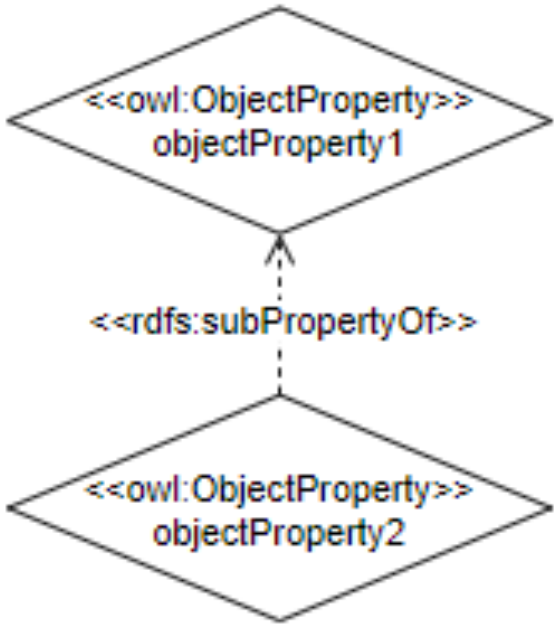
KPI Ontology



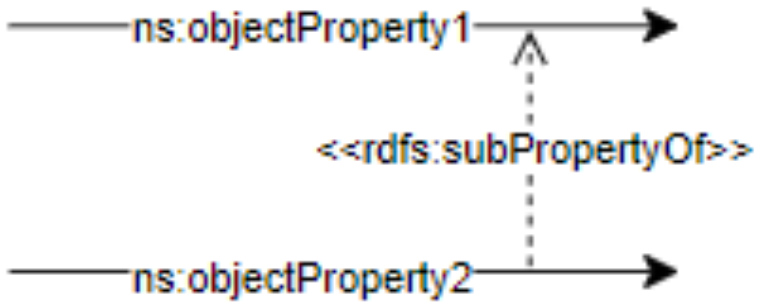
and so on …

- The converter only supports relationships between properties (e.g. subPropertyOf) using the option 2 of the visual notation.

Supported



Not Supported



Current Limitations

- The converter only supports relationships between properties (e.g. subPropertyOf) using the option 2 of the visual notation.
- If the model is big or complex, detecting and correcting syntax errors can be tedious.
- Comments, descriptions or additional metadata over the ontological elements is not supported.

Future Work

- The implementation of a plugin in diagrams.net to accelerate even more the construction of the ontology.
  - Also to detect and correct syntax errors rapidly.
- To explore the usage of the tool with other visualizations by adapting the detection module.
- To explore the reverse process, from an ontology to a diagram in diagrams.net.
- To integrate it into OnToology?
- Systematic comparisson of the visual notation with other existing ones + user-based evaluation.
- Submit Chowlk converter (+ lightweight description of notation) to ESWC resources

- Acknowledgments:
  - Thanks to Stephany Chávez for the great logo

Chowlk framework

María Poveda Villalón, Ontology Engineering Group
Serge Chávez-Feria, Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

[mpoveda@fi.upm.es]
[serge.chavez.feria@upm.es]

@MariaPovedaV
@SergeCF90

3rd December 2020
Thursday talk