Tools and Services Walkthrough Session - Ontology - a more detailed Guide

Dr. Jörg Waitelonis, FIZ Karlsruhe / Philipp v. Hartrott, Fraunhofer IWM

WP 4 Core Team

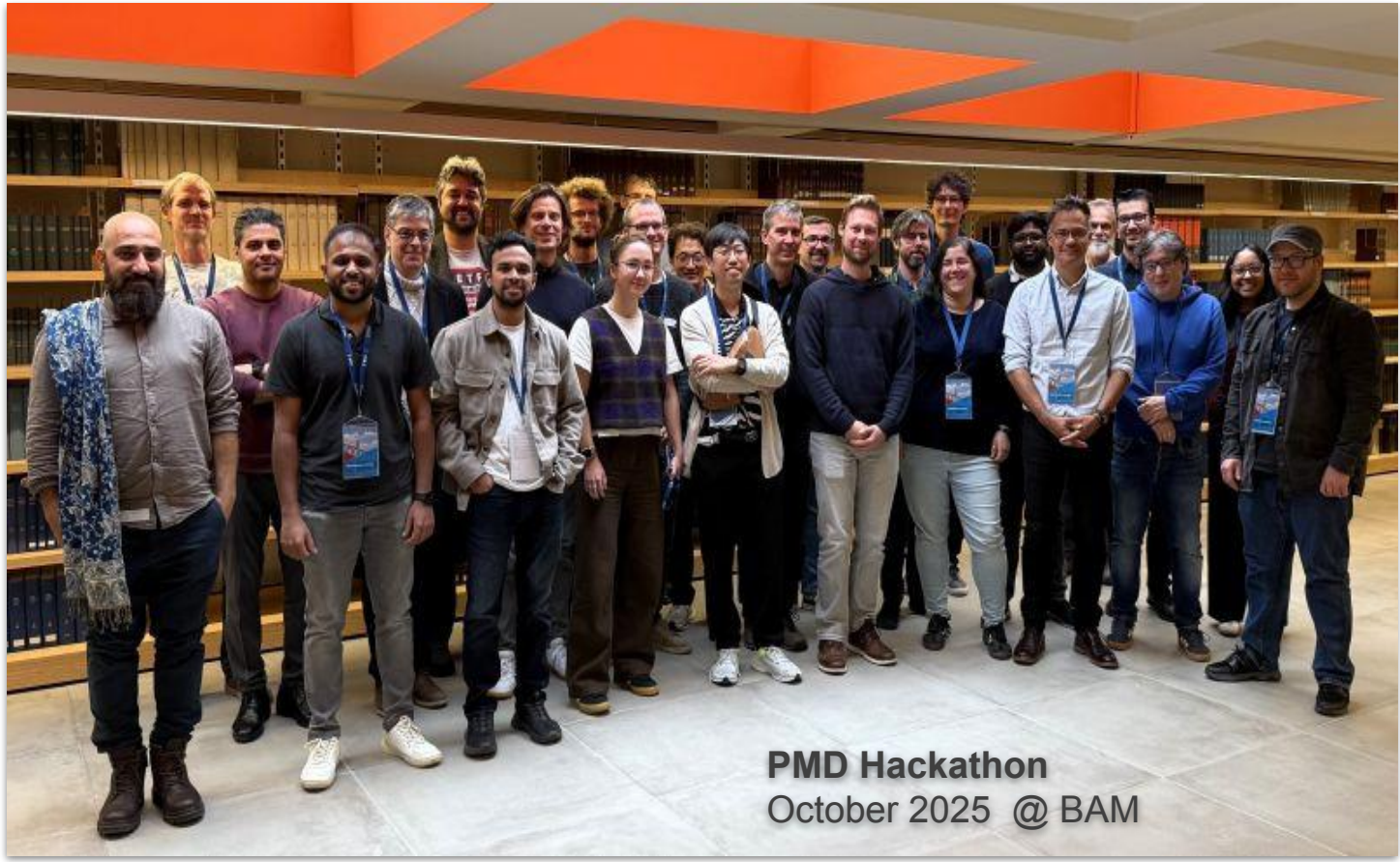
IWM *Fraunhofer Institut für Werkstoffmechanik*
Philipp von Hartrott
Dr. Thomas Hanke
Kamilla Zaripova

BAM *Bundesanstalt* für Materialforschung und -prüfung
Dr. Markus Schilling
Dr. Bernd Bayerlein
Dr. Khashayar Razghandi

FIZ *Leibniz-Institut für Informationsinfrastruktur*
Dr. Jörg Waitelonis
Dr. Hossein Beygi Nasrabadi
Dr. Gunjan Singh
Dr. Alsayed Algergawy

IWT *Leibniz Institut für Werkstofforientierte Technologien*
Dr. Henk Birkholz
Felix Thonagel
Divya Gosula

… with ongoing engagement and collaboration within the **community**…



Agenda

	Part 1:
09:00 – 10:30	**Fundamentals of ontology development with reference to PMDco**
10:30 – 11:00	Break
	Part 2:
11:00 – 12:00	**Example of describing material and process using PMDco**
12:00 – 13:00	Lunch Break (60min)
13:00 – 14:30	**Continuation**
14:30 – 15:00	Break
15:00 – 16:00	**Wrap-up & Feedback / Your modelling Q & our modelling A** Share your modelling challenges

PMDco:
Ontology Development

Content

- Recap
- Consensus Building Process
- Ontology Development Process
  - the 15 steps
  - best practices
  - do’s and don’t and pitfalls
- How To Transform My Data

Warmup

https://miro.com/app/board/uXjVJnJwyj8=/?moveToWidget=3458764676180921357&cot=14

Recap

Recap

We already learned about:

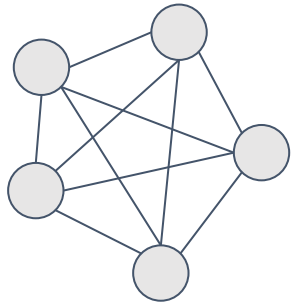
- Interoperability
- Ontology in philosophy and computer science
- RDF / OWL
- Triples and serialization
- Reasoning and triple stores
- Types of ontologies
- Protégé ontology editor
- BFO, RO & Co
- PMDco

Recap: Translation Problem

Solution 1:

- Between each dataset A and each dataset B a mapping is made

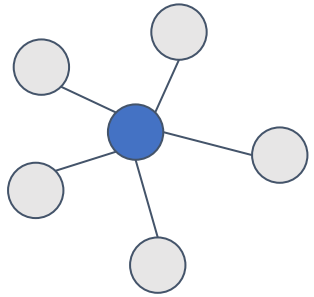
→ requires $\mathbf{n^2}$ “translators”



Solution 2:

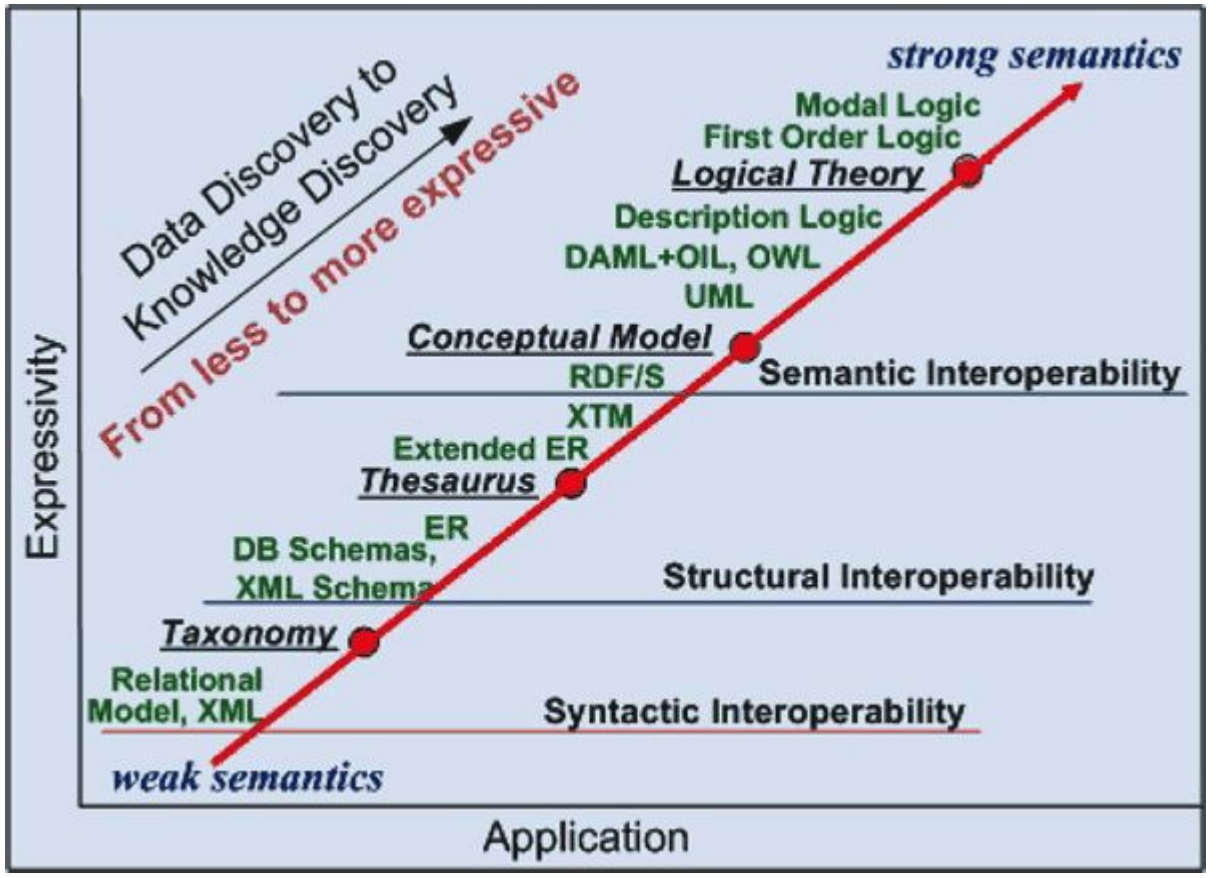
- Independent representation scheme (*Interlingua*)

→ requires **n** “translators”



Recap: The semantic ladder

- use a common language ...
- shared symbols and terms (**syntax**)
- agreement on their meaning (**semantics**)
- classification of terms (**taxonomy**)
- associations and connections between terms (**thesauri**)
- rules and knowledge about which connections are valid and meaningful (**ontologies**)



(Obrst, 2003)

Recap: Ontology in Computer Science

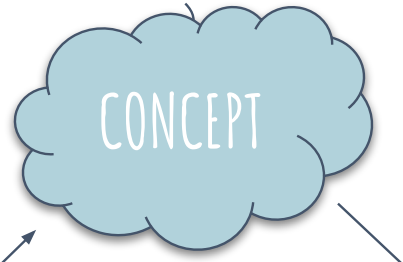
An **ontology** is an **explicit**, **formal** specification of a **shared conceptualization**. The term is borrowed from philosophy, where an ontology is a systematic account of existence. For AI systems, what 'exists' is that which can be represented.

(Thomas R. Gruber, 1993)

conceptualization:	abstract model (domain, relevant terms, relations)
explicit:	meaning of all terms is clearly and unambiguously defined
formal:	interpretable by machines
shared:	consensus

Recap: Conceptualization

shared concept



symbolizes

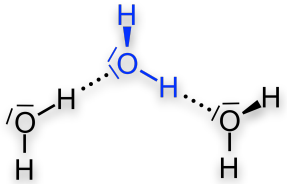
refers to

“Bridge”

SYMBOL

stands for

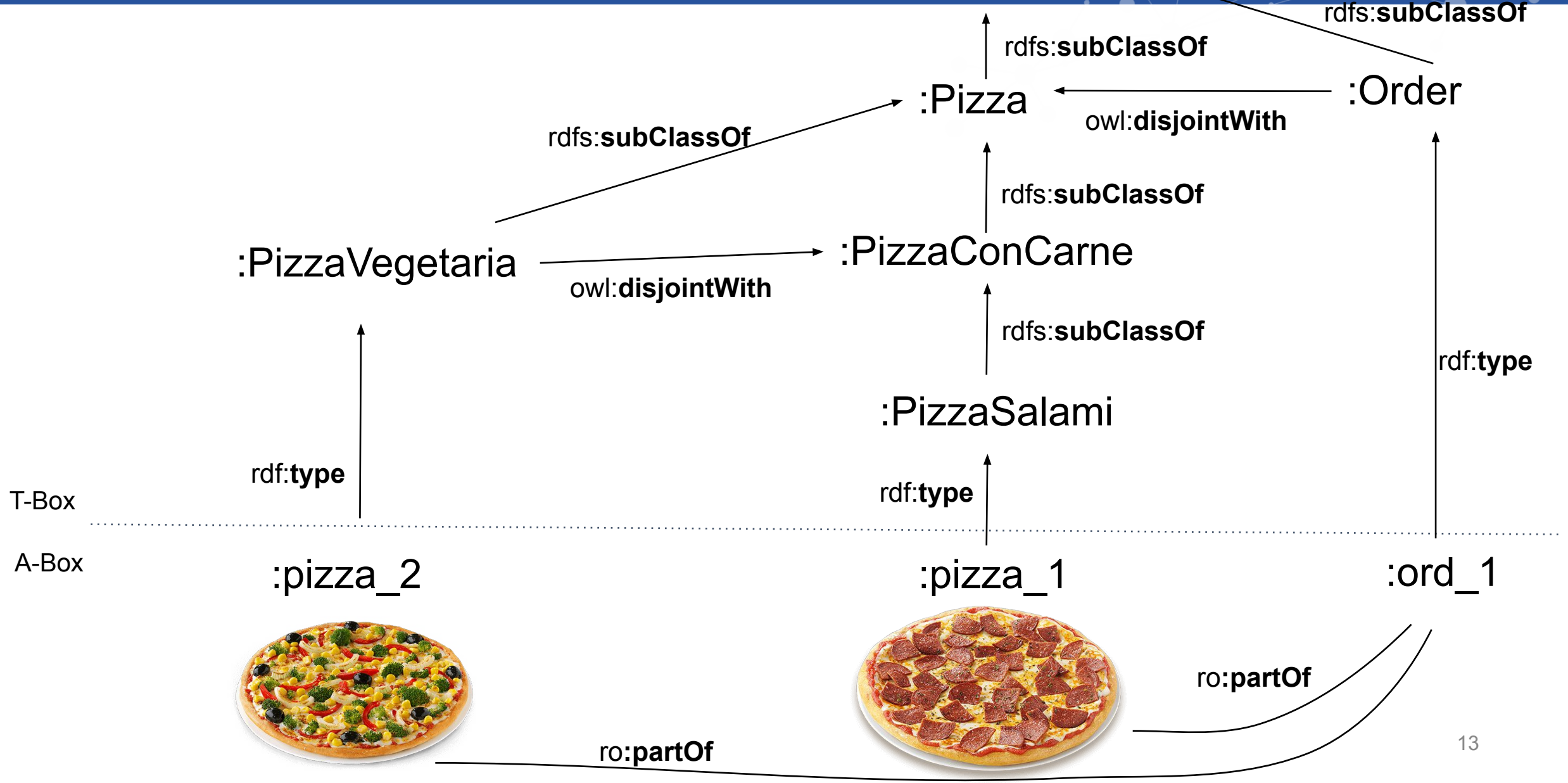
REAL THING



Assignment of symbol (word) to object only possible indirectly, people assign objects to concepts that they name

(Ogdens Triangle, 1923)

Recap: RDF and OWL



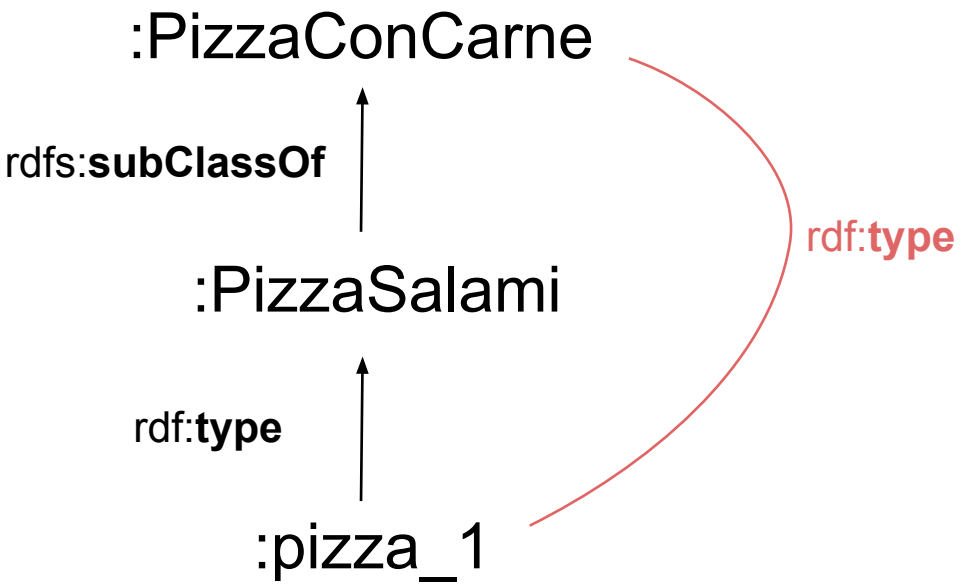
Recap: Reasoning

- Reasoner = (Theorem) Prover = Logical Inference Engine
- Can draw conclusions from logical axioms and statements
- Checks the consistency of ontologies
- Evaluates logical inference rules and thus generates information that is not explicitly contained in the ontology

RDFS Entailment Rules

Rule 9:

```
u rdfs:subClassOf x .
v rdf:type u .

----------------
v rdf:type x .
```



:pizza_1	rdf:type	:PizzaSalami .
:PizzaSalami	rdfs:subClassOf	:PizzaConCarne .
:pizza_1	rdf:type	:PizzaConCarne .

PMDco: Ontology Development Process

Today

Consensus Building!

Consensus building process

It is naive to think the ontology development process is just:

1. Gather domain knowledge from experts and texts
2. Translate concepts into RDF/OWL or formal logic
3. And all this fully automated

- Machines still have difficulties to process and "understand" natural language
- Also LLMs have their limits, because they mostly rely on text data, but not on emotions, facial-expressions, intention; and for real disambiguation we also need those, also in scientific or industrial contexts // use LLMs as inspiration, but don’t expect that they can do the work
- Ontology development is more about **trust** and **transparency** and **reproducibility**
- Success depends on **adoption by a community**

Motivation

- Ontologies define meanings of terms and relations
- **Consensus on meanings** reduces ambiguity
- Also experts often disagree on definitions and scope
- Differences emerge in **terminology** and **semantics**
- Even established sciences contain conceptual variation
- Example: multiple definitions of ‘cell’

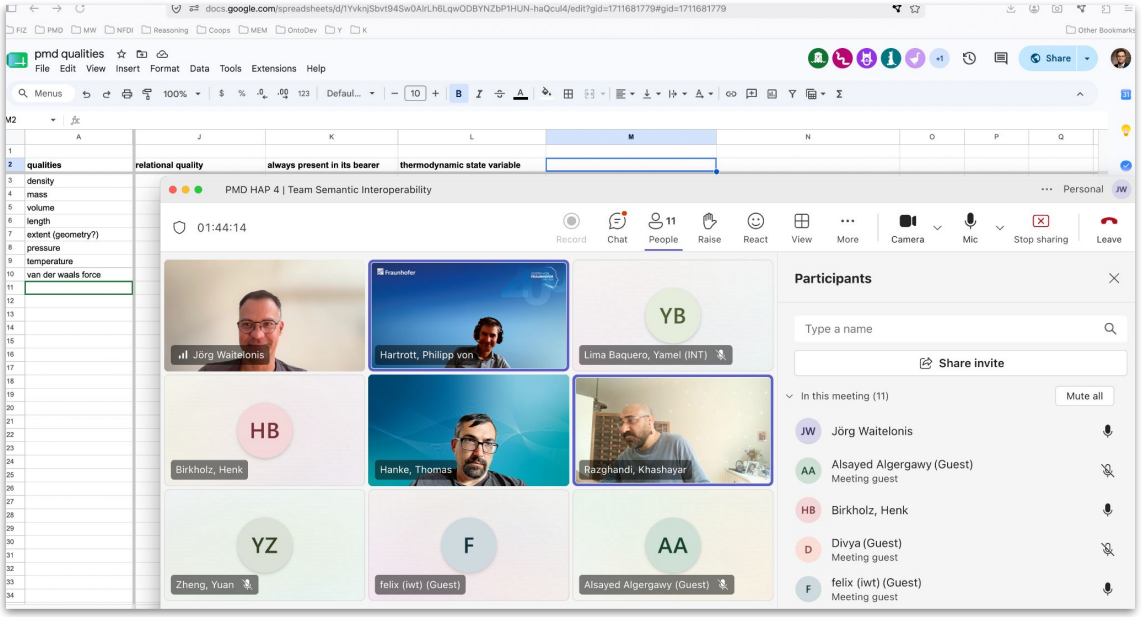
- Terms in an ontology need to be **unambiguously interpretable**
  - First by humans: **labels** and **textual definition**
  - By machines: **logical rules**

Consensus building process

How to build consensus ??

Consensus building process (micro)

- Ontology experts and domain experts **together**
- **Domain experts:**
  - **provide** and **document** their **knowledge**
  - techniques: inclusive participatory discussions, notes, diagrams, etc.
- **Ontology expert:**
  - **moderate** and **mediate** the consensus building process
  - techniques: naive questions, paraphrasing, summarizing



PMDco core team; regular meetings on Tuesdays and Thursday

[1]

Meeting protocol: https://docs.google.com/document/d/1f9WOIK18TZxJQDbQe0q1DYWtnDyLCuKlCp2868pdGVE/edit?tab=t.0

Consensus building process (macro)

- Encourage community-wide adoption
  - playgrounds
  - hackathons
  - conferences
- Open, extensible, and transparent ontology design
- Documentation and stakeholder engagement are essential
- Wider adoption increases interoperability value

Join our bi-weekly Fri. 1-2pm public **Ontology Playgrounds**

PMD Hackathon October 2025 @ BAM

[1]

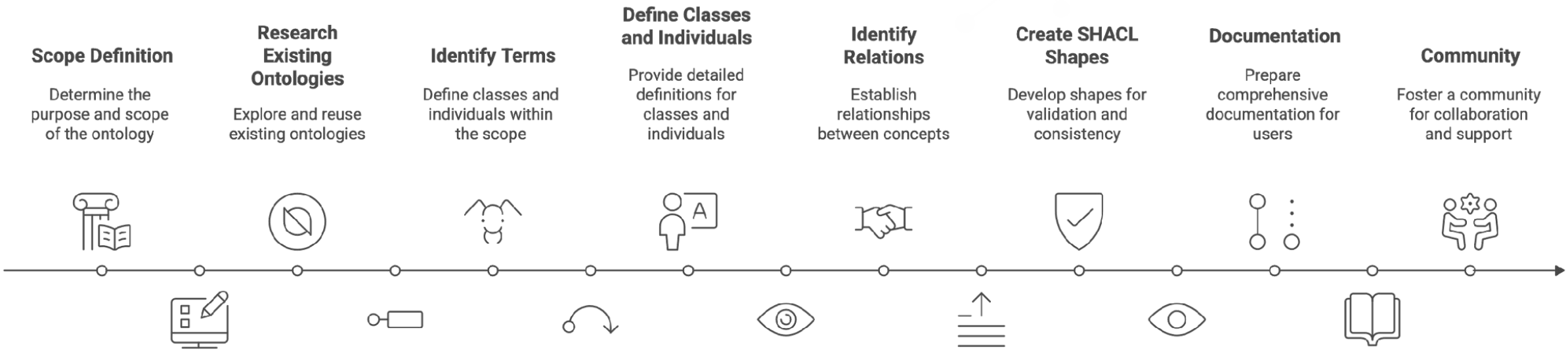
GitHub

Check & Contribute to our **GitHub PMDco** & **Application Ontologies**

Design of Ontologies

- Many Design Methods
  - **Methontology** [2], **Ontology 101** [3], **Extreme Design** [4], **NeOn** [5] , etc.
- Problem:
  - many degrees of freedom → leads to different solutions
- Follow Principles:
  - Reuse; IRIs, definitions, naming conventions, annotations, design
  - e.g.: OBO-Foundry principles
- Follow Existing Patterns:
  - Reusable modeling solution to a recurrent ontology engineering problem.
- Use Standards: deployment, import management, versioning → ODK

Ontology Development Process



Scope Definition

Determine the purpose and scope of the ontology

Research Existing Ontologies

Explore and reuse existing ontologies

Identify Terms

Define classes and individuals within the scope

Define Classes and Individuals

Provide detailed definitions for classes and individuals

Identify Relations

Establish relationships between concepts

Create SHACL Shapes

Develop shapes for validation and consistency

Documentation

Prepare comprehensive documentation for users

Community

Foster a community for collaboration and support

Competency Questions

Formulate questions the ontology should answer

Select Top-Level Ontology

Choose a foundational ontology for structure

Build Class Hierarchy

Organize classes into a hierarchical structure

Assign Concepts to TLO

Integrate concepts into the top-level ontology

Define Additional Rules

Create axioms and rules for the ontology

Evaluation

Assess the ontology's ability to answer questions

Publication

Release the ontology with necessary resources

1. Scope Definition

- What is the purpose of the ontology?
- What is the concrete scope or the “subject matter” of the ontology?
- Which type of ontology is being targeted?
- If there are multiple “scopes,” should these perhaps become several ontologies/modules?
- Top-down vs. bottom-up?
- PMDco scope:
  - https://materialdigital.github.io/core-ontology/docs/intro.html

2. Competency Questions

Questions that the ontology should be able to “answer”.

- What is the expected answer?
- Check whether the desired answer can actually be derived from the data.

- Understanding the scope
- Validating the ontology
- Alignment and understanding the upper level / alignments
- Use cases

- Examples from PMDco:
  https://materialdigital.github.io/core-ontology/docs/intro.html#competency-questions

3. Research of existing ontologies

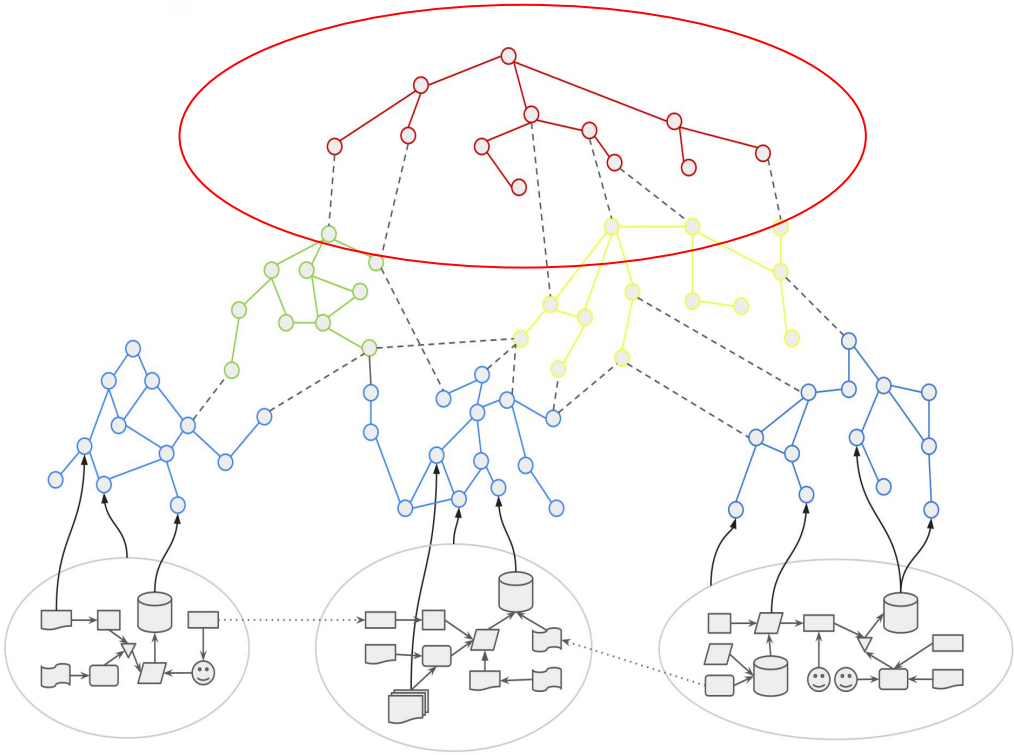
- **Re-use what already exists !!**
- Google
- Talk to people
- “Lookup” platforms and terminology services:
  - https://bartoc.org/
  - https://ontobee.org/
  - https://www.ebi.ac.uk/ols4/index
  - https://lov.linkeddata.es/dataset/lov/
  - https://terminology.tib.eu/ts
- Check suitability using the competency questions and scope(s).

Photo by Compagnons on Unsplash

4. Selection of a top-level ontology

- Provides structure,
- Promotes error prevention,
- Enables connection to other domains,
- Fosters reusability,
- Reduces uncontrolled growth,
- Increases acceptance,
- Supports community building,

- Decide also depending on what you can reuse
    - for PMDco we decided for BFO because we can easily reuse many parts from other ontologies, which we do not need to recreate



5. Identify terms: classes and individuals

- Discuss about their actual meaning
- Take **ambiguities** into account!
  - resolve them → consensus
- Even within a domain, there are many homonyms (same spelling + different meaning)
  - e.g. **phase**: thermodynamic phase, crystallographic phase, processing stage, project phase
- Think about different contexts and perspectives
- Be very precise
- Defined what level of granularity you would like to achieve
  - start with coarser grains
  - macro, meso, micro
- Sometimes it happens that you can't find a name for a meaning (to not get stuck on name finding: temporarily invent a "replacement" name for it " the A thing", "thing that relates to …")

6. Build the class hierarchy

- Organize your classes in a hierarchy.
  - “A subclassOf B” means: **all** instance of **A** are also instances of **B** (at all times)

- → Frequent source of mistakes (or bad practices)

  - `:HeatedSteel subclassOf :Steel`

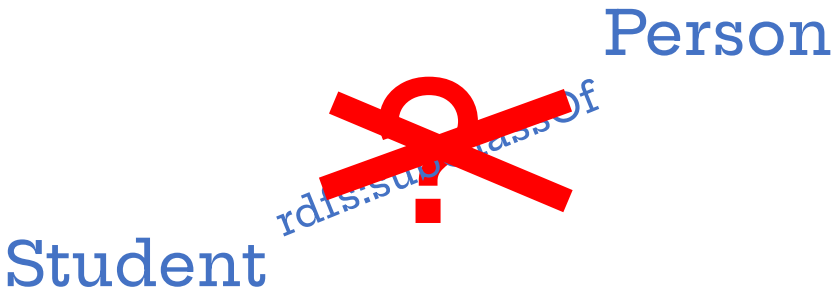
  - `:BrittleMaterial subClassOf :Ceramic .`
  - `:BrittleSteel subClassOf :BrittleMaterial .`

  - `:Alloy subClassOf :MetallicMaterial .`
  - `:MetallicMaterial subClassOf :Alloy .`

6. Build the class hierarchy: rigidity

Student vs. Person

- A person may be a member of the "Student" class **only for part of their life**.
- If "Student" is used as a superclass or central category in the ontology, reasoning about the individual's properties and identity can become inconsistent as **their status changes over time**. [6,9]
- Use a role pattern.

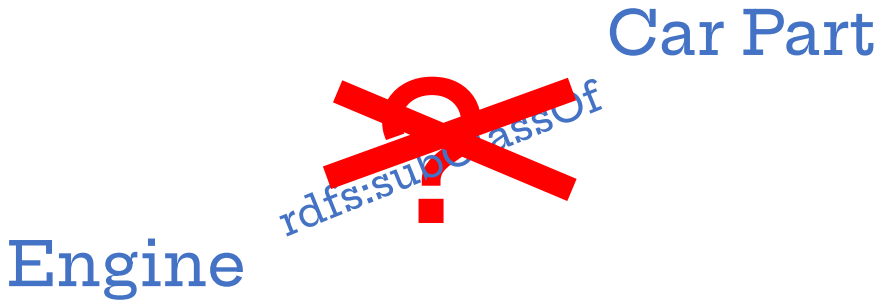


6. Build the class hierarchy: rigidity

Car Part vs. Engine

An "engine" is rigid (always an engine if it exists), but "car part" is non-rigid as an engine could cease being part of a car (e.g., moved to a boat).

If "car part" is modeled as a superclass for "engine," this creates a conflict because an engine can be an engine outside of being a car part. [7,9]



6. Build the class hierarchy: rigidity

→ Temporal Reasoning Burden

Non-rigid class membership is often time-dependent, requiring ontologies to track context or temporal attributes to preserve correct reasoning, significantly increasing modeling and inferencing complexity.

6. Build the class hierarchy: rigidity

Check the **rigidity** !!

If **i** is instance of **c**,

does this hold for **every time** and in **every situation** throughout the existence of **i**?

6. Build the class hierarchy: rigidity

Check the **identity** !!

If **p** is a subclass of **q**,

then **p** cannot define any new way of identifying its instances that **q** does not already define.

(e.g. Person identified by name vs. Student identified by matriculation number)

6. Build the class hierarchy: rigidity

Check the **unity** !!

For some classes, entities can be decomposed into instances of the same class → **anti-unity classes**

(e.g. group into two groups, portion of water into two portions of water)

Other classes only have “whole” instances → **unity classes**

For "whole" individuals, there is always a functional relation unambiguously relating a part to the whole

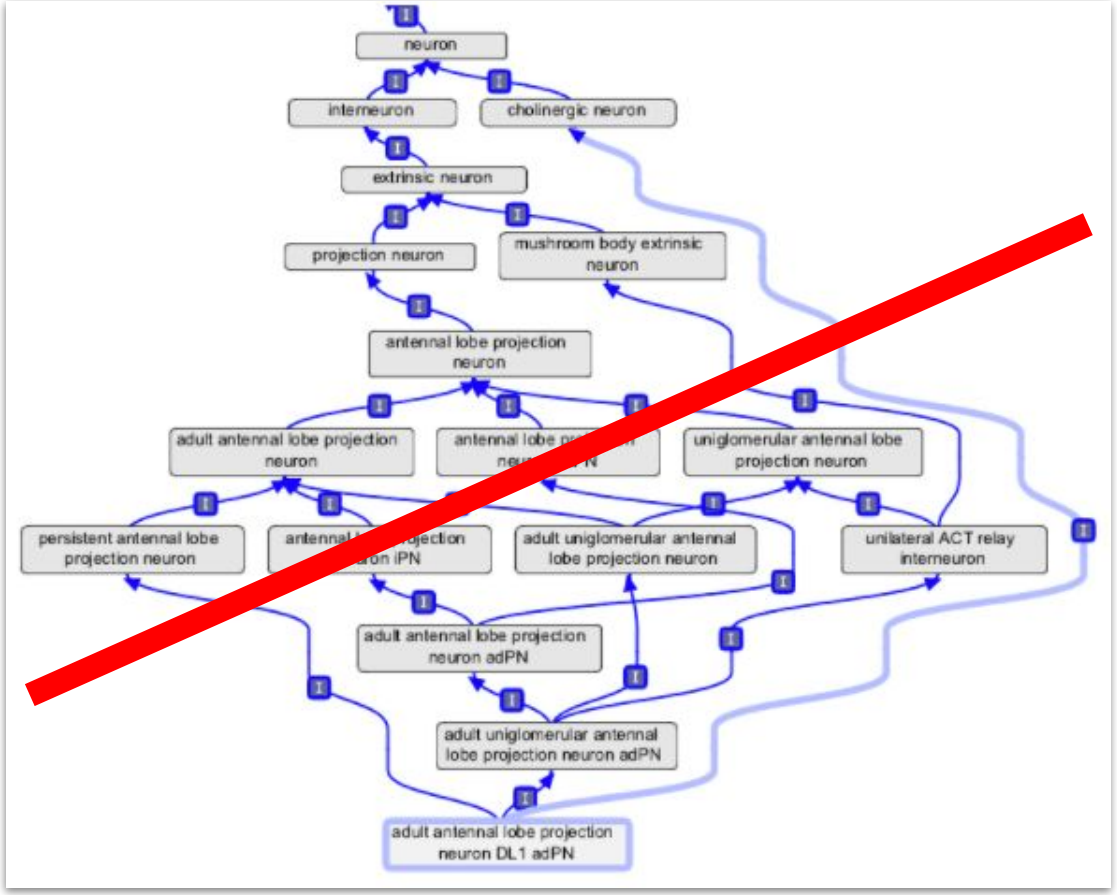
(e.g. person has some bodyparts, city has districts)

Unity classes may **only** have **unity classes** as their subclasses

Anti unity classes may only have **anti unity classes** as their subclasses [9]

6. Build the class hierarchy

- Avoid poly-hierarchies!
- Very hard to maintain:
  - keeping track of multiple classification chains
  - avoid redundancy



[11]

- Keep a mono-hierarchy in the edit version and use the reasoner to infer the resulting polyhierarchy in release artefacts
- Rector normalization (e.g. via roles) [10]

7. Define classes and individuals

- A class without a definition is possible, but “meaningless”.
- What is a good definition?

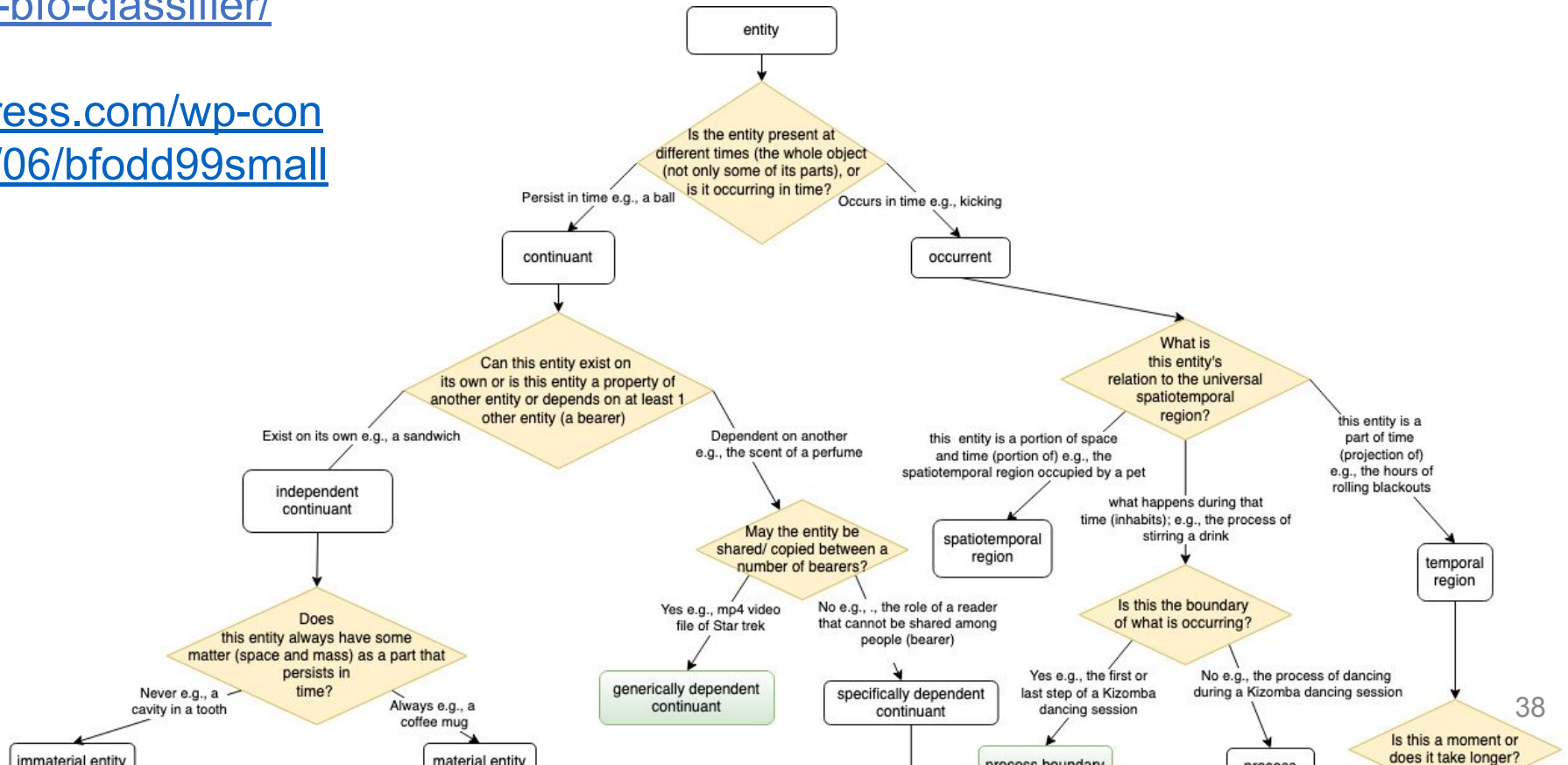
- Aristotelian definition:
  - If A is a subclass of B, then the definition is:
    - A := B + Δ
    - A is a B with the restriction that Δ
    - Example: *PizzaConCarne* is a *Pizza* that contains *Meat*.
  - Inconsistencies in the class hierarchy often become apparent here.

8. Assign concepts to the TLO

- If necessary, adapt and refine definitions again.

https://keet.wordpress.com/2023/06/29/improved-the-bfo-classifier/

https://keet.wordpress.com/wp-content/uploads/2023/06/bfodd99small.png



9. Identify relations between concepts

- If possible, (re)use relations from the top-level ontology.
  - avoid creating new ones; use classes to express relations
  - you cannot simply add additional info to properties (provenance: who, when, etc.)
    - except via formal reification (expensive) / rdf-star (not all tools support that)
- Special relations include: “part-whole” relationships, existential dependency, participation relationships, etc.
- Consider the temporal-spatial context of the relations (possibly define roles)
- Be very clear on use of “datatype property” vs. “object property”

10. Define additional rules (axioms)

- "If → Then," e.g., using **SWRL rules**: https://www.w3.org/submissions/SWRL/
- **Property chains**:
  - owns **o** hasPart → owns
  - I own my car, my car hasPart engine → I own engine.

- SPARQL **construct queries** for expansion of shortcut relations [12]
  - defined by construct (OMO_0002000)
- You can use this for mapping to external ontologies, without making an explicitly commitment

11. Create SHACL Shapes

- Shapes Constraint Language (SHACL)
- **For validation (consistency checking) of A-Box/T-Box**
- **Documentation (Patterns)**
  - Puzzle pieces to structure your data
- Ontology → describes what **IS**
- SHACL Shapes → describes what **SHOULD BE**

Photo by Jonny Gios on Unsplash

12. Evaluation

- Check whether the competency questions can be answered
- If not: return to step 1

13. Documentation

- Usage guide for users: how to reuse it?
- Provide example data
- Provide best practices and design guidelines for structuring concepts, relations, and axioms (ODPs).
- Formulate design principles and conventions (development guide)
- Prepare a concept for maintenance and further development

14. Publication of the ontology

- Namespace (e.g., via w3id.org or purl),
  - Prefix
  - Make sure you own it
- Dereferencing (can be implemented with GitHub)
  - the IRI should deliver information about the resource → Semantic Web
  - implement content negotiation: https://www.w3.org/TR/cooluris/
    - curl -L https://w3id.org/pmd/co/3.0.0 → Website
    - curl -L -H "Accept: text/turtle" https://w3id.org/pmd/co/3.0.0 → Turtle-File

- Publish a technical or research paper
- Registration with lookup portals and terminology services
- It is **not** enough to put an owl/ttl file to Github

15. Community

- Communication:
  - regular meetings; online and offline
- Form working groups; distribute responsibilities
- Provide development infrastructure, e.g., via GitHub, Miro, etc.
- Maintain and incentivize the community, e.g., academic publishing, funding opportunities, commercialization opportunities, recognition, and reputation
- Provide “playgrounds”, workshops, hackathons
- Governance
- Dissemination

Summary

- Ontology development is primarily a consensus-building activity, not just a technical one.
- The hardest part is agreeing on meanings, definitions, and scope, not writing OWL/RDF.
- Good ontologies emerge through discussion, negotiation, and reuse of existing standards.
- Reuse existing ontologies, standards, and design patterns whenever possible.
- Community adoption and governance are essential for success.
- It is normal to make mistakes:
  - ontology development is iterative, and continuous refinement is part of the process.

References

[1] Neuhaus, Fabian & Hastings, Janna. (2022). Ontology Development is Consensus Creation, Not (Merely) Representation. 10.48550/arXiv.2210.12026.
[2] Fernández-López, Mariano & Gomez-Perez, Asuncion & Juristo, Natalia. (1997). METHONTOLOGY: from ontological art towards ontological engineering. Engineering Workshop on Ontological Engineering (AAAI97).
[3] Noy, N. & Mcguinness, Deborah. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Knowledge Systems Laboratory. 32.
[4] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. 2009. EXtreme design with content ontology design patterns. In Proceedings of the 2009 International Conference on Ontology Patterns - Volume 516 (WOP'09). CEUR-WS.org, Aachen, DEU, 83–97.
[5] Suárez-Figueroa, Mari Carmen (2010). NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. Thesis (Doctoral), Facultad de Informática (UPM) https://doi.org/10.20868/UPM.thesis.3879
[6] https://basic-formal-ontology.org/documents/seyed-shapiro-fois-2012.pdf
[7] https://www.jarrar.info/courses/AAI/Jarrar.LectureNotes.AAI.2012S.OntologyModelingChallenges.pdf
[8] https://software-lab.org/publications/ontology_composition_gb.pdf
[9] Guarino, Nicola and Chris Welty. 2002. Evaluating Ontological Decisions with OntoClean. Communications of the ACM. 45(2):61–65. New York:ACM Press
[10] https://douroucouli.wordpress.com/2019/06/29/ontotip-learn-the-rector-normalization-technique/
[11] https://oboacademy.github.io/obook/explanation/intro-to-ontologies/
[12] C. Mungall, A. Ruttenberg, D. Osumi-Sutherland, Taking shortcuts with owl using safe macros, Nature Precedings (2010) 1–1. doi:https://doi.org/10.1038/npre.2010.5292.1.

PMDco:
Data Transformation

How to transform my data?

1. **Work out the patterns**
2. **Implement transformation method**

Example: tensile-strength = 520 MPa

How to transform my data?

tensile-strength = 520MPa

- Find out what your data actually describes:
  - **when, how, who, what (thing, property, value), where**

when and where → **process** (occupies temporal region, occurs in location)
how → **plan specification** (e.g. document of a norm)
who → **agent** (persons, running software, or devices)
what thing → **object** or **information**
what property → **quality**, **disposition**, or **process-attribute**
what value → **value specification** and **datum** (setting/measurement)

How to transform my data?

tensile-strength = 520MPa

- Find out what your data actually describes:
    - Setting or Measurement? (“should be” or “is”)

What we know from the ontology:

- measured quality is a tensile strength
- a quality is inherent in some object (with role test piece)
- the quality has been measured
- the measurement has been carried out by a process
- the process has the object as a participant
- the measurement datum is the output of the process
- the measurement datum refers to a value specification, which describes the quality (tensile strength) and has value and unit

How to transform my data?

- the quality is a tensile strength
- a quality is inherent in some object (with role test piece)

```
ex:qual_1  rdf:type  tensile_strength: .

ex:obj_1  has_quality:  ex:qual_1 .
ex:obj_1  has_role:  ex:role_1 .
ex:role_1 a :test_piece_role .
```

- the quality has been measured

```
ex:datum_1  rdf:type  measurement_datum: .
ex:datum_1 is_quality_measurement_of: ex:qual_1 .
```

- the measurement has been carried out by a process
- the process has the object as a participant and realizes it test piece role

```
ex:datum_1  specified_output_of:  ex:proc_1 .
ex:proc_1  has_participant:  ex:obj_1 .
ex:proc_1  realizes:  ex:role_1 .
```

- the measurement datum refers to a value specification, which describes the quality (tensile strength) and has value and unit

```
ex:datum_1  has_value_specification:  ex:spec_1 .
ex:spec_1  specifies_value_of:  ex:qual_1 .
ex:spec_1  has_specified_numeric_value:  520 .
ex:spec_1  has_measurement_unit_label:  qudt:MegaPA .
```

How to transform my data?

RDF Triple

```
ex:qual_1  rdf:type  tensile_strength: .

ex:obj_1  has_quality:  ex:qual_1 .
ex:obj_1  has_role:  ex:role_1 .
ex:role_1 a :test_piece_role .

ex:datum_1  rdf:type  measurement_datum: .
ex:datum_1 is_quality_measurement_of: ex:qual_1 .

ex:datum_1  specified_output_of:  ex:proc_1 .
ex:proc_1  has_participant:  ex:obj_1 .
ex:proc_1  realizes:  ex:role_1 .

ex:datum_1  has_value_specification:  ex:spec_1 .
ex:spec_1  specifies_value_of:  ex:qual_1 .
ex:spec_1  has_specified_numeric_value:  520 .
ex:spec_1  has_measurement_unit_label:  qudt:MegaPA .
```

Prefixes

```
@prefix tensile_strength: <https://w3id.org/pmd/tto/TTO_0000053> .
@prefix has_quality: <http://purl.obolibrary.org/obo/RO_0000086> .
@prefix has_role: <http://purl.obolibrary.org/obo/RO_0000087> .
@prefix has_value_specification: <http://purl.obolibrary.org/obo/OBI_0001938> .
@prefix specifies_value_of: <http://purl.obolibrary.org/obo/OBI_0001927> .
@prefix has_participant: <http://purl.obolibrary.org/obo/RO_0000057> .
@prefix specified_output_of: <http://purl.obolibrary.org/obo/OBI_0000312> .
@prefix realizes: <http://purl.obolibrary.org/obo/BFO_0000055> .
@prefix has_specified_numeric_value: <http://purl.obolibrary.org/obo/OBI_0001937> .
@prefix has_measurement_unit_label: <http://purl.obolibrary.org/obo/IAO_0000039> .
@prefix measurement_datum: <http://purl.obolibrary.org/obo/IAO_0000109> .
```

How to transform my data?

Demo: https://github.com/materialdigital/walkthrough-demos

- Python tool with pattern replace
- Python tool with RDFlib: https://rdflib.readthedocs.io/en/stable/
- RML:
  - YARRRML mapping: https://rml.io/yarrrml/
  - (https://github.com/Mat-O-Lab/RDFConverter)
  - (https://ontop-vkg.org/)
- Robot Spreadsheet: https://robot.obolibrary.org/template
- OTTR template: https://ottr.xyz/

How to transform my data?

Anti-Patterns

- Don’t **just change the structure** of your data.

- If you have “key” - “value” pairs in your data and you create properties “hasKey” and “hasValue”
  - :data a :Entry .
  - :data hasKey "Tensile Strength" .
  - :data hasValue 520 .
  - → you will not gain interoperability

- If you just transform you tabular data into CSVW
  - ex:col2 a csvw:Column ;
  - csvw:name "tensile_strength_MPa" ;
  - csvw:value "520" .
  - → you will not gain interoperability

Thank you!!

PMDco: Ontology pattern and usage guide

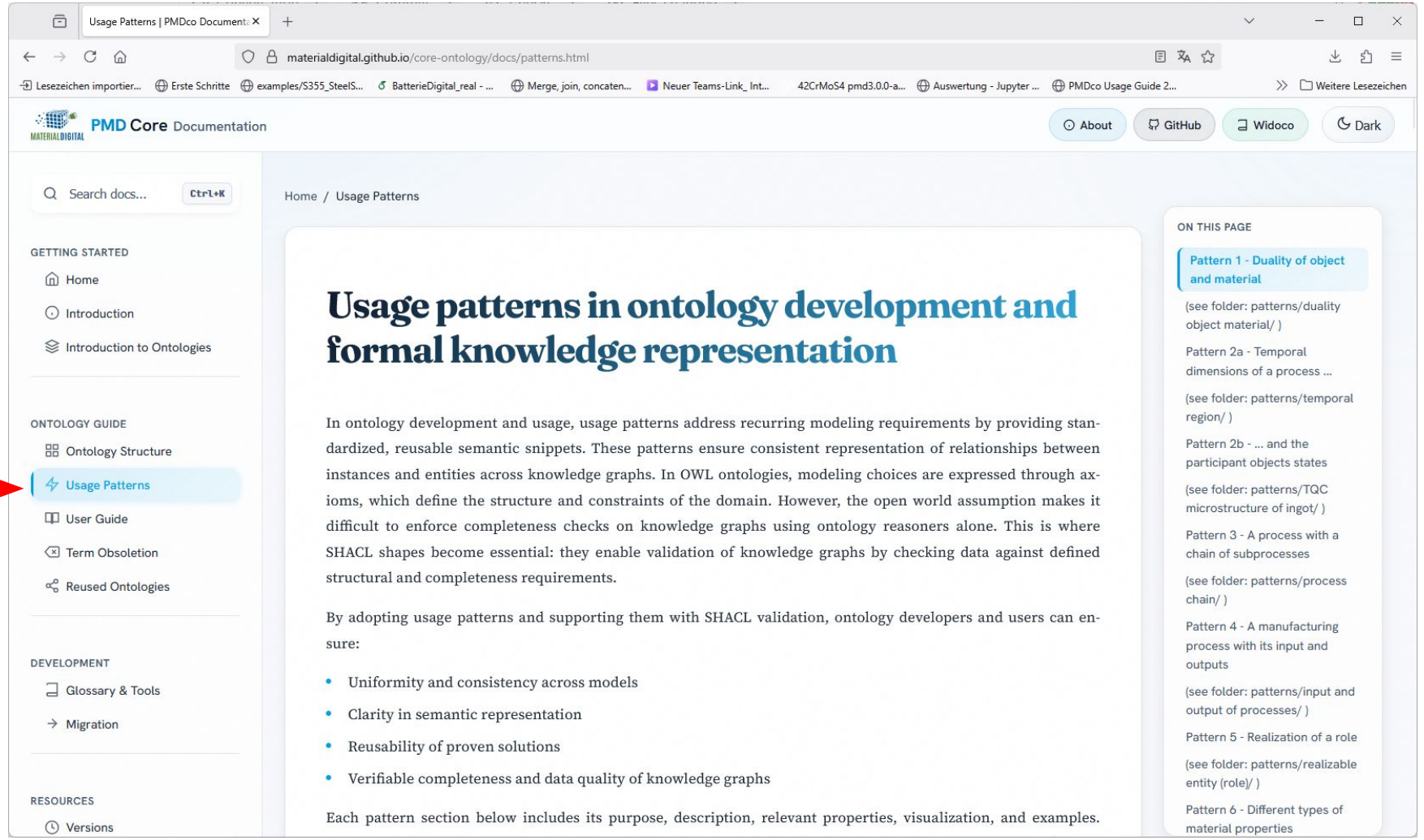
Usage patterns

Usage patterns:

- are a means of addressing recurring modelling situations
- allow for uniformity and consistency across different models
- provide clarity in semantic representation
- facilitate reusability of proven solutions
- give us a tool to verify completeness and data quality of knowledge graphs (to a certain extent)
- are usually expressed in the context of a “closed world” assumption
- can easily be expressed with the SHACL-language
- are **not** meant to replace the (OWL-) axioms that formulate classes and rely on the open world assumption

Lets learn to play piano by studying the piano score :)

Documentation reference

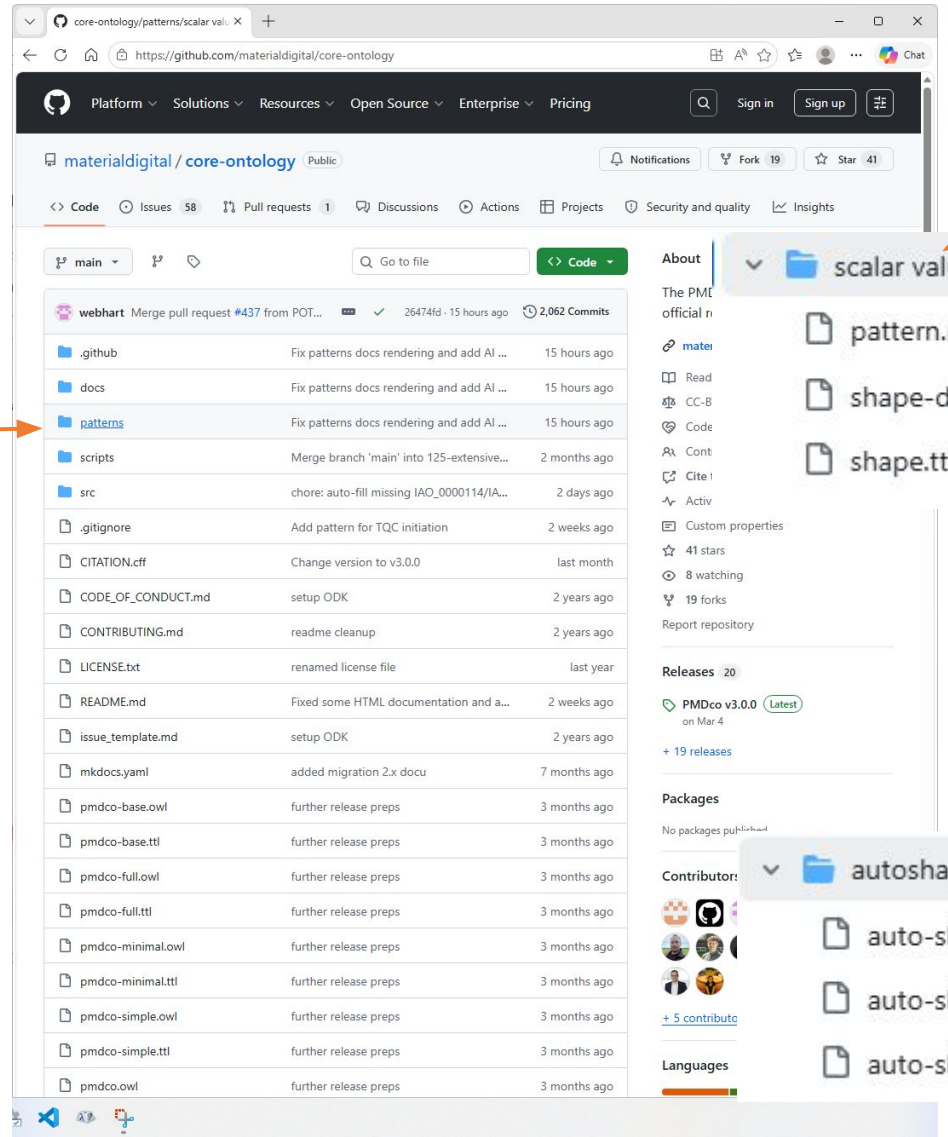


https://materialdigital.github.io/core-ontology/docs/patterns.html

Reading the pattern: @PREFIX-trick

```
# Typical usage:

@PREFIX schema: <https://schema.org/> .

<#philipp> a schema:person .
# … is expanded to:
<#philipp> a <https://schema.org/person> .



# Readability trick in pattern files:

@PREFIX scalar_value_specification: <http://purl.obolibrary.org/obo/OBI_0001931> .

<#mySVS> a scalar_value_specification: .
# … is expanded to:
<#mySVS> a <http://purl.obolibrary.org/obo/OBI_0001931> .
```

Organization of the pattern



Scope

The **Platform MaterialDigital Core Ontology (PMDco)** is a mid-level semantic framework for Materials Science and Engineering (MSE). Aligning with the ISO/IEC 21838-2:2021 standard, PMDco is built on the *Basic Formal Ontology (BFO)* and reuses several BFO-aligned ontologies such as **RO**, **IAO**, and **OBI**. Its scope follows the fundamental paradigm of MSE — **processing, structure, and properties** — and provides general semantics for entities commonly required across MSE disciplines, such as devices, roles, functions, and plans.

Processes	Structure & State	Properties
MSE-related process chains, including materials manufacturing, characterization, and simulation processes.	Substances, engineered materials, their composition, and multiscale structural features.	Material properties and qualities, representing processing–structure–property dependences.

“Represent our knowledge of the MSE domain”

The steel ingot story

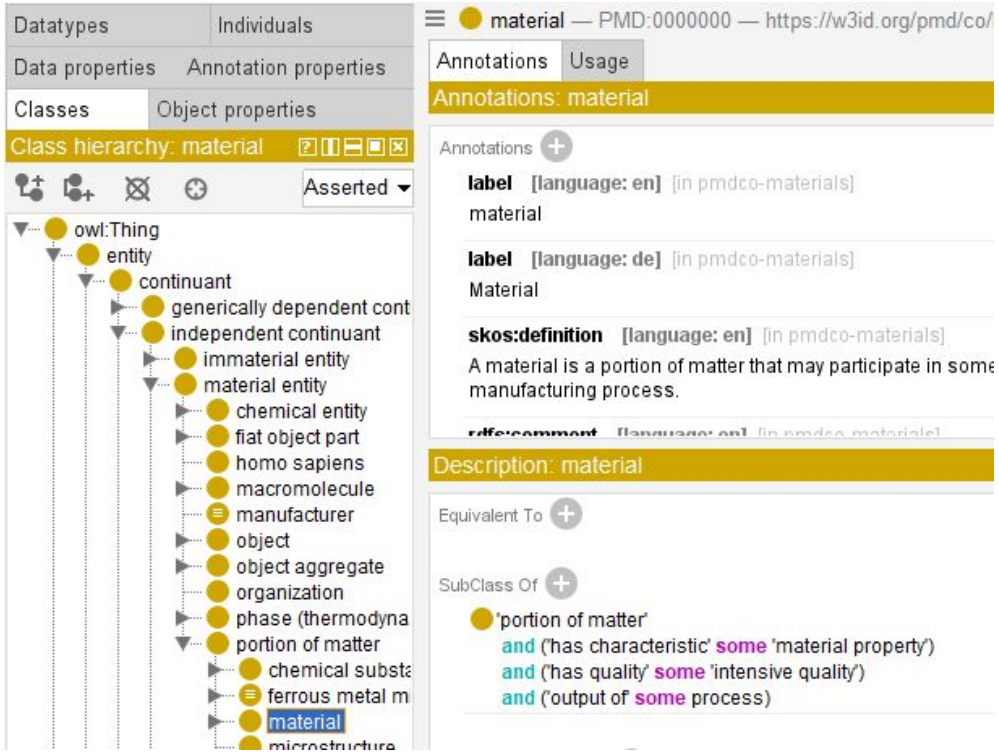
- **Pattern 1 - Duality of object and material**
- **Pattern 2 - Temporal dimensions of a process and the participant objects states**
- **Pattern 3 - A process with (a chain of) subprocesses**
- **Pattern 4 - A manufacturing process with its input and outputs**
- **Pattern 5 - Realization of a role (in a measurement)**

https://www.lohmann-stahl.de/

Two fundamental classes

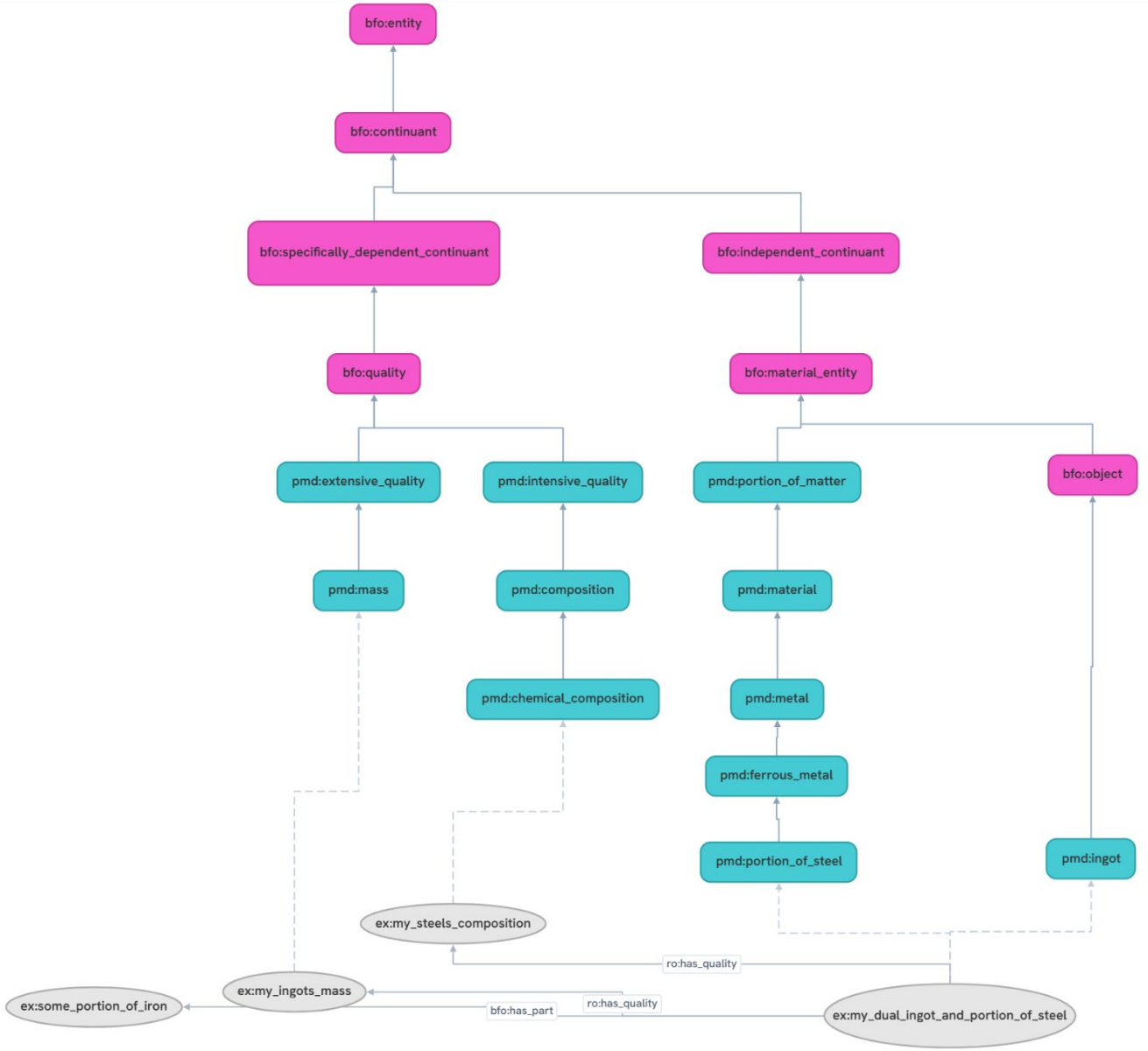
portion of matter (pmd:PMD_0000001)
skos:definition “A material entity that is not demarcated by any physical discontinuities. At some finer level of granularity it is an object aggregate, at some coarser level of granularity it is a fiat object part,but at this level of granularity it is neither.”

material (pmd:PMD_0000000)
skos:definition “A material is a portion of matter that may participate in some manufacturing process and whose shape is not relevant for its participation in the manufacturing process.”



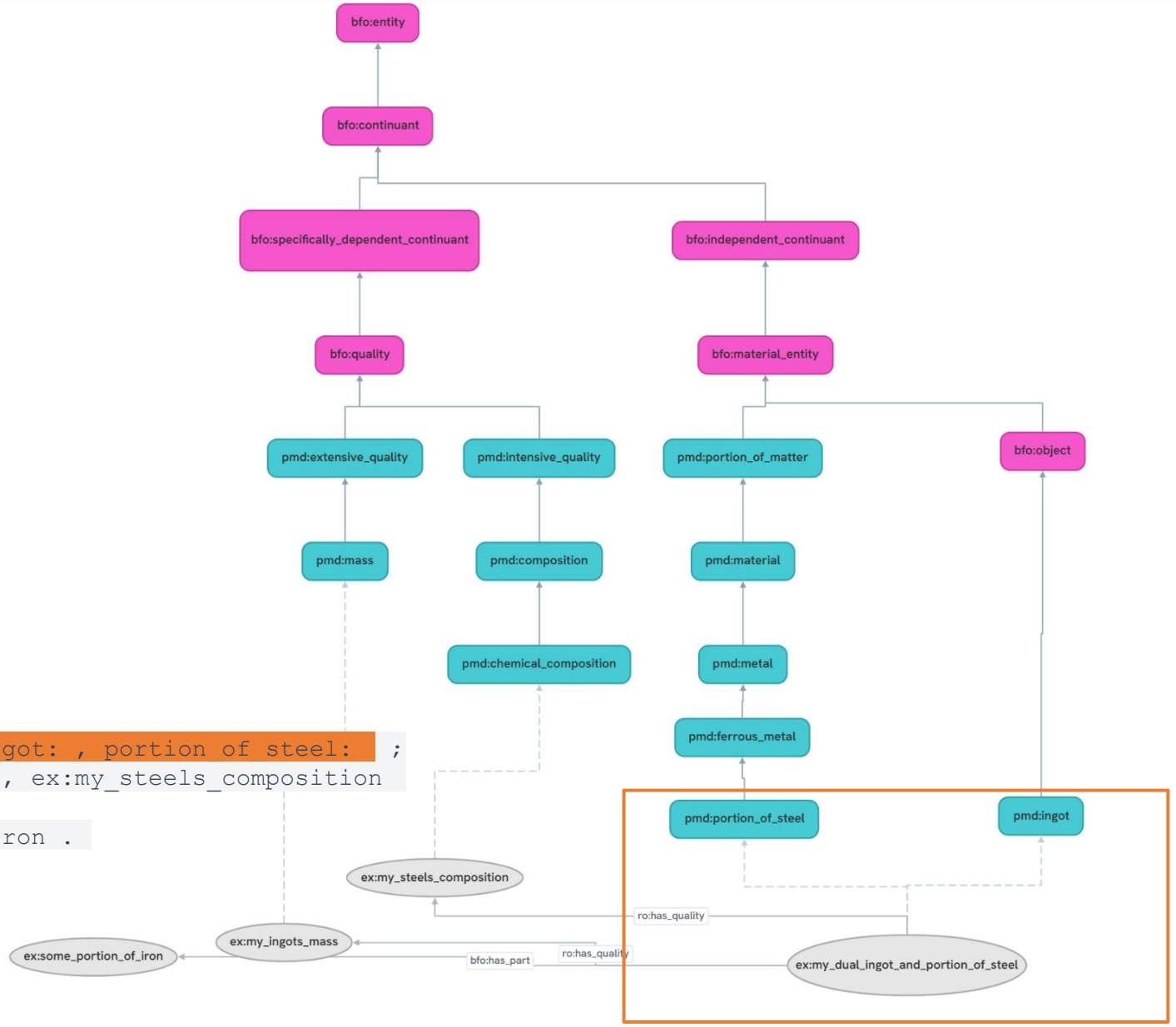
Pattern 1 - Duality of object and material

Purpose: Represents how objects and portions of matter are the same material entity, like a statue that is at the same time a portion of clay



Pattern 1 - Duality of object and material

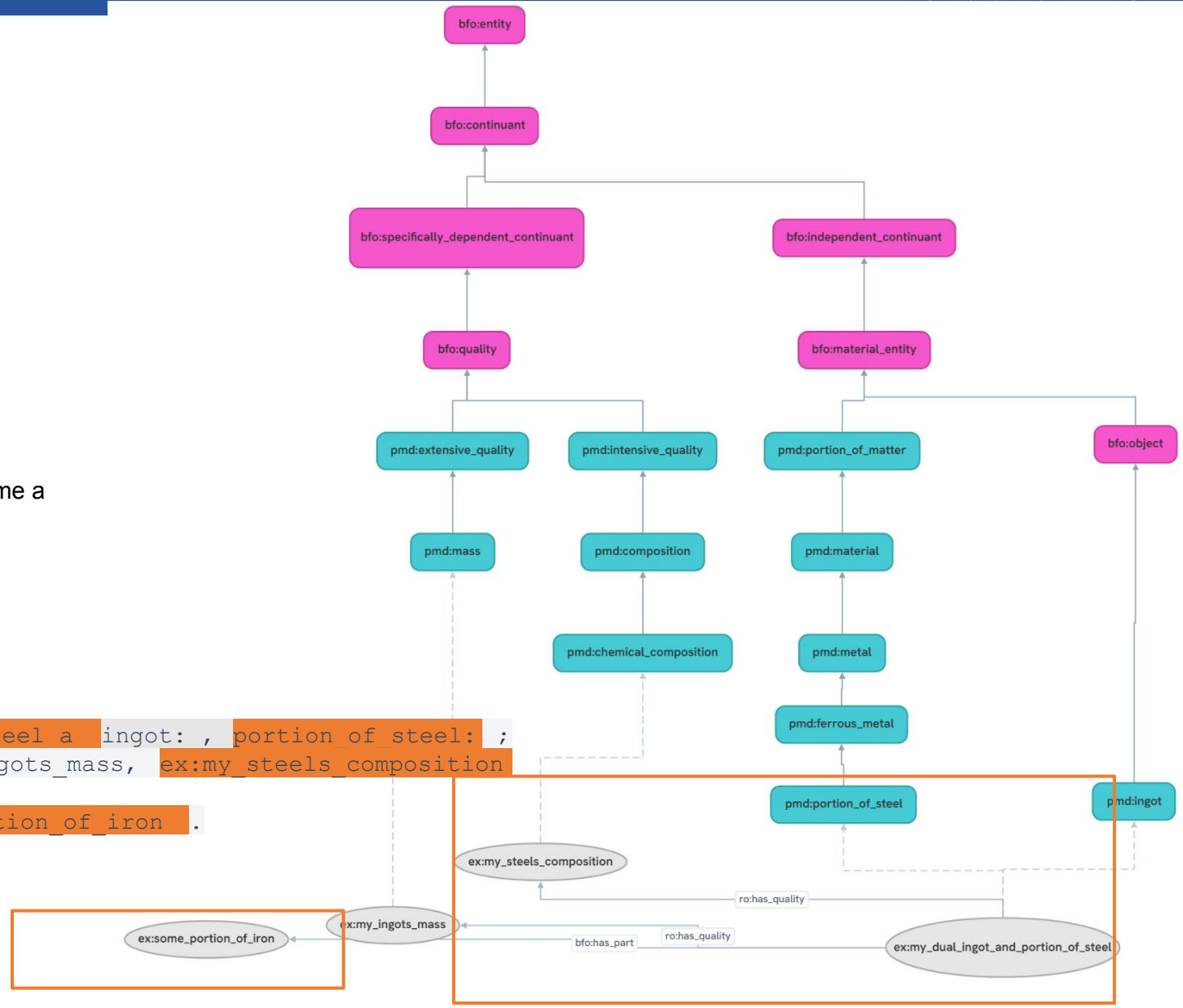
Purpose: Represents how objects and portions of matter are the same material entity, like a statue that is at the same time a portion of clay




```
ex:my_dual_ingot_and_portion_of_steel a ingot: , portion of steel: ;
            has_quality: ex:my_ingots_mass, ex:my_steels_composition
;
            has_part: ex:some_portion_of_iron .
```

Pattern 1 - Duality of object and material

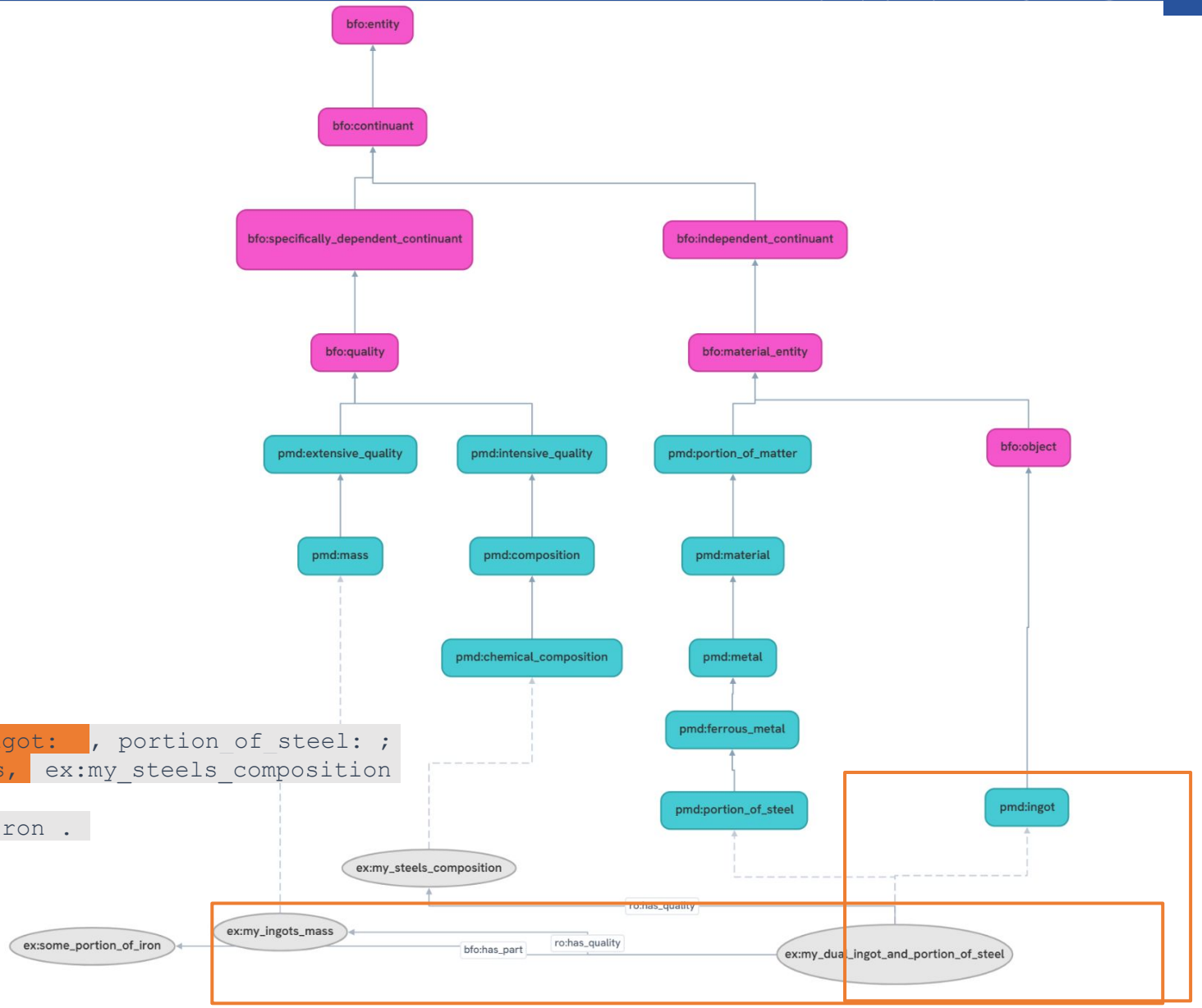
Purpose: Represents how objects and portions of matter are the same material entity, like a statue that is at the same time a portion of clay




```
ex:my_dual_ingot_and_portion_of_steel a ingot: , portion_of_steel: ;
            has_quality: ex:my_ingots_mass, ex:my_steels_composition
;
            has_part: ex:some_portion_of_iron .
```

Pattern 1 - Duality of object and material

Purpose: Represents how objects and portions of matter are the same material entity, like a statue that is at the same time a portion of clay

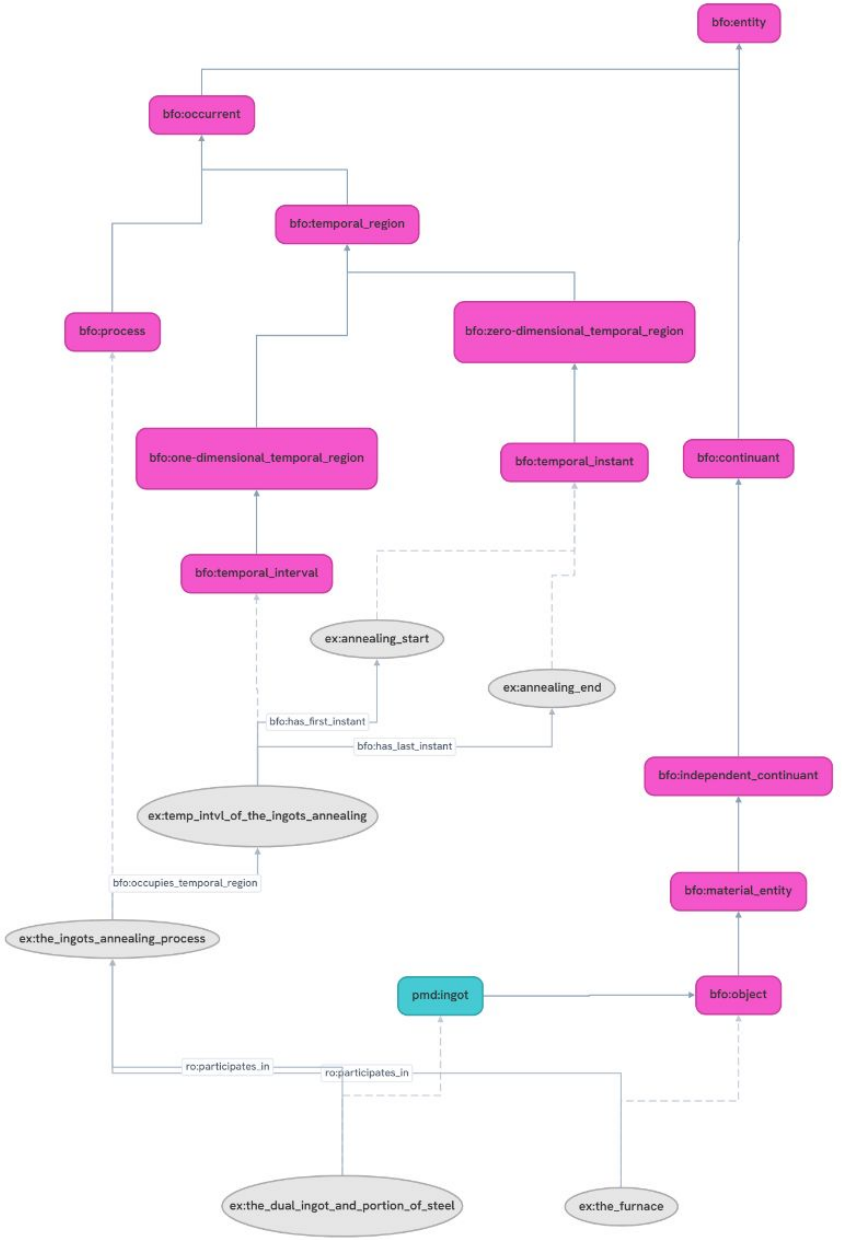



```
ex:my_dual_ingot_and_portion_of_steel a ingot:    , portion_of_steel: ;
            has_quality: ex:my_ingots_mass,  ex:my_steels_composition
;
            has_part: ex:some_portion_of_iron .
```

Pattern 2 - Temporal dimensions of a process and the participant objects states

Purpose: Specify the temporal extend and the temporal boundaries of a process on the time axis.

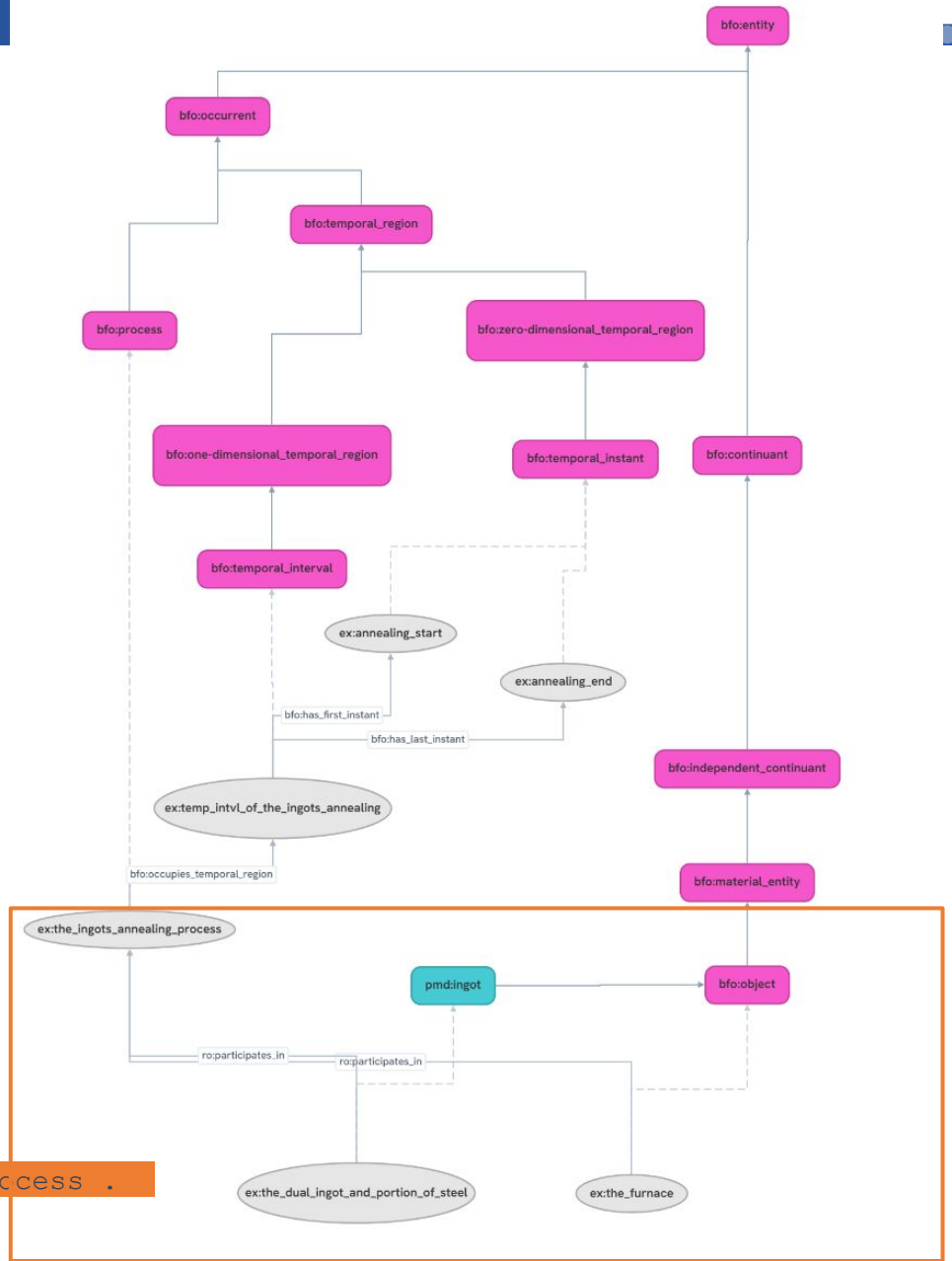
https://www.wlw.de/de/firma/industrieofen-haertereizubehoer-gmbh-unna-493968/produkte/individuell-gefertigte-haerteoefen-fuer-stahl-schacht-vakuum-und-schutzgas oefen-bis-1100c-33404359



Pattern 2 - Temporal dimensions of a process and the participant objects states

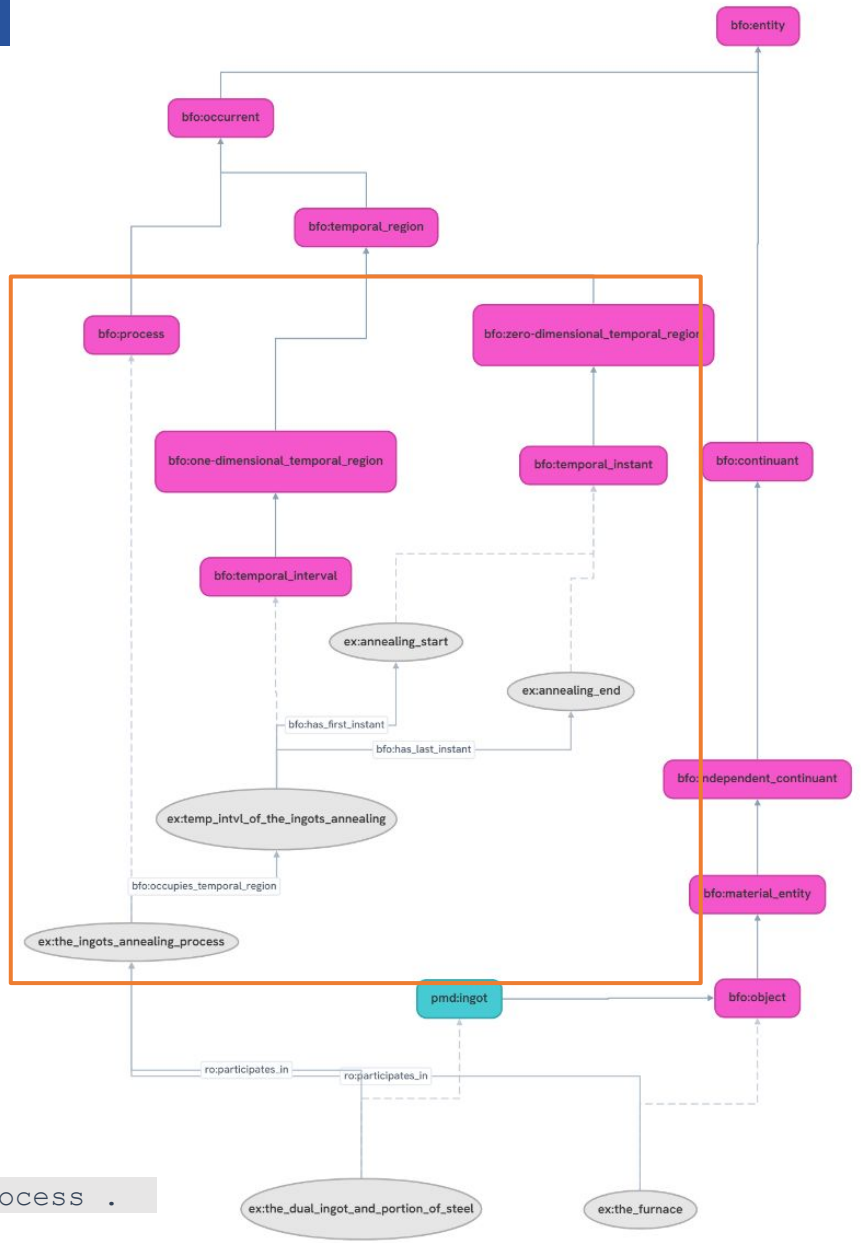
Purpose: Specify the temporal extend and the temporal boundaries of a process on the time axis.

```
ex:the_ingots_annealing_process a process: .
ex:temp_intvl_of_the_ingots_annealing a temporal_interval: .
ex:the_furnace a object: .
ex:the_dual_ingot_and_portion_of_steel a ingot: .
ex:annealing_start a temporal_instant: .
ex:annealing_end a temporal_instant: .

ex:the_ingots_annealing_process occupies_temporal_region:
ex:temp_intvl_of_the_ingots_annealing .
ex:temp_intvl_of_the_ingots_annealing has_first_instant: ex:annealing_start .
ex:temp_intvl_of_the_ingots_annealing has_last_instant: ex:annealing_end .
ex:the_furnace participates_in: ex:the_ingots_annealing_process .
ex:the_dual_ingot_and_portion_of_steel participates_in: ex:the_ingots_annealing_process .
```



Pattern 2 - Temporal dimensions of a process and the participant objects states

Purpose: Specify the temporal extend and the temporal boundaries of a process on the time axis.

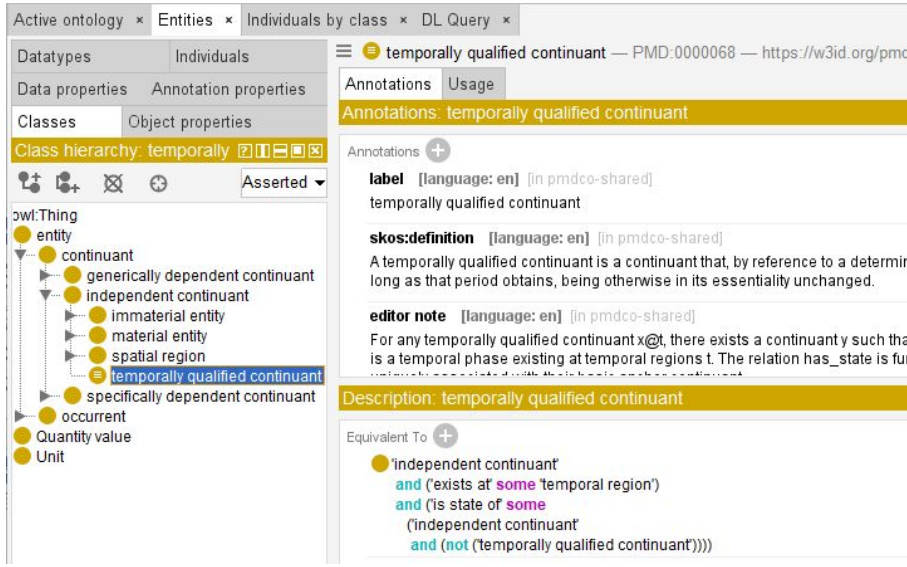



```
ex:the_ingots_annealing_process a process: .
ex:temp_intvl_of_the_ingots_annealing a temporal_interval: .
ex:the_furnace a object: .
ex:the_dual_ingot_and_portion_of_steel a ingot: .
ex:annealing_start a temporal_instant: .
ex:annealing_end a temporal_instant: .

ex:the_ingots_annealing_process occupies_temporal_region:
ex:temp_intvl_of_the_ingots_annealing .
ex:temp_intvl_of_the_ingots_annealing has_first_instant: ex:annealing_start .
ex:temp_intvl_of_the_ingots_annealing has_last_instant: ex:annealing_end .
ex:the_furnace participates_in: ex:the_ingots_annealing_process .
ex:the_dual_ingot_and_portion_of_steel participates_in: ex:the_ingots_annealing_process .
```

Another fundamental class …

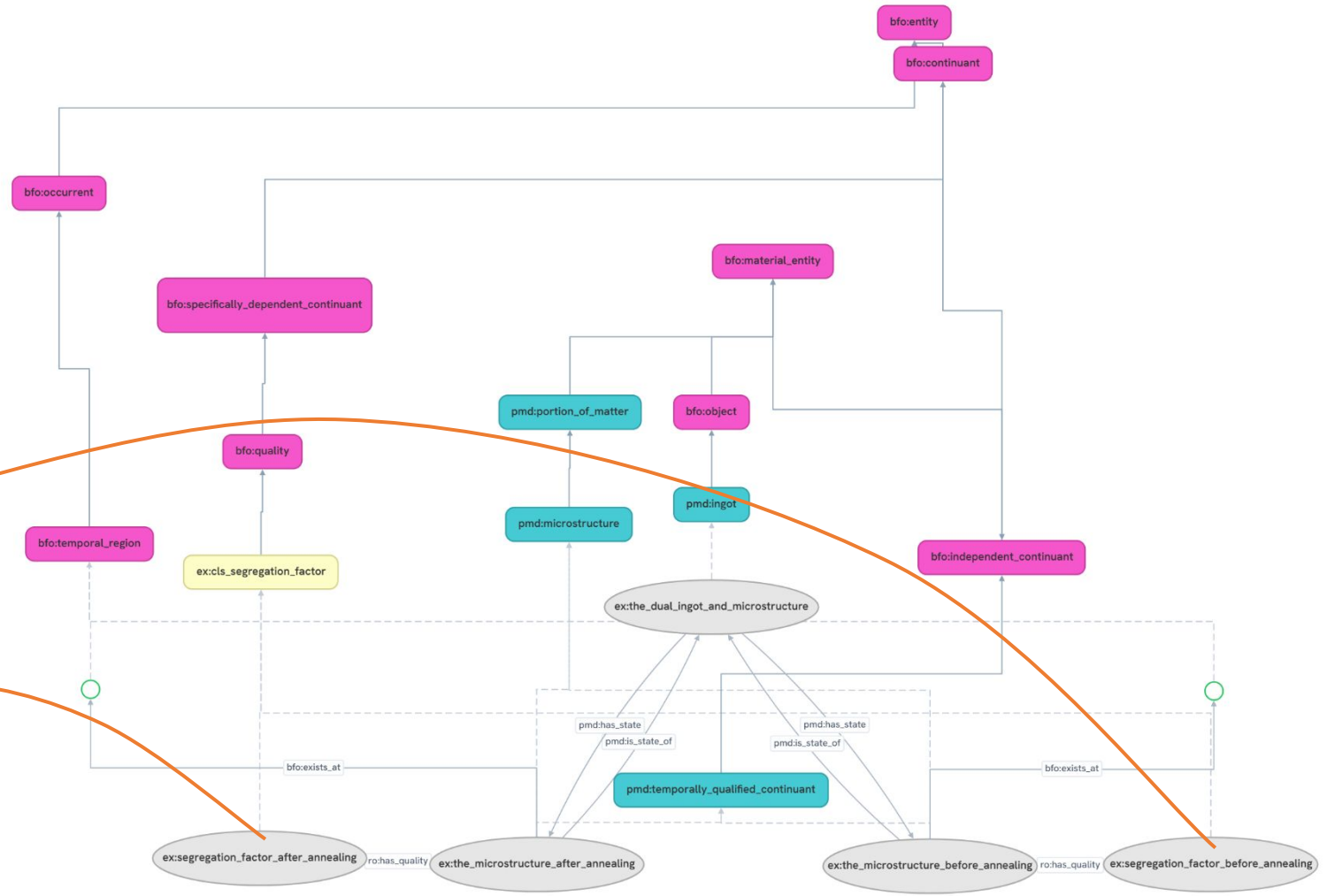
temporally qualified continuant (pmd:PMD_0000068)
skos:definition “A temporally qualified continuant is a continuant that, by reference to a determinate temporal region, is such as to possess its properties only for so long as that period pertains, being otherwise in its *essentiality* unchanged.”

Side-note:
“anchor” === ('independent continuant'
and (not ('temporally qualified continuant')))



Pattern 2 - Temporal dimensions of a process and the participant objects states

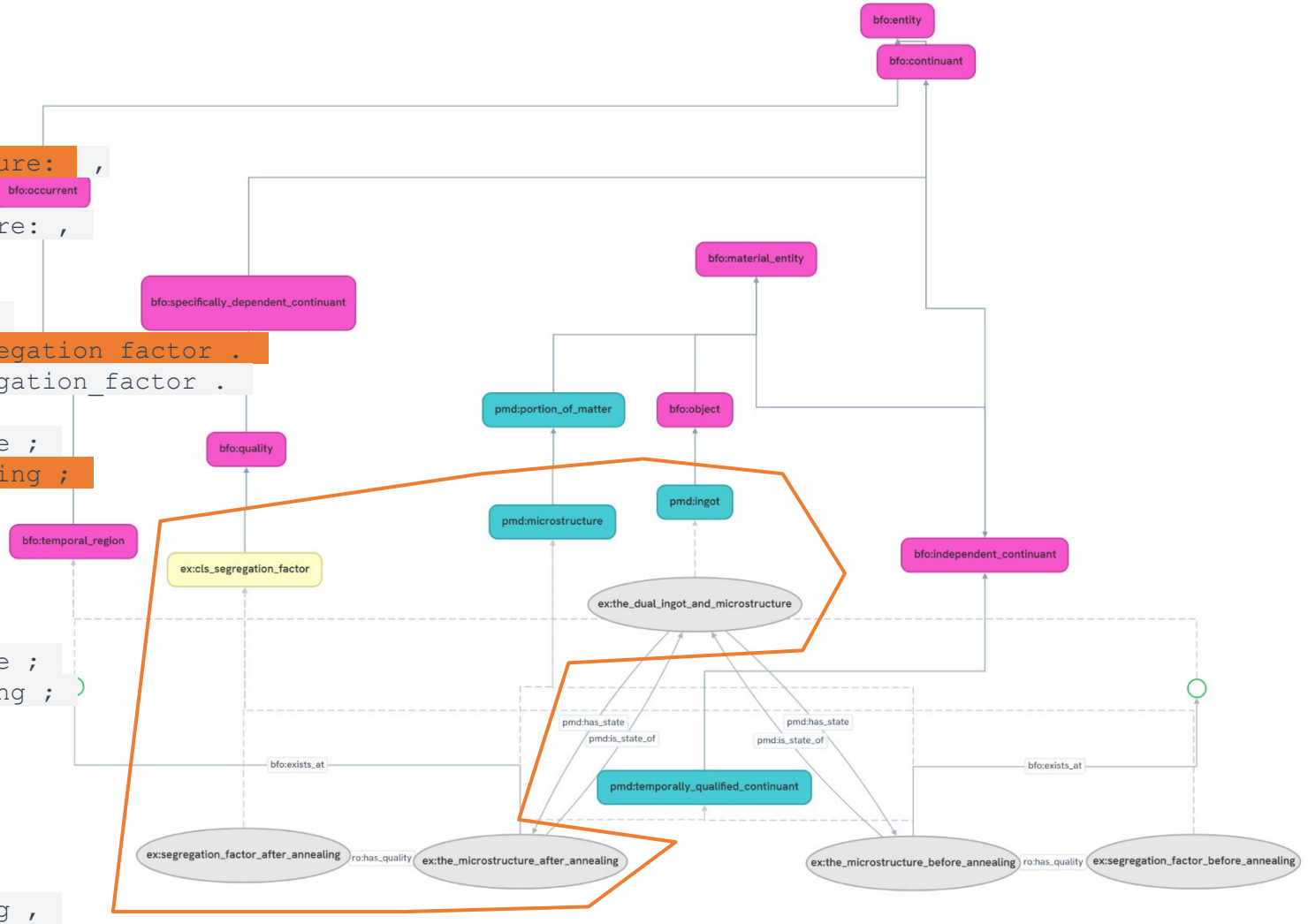
https://www.doitpoms.ac.uk/tlplib/casting/microsegregation.php



Simulation and experimental validation of the effect of superheat on macrosegregation in large-size steel ingots - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Segregation-ratio-patterns-of-carbon-during-the-casting-process-predicted-by-75M-upper_fig1_339273752 [accessed 22 Jun 2026]

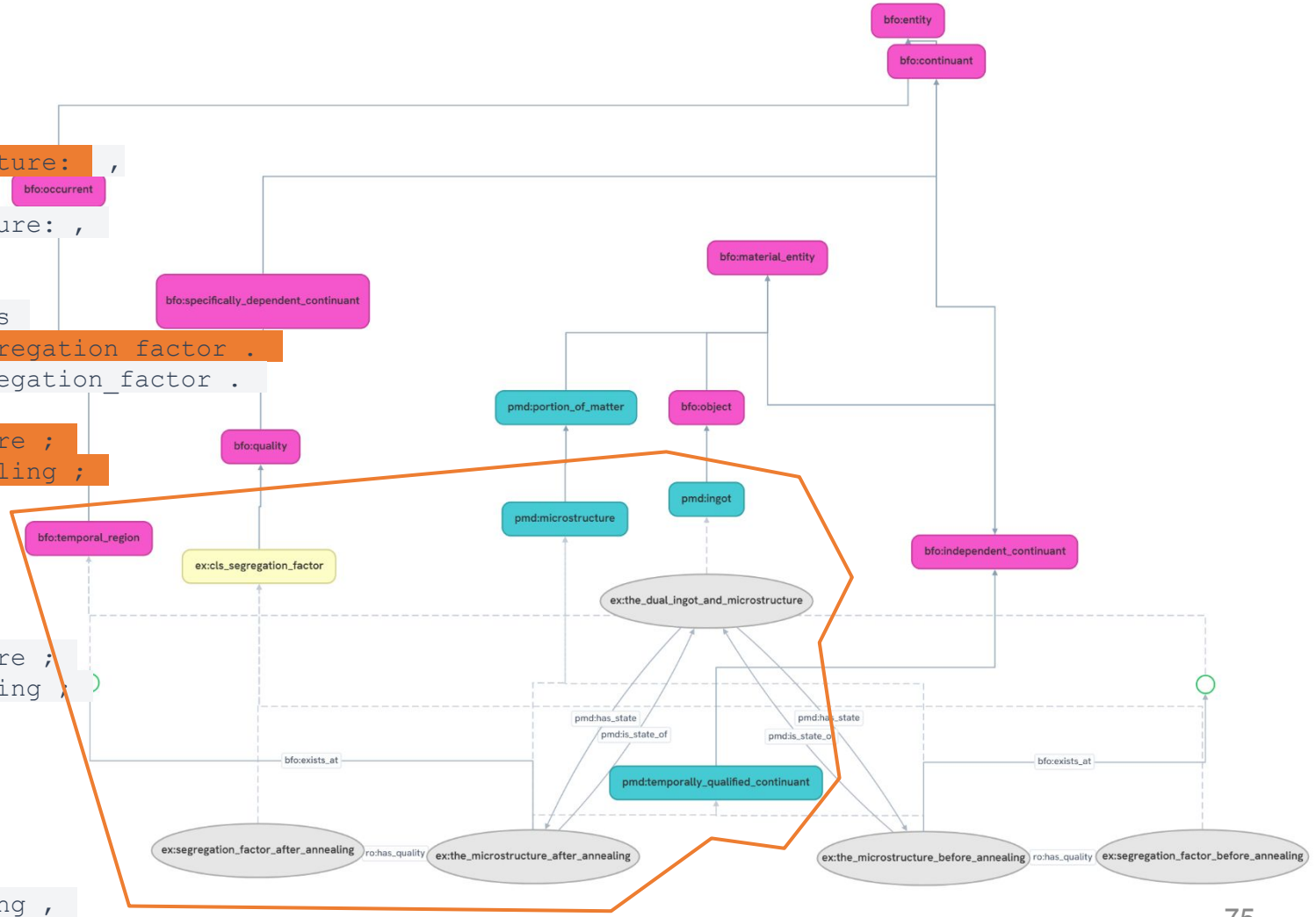
Pattern 2 - Temporal dimensions of a process and the participant objects states

```
# Define Object (Anchor)
ex:the_dual_ingot_and_microstructure a ingot: .
# ...and its variable states:
ex:the_microstructure_before_annealing a microstructure: ,
    temporally_qualified_continuant: .
ex:the_microstructure_after_annealing a microstructure: ,
    temporally_qualified_continuant: .

# Describe the states with their individual qualites
ex:segregation_factor_before_annealing a ex:cls_segregation_factor .
ex:segregation_factor_after_annealing a ex:cls_segregation_factor .
ex:the_microstructure_before_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_before_annealing ;
    exists_at: [
        a temporal_region:
    ] .

ex:the_microstructure_after_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_after_annealing ;
    exists_at: [
        a temporal_region:
    ] .

# Link the states to their anchor and vice versa
ex:the_dual_ingot_and_microstructure
    has_state: ex:the_microstructure_before_annealing ,
        ex:the_microstructure_after_annealing .
```



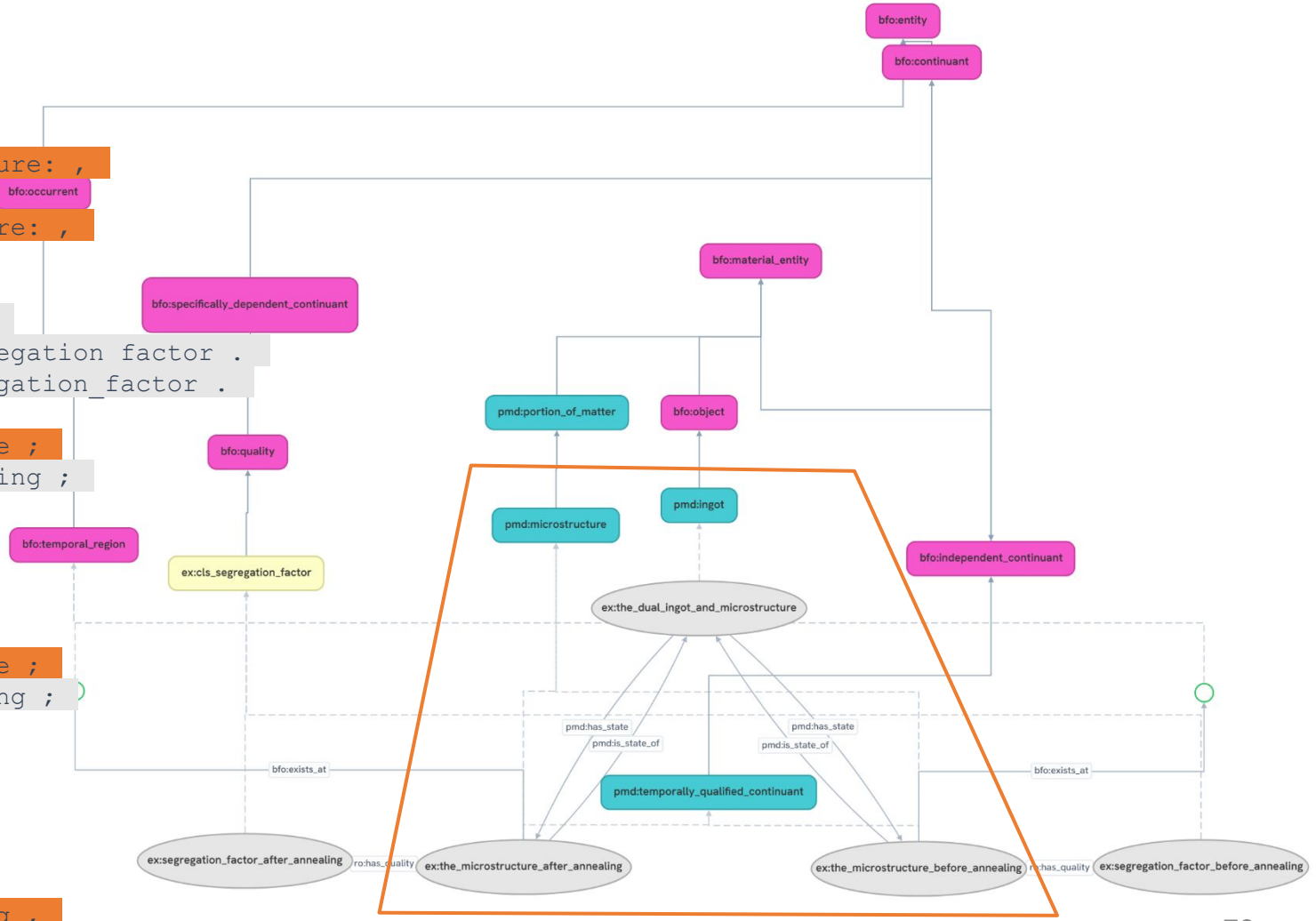
Pattern 2 - Temporal dimensions of a process and the participan objects states

```
# Define Object (Anchor)
ex:the_dual_ingot_and_microstructure a ingot: .
# ...and its variable states:
ex:the_microstructure_before_annealing a microstructure: ,
    temporally_qualified_continuant: .
ex:the_microstructure_after_annealing a microstructure: ,
    temporally_qualified_continuant: .

# Describe the states with their individual qualites
ex:segregation_factor_before_annealing a ex:cls_segregation_factor .
ex:segregation_factor_after_annealing a ex:cls_segregation_factor .
ex:the_microstructure_before_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_before_annealing ;
    exists_at: [
        a temporal_region:
    ] .

ex:the_microstructure_after_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_after_annealing ;
    exists_at: [
        a temporal_region:
    ] .

# Link the states to their anchor and vice versa
ex:the_dual_ingot_and_microstructure
    has_state: ex:the_microstructure_before_annealing ,
        ex:the_microstructure_after_annealing .
```



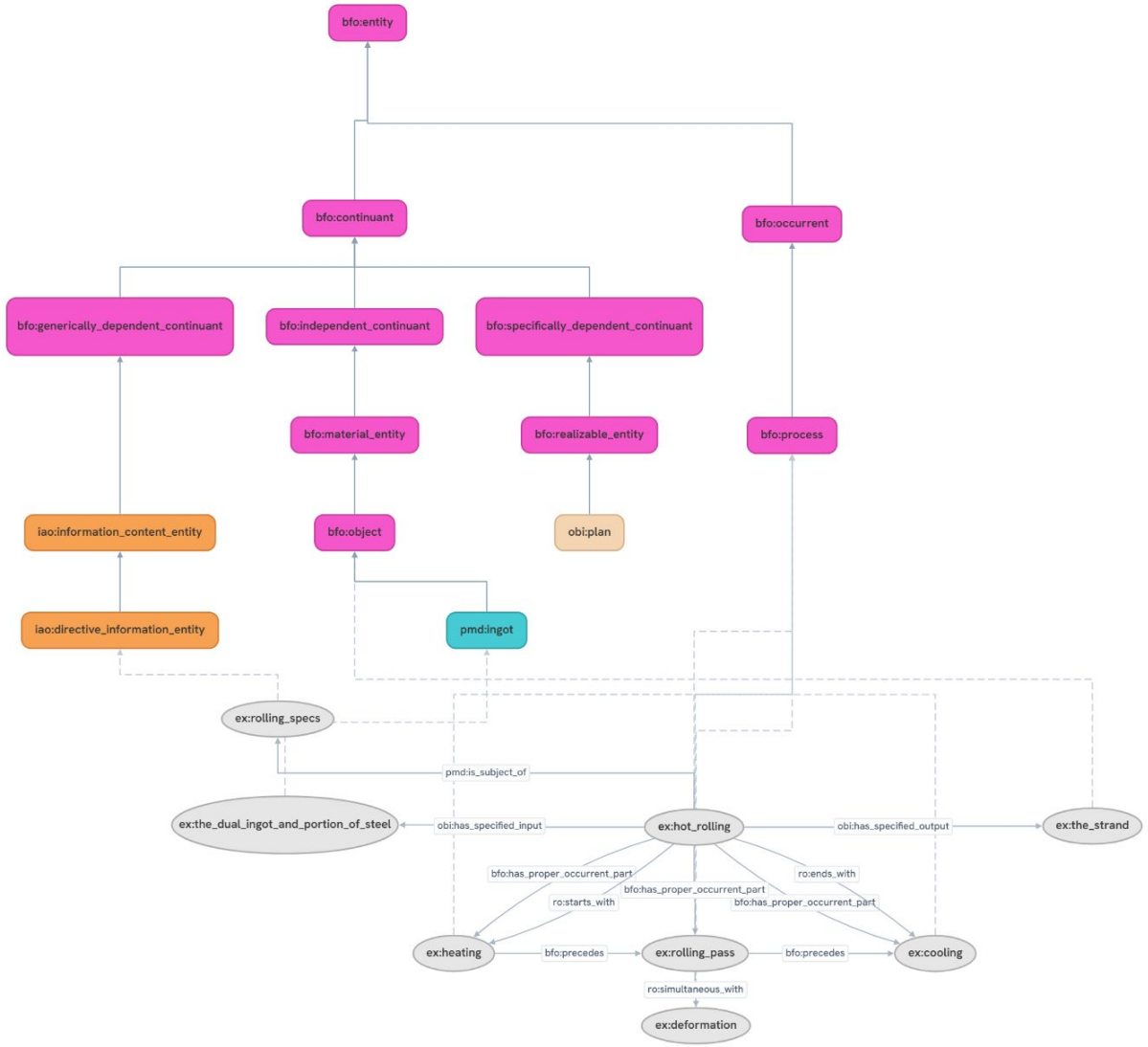
Pattern 2 - Temporal dimensions of a process and the participan objects states

```
# Define Object (Anchor)
ex:the_dual_ingot_and_microstructure a ingot: .
# ...and its variable states:
ex:the_microstructure_before_annealing a microstructure: ,
    temporally_qualified_continuant: .
ex:the_microstructure_after_annealing a microstructure: ,
    temporally_qualified_continuant: .

# Describe the states with their individual qualites
ex:segregation_factor_before_annealing a ex:cls_segregation_factor .
ex:segregation_factor_after_annealing a ex:cls_segregation_factor .
ex:the_microstructure_before_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_before_annealing ;
    exists_at: [
        a temporal_region:
    ] .

ex:the_microstructure_after_annealing
    is_state_of: ex:the_dual_ingot_and_microstructure ;
    has_quality: ex:segregation_factor_after_annealing ;
    exists_at: [
        a temporal_region:
    ] .

# Link the states to their anchor and vice versa
ex:the_dual_ingot_and_microstructure
    has_state: ex:the_microstructure_before_annealing ,
        ex:the_microstructure_after_annealing .
```



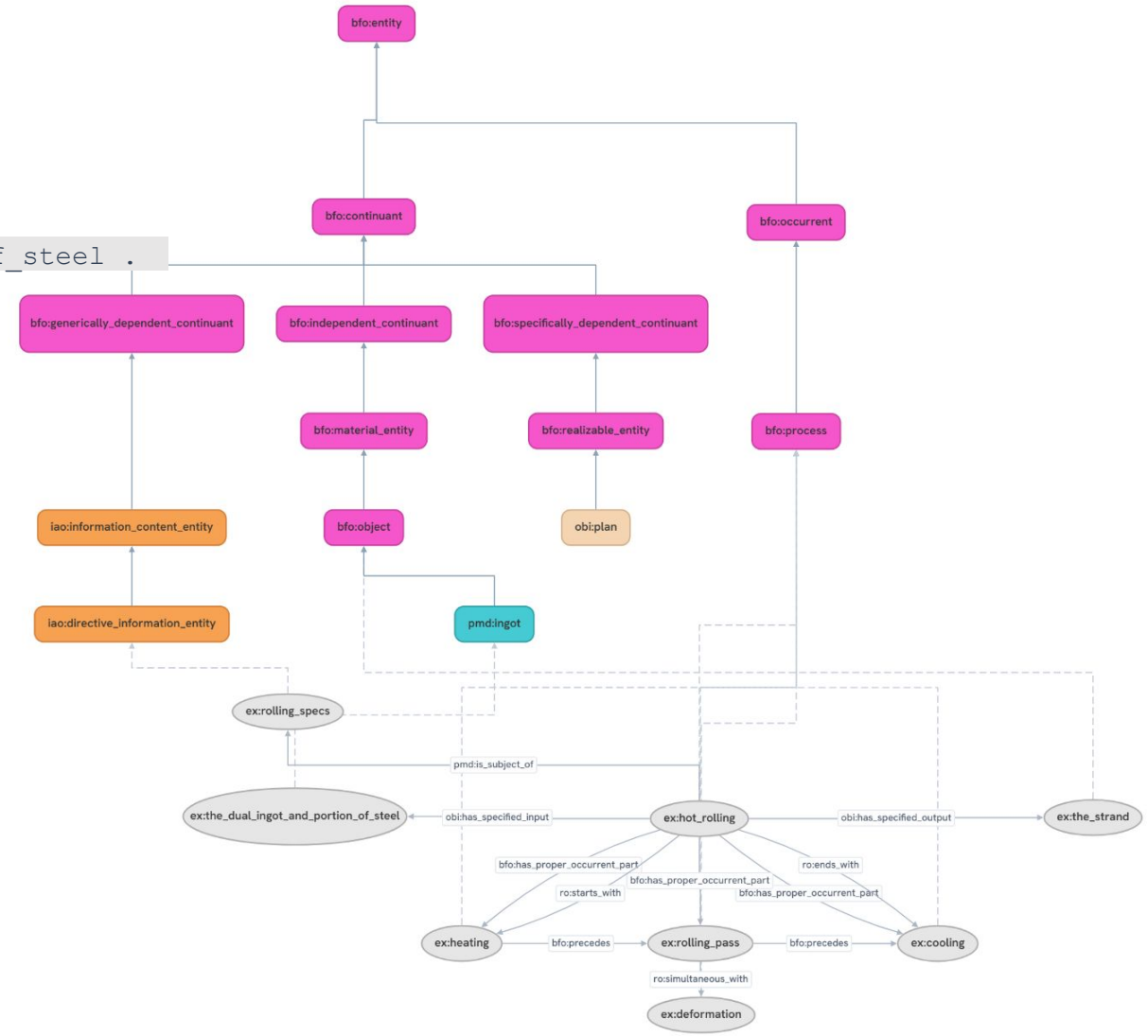
Pattern 3 - A process with in/output and a chain of subprocesses

https://www.wshampshire.com/industries/steel-aluminum/



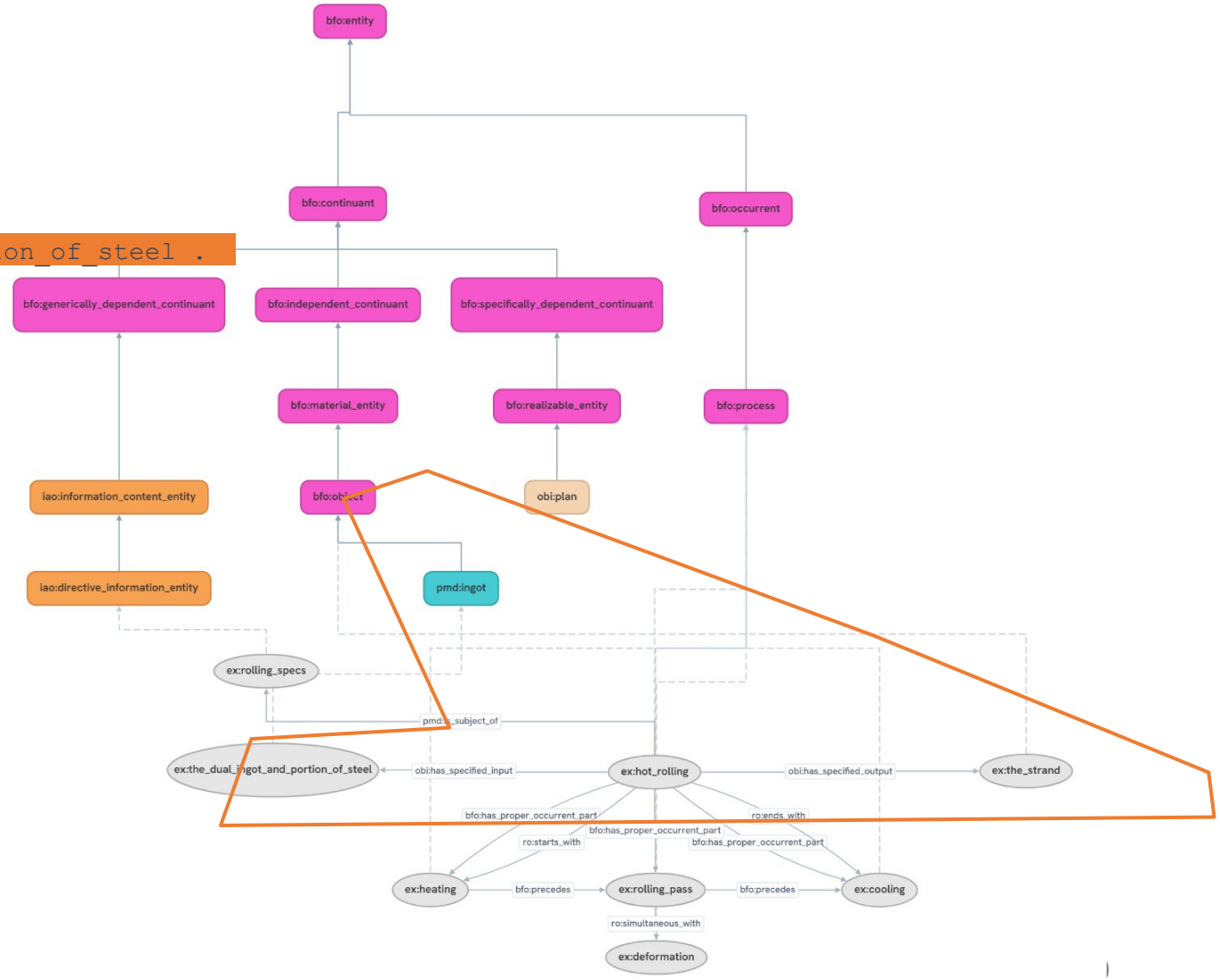
Pattern 3 - A process with in/output and a chain of subprocesses

```
ex:the dual ingot and portion_of_steel a ingot: .
ex:the_strand a object: .

ex:hot_rolling a process: .
ex:hot rolling has specified input: ex:the dual ingot_and_portion_of_steel .
ex:hot_rolling has_specified_output: ex:the_strand .

ex:heating a process: .
ex:rolling_pass a process: .
ex:deformation a process: .
ex:cooling a process: .

ex:rolling_specs a directive_information_entity: .

ex:hot rolling realizes: [
        a plan: ;
        concretizes: ex:rolling_specs
    ] ;
    is subject of: ex:rolling specs ;
    starts with: ex:heating ;
    ends with: ex:cooling ;
    has proper occurrent part: ex:heating,
        ex:rolling_pass, ex:cooling .

ex:heating precedes: ex:rolling pass .
ex:rolling_pass precedes: ex:cooling .

ex:rolling_pass simultaneous_with: ex:deformation .
```



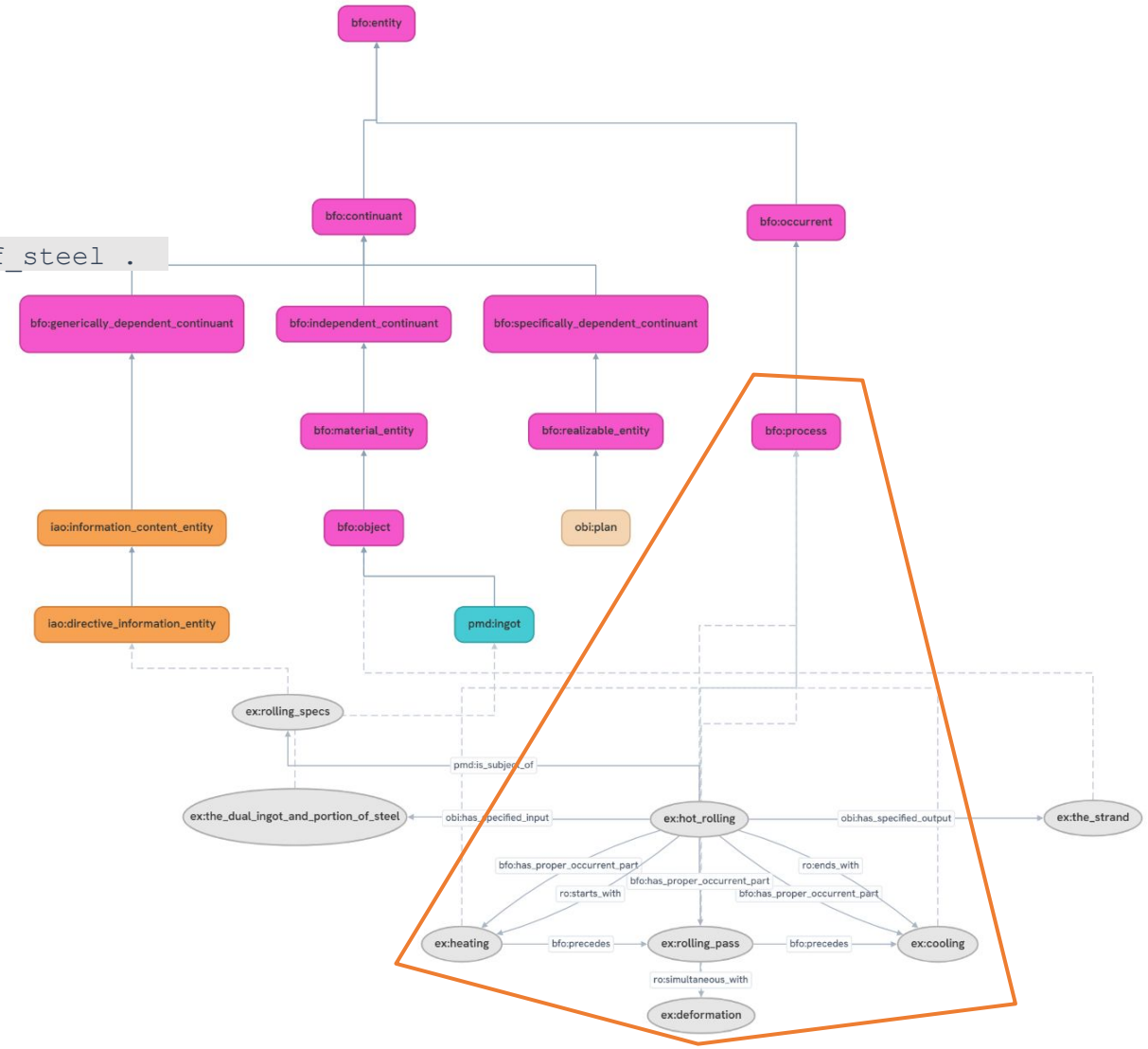
Pattern 3 - A process with in/output and a chain of subprocesses

```
ex:the_dual_ingot_and_portion_of_steel a ingot: .
ex:the_strand a object: .

ex:hot_rolling a process: .
ex:hot_rolling has specified input: ex:the_dual_ingot_and_portion_of_steel .
ex:hot_rolling has specified output: ex:the_strand .

ex:heating a process: .
ex:rolling_pass a process: .
ex:deformation a process: .
ex:cooling a process: .

ex:rolling_specs a directive_information_entity: .

ex:hot_rolling realizes: [
        a plan: ;
        concretizes: ex:rolling_specs
    ] ;
    is_subject_of: ex:rolling_specs ;
    starts_with: ex:heating ;
    ends_with: ex:cooling ;
    has_proper_occurrent_part: ex:heating,
        ex:rolling_pass, ex:cooling .

ex:heating precedes: ex:rolling_pass .
ex:rolling_pass precedes: ex:cooling .

ex:rolling_pass simultaneous_with: ex:deformation .
```



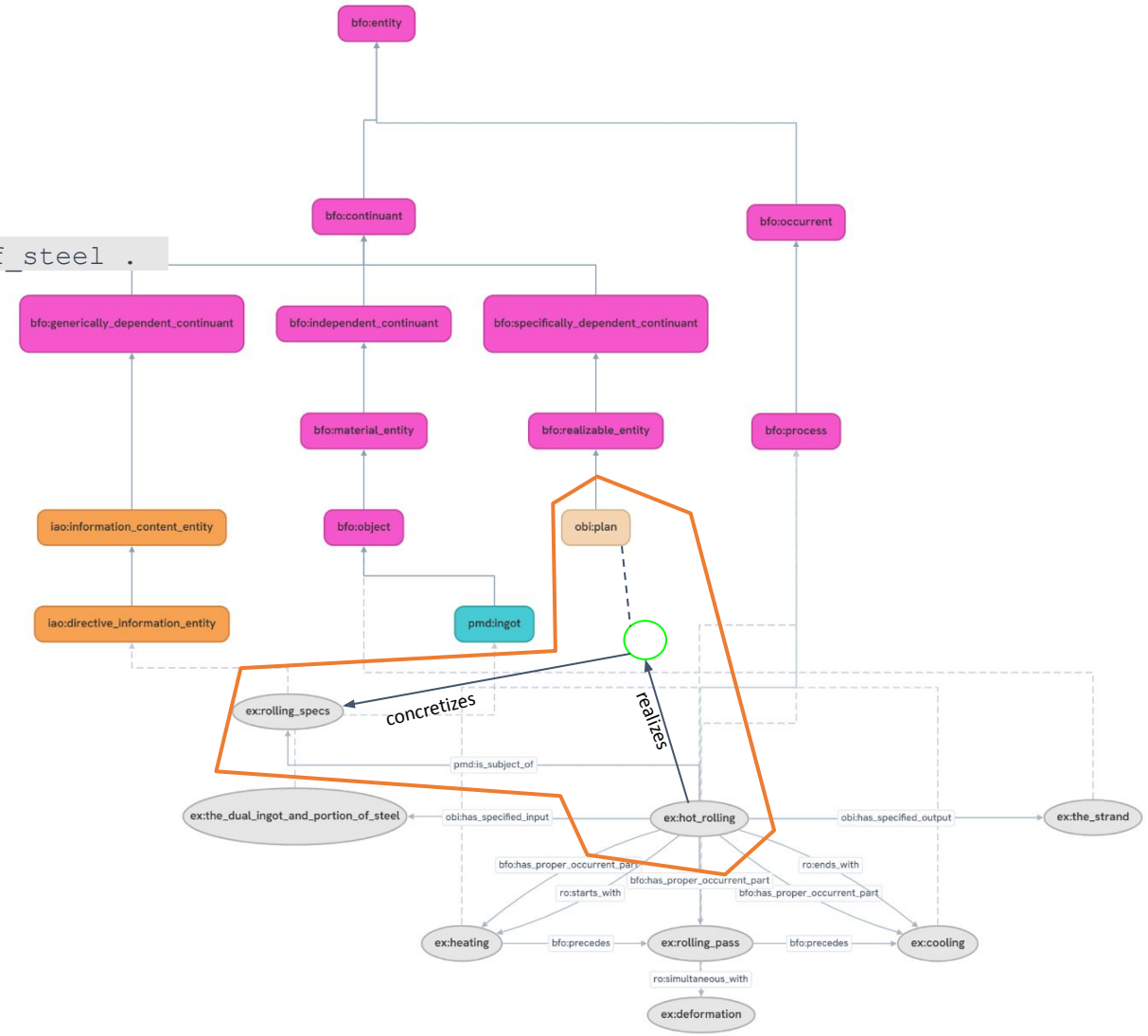
Pattern 3 - A process with in/output and a chain of subprocesses

```
ex:the dual ingot and portion_of_steel a ingot: .
ex:the_strand a object: .

ex:hot rolling a process: .
ex:hot rolling has specified input: ex:the dual ingot_and_portion_of_steel .
ex:hot_rolling has_specified_output: ex:the_strand .

ex:heating a process: .
ex:rolling pass a process: .
ex:deformation a process: .
ex:cooling a process: .

ex:rolling_specs a directive_information_entity: .

ex:hot rolling realizes: [
        a plan: ;
        concretizes: ex:rolling_specs
    ] ;
    is subject of: ex:rolling specs ;
    starts with: ex:heating ;
    ends with: ex:cooling ;
    has proper occurrent part: ex:heating,
        ex:rolling_pass, ex:cooling .

ex:heating precedes: ex:rolling pass .
ex:rolling_pass precedes: ex:cooling .

ex:rolling_pass simultaneous_with: ex:deformation .
```

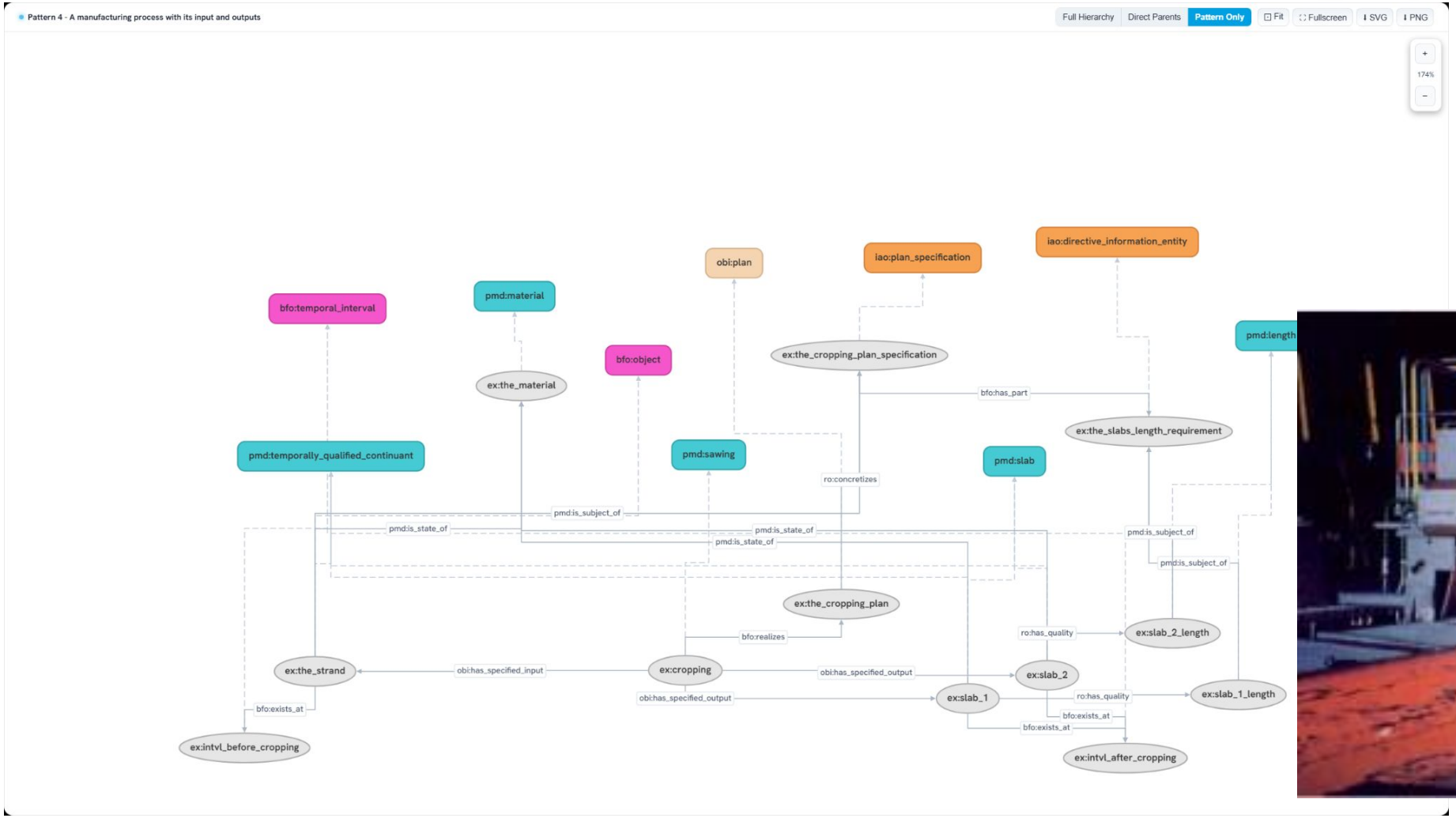


Pattern 3 - A process with in/output and a chain of subprocesses

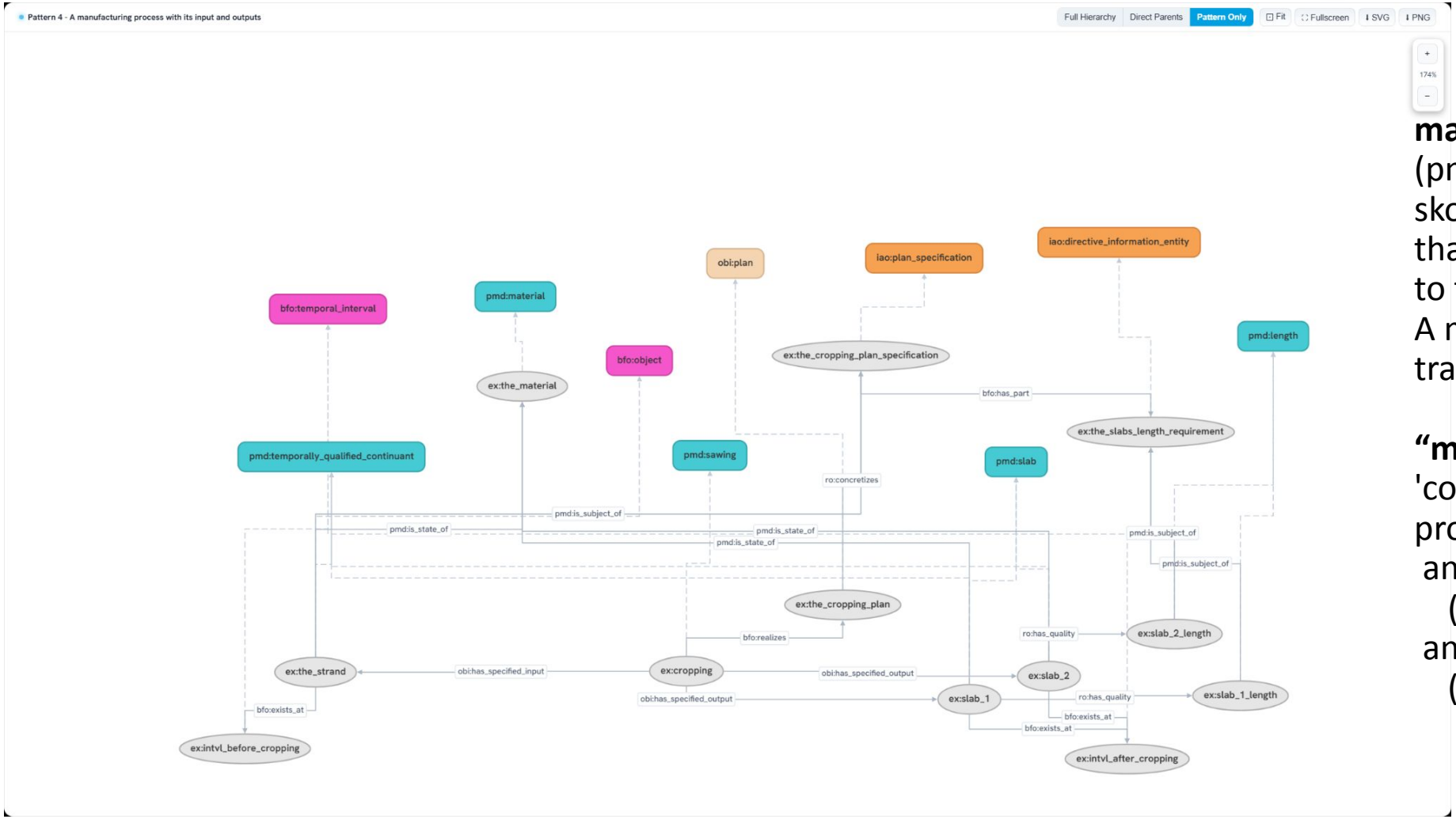
```
ex:the dual ingot and portion_of_steel a ingot: .
ex:the_strand a object: .

ex:hot_rolling a process: .
ex:hot rolling has specified input: ex:the dual ingot_and_portion_of_steel .
ex:hot_rolling has_specified_output: ex:the_strand .

ex:heating a process: .
ex:rolling_pass a process: .
ex:deformation a process: .
ex:cooling a process: .

ex:rolling_specs a directive_information_entity: .

ex:hot rolling realizes: [
        a plan: ;
        concretizes: ex:rolling_specs
    ] ;
    is subject of: ex:rolling_specs ;
    starts with: ex:heating ;
    ends with: ex:cooling ;
    has proper occurrent part: ex:heating,
        ex:rolling_pass, ex:cooling .

ex:heating precedes: ex:rolling pass .
ex:rolling_pass precedes: ex:cooling .

ex:rolling_pass simultaneous_with: ex:deformation .
```



Pattern 4 - A manufacturing process with its input and outputs



Pattern 4 - A manufacturing process with its input and outputs



manufacturing process (pmd:PMD_0000833)
skos:definition “A planned process that is driven by the primary intent to transform objects
A manufacturing process is always a transformative process.”

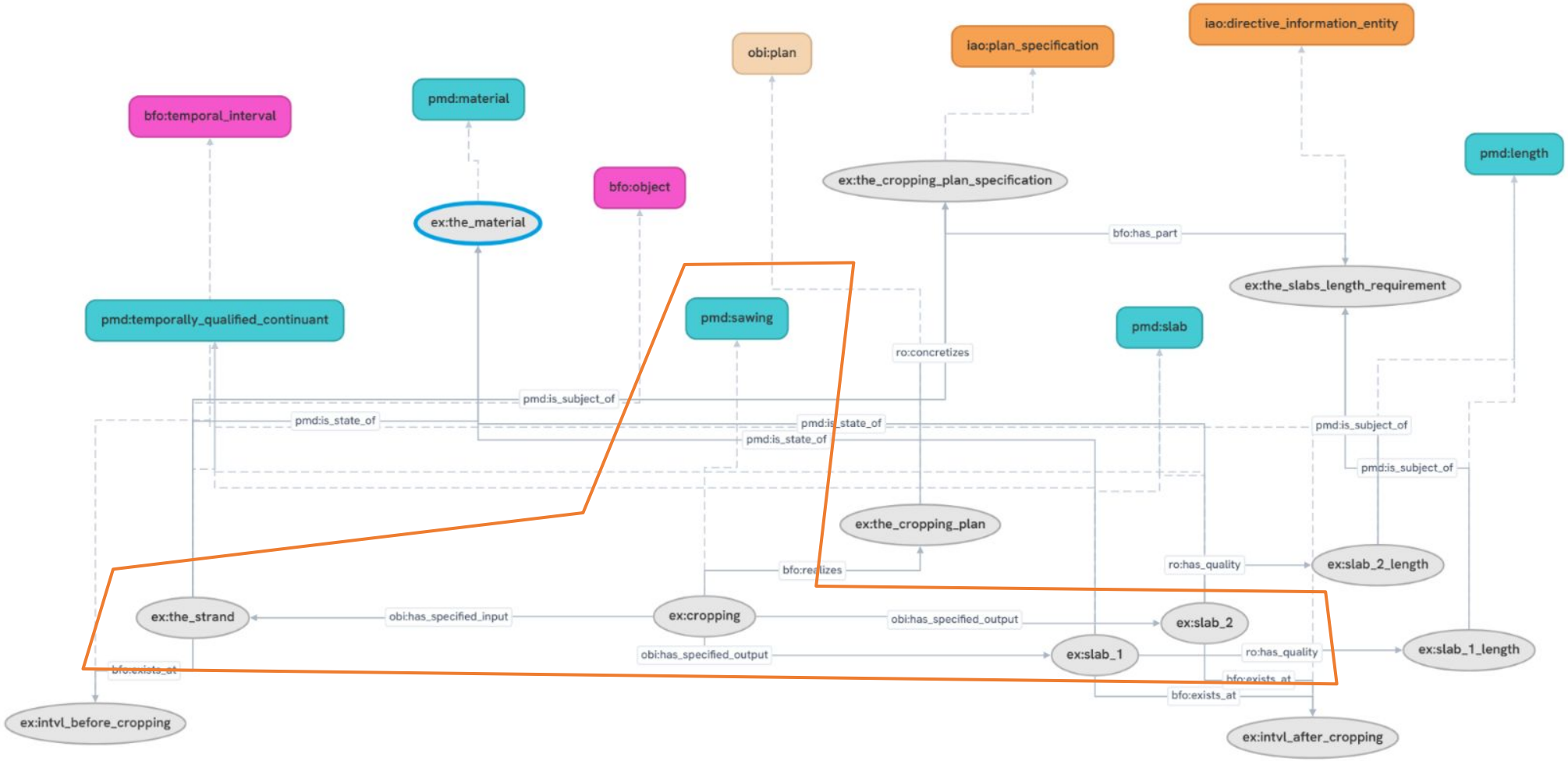
“manufacturing process” === 'completely executed planned process'
and ('has specified input' some ('object aggregate' or object))
and ('has specified output' some ('object aggregate' or object))

Pattern 4 - A manufacturing process with its input and outputs

Full Hierarchy | Direct Parents | Pattern Only | Fit | Fullscreen | SVG | PNG

174%

in- and output

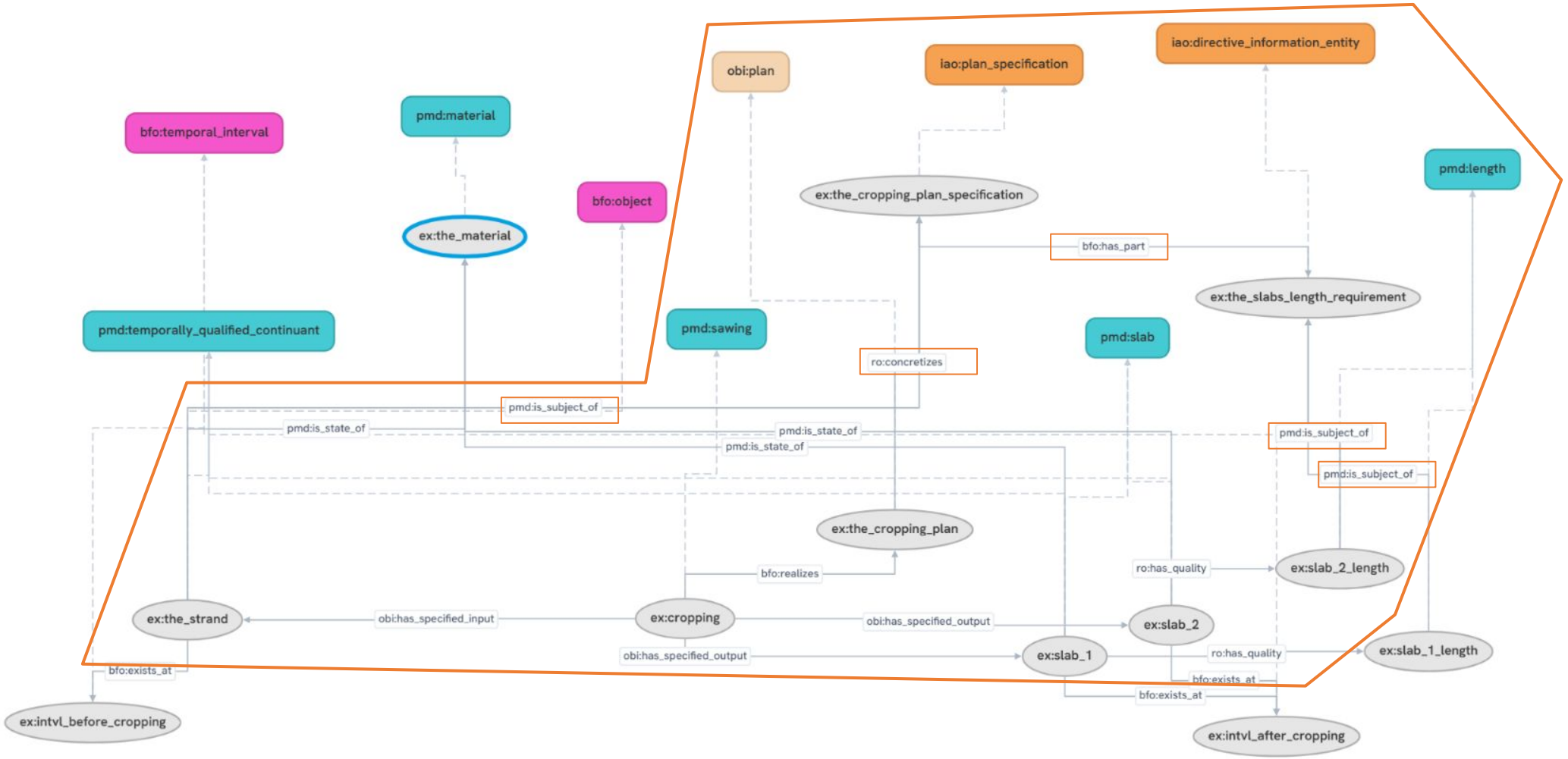


Pattern 4 - A manufacturing process with its input and outputs

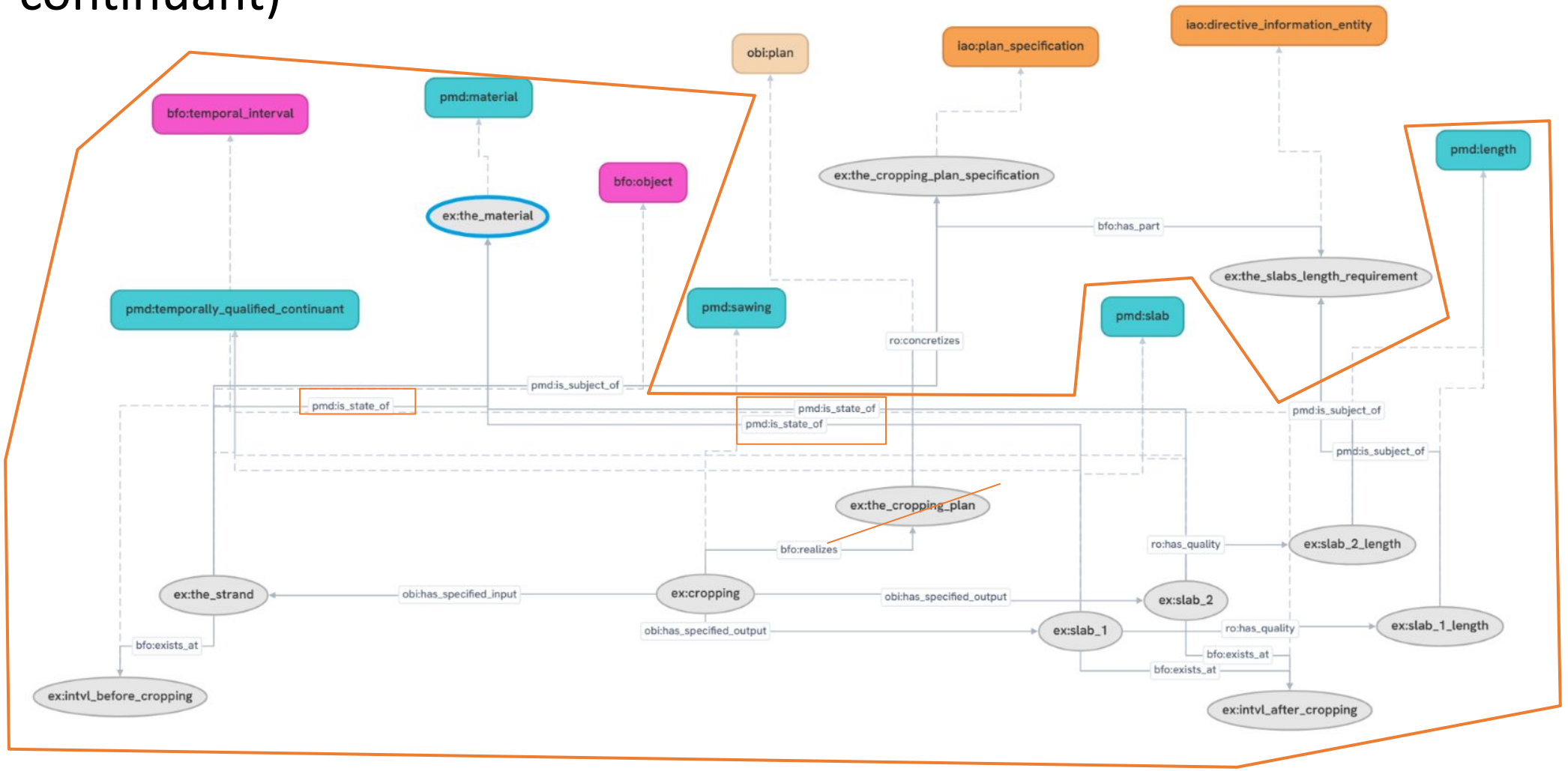
Full Hierarchy | Direct Parents | Pattern Only | Fit | Fullscreen | SVG | PNG

174%

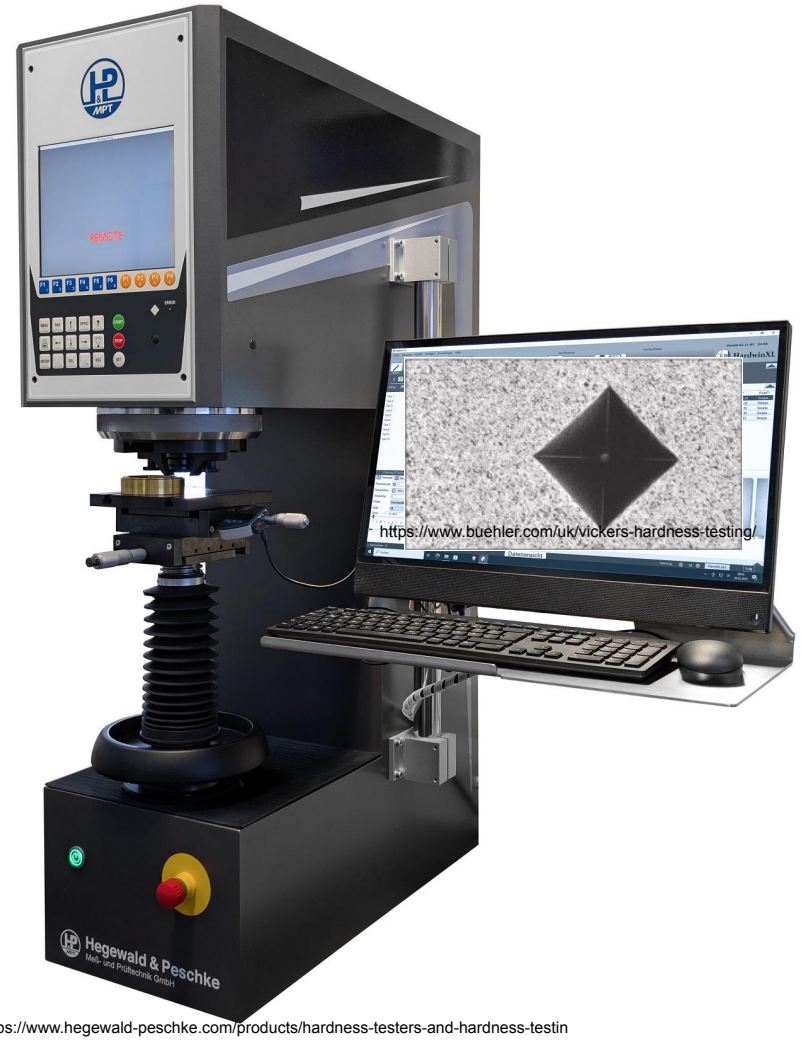
the plan with its parts and aboutness



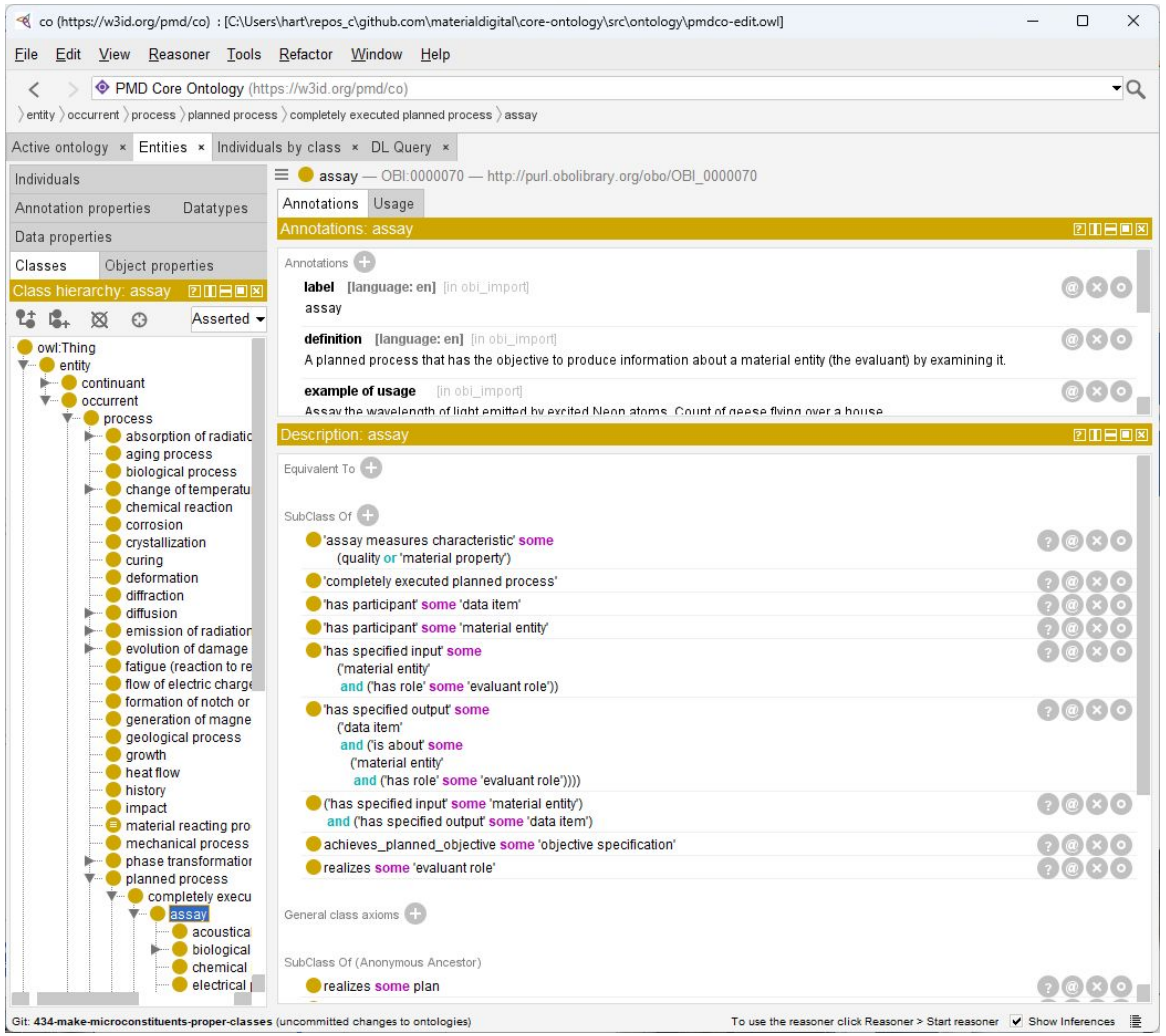
the material as “anchor” and the objects as TQC (temporally qualified continuant)



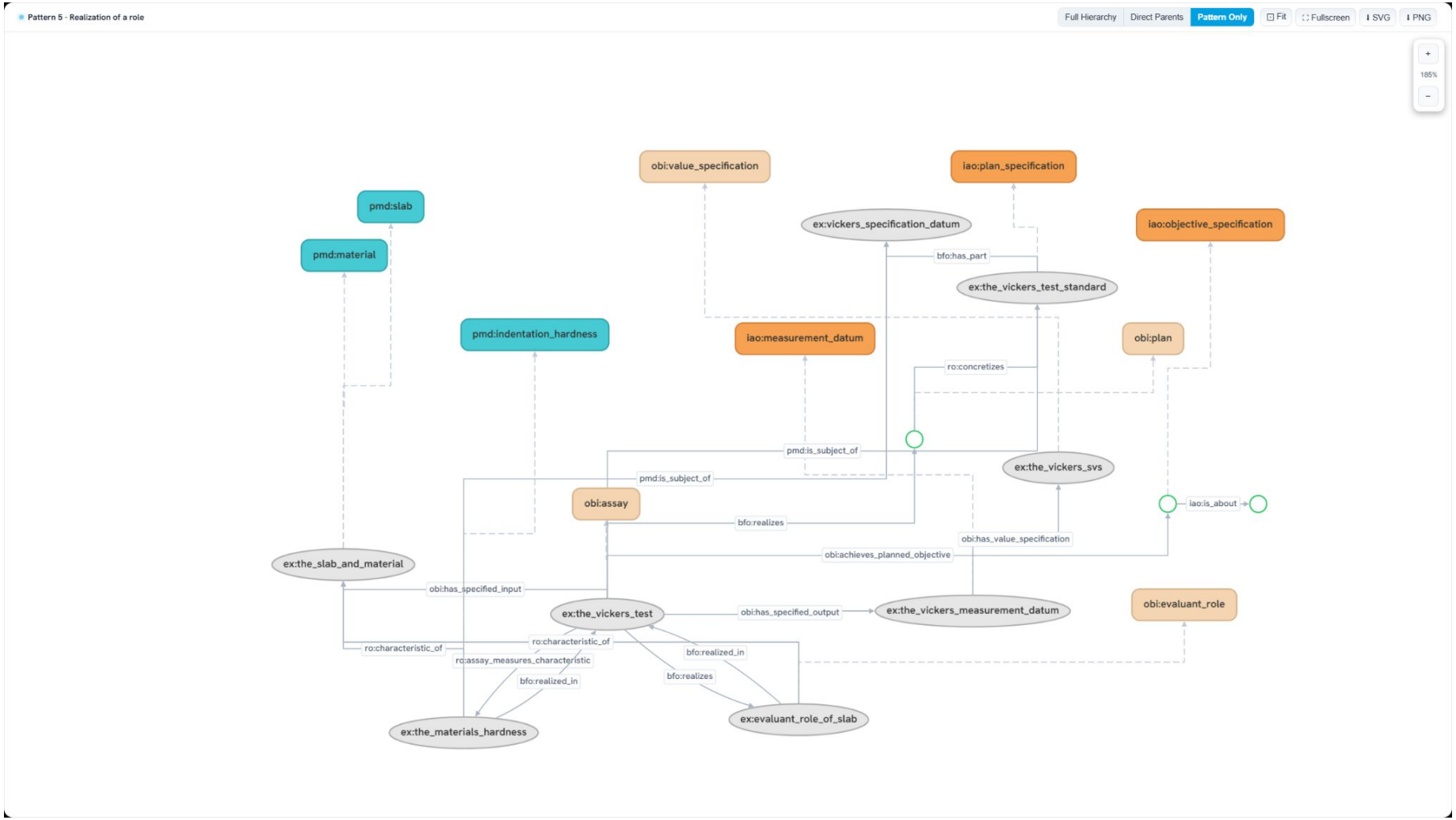
Pattern 5 - Assay and the realization of a evaluant role



https://www.hegewald-peschke.com/products/hardness-testers-and-hardness-testing-machines/universal-hardness-testers/universal-hardness-tester-hpuniversal-250-750-3000-1/

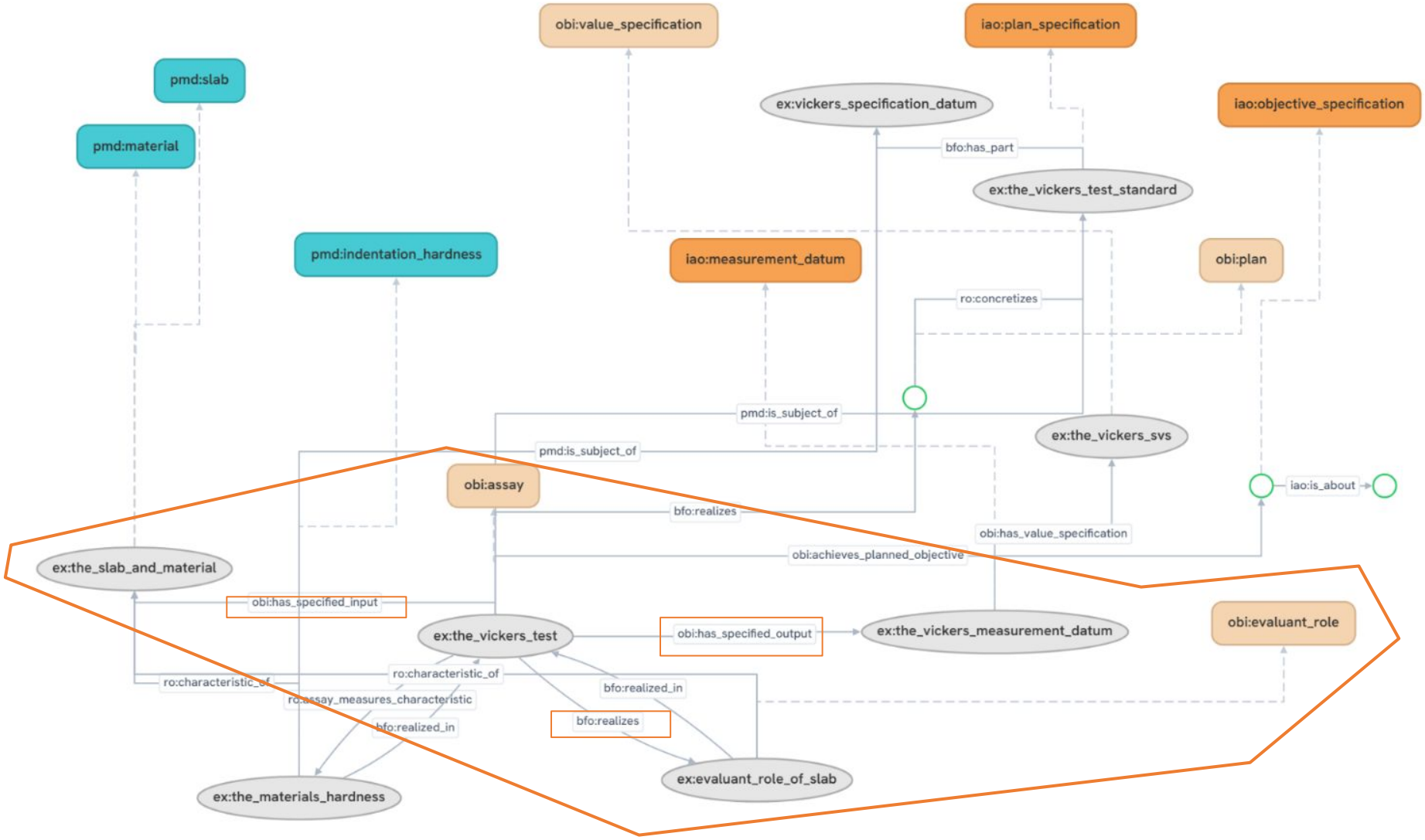


Pattern 5 - Assay and the realization of a evaluant role

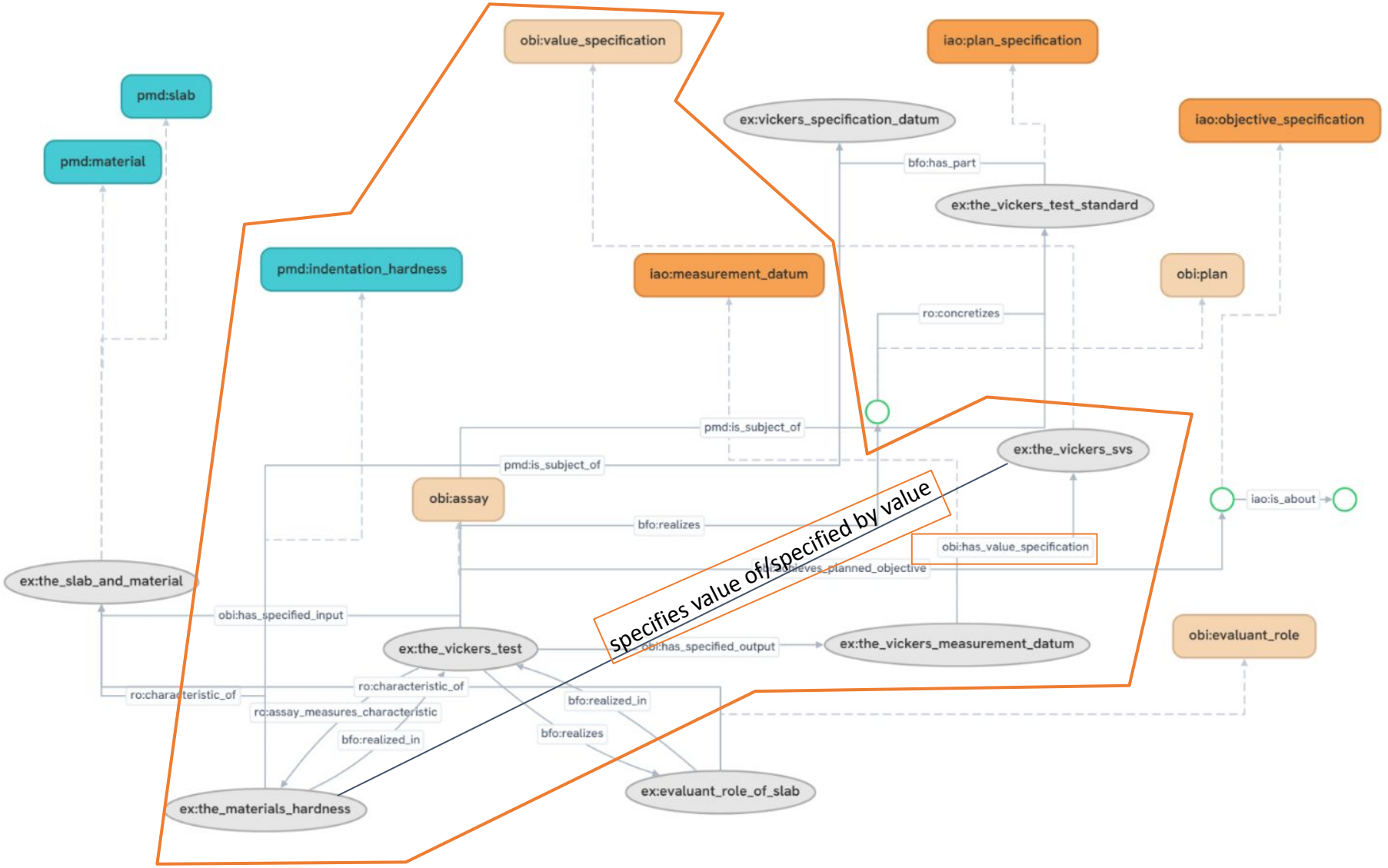


'has specified input' some
('material entity'
and ('has role' some 'evaluant role'))
'has specified output' some
('data item'
and ('is about' some
('material entity'
and ('has role' some 'evaluant role'))))

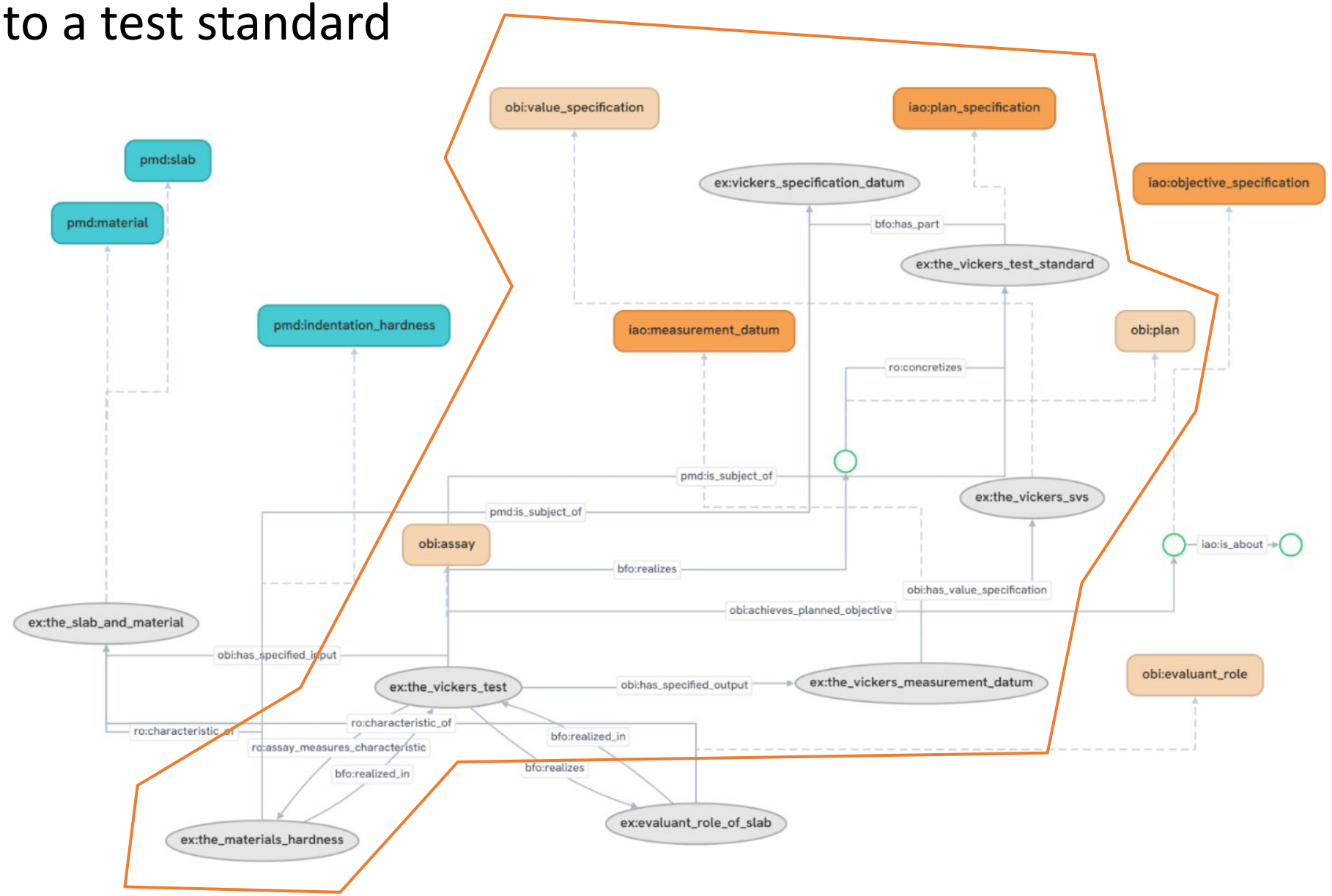
the assays input bearing an evaluant role and output data item



experimentally established value of a property



Reference to a test standard



Frequently applied pattern

These patterns are so frequent that they can be considered as “fixed shapes”

- Pattern 8a - Expressing a (scalar) value with "Scalar Value Specification"
- Pattern 6a - fundamental property
- Pattern 6c - relational property
- Pattern 7 - Measurement and setpoint

Usage Patterns | PMDco Documentation

Thank you!!

PMDco:
AI outlook

AI techniques

- **Ontologies & Logic Programming** – Explicitly encode knowledge using formal logic rules and structured relationships to enable symbolic reasoning (e.g., OWL).
- **Rule-Based Expert Systems** – Apply hand-crafted IF-THEN rules to mimic human expertise and make decisions through logical inference.
- **Traditional Machine Learning** – Learn patterns from labeled data using explicit features and statistical methods like decision trees.
- **Deep Neural Networks** – Automatically learn hierarchical feature representations through many interconnected layers trained on large datasets
- **Transformers** – Learn complex patterns directly from raw data using self-attention mechanisms to capture relationships across entire sequences (e.g., GPT, BERT)

Ontologies and LLMs

- Ontologies could be used as ground truth for LLMs
- LLMs could help to extract structured knowledge representations from ambiguous (natural language representations) and fill knowledge graphs
- LLMs can interact with ontologies via MCP (GitHub - ai4curation/owl-mcp: MCP server for OWL applications · GitHub)