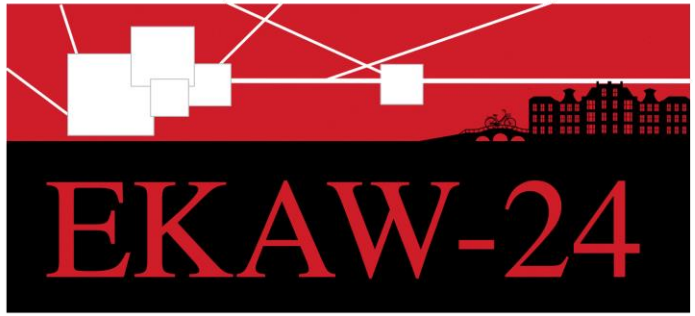
metaphacts

Smart Data. Smart Apps. Smart Decisions.

Tutorial: Semantic Knowledge Modeling - Ontologies & Vocabularies



Who we are

Dr. Peter Haase

Founder & Chief Scientific Officer

Irina Schmidt

Chief Marketing Officer

metaphacts

W3C Recommendation

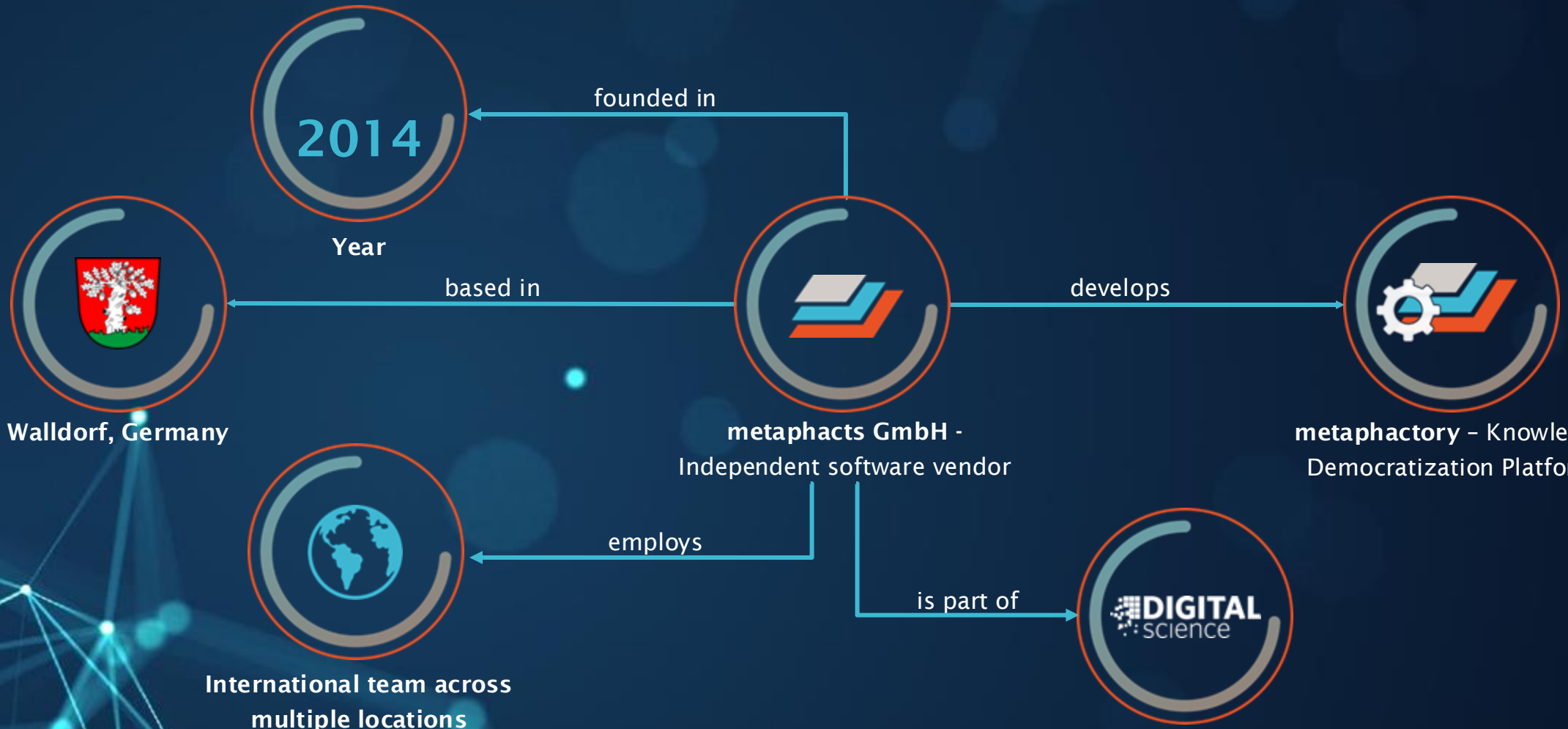
OWL 2 Web Ontology Language
Structural Specification and Functional-Style Syntax (Second Edition)

W3C Recommendation 11 December 2012

This version:
http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
Latest version (series 2):
http://www.w3.org/TR/owl2-syntax/
Latest Recommendation:
http://www.w3.org/TR/owl-syntax
Previous version:
http://www.w3.org/TR/2012/PER-owl2-syntax-20121018/
Editors:
Boris Motik, University of Oxford
Peter F. Patel-Schneider, Nuance Communications
Bijan Parsia, University of Manchester
Contributors: (in alphabetical order)
Conrad Bock, National Institute of Standards and Technology (NIST)
Achille Fokoue, IBM Corporation
Peter Haase, FZI Research Center for Information Technology
Rinke Hoekstra, University of Amsterdam
Ian Horrocks, University of Oxford
Alan Ruttenberg, Science Commons (Creative Commons)
Uli Sattler, University of Manchester
Michael Smith, Clark & Parsia

metaphacts GmbH - Knowledge Democratization and AI

Company Snapshot



Agenda

- Introduction
- What is semantic knowledge modeling – The metaphacts approach
- Visual modeling basics – Hands-on tutorial

- *Coffee break*

- Examples and experiences from industry
- Model-driven app building
- Advanced considerations in visual modelling: collaborative workflows, versioning
- Conclusion and Q&A

Setup & Supporting Material

- The tutorial is an interactive learning experience with hands-on exercises
- Trainer will show live examples and guide participants through them
- Participants have their individual metaphactory setup for hands-on exercises
- Slides and exercise material will be made available to the participants for their personal use

Watch Out For…

"Documentation"

- Links content being presented to built-in metaphactory 'Help' sections for more technical details

"Walkthrough"

- Links Example being presented to re-usable and complete pre-defined code
- Description of planned Hands-on exercises
- Code Snippets to carry out exercise, and entire solutions

Walkthrough: **https://**#yourmetaphactory#.tutorial.metaphacts.cloud

Documentation: **http://help.metaphacts.com/**resource/Help:Start

Interact with the trainers!

I'm Ready!

Raise a hand in **Checkpoints** to signify **readiness/completion**, e.g. ready to proceed, task complete, etc.

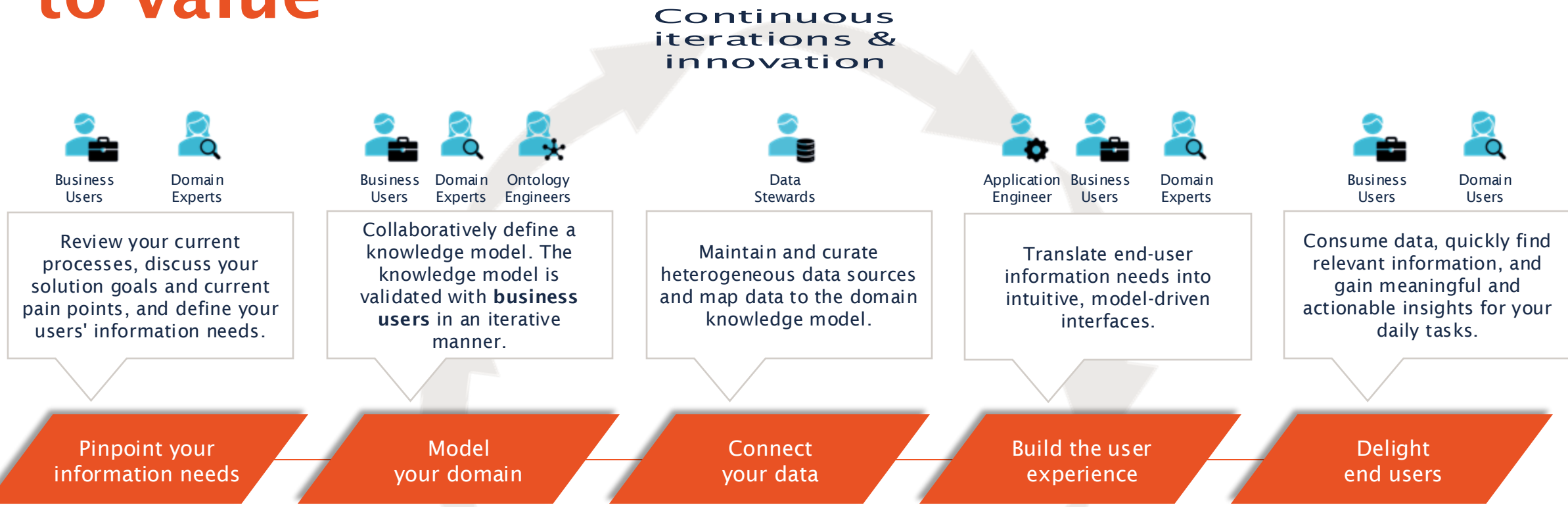
Speak out to flag **issues**, ask **questions**, request **support** with exercises, etc.

metaphacts

Smart Data. Smart Apps. Smart Decisions.

Semantic Modeling – The metaphacts Approach

Rapid application building – Fast path to value



Today **Week 2-4**

Powered by metaphactory

metaphactory - Knowledge Democratization Platform

metaphacts

KNOWLEDGE GRAPH MANAGEMENT

Visual authoring, visualization, versioning & cataloging of ontologies, vocabularies, datasets & queries

Data validation, provenance & lineage

END-USER ORIENTED INTERACTION

Abstracted view

One-stop knowledge hub

Intuitive UI for knowledge discovery, exploration, analytics, editing

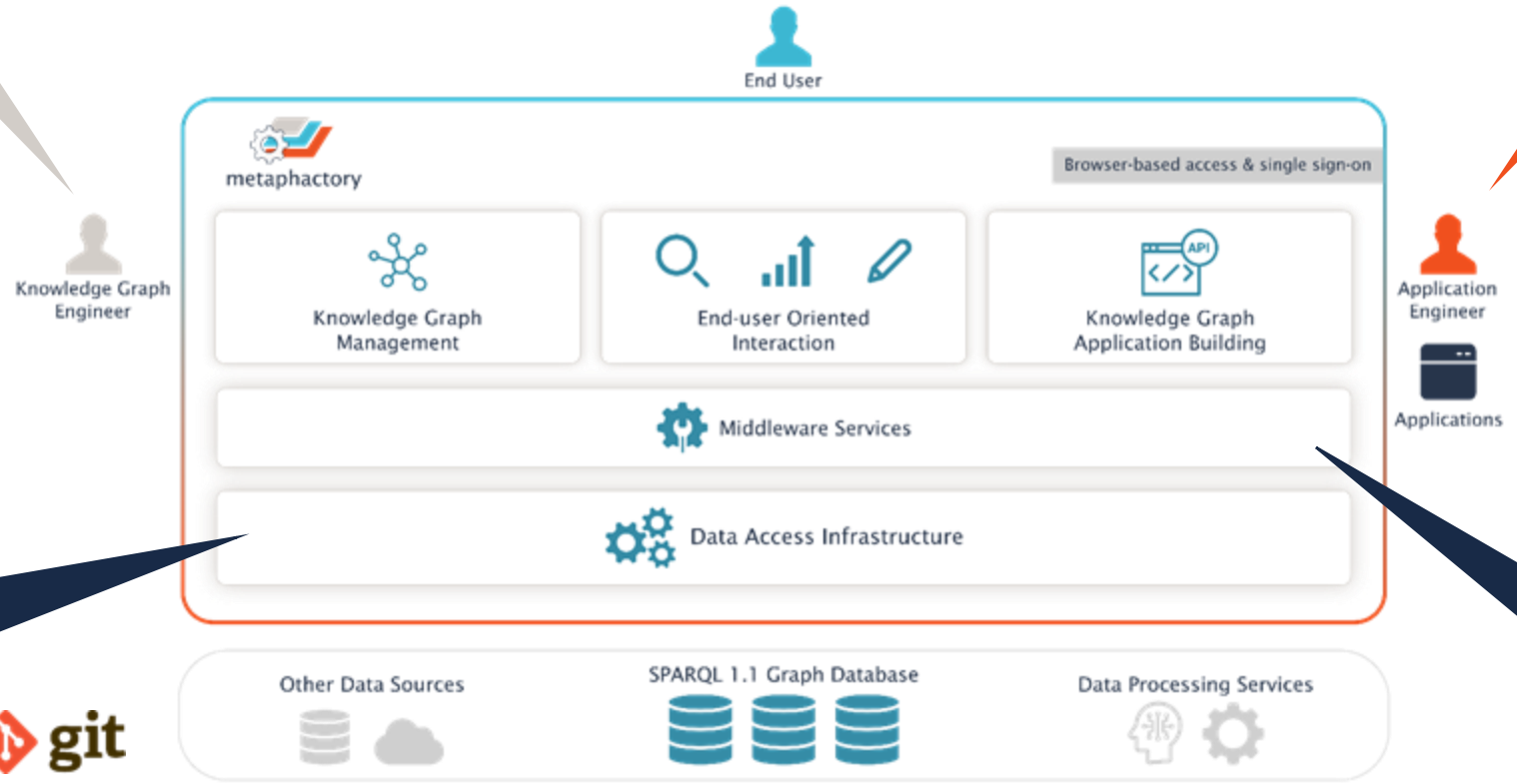
KNOWLEDGE GRAPH APPLICATION BUILDING

Low-code platform

Powerful template engine

Large library of Web components

Easy customization



DATA INTEGRATION & FEDERATION

Unified view on distributed and heterogenous data sources: graph databases, relational databases, REST APIs, machine learning algorithms

Transparent SPARQL federation

MIDDLEWARE SERVICES

Dynamic data-driven REST APIs based on queries

Role-based access control

Lookup & Reconciliation

Tableau – Web Data Connector Endpoint

W3C

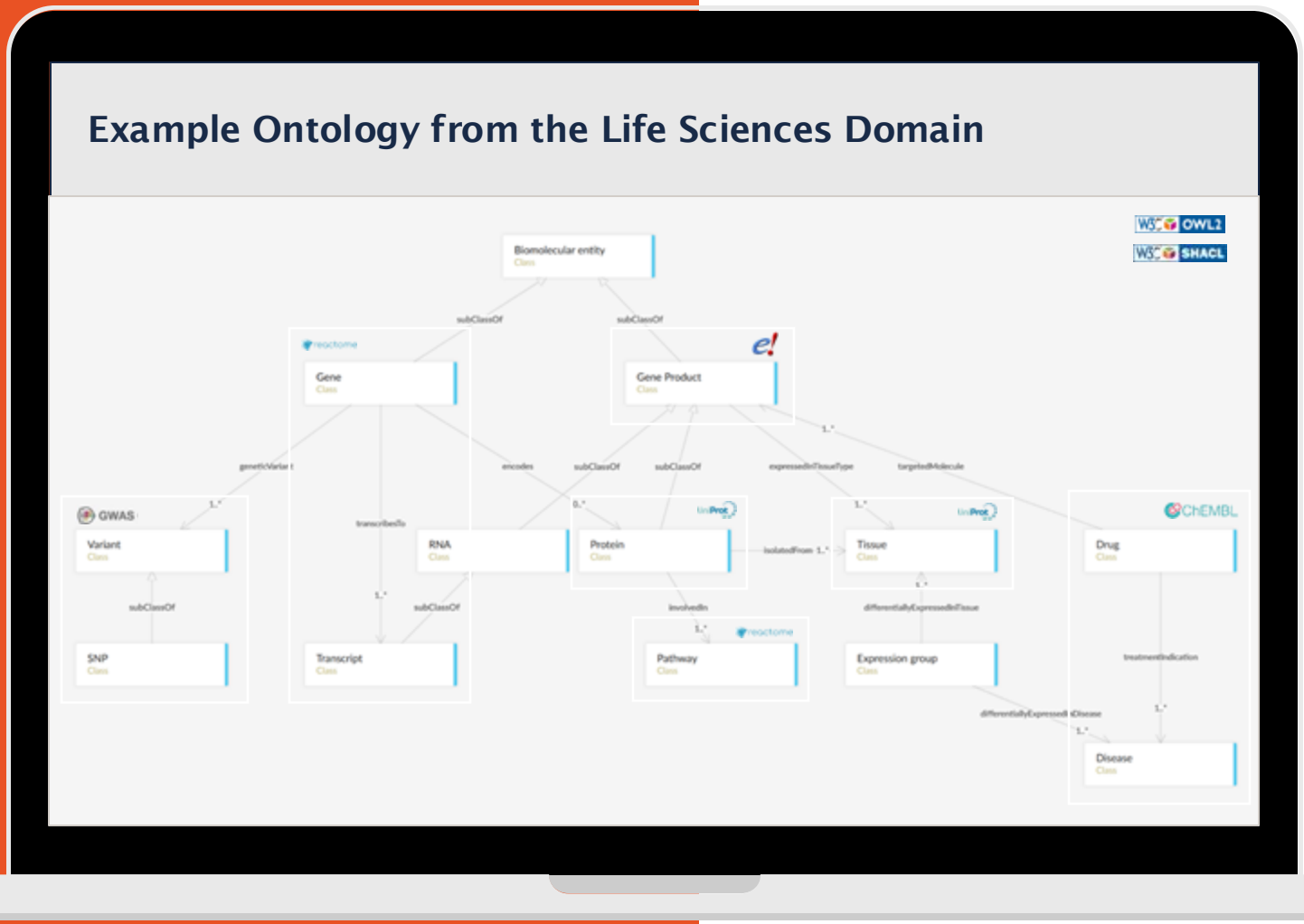
Run anywhere

git

FAIR Findable Accessible Interoperable Reusable

metaphacts

Building the knowledge graph



Knowledge Graph Engineer

Knowledge Steward (Domain Expert)

Data Steward

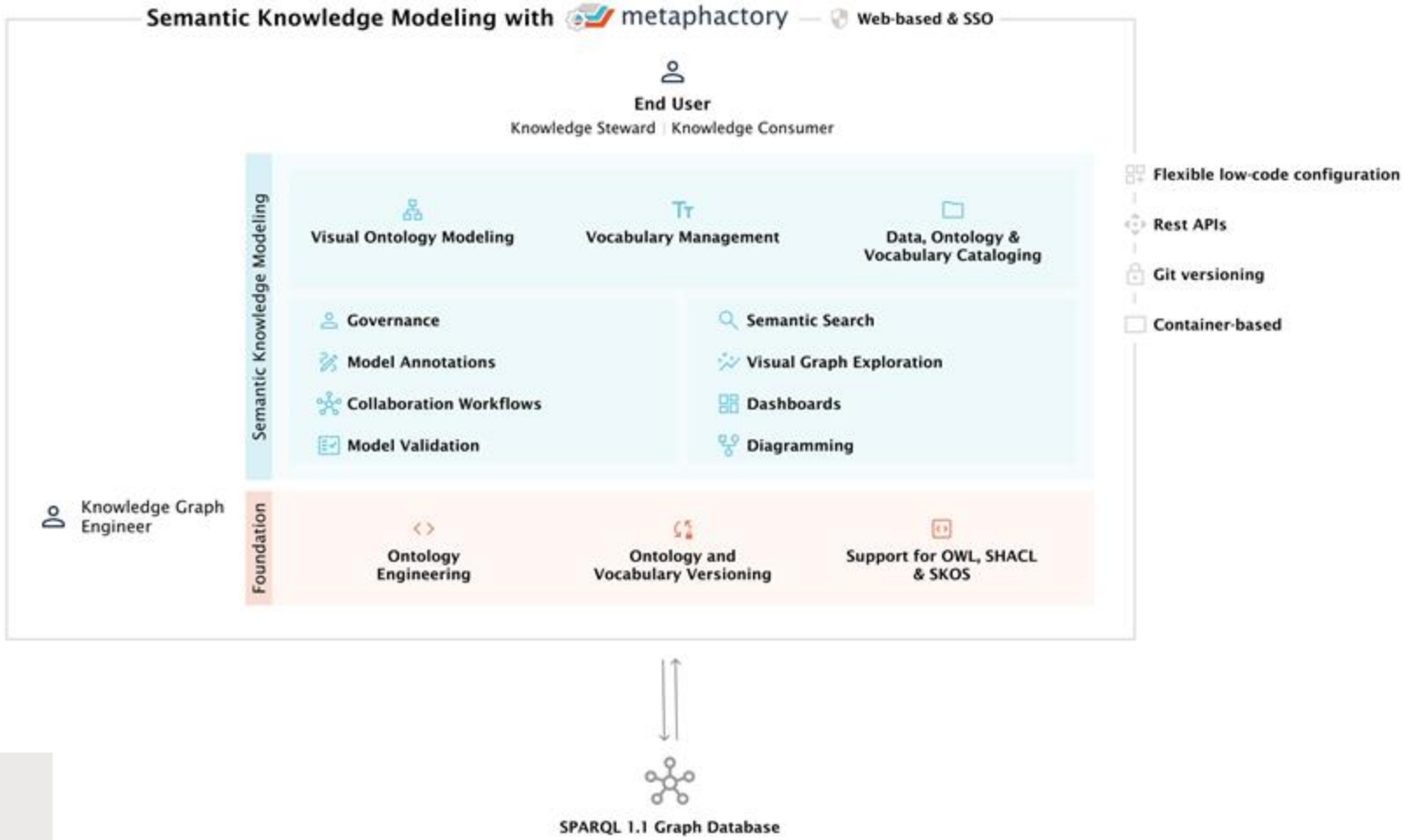
Visual Ontology Modeling

- All stakeholders are empowered to actively participate in the modeling process
- Agile processes for ontology design, implementation, documentation and governance

FAIR Findable Accessible Interoperable Reusable

Semantic Knowledge Modeling

- Capture domain knowledge & expertise in **reusable & extensible semantic models**
- Boost data literacy by enabling **domain experts** & business users to contribute to the modeling process
- Support **expert users** with advanced features for ontology engineering & governance workflows

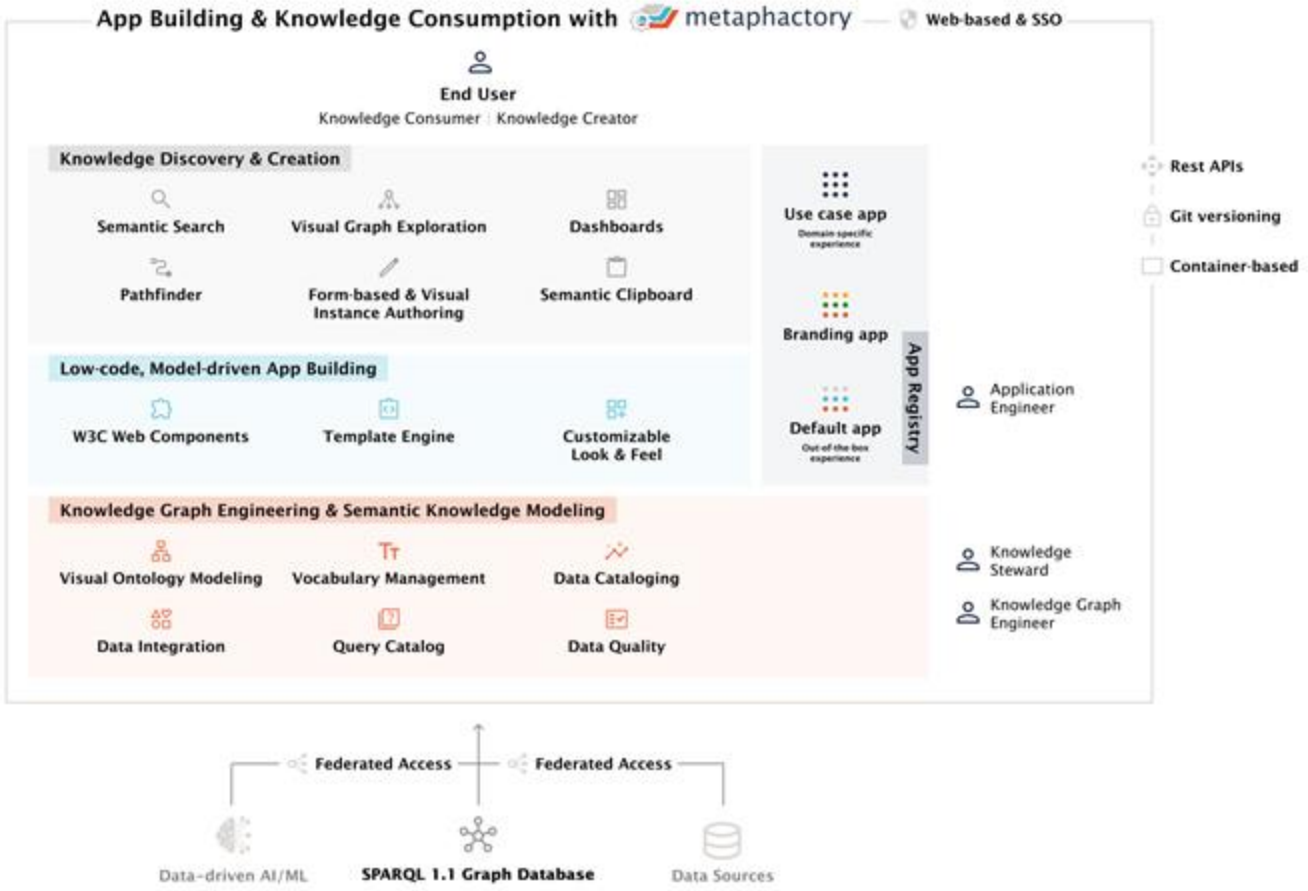


Based on open standards

RDF W3C OWL2 W3C SHACL W3C SKOS W3C SPARQL

Model-driven App Building & Knowledge Consumption

- Build **use-case specific knowledge graph apps** that meet enterprise requirements
- Drive knowledge democratization by empowering **end users** to discover, consume & create knowledge & insigts in a self-service manner
- Support **application engineers** with a model-driven, low-code approach for app building



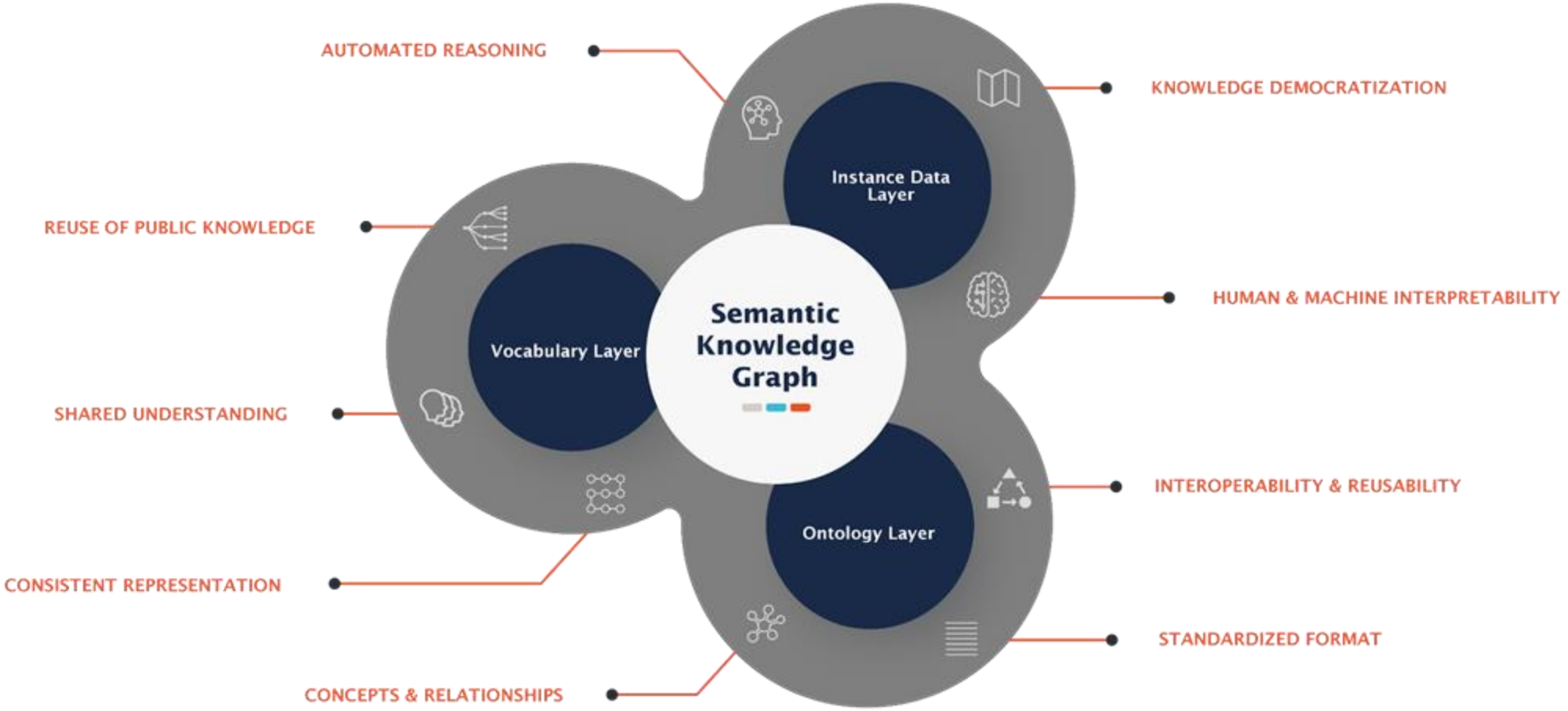
Based on open standards

RDF W3C OWL2 W3C SHACL W3C SKOS W3C SPARQL

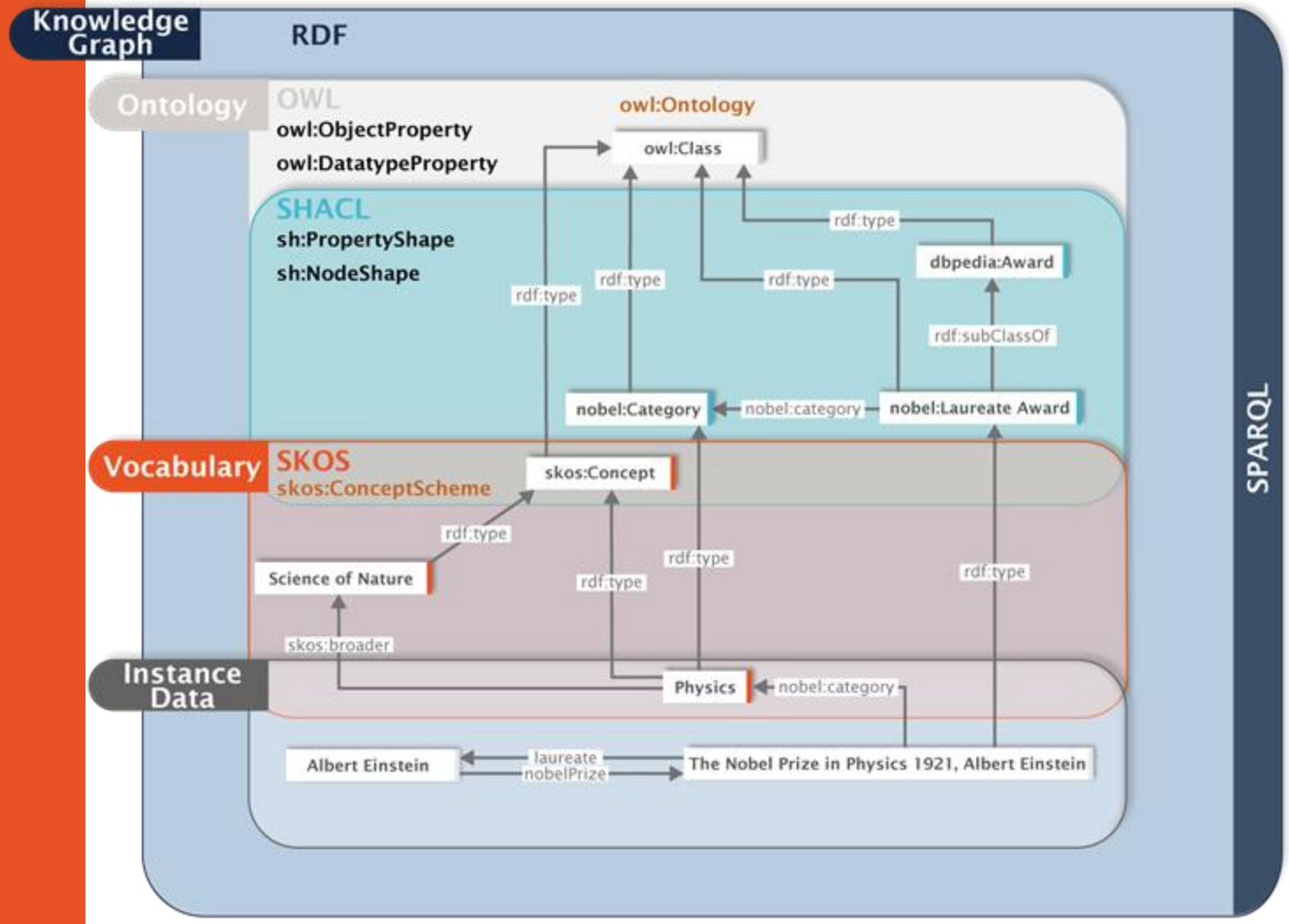
metaphacts

The three layers of a semantic knowledge graph

FAIR: Findable, Accessible, Interoperable, Reusable



Connecting ontologies, vocabularies & instance data



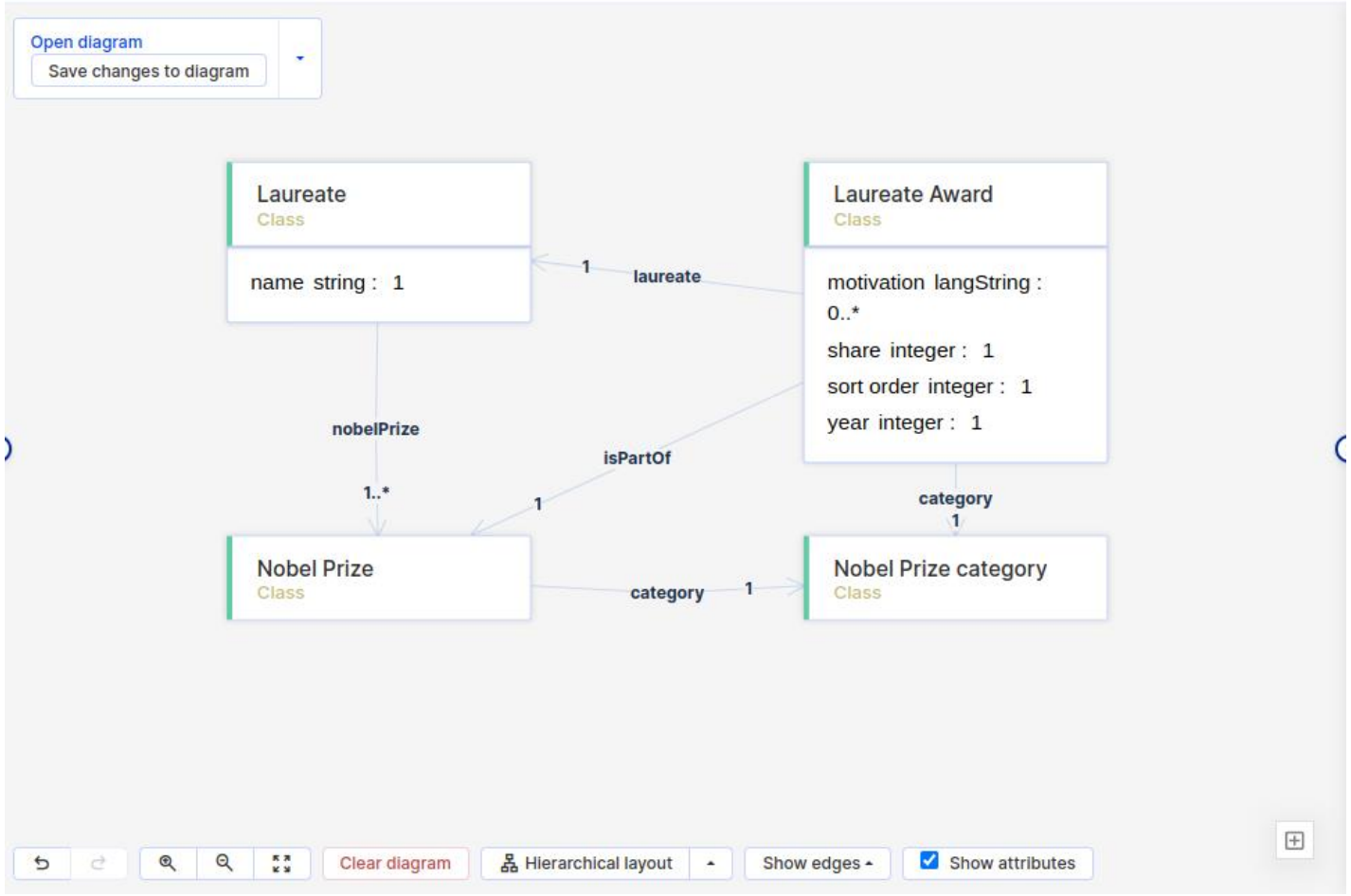
metaphactory knowledge graph approach - Layering of open W3C semantic knowledge graph standards as utilized & applied by metaphactory

Benefits

- Interlink ontologies & vocabularies to support reuse while separating management & governance tasks
- Improve stakeholder communication, asset documentation & governance
- Enable model-driven applications with e.g., auto-suggestions in semantic forms, runtime validation of user interaction, hierarchical facets in search, etc.
- Ensure data quality by running checks & validations against business logic

Visual Notation for Ontology Modeling

- Classes
  - represent the abstract or concrete categories, types, or groups of entities within the domain of discourse.
- Attributes
  - are specific properties or characteristics assigned to an instance (individual) of a class.
- Relations
  - define how entities or classes are related to each other within the ontology.



Behind the scenes: SKOS & OWL+SHACL

- **Web Ontology Language (OWL)** provides more ontological constructs including:
  - distinction between OWL ObjectProperties (resources as values) and OWL DatatypeProperties (literals as values).
  - Able to define a named graph as an ontology, e.g.
    - `<http://data.nobelprize.org/terms/> a owl:Ontology; dcterms:title "Linked Nobelprize Ontology".`

Complexity **hidden** behind visual editing in metaphactory!

- **SHACL** is a language to describing and validating RDF graphs against a set of conditions through shapes:
  - **Node shapes**: constraints about a given focus (target) node.
  - **Property shapes**: constraints about a given property and its values for the focus node.

Documentation: **https://www.w3.org/TR/owl2-overview/**
https://www.w3.org/TR/shacl/

OWL: Example

metaphacts

Class definitions:

```
# prefixes omitted
:Laureate a owl:Class ;
        rdfs:label "Laureate" .

:LaureateAward a owl:Class ;
        rdfs:label "Laureate Award" ;
        rdfs:subClassOf dbpedia:Award .
```

Property definitions:

```
:isPartOf a owl:ObjectProperty;
        rdfs:label "is part of" .

:name a owl:DatatypeProperty;
        rdfs:label "name" .
```

SHACL: Example

metaphacts

SHACL definitions:

```
# prefixes omitted
:LaureateShape a sh:NodeShape ;
sh:targetClass :Laureate ;
```

Node shape

```
sh:property [
        sh:path  :nobelPrize;
        sh:class :NobelPrize;
        sh:minCount 1 ;
] ;
sh:property [
        sh:path :name;
        sh:datatype xsd:string;
        sh:minCount 1 ;
        sh:maxCount 1 ;
 ].
```

Property shapes

Knowledge Graph Assets

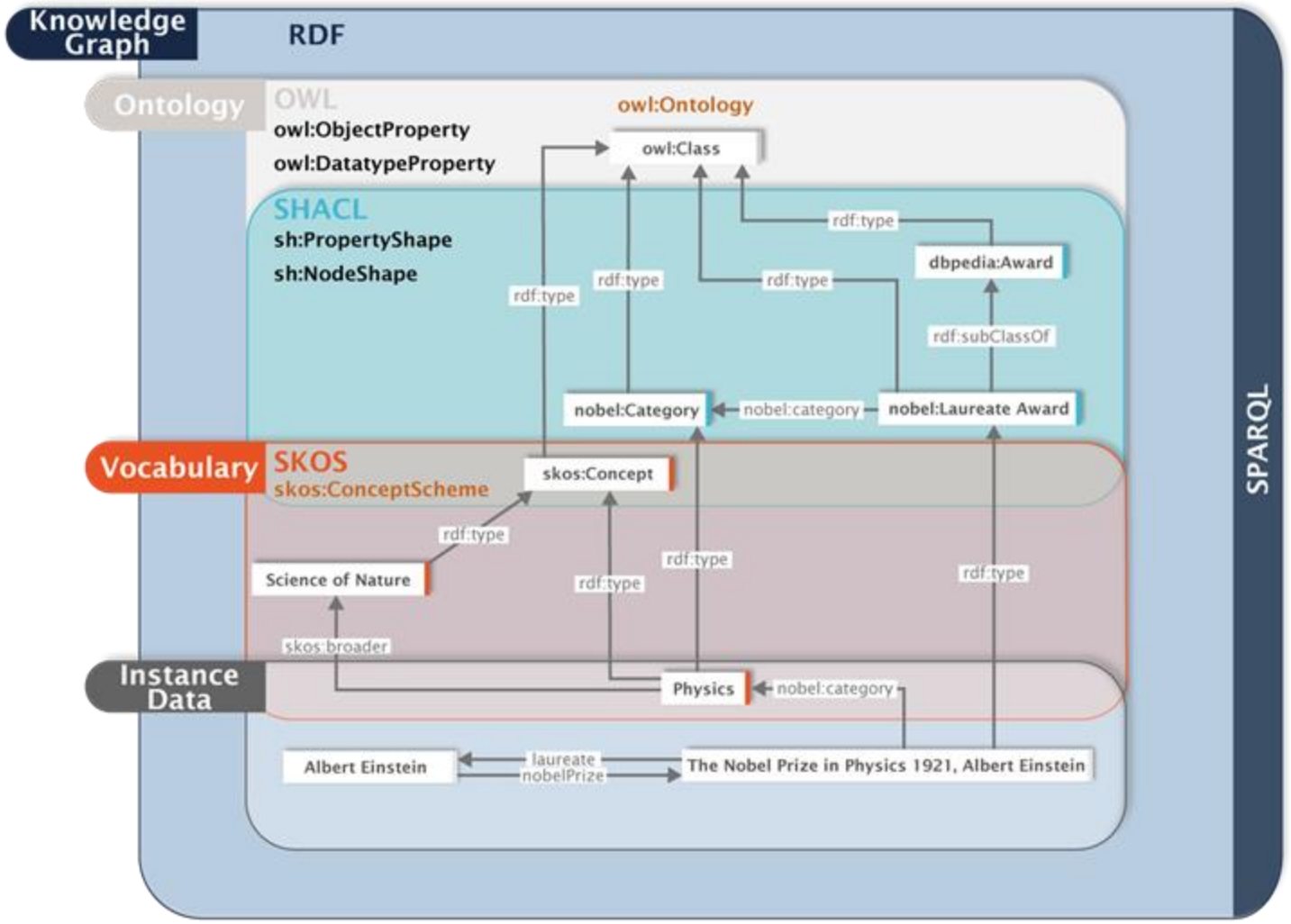
metaphacts

A. Ontologies
- the model of your domain
- the schema/structure of the data
- Modeling language: OWL + SHACL

B. Vocabularies
- Extends structure introduced by ontologies
- Definitions of relevant terms / concepts for a given domain
- Modeling language: SKOS

C. Instance Data
- Data about actual entities in the domain
- Concrete "ground level" information: particular people, specific products, or individual events with their associated attributes and relationships

- *Notes:*
  - Knowledge Graph = collective assets
  - SPARQL is the entry point to the Knowledge Graph as a whole (Read/Write at all levels)
  - Vocabularies considered as Instance Data with special characteristics & function
  - Vocabulary definition is very subjective, use-case dependent
  - Blurred lines / overlap in asset role within the KG (what constitutes the model/structure VS what constitutes data)



Behind the scenes: SKOS & OWL+SHACL

- **Simple Knowledge Organization System (SKOS)** provides vocabulary definitions:
  - Concept schemes, informal hierarchies and association networks

```
nobel:Physics a skos:Concept .
nobel:Chemistry a skos:Concept .
```

- **SKOS** allows for the definition of term hierarchies (**skos:hasTopConcept**, **skos:broader**, and **skos:narrower**) e.g.

```
nobel:vocabulary a skos:ConceptScheme ;
   skos:hasTopConcept nobel:Categories.
nobel:Physics a skos:Concept ;
   skos:inScheme nobel:Categories ;
   skos:broader nobel:Science-of-Nature .
```

Reference: **http://www.w3.org/TR/skos-reference/**

Modeling Guidelines

metaphacts

Contents

https://metaphacts.com/images/PDFs/metaphacts_Semantic_Modeling_Guidelines_1.0.pdf

Modeling Guidelines

metaphacts

Class Naming Tips

- A class name is usually singular (e.g., **Person** not People).
- Class names are usually capitalized (e.g., **Organization**).
- Classes generally use nouns or noun phrases (e.g., **Organization** or **Software Business**).
- For classes named with noun phrases, you typically use title case (e.g., **Software Business**).

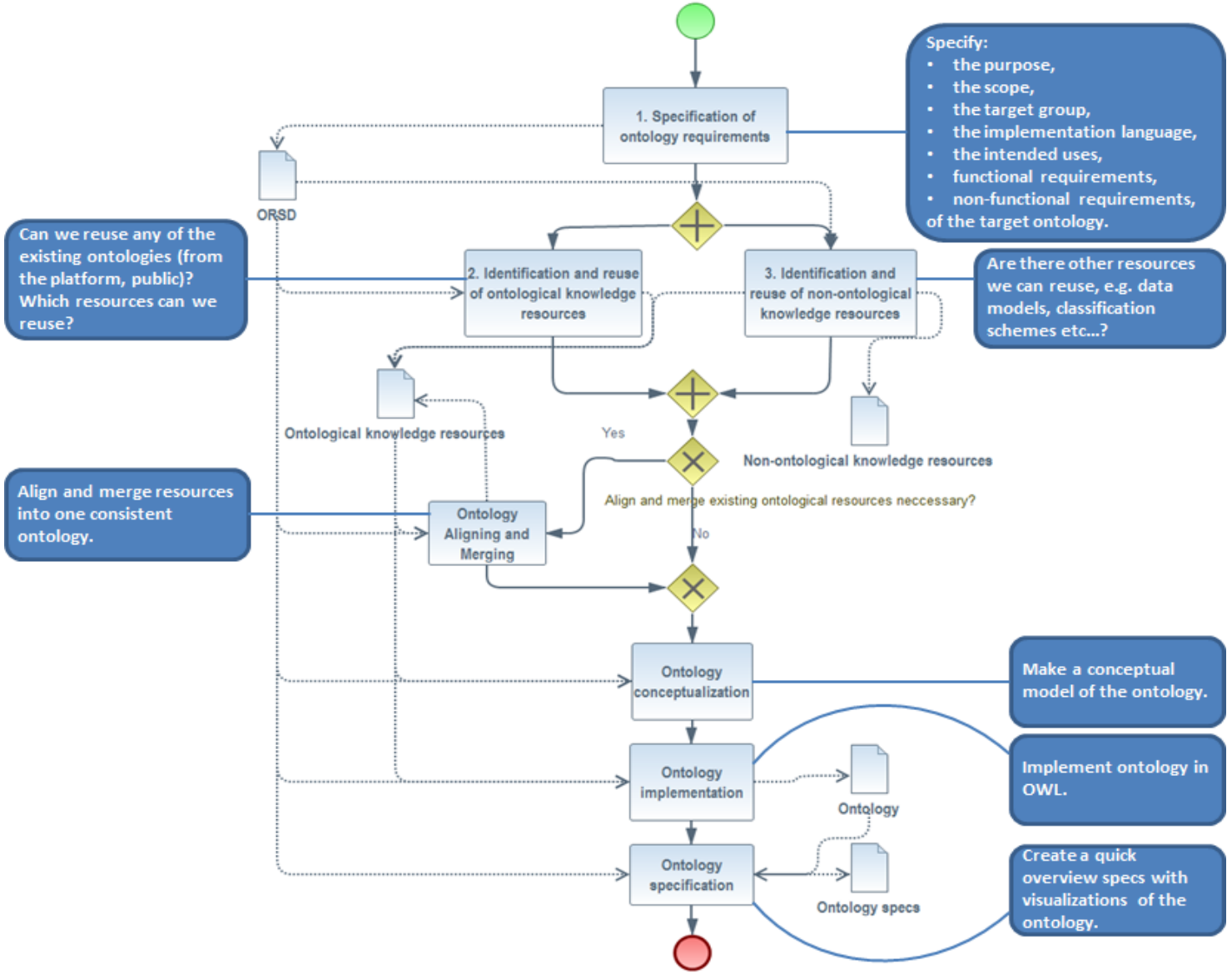
Attribute Naming Tips

- Attributes are typically named after the internal properties of a class (e.g., `length`, `birth date`, `color`, etc.).
- Attribute names are usually lowercase.
- Grammatically, attributes are usually either:
  - Nouns (e.g., `length`) - most common
  - Verbs/verb phrases (e.g., `has length of`) - less common
- When a verb is used for an attribute name, it should be present tense (e.g., `has`, not had).
- Multi-word attributes should use spaces to enable human-readability, although some prefer camel-case.
- No matter which style you choose, you should always be consistent!

Relation Naming Tips

- Relations use mostly the same naming conventions as attributes.
  - The exception is that grammatically, you usually want verbs and verb phrases for relations rather than nouns (e.g., "A **Person** `knows` a **Person**", "A **Person** `lives in` a **House**", etc.).
- A relation should derive its name from the intended connection between the two classes and indicate the semantics.
  - This allows us to distinguish between different connections between the same classes (e.g. "A **Person** `eats` **Food**" vs "A **Person** `loves` **Food**").

Ontology Development Methodology – Inspired by NeOn



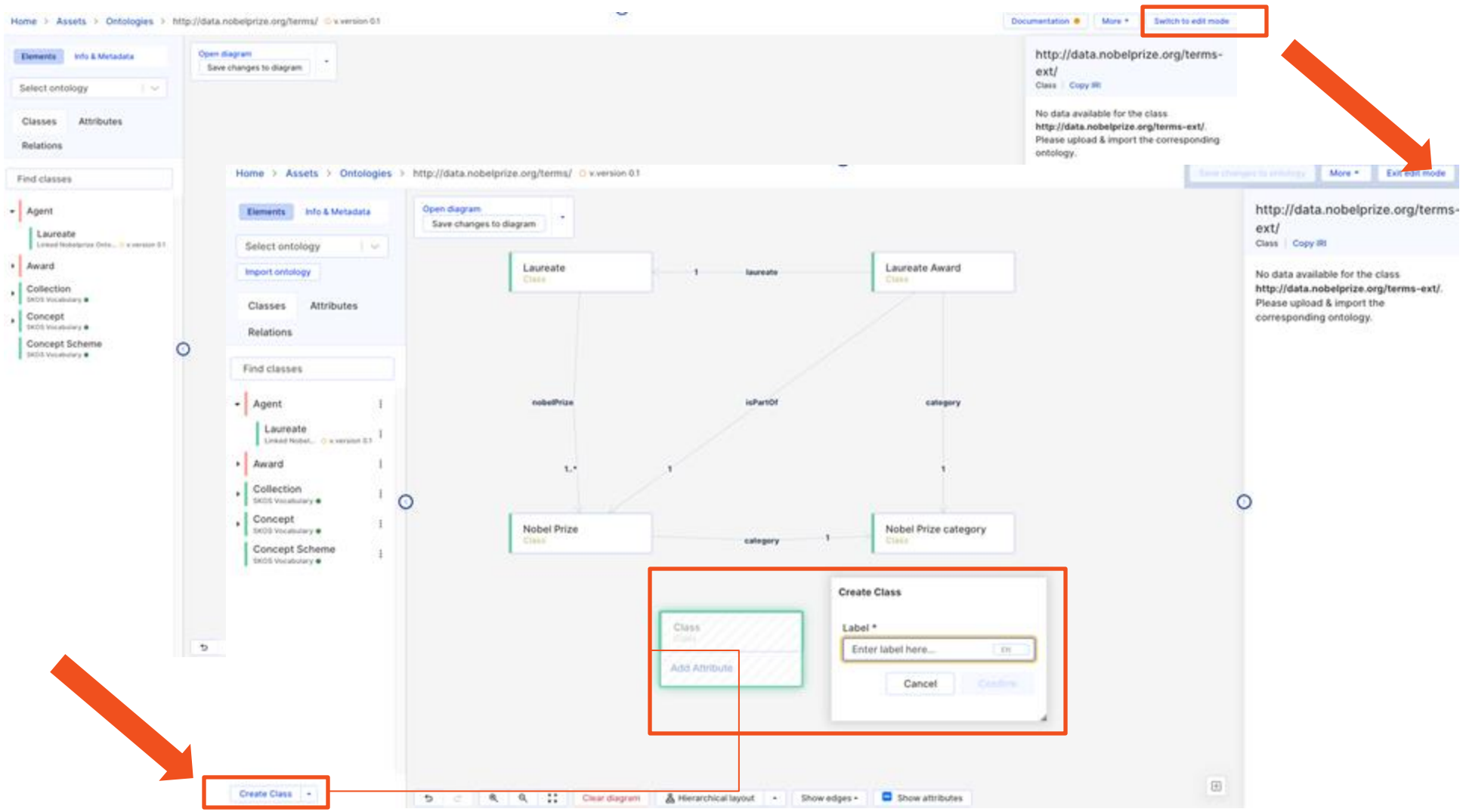
metaphacts

Smart Data. Smart Apps. Smart Decisions.

Visual ontology modeling

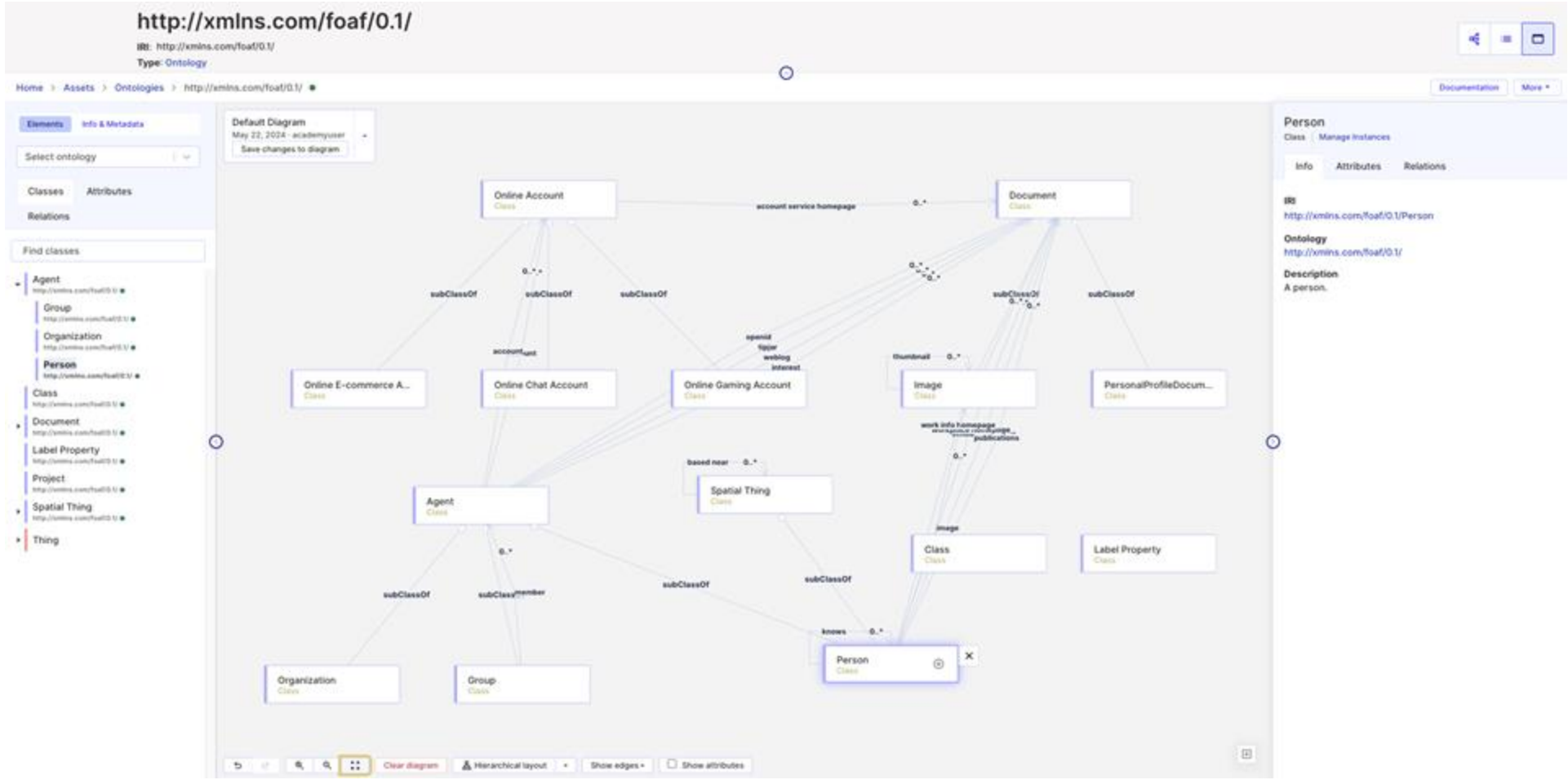
- Explore catalog
- Create a new ontology
- Create classes/relations/attributes
- Reuse
- Vocabulary restriction

Toggling Edit Mode

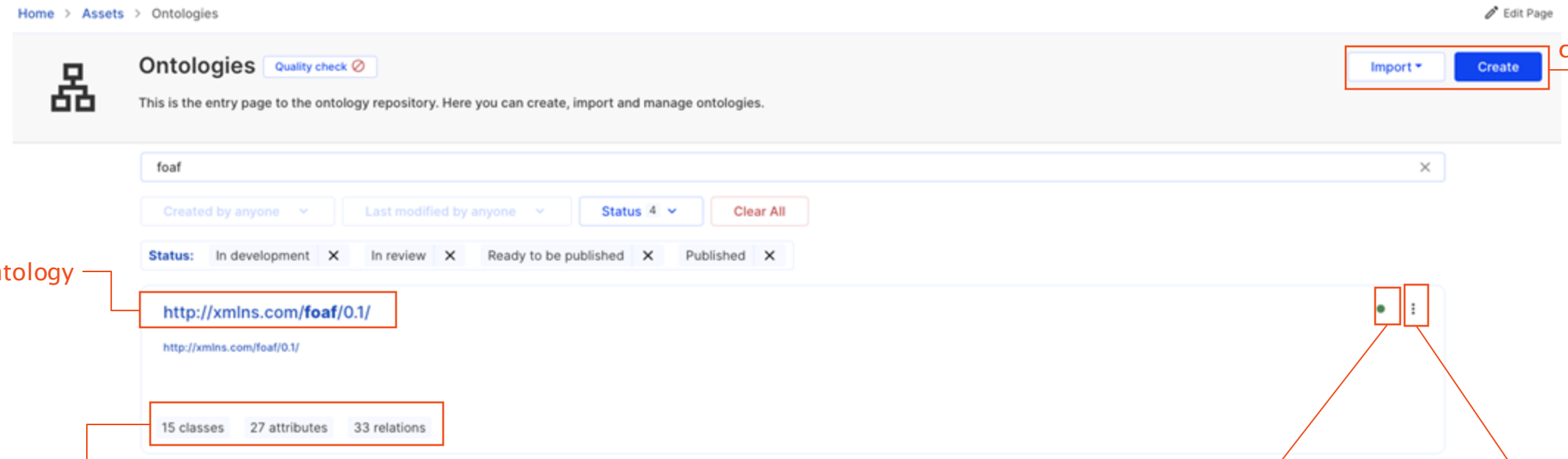


Visual Vocabulary & Ontology Management

- Visual diagrams as default means to interact with vocabularies & ontologies
  - exploration & documentation
  - modification



Vocabulary & Ontology Management: Basics



create or import

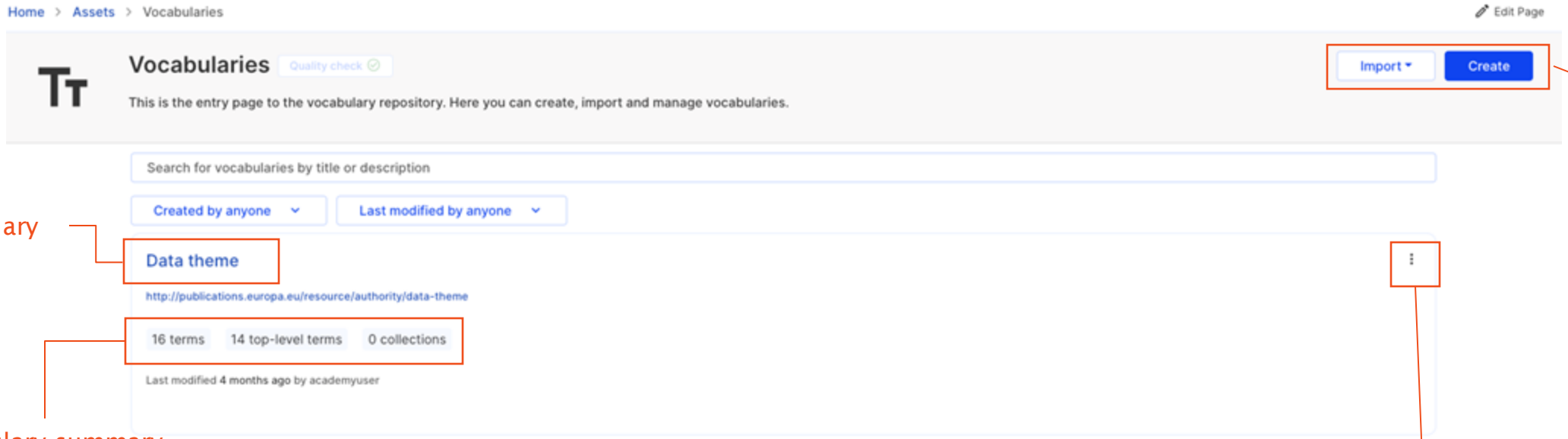
view ontology

ontology summary

status

Create version, Export, delete, copy, RDF serialized, validate database, save to git, set status, Role Assignment

Vocabulary & Ontology Management: Basics



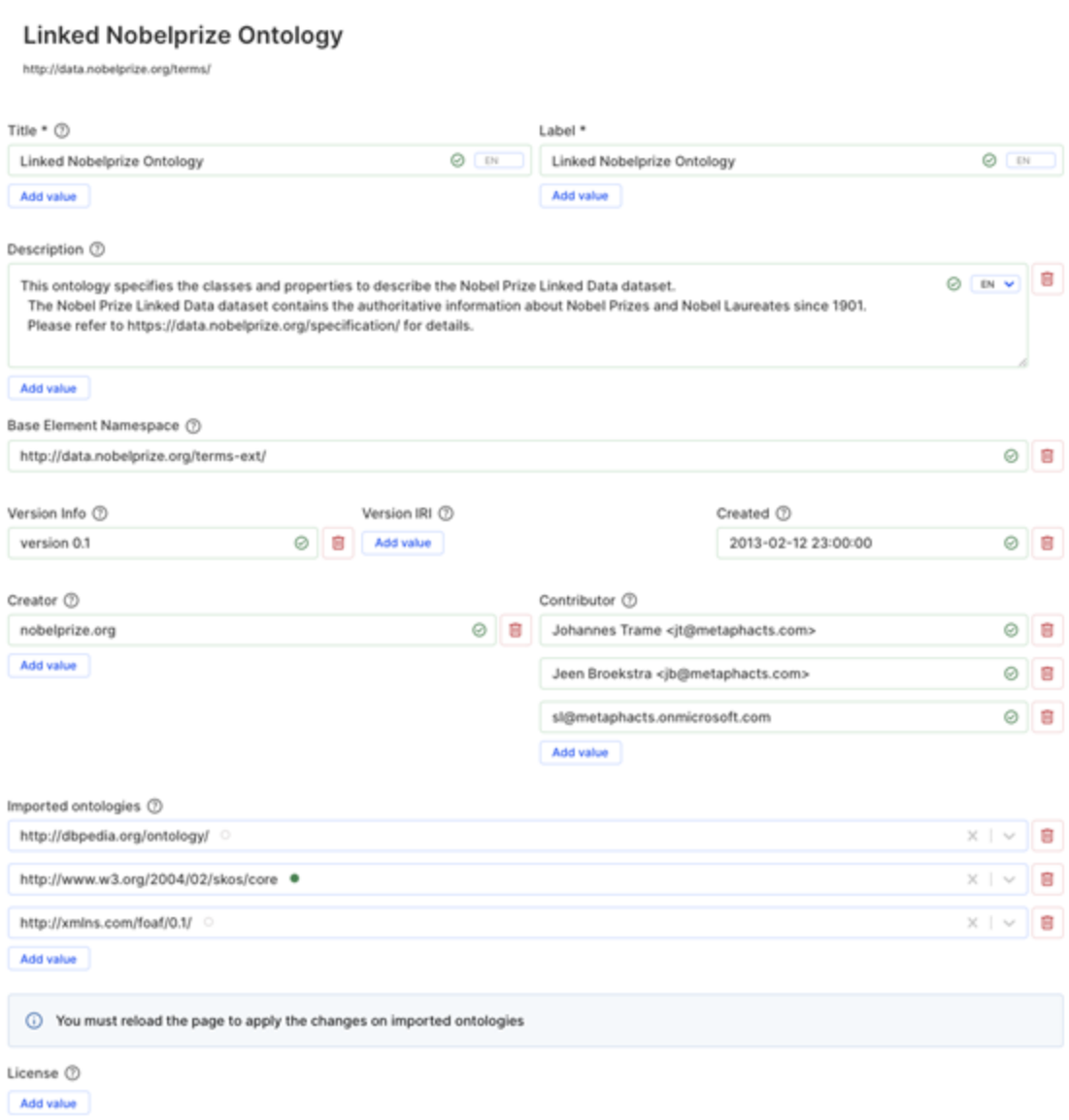
create or import

view vocabulary

vocabulary summary

download, delete,
Save to git,
Role assignment

Versioning Info as Metadata



- Title (dcterms:title)
- Label (rdfs:label)
- Description (dcterms:description)
- Base Element Namespace (IRI) (vaem:namespace)
- Version info (owl:versionInfo)
- Version IRI (owl:versionIRI)
- Created (dcterms:created)
- Creator (dcterms:creator)
- Contributor (dcterms:contributor
- Imported ontologies (owl:imports)
- License (dcterms:license)

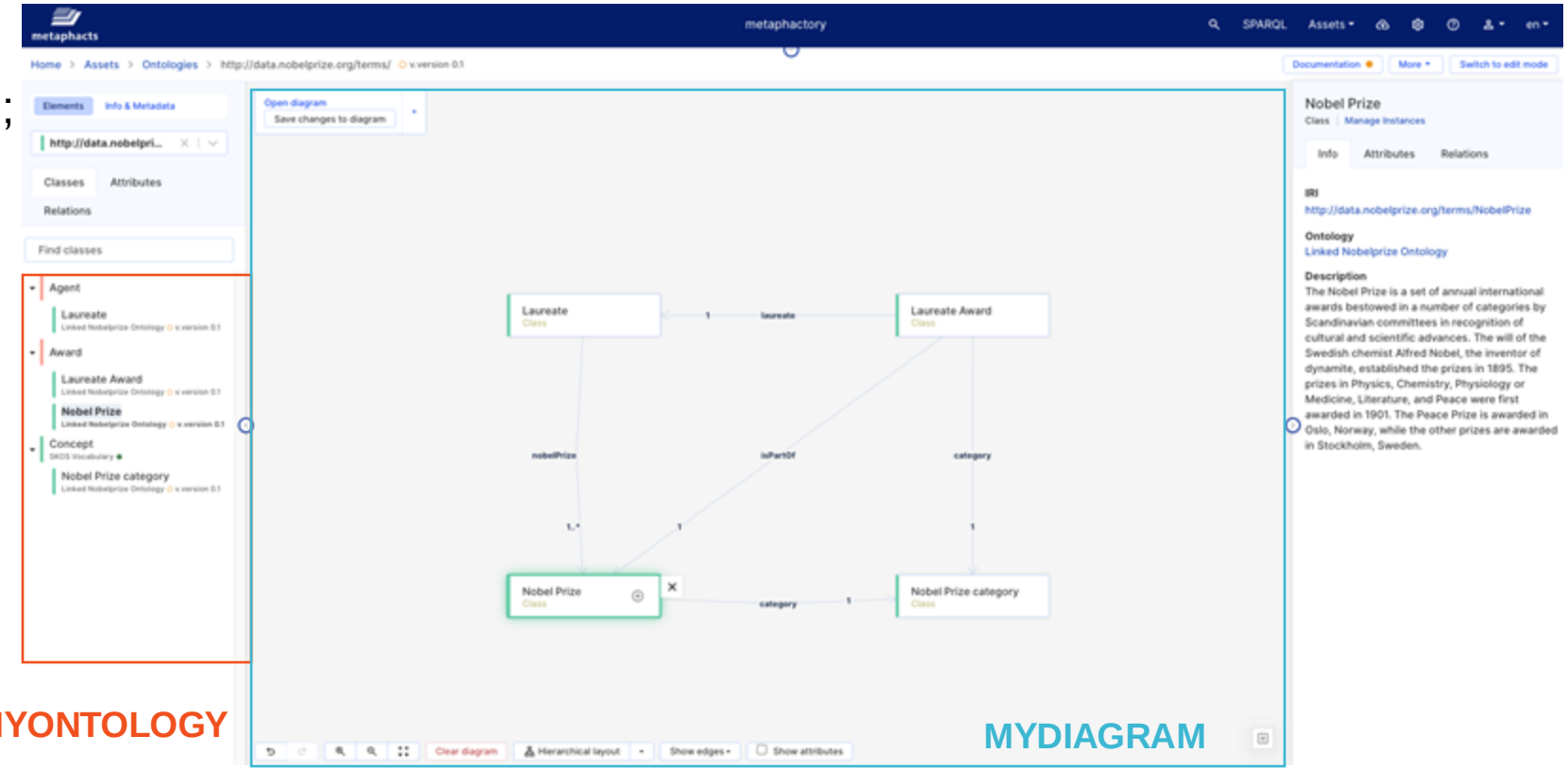
Ontology VS Diagrams: What to keep in mind!

MYDIAGRAM – saved as a Diagram

- Apart from default diagrams; unlimited amount of diagrams can be created as views
- Each ontology can have 1+

MYONTOLOGY – model corresponding to MYDIAGRAM saved as Ontology

- elements inherited from other ontologies NOT included, only references
- Exactly 1

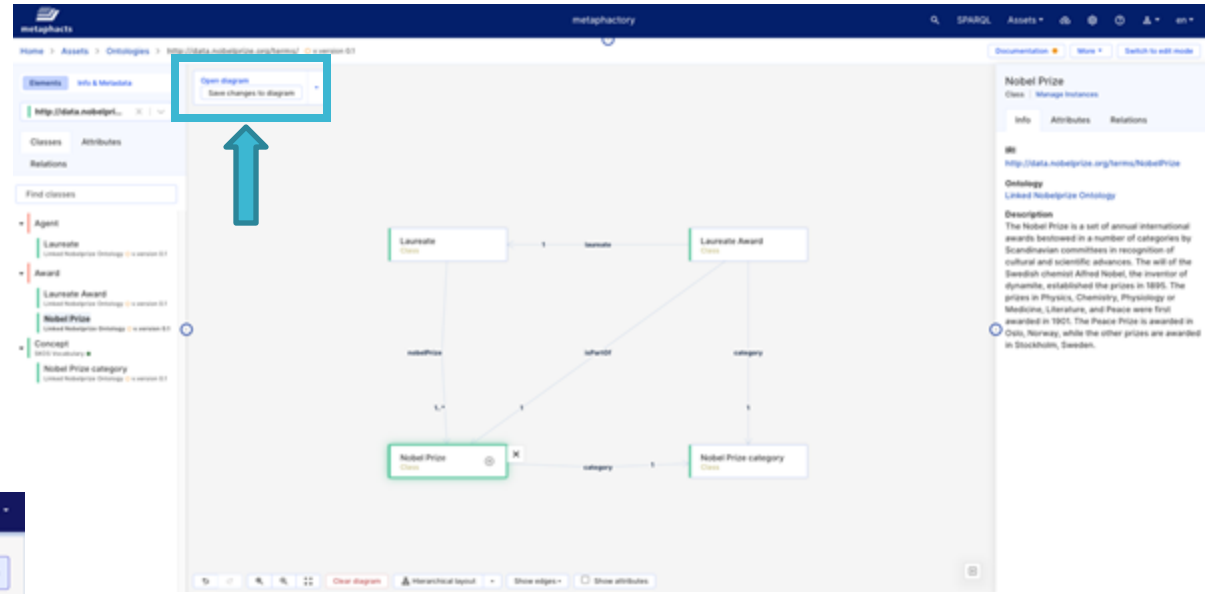


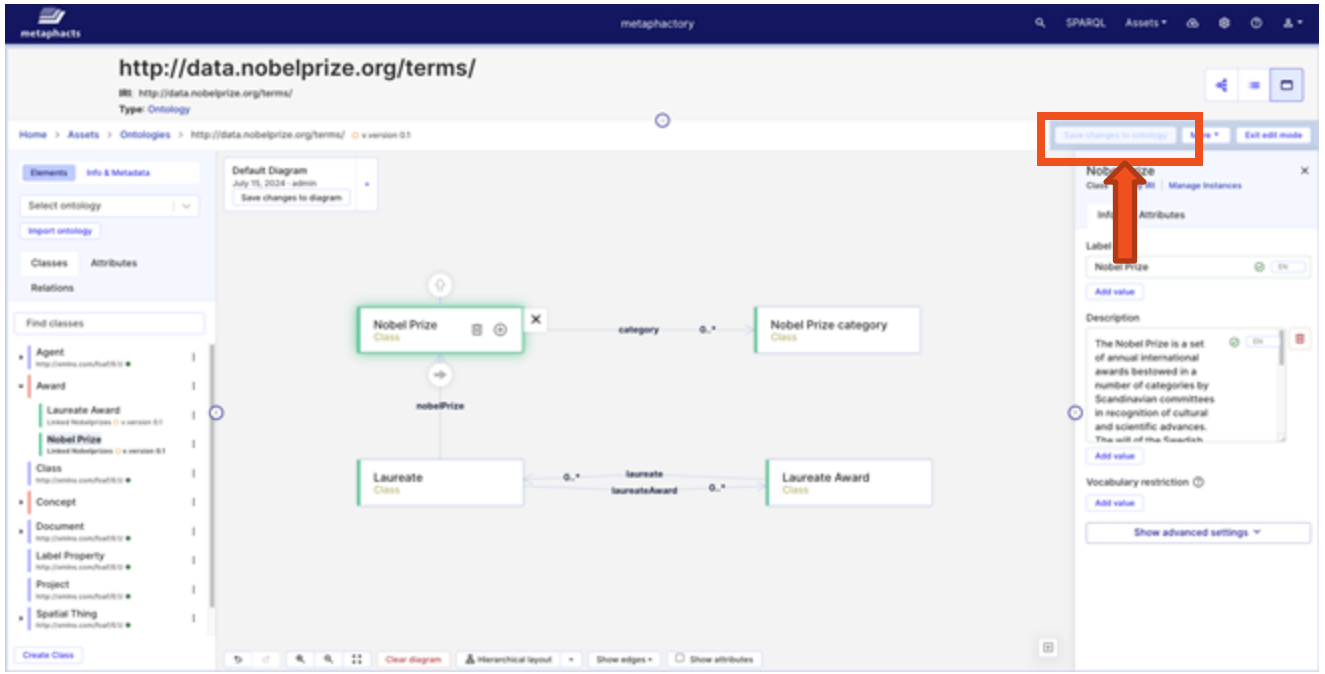
Ontology VS Diagrams: What to keep in mind!

metaphacts

Default Ontology View

- Save changes to Diagram!





ONTOLOGY EDIT MODE

- Saves changes to ontology AND current diagram!

Combining multiple ontologies

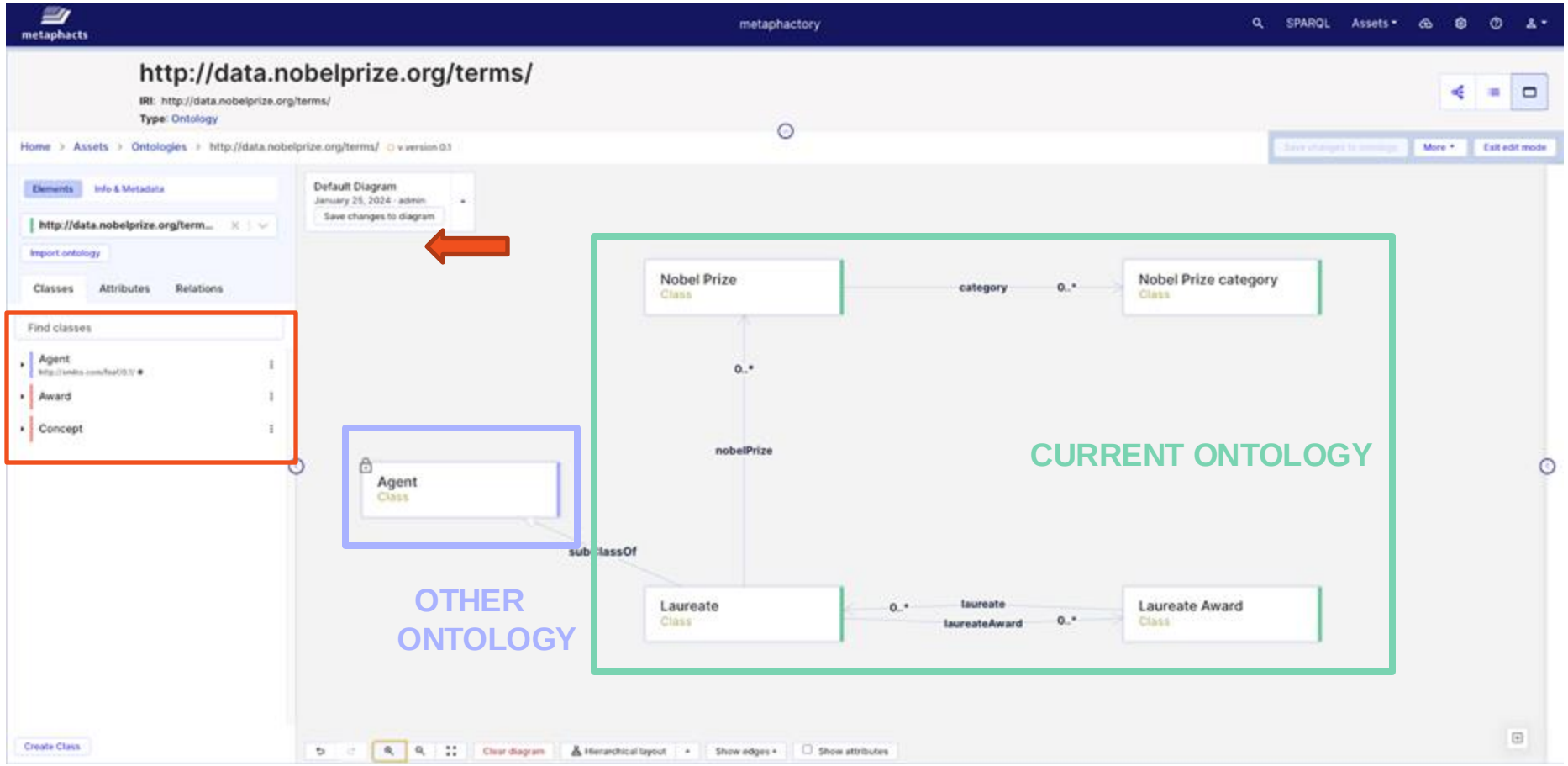
Combination supported and encouraged!

Side bar:
Switch between ontologies without resetting Canvas view

Canvas:
Drag/drop and connect entities from multiple ontologies

Colour scheme:
Same colour = same ontology!

Edit mode:
can only edit entities in current ontology



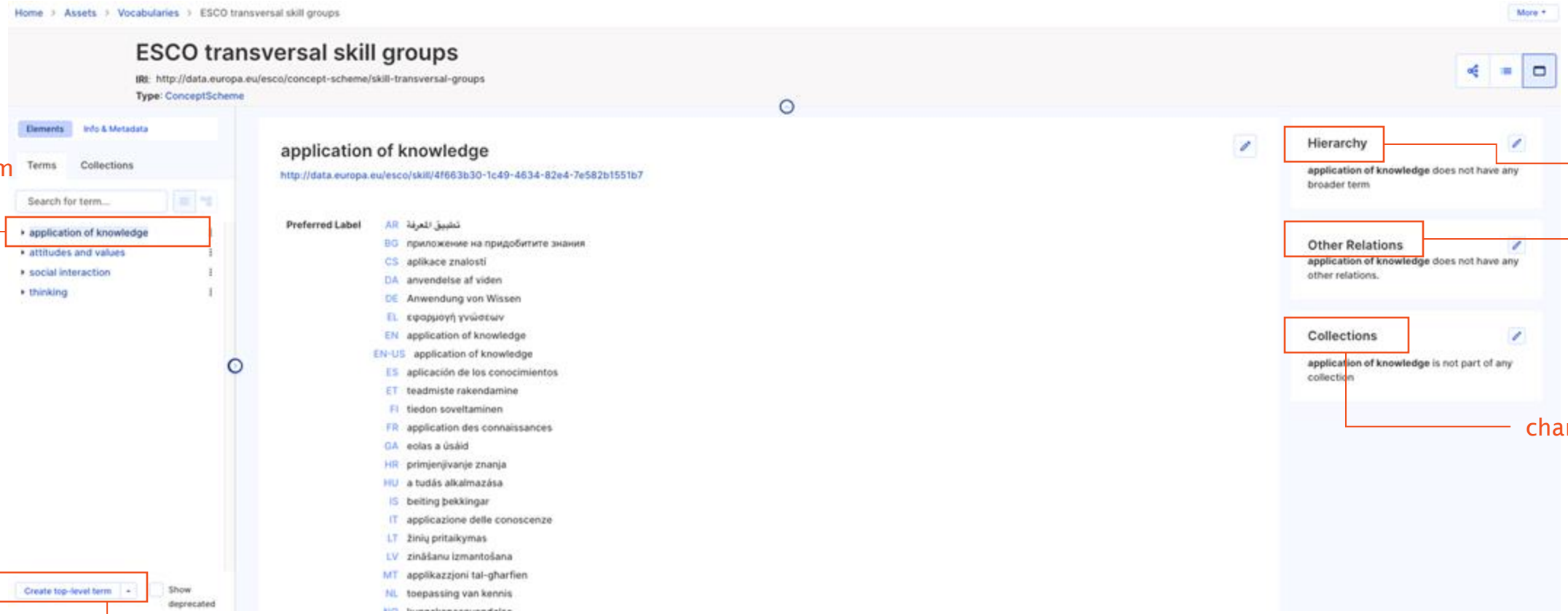
metaphacts

Smart Data. Smart Apps. Smart Decisions.

Vocabulary & taxonomy management

- Explore catalog
- Create a new vocabulary
- Create/edit/import terms
- Create a collection

Vocabulary & taxonomy management



create term in branch

change hierarchy

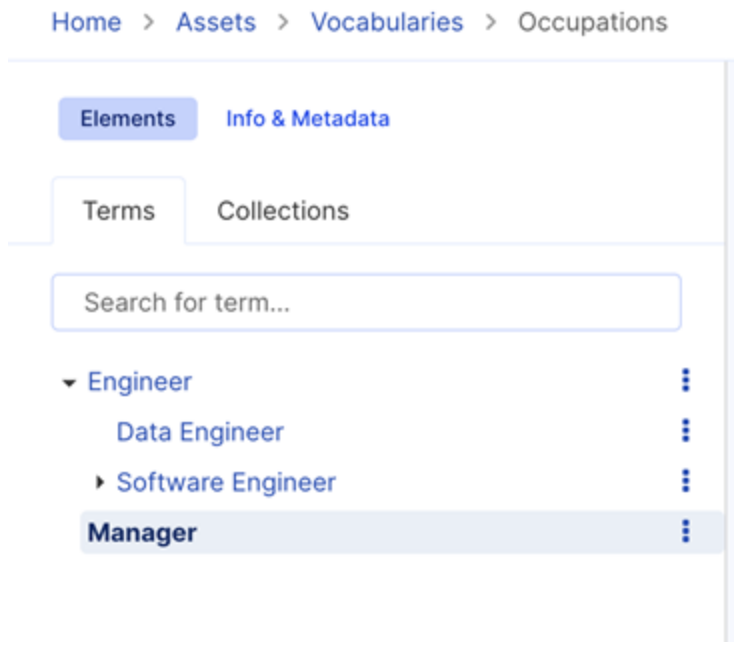
set additiona skos relation

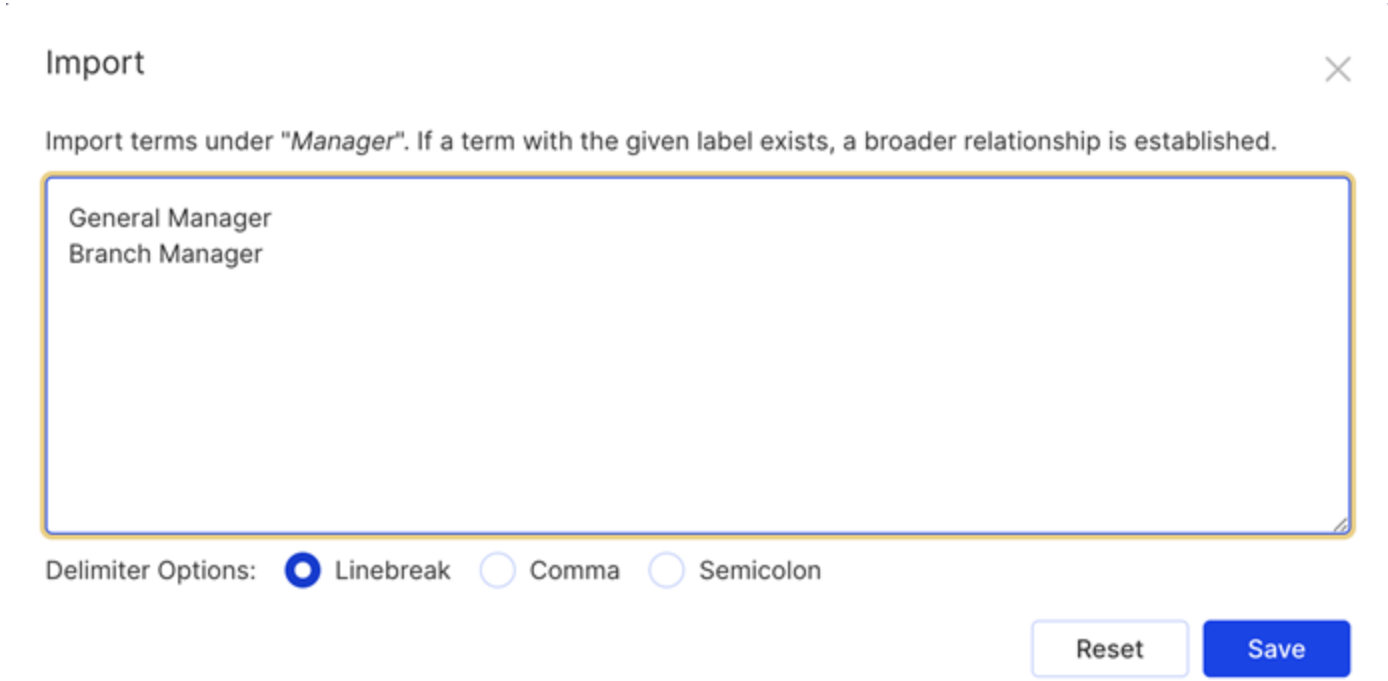
change collection

create a top level term

Vocabulary & taxonomy management

- Create a new ‘Occupations’ Vocabulary
- Create top level terms
- Import narrower terms





metaphacts

Smart Data. Smart Apps. Smart Decisions.

Hands-On: Go to your tutorial instance!

metaphacts

Smart Data. Smart Apps. Smart Decisions.

Coffee Break

metaphacts

Smart Data. Smart Apps. Smart Decisions.

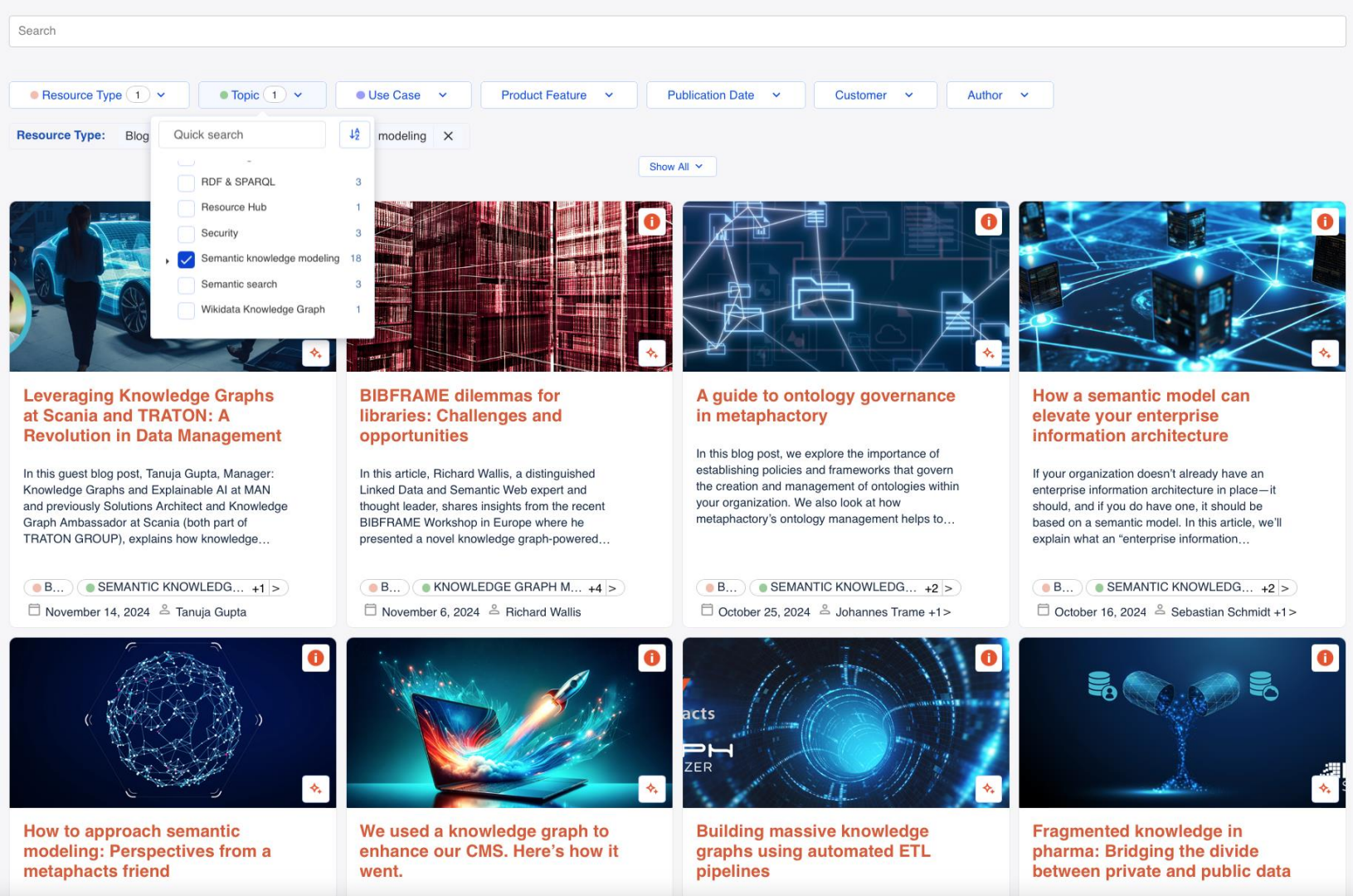
Examples and Experiences from Industry

metaphacts

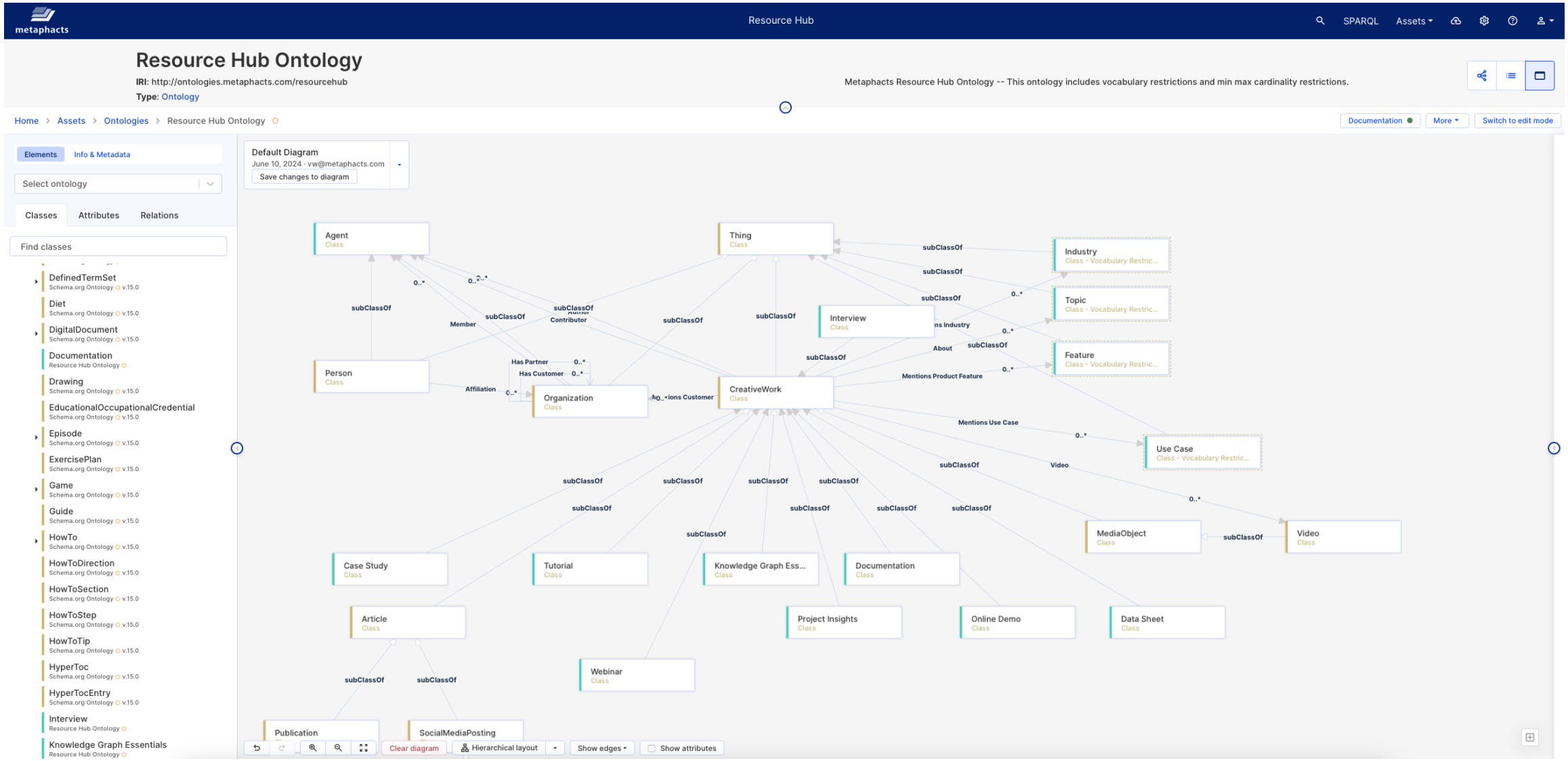
Smart Data. Smart Apps. Smart Decisions.

Resource Hub

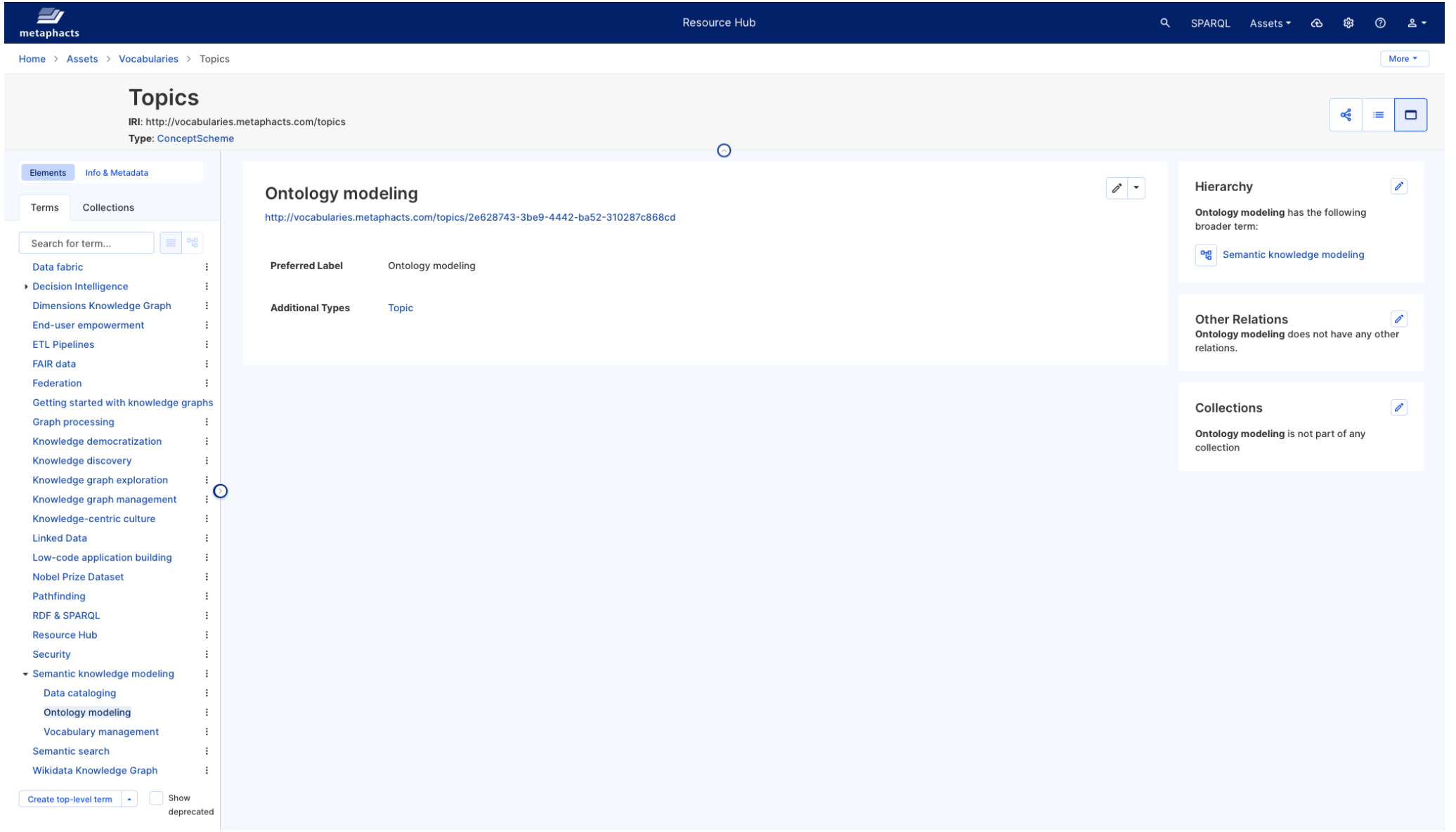
metaphacts Resource Hub



Resource Hub Ontology



Resource Hub Vocabularies



metaphacts

Smart Data. Smart Apps. Smart Decisions.

Enterprise Information Architecture

What is Enterprise Information Architecture (EIA) ?

Critical component for (strategic) data, IT and technology management and planning in support of the business.
(often embedded into Enterprise Architecture – EA)

model, assess, structure, analyze, organize, manage & visualize an organization's data & information assets across various systems, technologies, departments, processes & stakeholders

enable efficient decision-making, information sharing, & process optimization within an organization while ensuring security and privacy

common elements include metadata, taxonomies, content management, data modeling, data governance, & API integration

enable stakeholders within to organization to communicate on data (with a common business language)

Why Semantics in EIA*?

» Connect business and IT (drive data democratization)

» Connect data governance silos (data catalog, GRC, business glossary, process catalog, …)

» Connect company strategy and digital execution

* see also https://www.forrester.com/blogs/enterprise-architecture-**from-design-driven-to-data-driven**/

Semantic layer approach to EIA

Modeling of the data landscape in **two layers**:

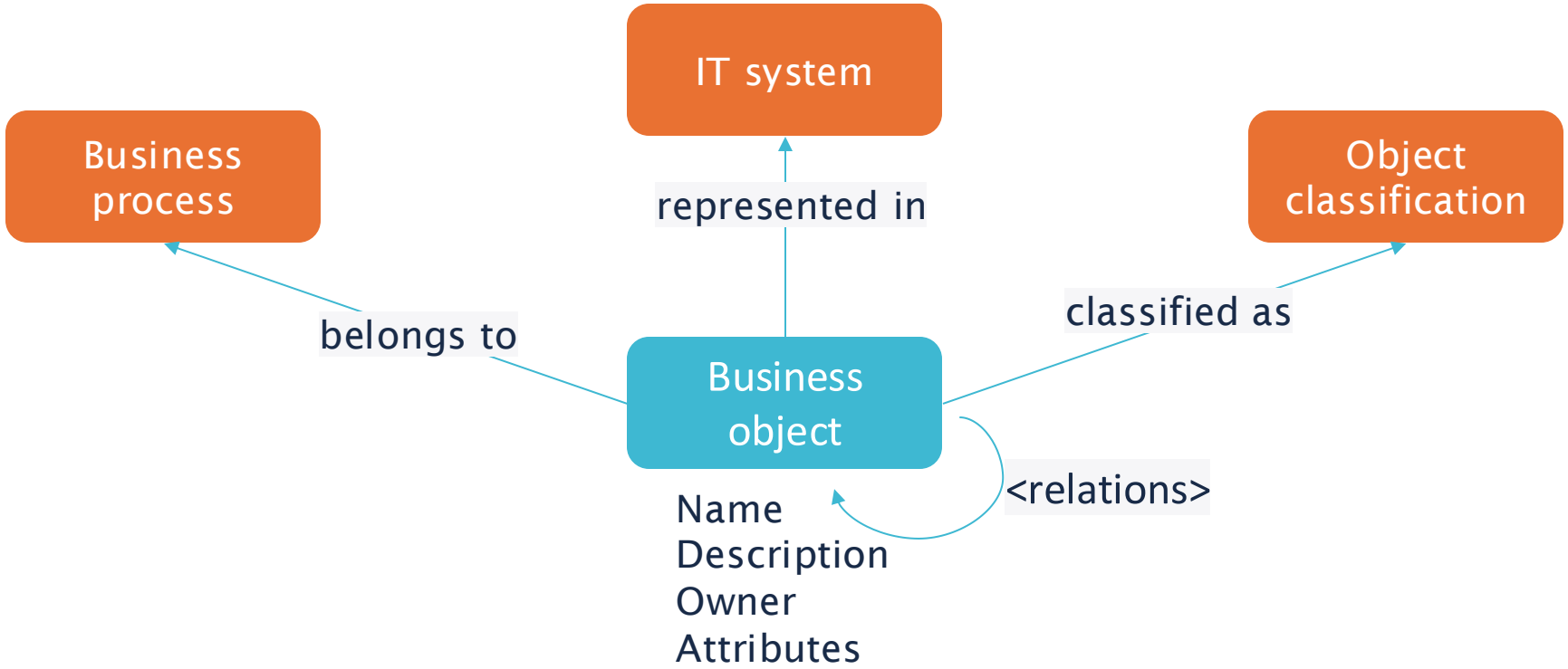
Business objects

- Real or virtual objects of the business world (e.g., sales order, packaging material, plant)
- Input, output or interim result of a business process
- Named by explicit and harmonized terms
- Represented and persisted by data within IT systems

Physical representations

- Physical data models representing the physical structure of data
- Part of the IT landscape
- Cover complete or parts of business objects (in a specific context = IT system, database, data warehouse, data lake, data transport technology)

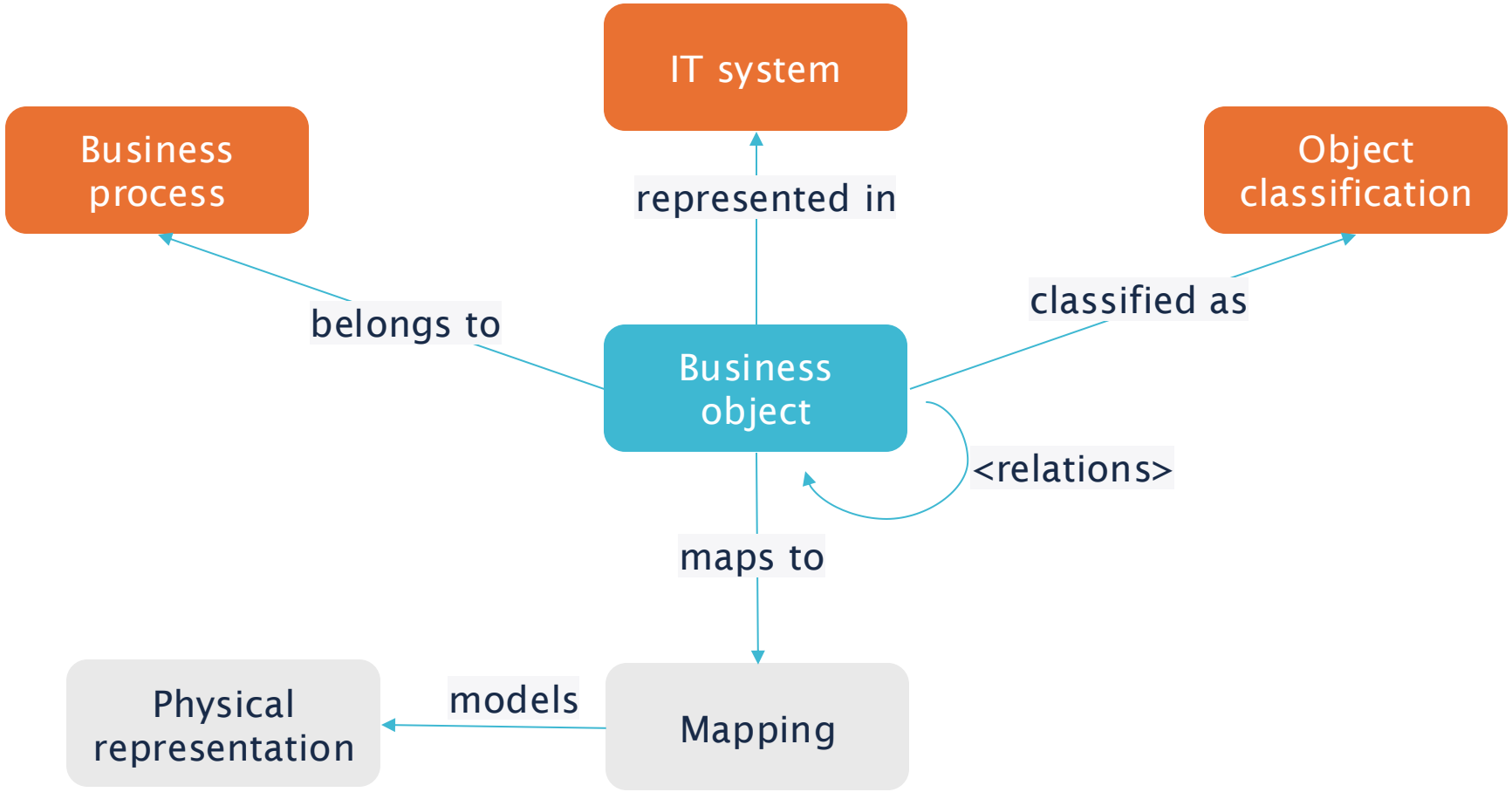
The business object in the semantic layer



Defined terms

- Data Owner
- Data Steward
- Data Engineer
- Information Architect
- Information Architecture Governance
- User

Expanding the semantic layer to physical layer



Defined terms

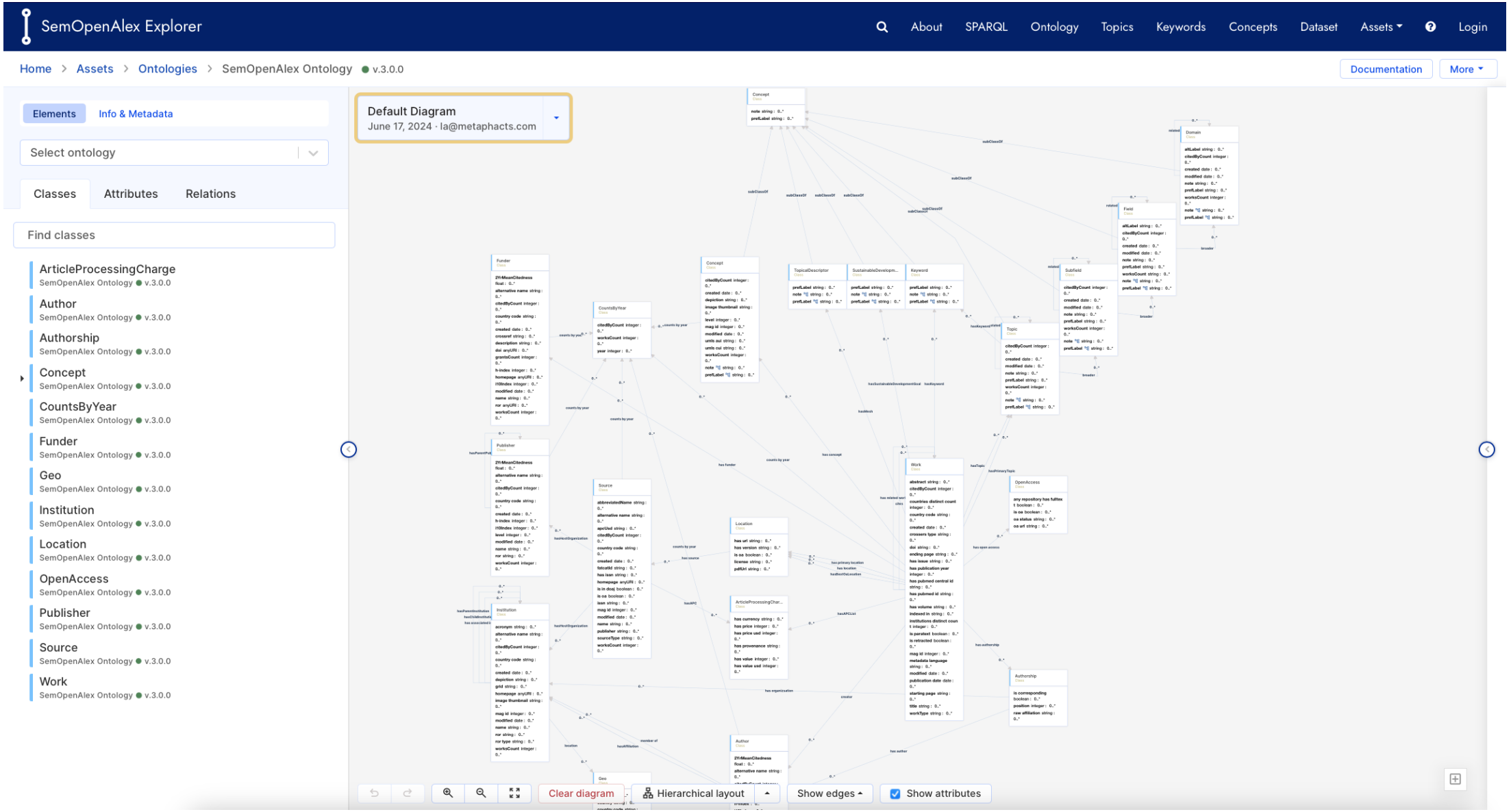
- Data Owner
- Data Steward
- Data Engineer
- Information Architect
- Information Architecture Governance
- User

metaphacts

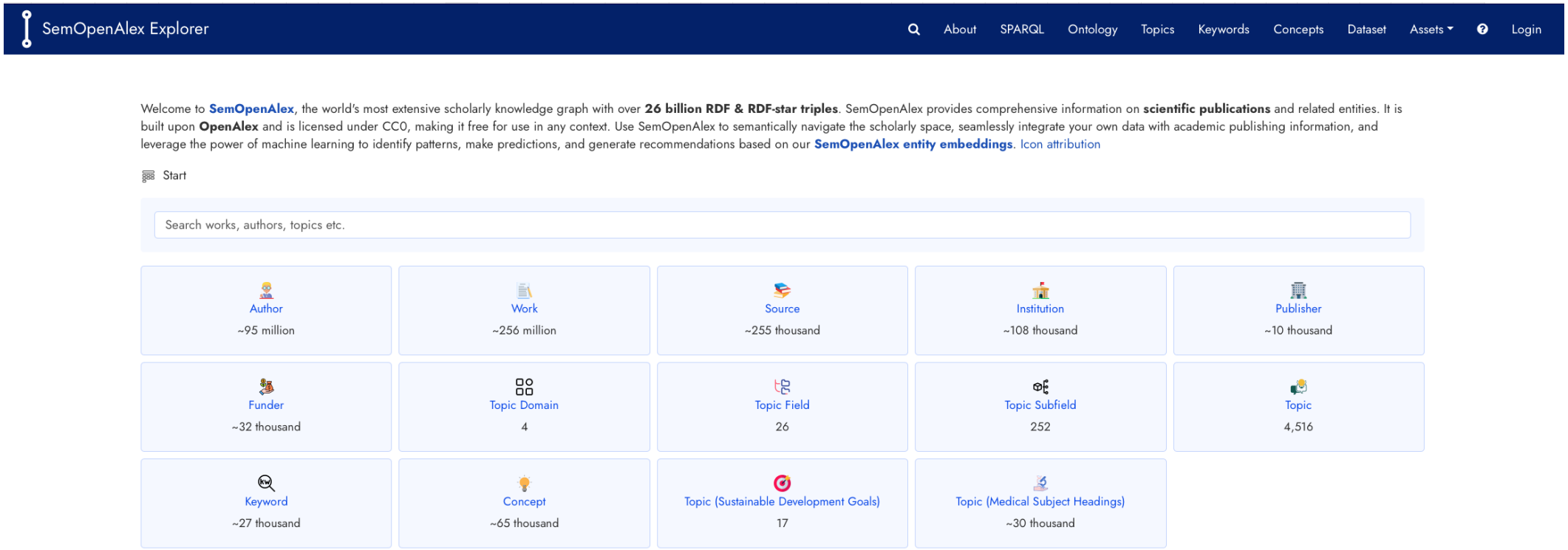
Smart Data. Smart Apps. Smart Decisions.

SemOpenAlex

SemOpenAlex Ontology



SemOpenAlex Explorer

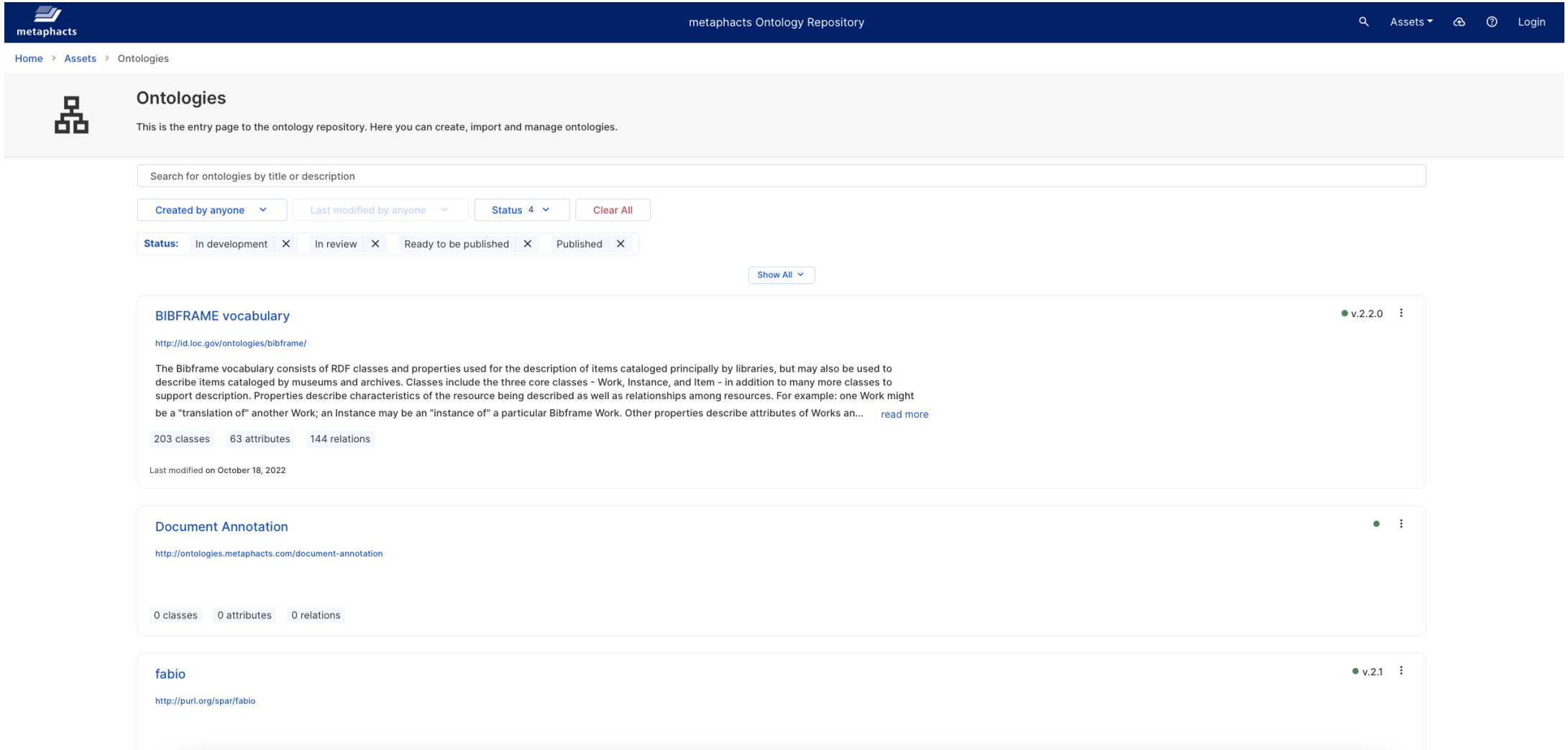


metaphacts

Smart Data. Smart Apps. Smart Decisions.

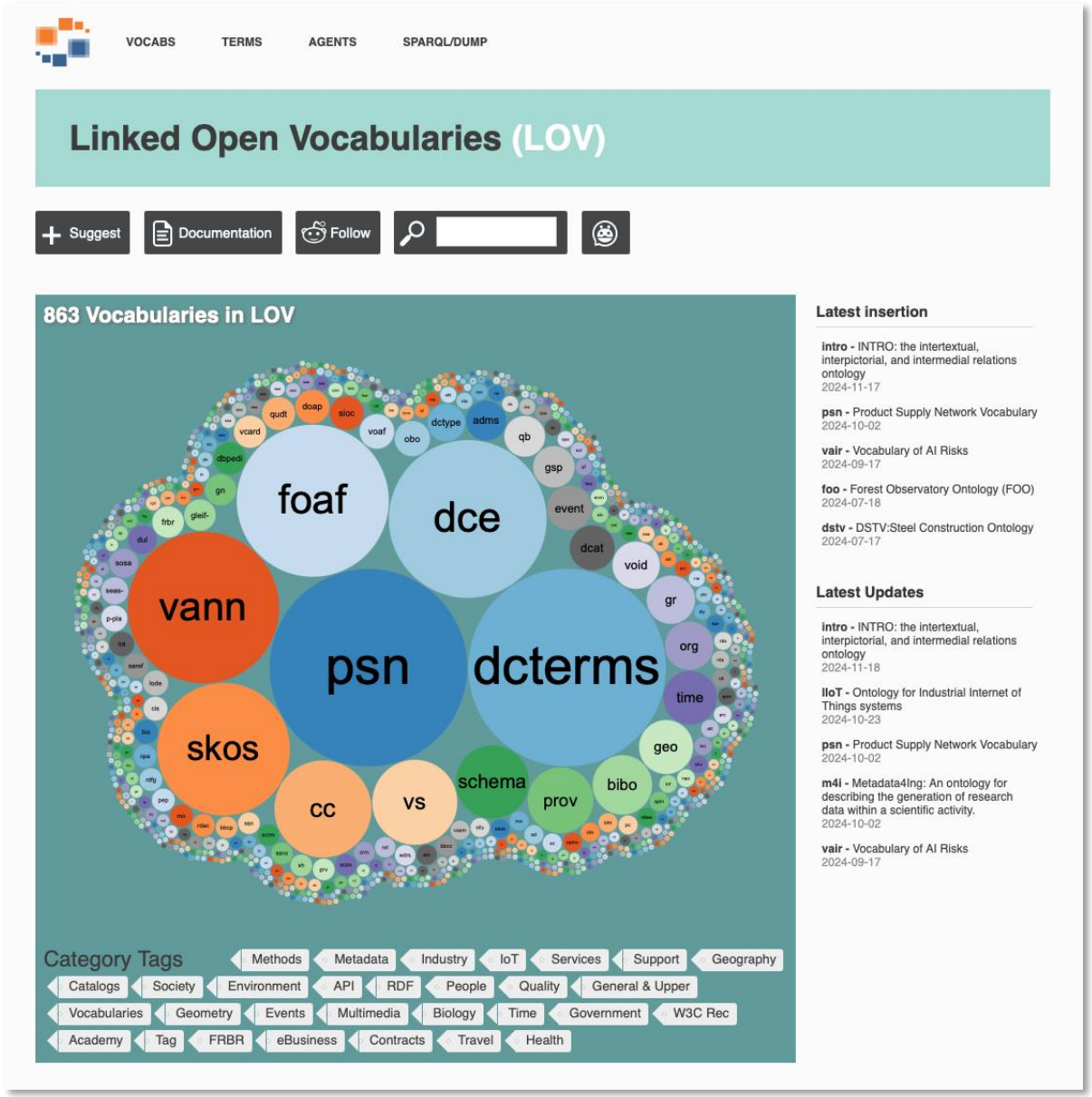
Ontology Repository

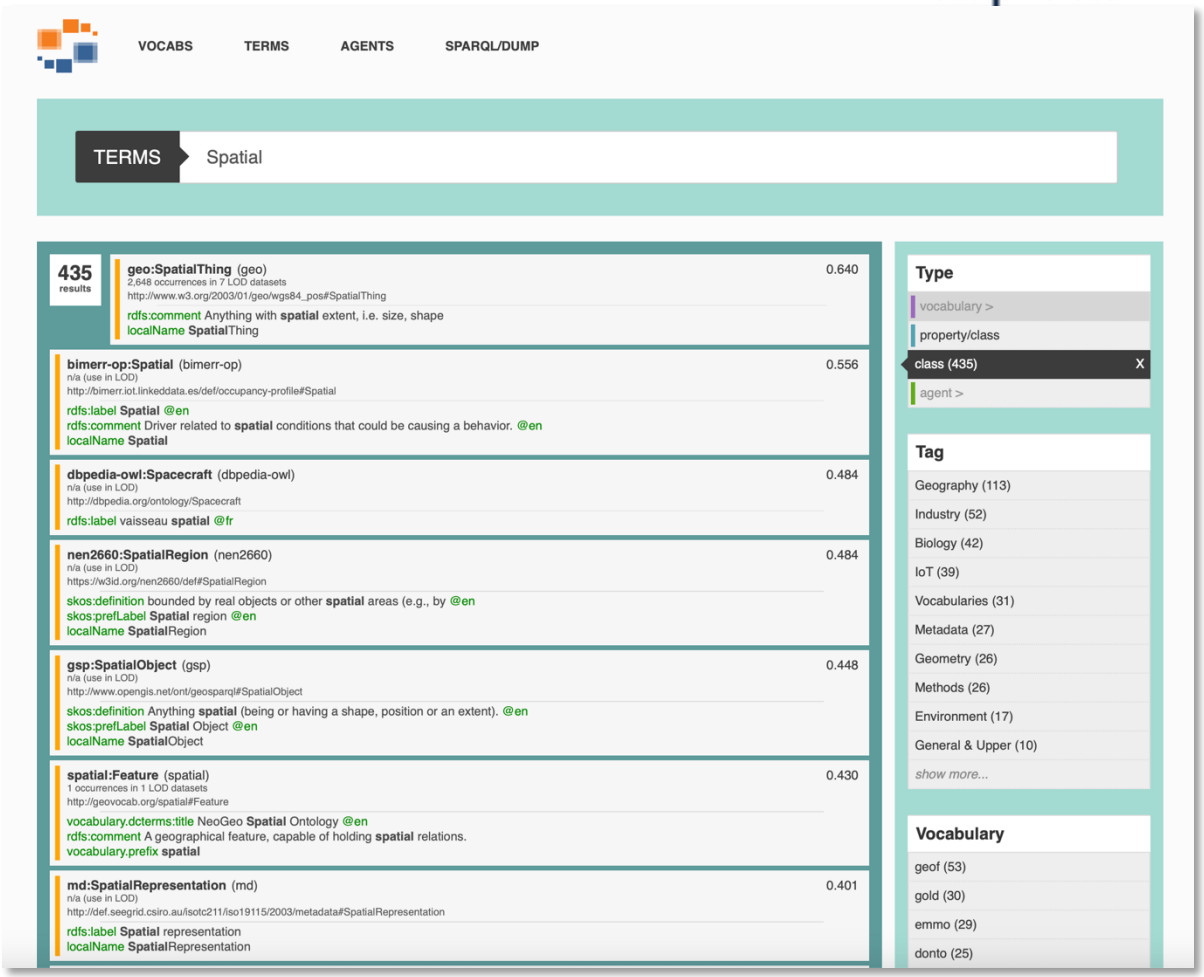
metaphacts Ontology Repository



https://ontologies.metaphacts.com/

LOV Repository





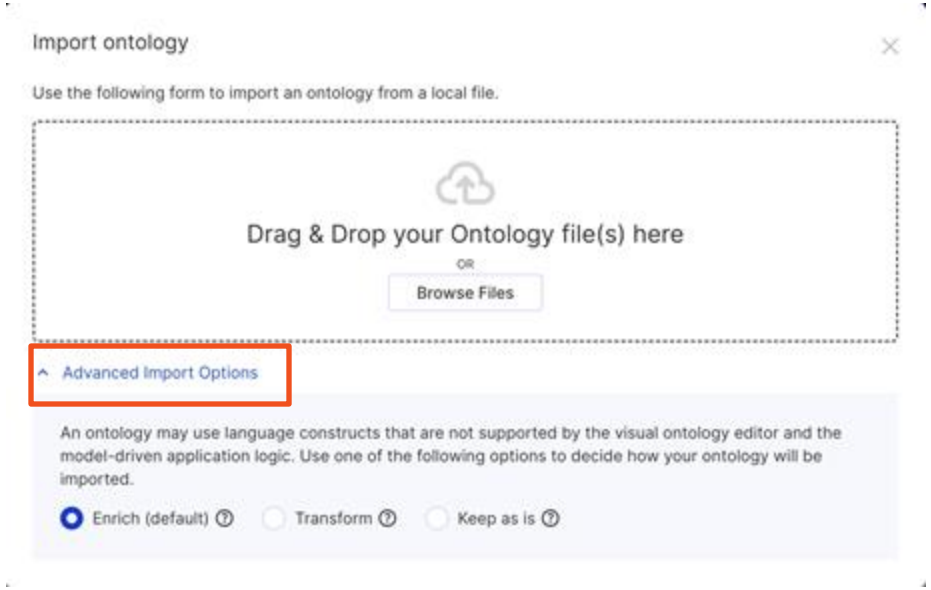
https://lov.linkeddata.es

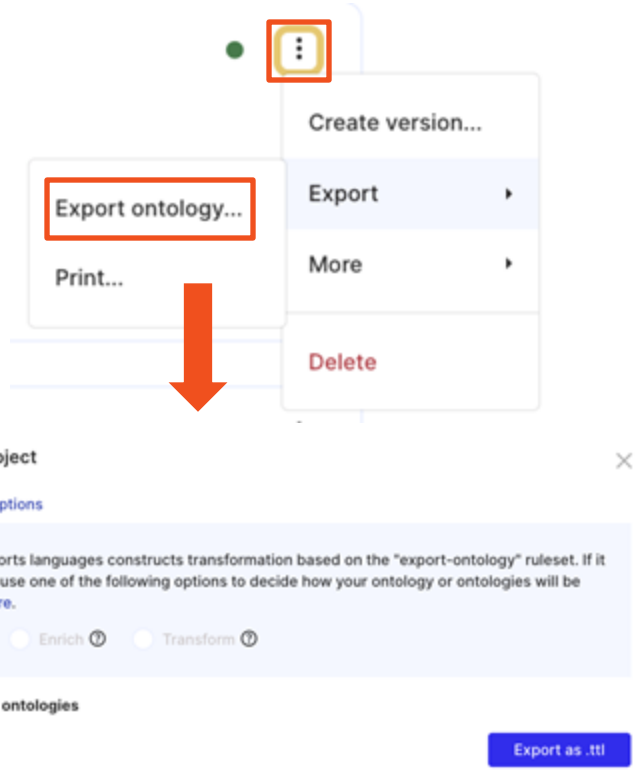
Import existing Ontology/Vocabulary - Ontology transformations

***Warning!**: transformation to non-SHACL ontology in metaphactory can be automatically applied.*

Check *Admin > Assets System Configuration* to set the appropriated option. The following three options are available for import:

- **enrich**: Augment the original ontology with language constructs that are compatible with the visual ontology editor and the model-driven application logic.
- **transform**: Transform the original ontology to language constructs that are compatible with the visual ontology editor and the model-driven application logic. Original language constructs may be dropped as part of transformation.
- **keep**: Import the ontology as is. Incompatible language constructs will not be transformed.



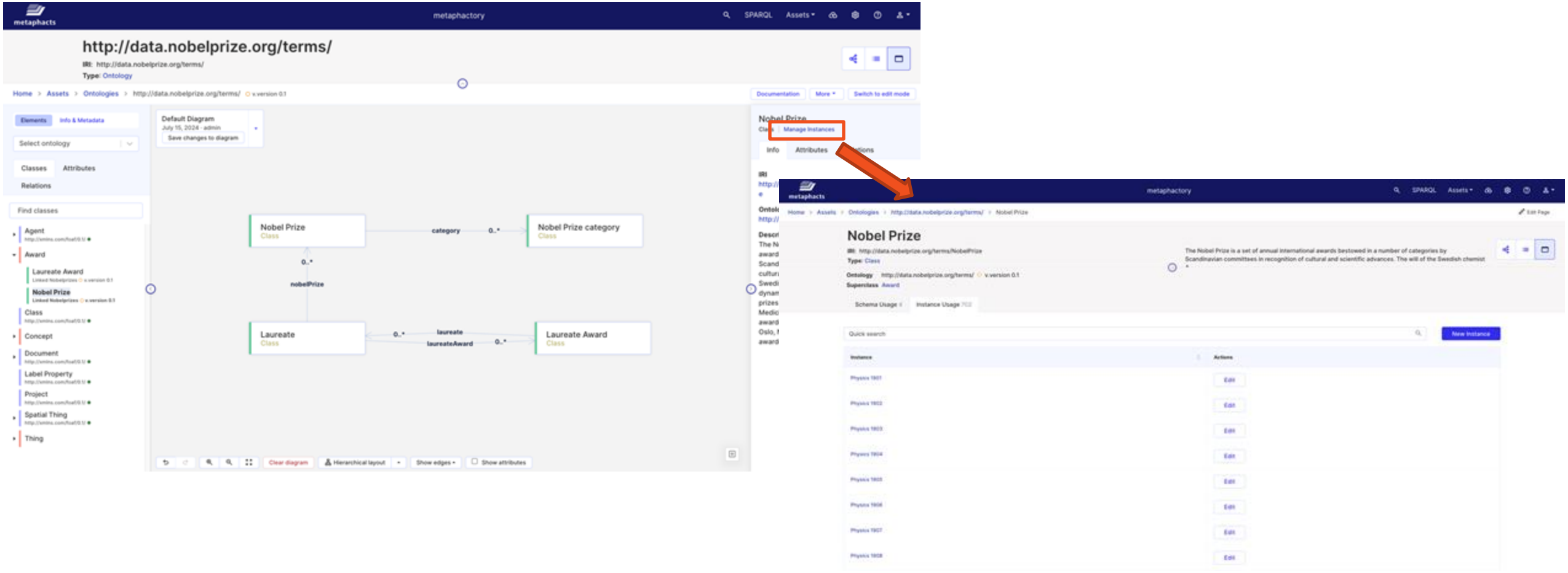


metaphacts

Smart Data. Smart Apps. Smart Decisions.

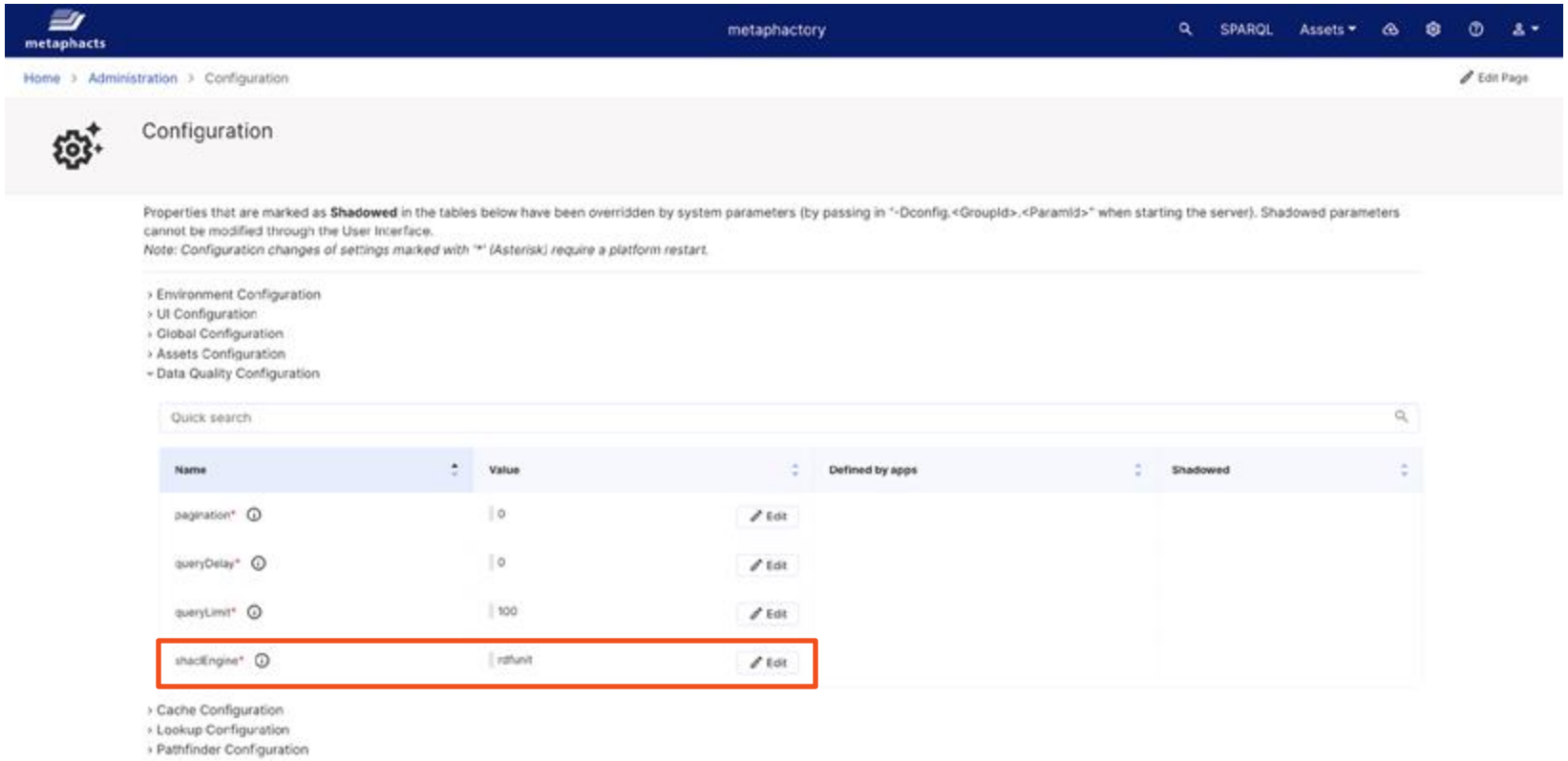
Instance Data Management

Instance data management



Model-driven Data Validation

- **metaphactory** uses an internal SHACL engine by default
  - Possible to set up database's own engine (Stardog or GraphDB)



Documentation: **https://help.metaphacts.com/resource/Help:DataQuality/**

metaphacts

Smart Data. Smart Apps. Smart Decisions.

Model-driven Development

Model-driven Wizards

Search Wizard

Configuration process

1. **Select search features**
2. **Select classes**
   3 classes
3. Select relations and attributes
4. Generate configuration

Download | Discard all

This wizard helps you create configuration a for your search, which you can then insert into the page and fine-tune to your specific needs by following the examples in the metaphactory documentation.

Based on your selection, the wizard will generate the configuration for the metaphactory search component.

Select search features (1 of 4)

Configure your search based on your needs and available data.

- [x] **Enable facets**
  If enabled, additional contextual filters will be generated on top of the results which allow the user to refine them without changing the initial query.
  - [x] **Show exploration facets**
    If enabled, facets will be shown when exploring all entities of a domain without further search conditions being formulated. Consider to disable this option to improve performance on large datasets with millions of entities.
- [x] **Model-Driven Search Profile**
  If enabled, you will only be asked to identify and select the classes relevant as domain for your search. The selection of relations and attributes will be skipped as the search profile is created automatically at runtime based on the selected classes.

Count Query Optimization

Some database vendors offer built-in optimizations for count queries (e.g. for instances of a given type). In order to efficiently render the counts in the domain cards, it is recommended to use the database specific optimization.

- ( ) GraphDB
- ( ) Stardog
- (•) Generic SPARQL

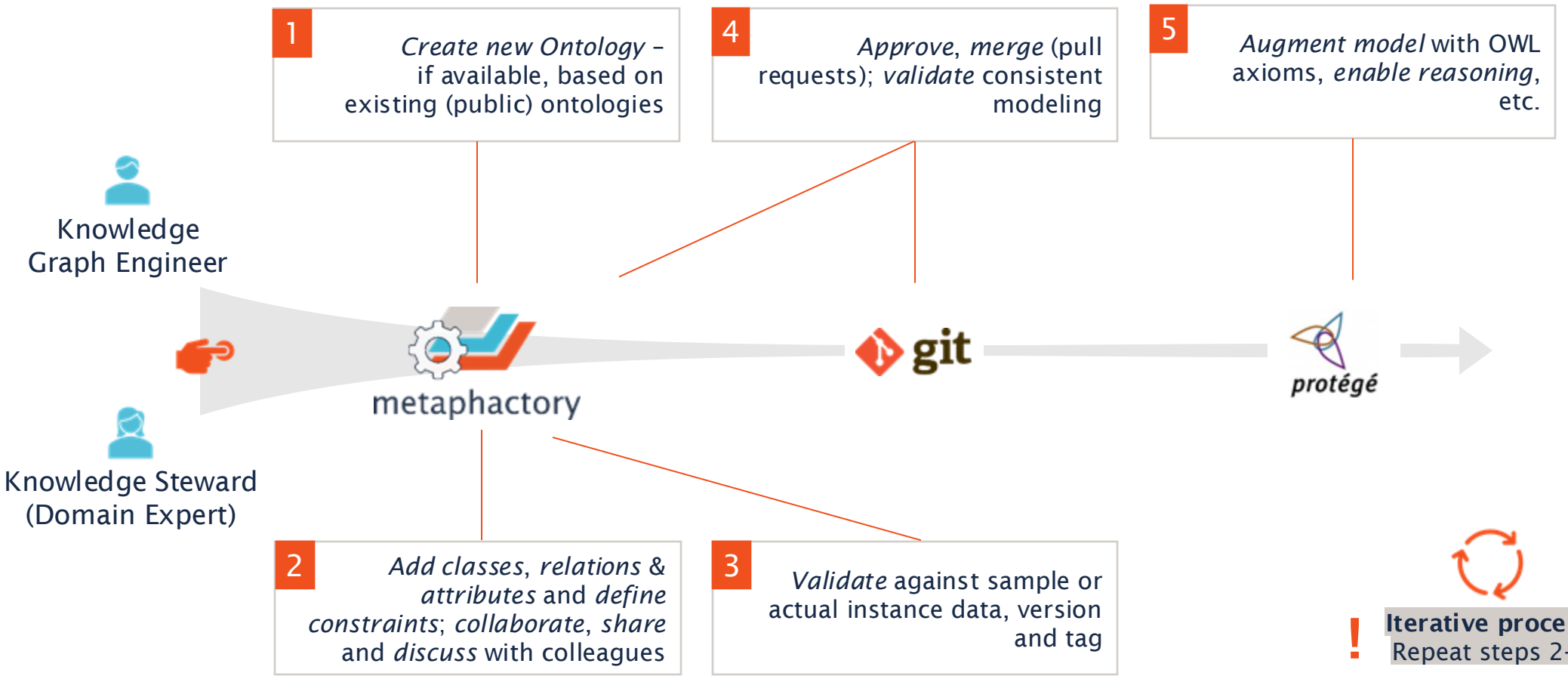
Next: Select classes

metaphacts

Smart Data. Smart Apps. Smart Decisions.

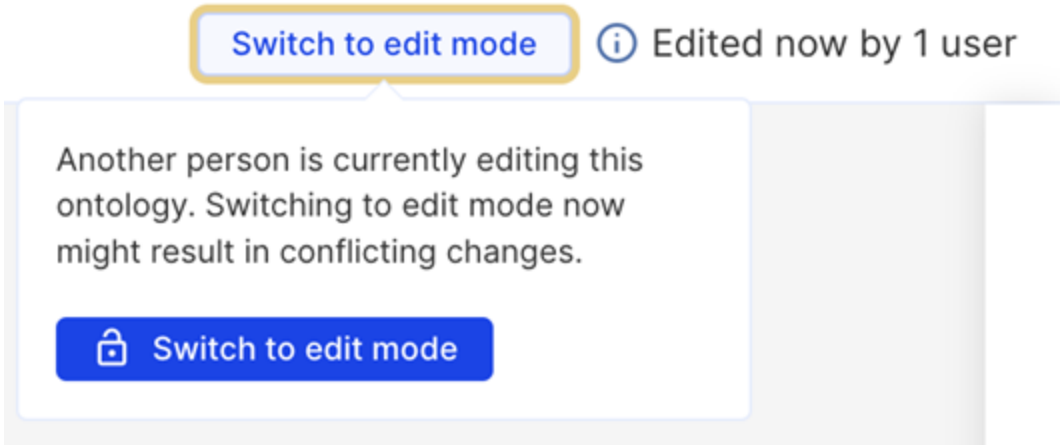
Advanced Modeling Features

Collaborative semantic knowledge modeling



Collaborative Editing notifications

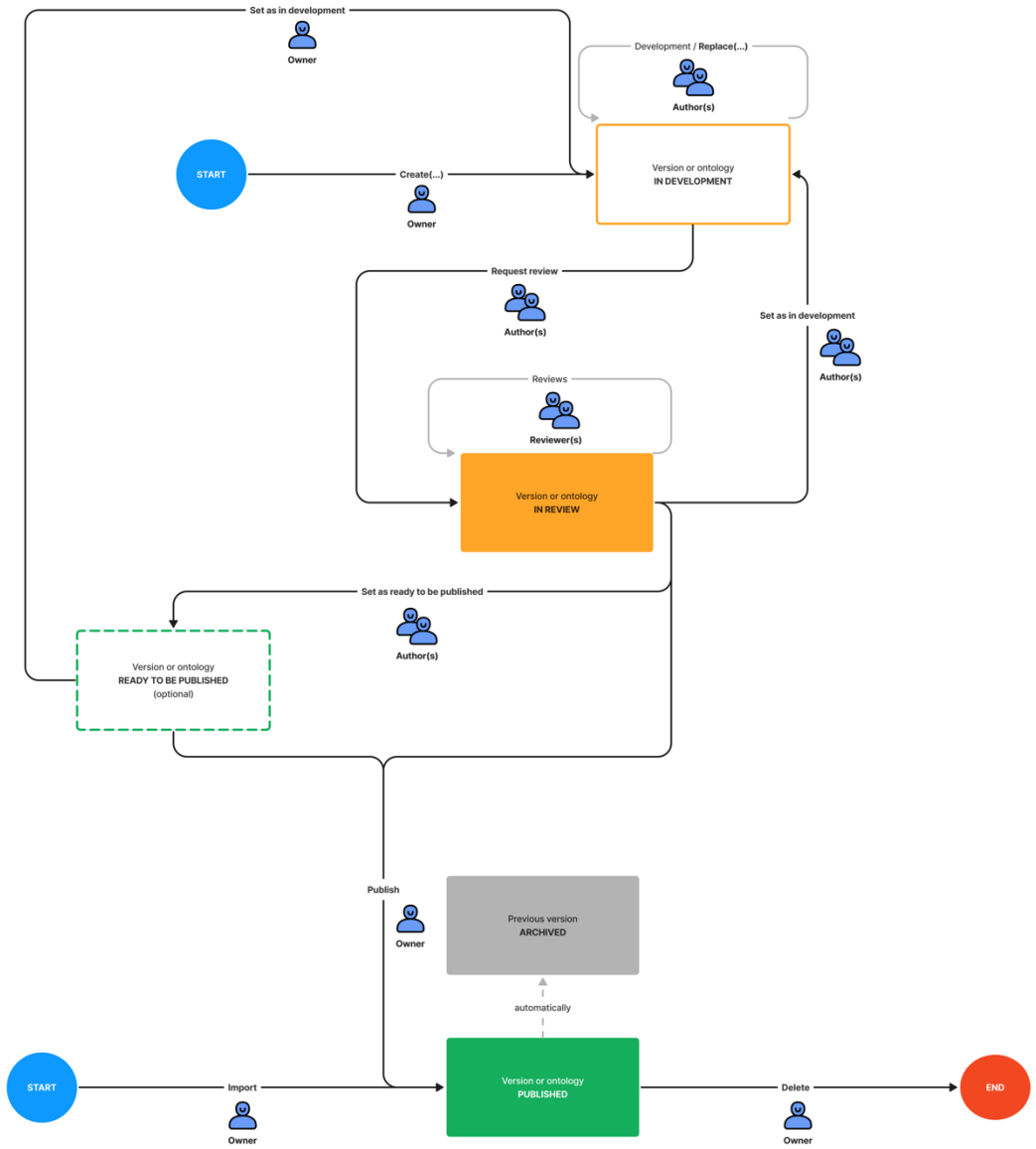
- ‘Soft Locking mechanism’: Alerts that other engineers are editing the asset
  - No. of active editors/edit sessions
  - Users are anonymized by default (data privacy) – can be overridden
    - Admin > Sys. Settings > UI config (anonymizeAssetLocks = false)



- Important Notes: 1 editing session per ontology supported
  - Notifications serve informational purpose, no hard locking
  - Saving changes persists/replaces the entire graph of the ontology
  - Offline checks amongst engineers required in case of questionable locks (e.g. other user has a stale session, or same user in another Web browser tab)

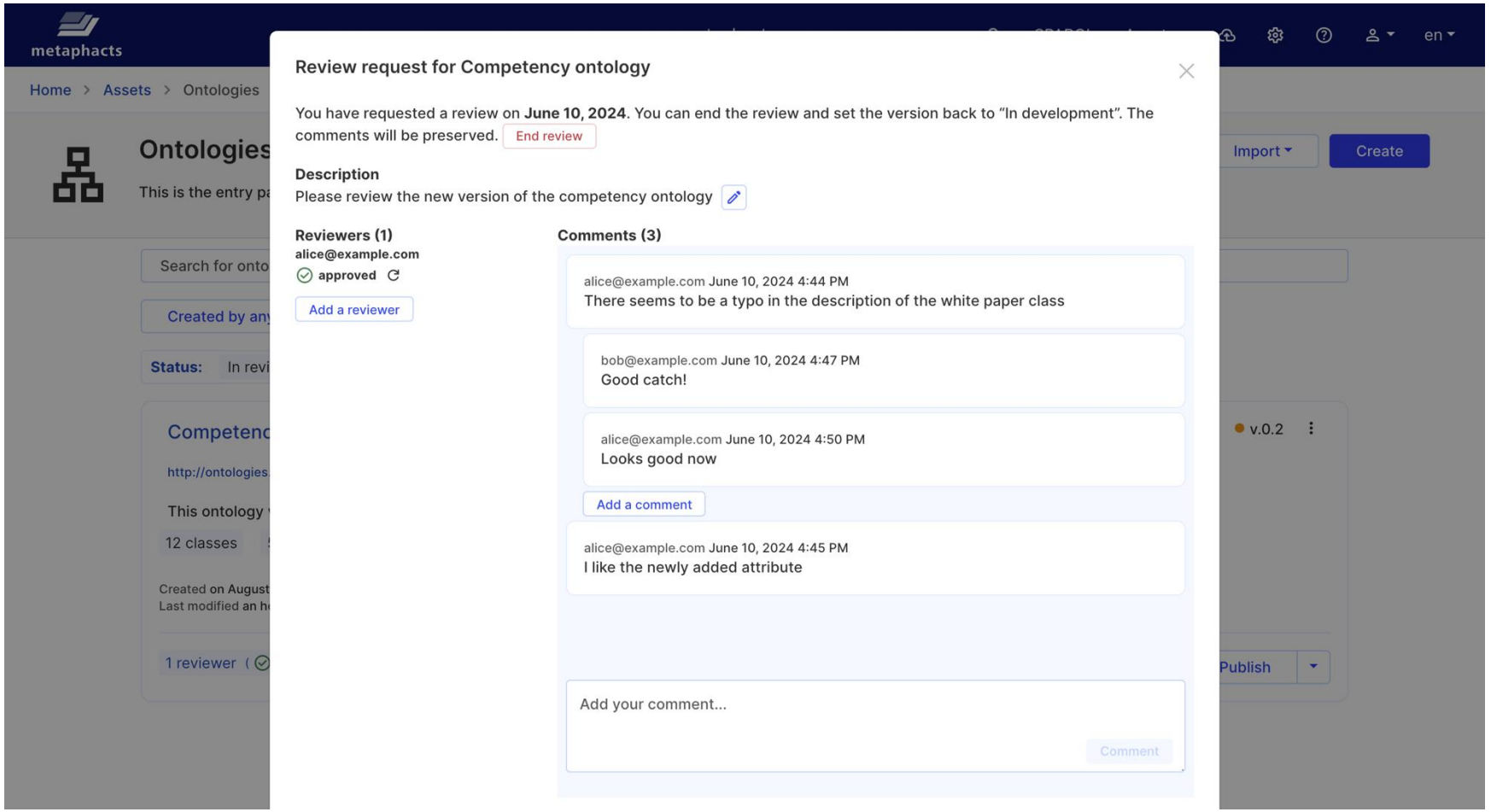
Documentation: http://help.metaphacts.com/resource/Help:VisualOntologyEditing#collaborative-and-concurrent-editing

Collaborative Editorial Workflows



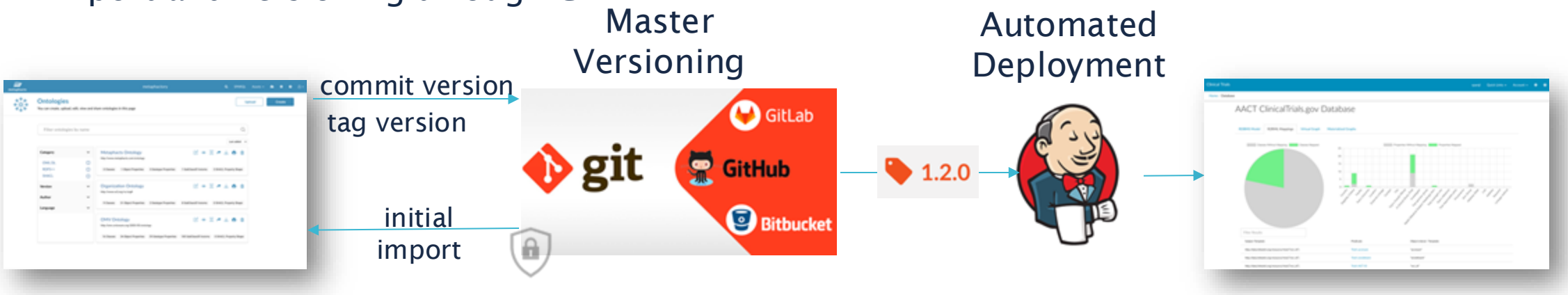
https://help.metaphacts.com/resource/Help:EditorialWorkflowsAndVersioning

Review Requests

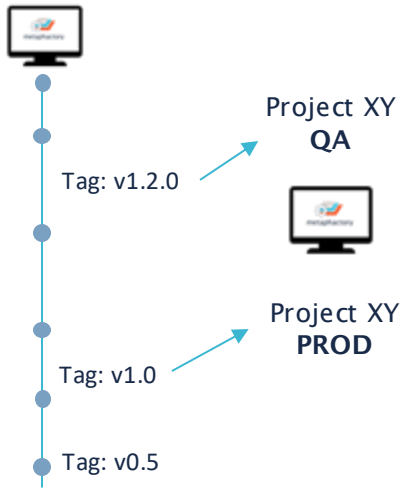


Ontology & Vocabulary Versioning with Git

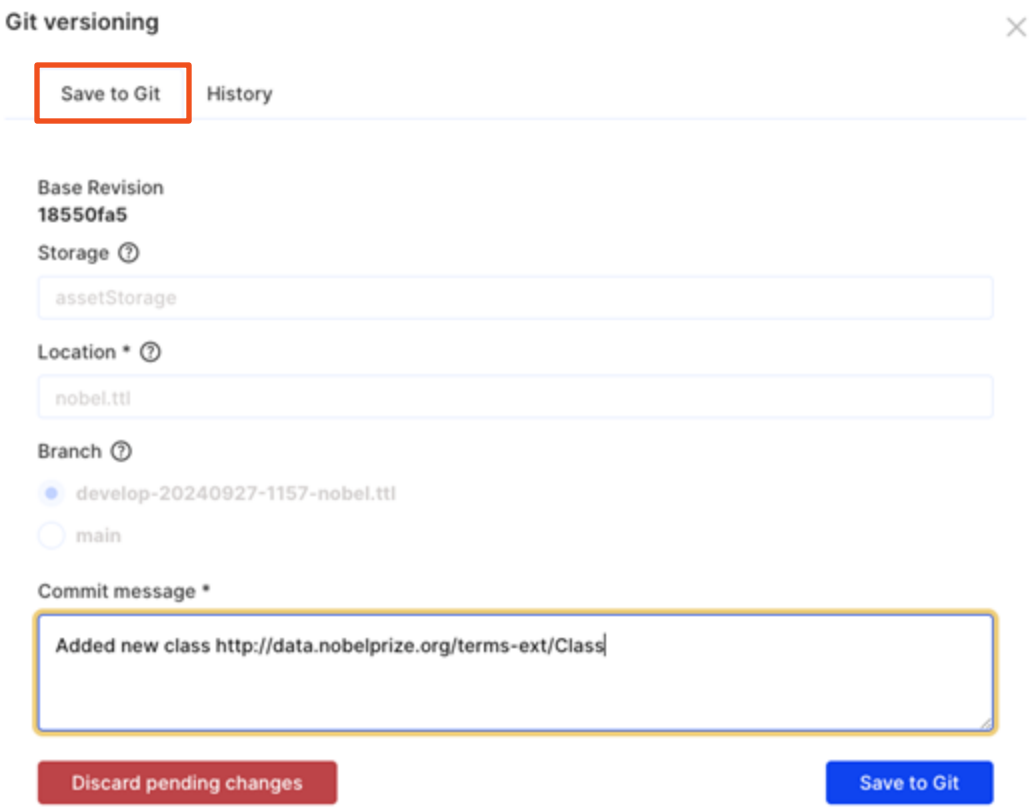
- Import and versioning through GIT

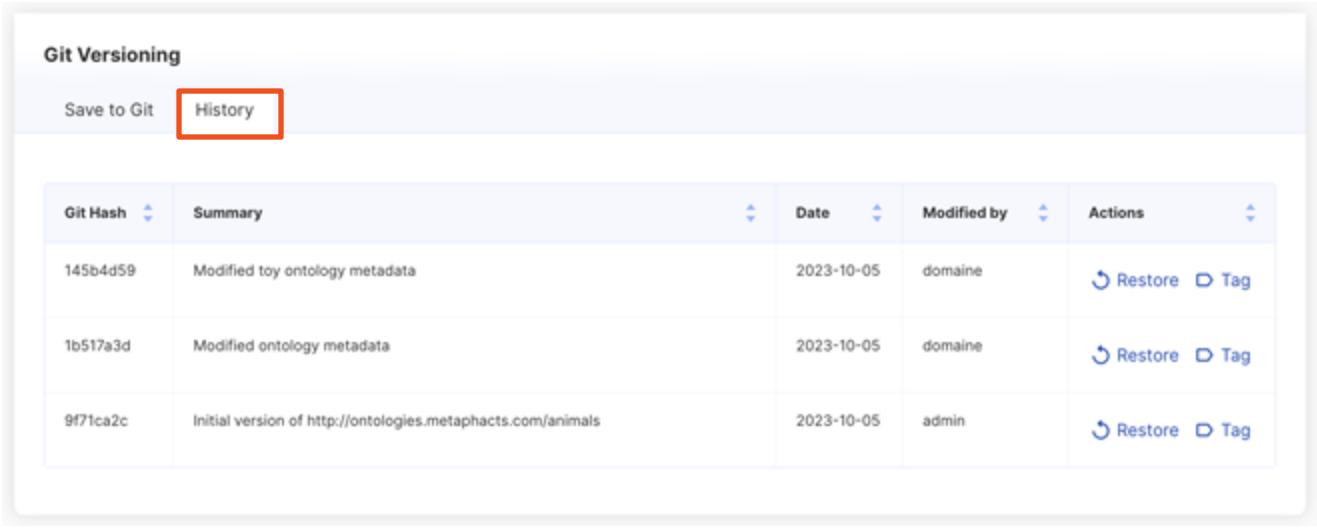


- Works on private and public GIT
  - Support for ssh key handling
- Support for tagging



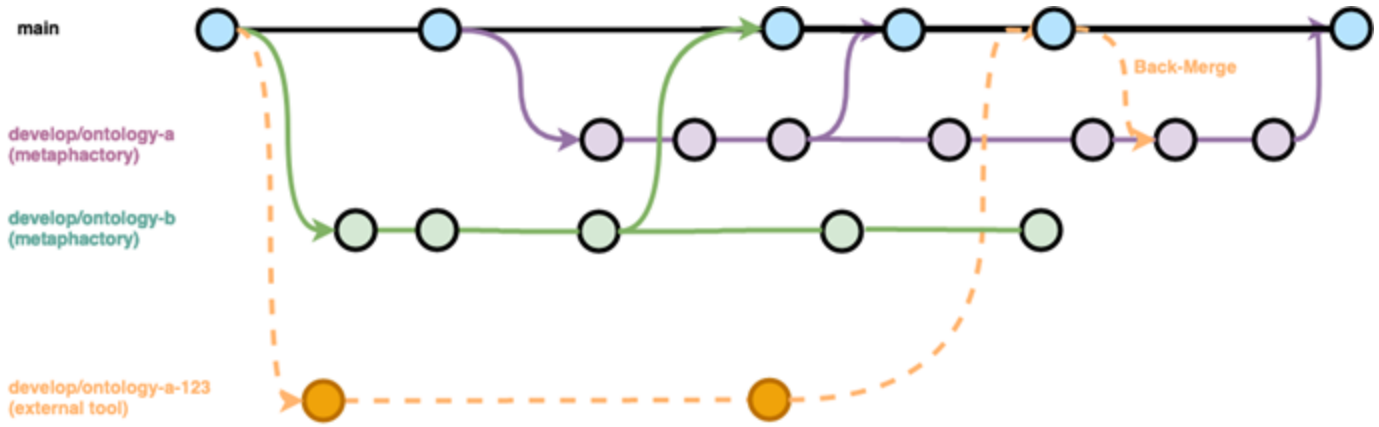
Saving Changes & Documentation, History





Asset Git Branching

- **Recommended workflow**
  - Main Branch: contains stable (release) versions of assets
  - Development Branch(es): Ontology editor operates on revisions
  - Small badge indicates asset connection to main branch (Main) or develop branch (Dev).



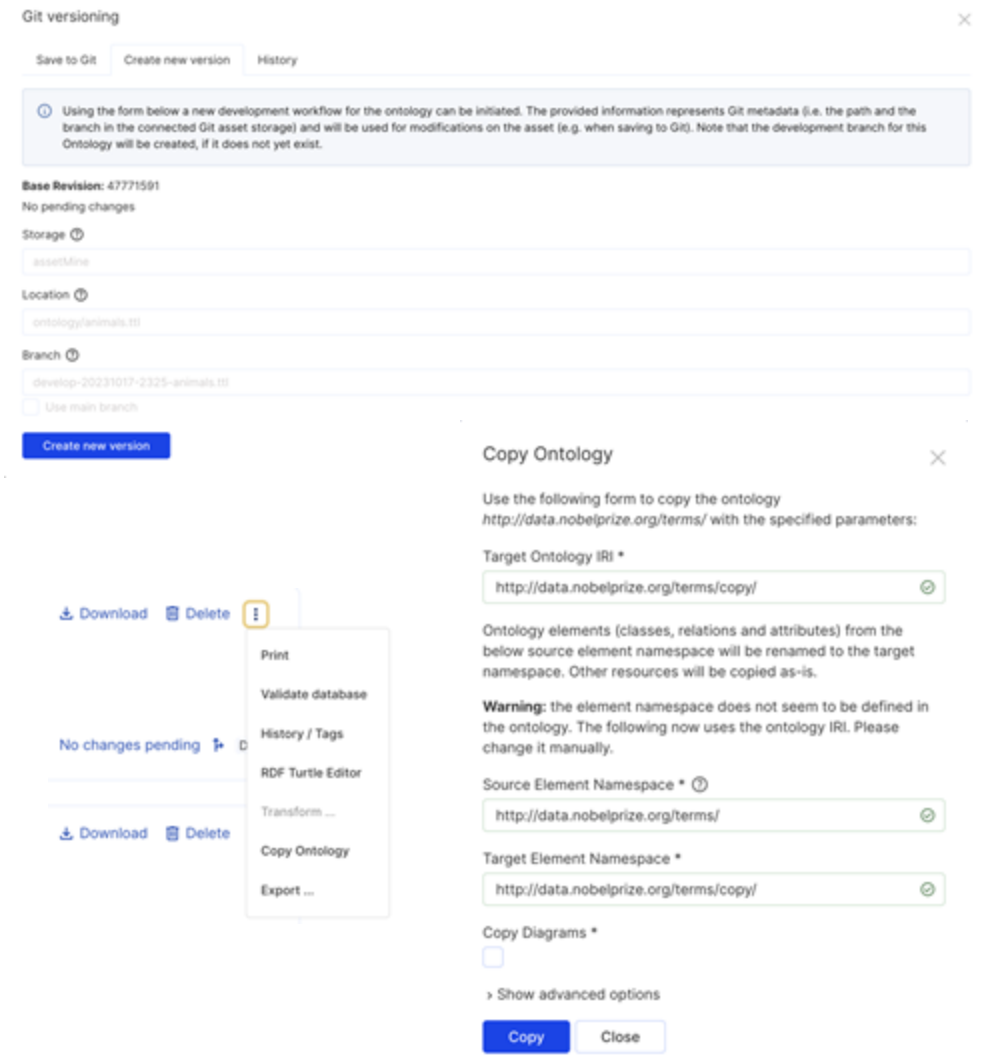
- **One branch per Ontology or Vocabulary ensures isolated/reviewable changes**
  - easy integration in external Git workflows
    - See ontology changes as diff with main branch
    - PRs can be used for reviews
    - Merge + Git approvals can be employed for publishing
  - Automation with Git-Actions or Webhooks possible , e.g., consistency checks, publish metadata to other DBs, injecting asset to another system - staging systems.
- **Important Notes**
  - Asset scanning on startup operates on main branch, i.e., only ontologies/vocabularies in main branch can be imported
  - Main branch = Configured branch in the Asset Storage (independent of name)

Documentation: http://help.metaphacts.com/resource/Help:AssetManagement#git-branching-model

Asset Git Branching

- **Creating new version (branch) of Ontology/Vocabulary**
  1. Existing ontology imported from main branch
  2. Specify and confirm the Development Git metadata (location, branch, etc).
  3. Branch is created in the Git repository and is applied on save

- Entry points & implications:
  - Status badge of an asset (Main branch) opens a form to create a new dev version
    - Implication: Asset identity not changed – Ontology/Vocabulary editor operates on existing loaded Asset
  - Copying an existing ontology
    - Implication: New asset identity (i.e., separate/new ontology identifier etc).
    - Parameters pre-set with computed defaults based on original



Documentation: http://help.metaphacts.com/resource/Help:AssetManagement#git-branching-model

metaphacts

Smart Data. Smart Apps. Smart Decisions.

Free Exercises

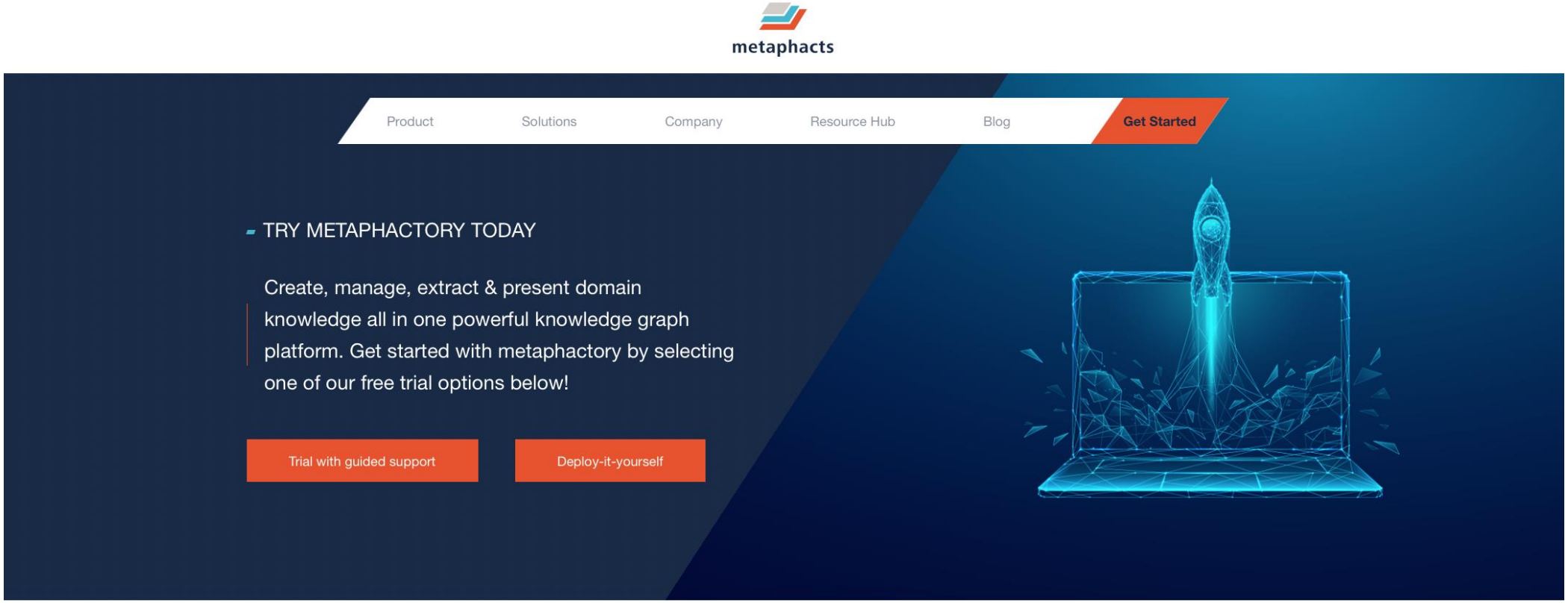
After the tutorial

- We will keep the tutorial instances running for one week
- Upon email request we can extend to four weeks

- Then there are various options to continue / get started with your own metaphactory

To Continue with metaphactory: https://metaphacts.com/get-started

metaphacts



Select your trial option

Whether you prefer to do-it-yourself or receive guided support, we have a range of trial options to choose from.

The metaphacts Academy

metaphacts

Your learning path..

metaphactory Basics

Covers out-of-the-box features and introduces you to Knowledge Graph search, exploration and navigation using metaphactory.

- Intro to metaphactory
- Overview of platform features
- Basics

Building Knowledge Graph Applications

App Building Basics

Covers platform customization topics to support build apps for data visualization and discovery.

» Templates & Customization
» Semantic components
» Events

App Building Advanced

Covers advanced features for templating and interactions involving multiple components.

» Security & permissions
» App packing and lifecycle
» Advanced data authoring, search framework and federation.

Building models for domain experts and data engineers

Visual Modelling Basics
~ Ontology & Vocabulary ~

A self-guided business user tutorial to semantic knowledge modeling.

» Vocabulary & Taxonomy
» Ontology
» Data cataloging

Visual Modelling Advanced
~ Ontology & Vocabulary ~

Advanced features for knowledge modeling & assets management.

» Editorial workflows
» Management
» Versioning

Data Authoring

Focuses on creating forms for data authoring and editing, and how to use them in data curation workflows.

» Forms & Model-driven Forms
» Outlook Visual Editing
» Workflows

Search

Provides a deep dive into configuring search interfaces and search results.

» Semantic search framework
» Facet filtering
» Lookup Service
» Result visualization

Federation

Focuses on the setup and querying of external data sources to augment the knowledge graph beyond the main graph database.

» Ephedra & FedX
» Rest API
» SQL

Platform extensions

Covers the development of custom extension for custom UI or backend requirements.

» App packing and deployment
» Frontend extensions as custom components in JS
» Backend extensions in Java

Certification (optional)

metaphactory

★ Knowledge Graph Engineer (Associate)

★ Knowledge Graph Application Engineer (Associate)

★★ Knowledge Graph Solution Architect (Advanced)

★★★ Knowledge Graph Solution Architect (Expert)

The concepts and standards behind Knowledge Graphs

OWL + SHACL

Covers ontology modelling with OWL and the creation of SHACL constraints using metaphactory.

» Classes, Properties
» OWL in RDF syntax
» SHACL node shapes

RDF + SPARQL

Provides an overview of the RDF data model and the use of the SPARQL query language with metaphactory.

» RDF triples, Named Graphs
» SPARQL query forms
» Outlook RDF-star, SPARQL-star

Production readiness

DevOps

Covers the deployment and configuration of metaphactory for production setups.

» Docker deployment
» Prod-QA-DEV setup, Git storage
» Connect to graph databases
» Single-Sign-On

› A training for your custom topic can be arranged upon request.

QA & Feedback

metaphacts