IOF for Digital Thread Tutorial

Organizer: Dušan Šormaz[1]

Presenters: Arkopaul Sarkar[2] and Saruda Seeharit[1]

[1] Industrial and Systems Engineering Department , Ohio University, Athens, OH, USA
[2] The National Engineering School of Tarbes (ENIT) , Tarbes, France

Prepared by IOF Production Planning and Scheduling Working Group

AGENDA

01 **Purposes of Tutorial**
Expected Outcomes

02 **Overview of Digital Thread**
Digital Model, Digital Shadow, Digital Twin
IOF and Approaches

03 **Case Studies**
Jet Engine, Bio Process, Machining (plastics case)
Competency Question and Capability

04 **Tools for Ontology Development, Reasoning, and Testing**
Protégé, GraphDB, SPARQL

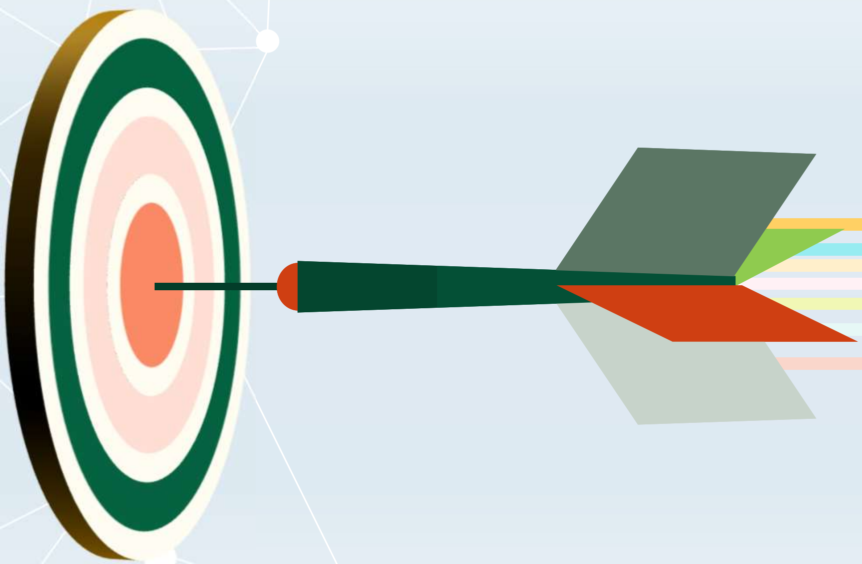
05 **Practical Work**
Hands-on Experience

Purposes

Purposes of Tutorial

Expected Outcomes

What is this tutorial about?



- ❑ Explain development of design and planning entities using IOF ontologies
- ❑ Compare various approaches to model digital artifact ontology
- ❑ Utilize IOF to build digital artifact ontologies
- ❑ Practice using Protégé and GraphDB to develop use cases
- ❑ Practice using GraphDB to explore knowledge graph
- ❑ Practice using GraphDB to run use case SPARQL queries
- ❑ Build instance data sets from external sources (csv files)

Understanding of importance of modeling digital artifacts and digital models

Enhanced skills in using GraphDB and SPARQL to obtain answers to the competency questions related to design and planning

Enhanced Protégé skills for managing imports, developing ontologies, and building instance data

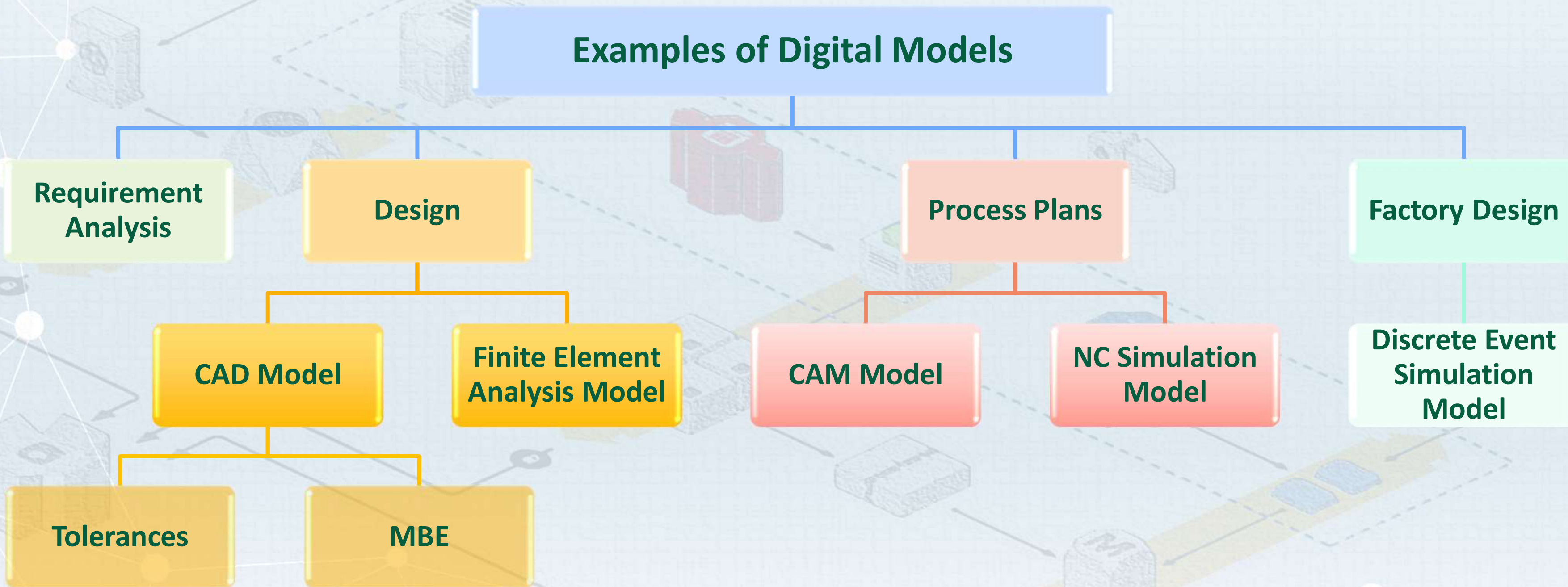
Practical skills for developing relations between digital artifacts and produced artifacts

What are the expected outcomes?

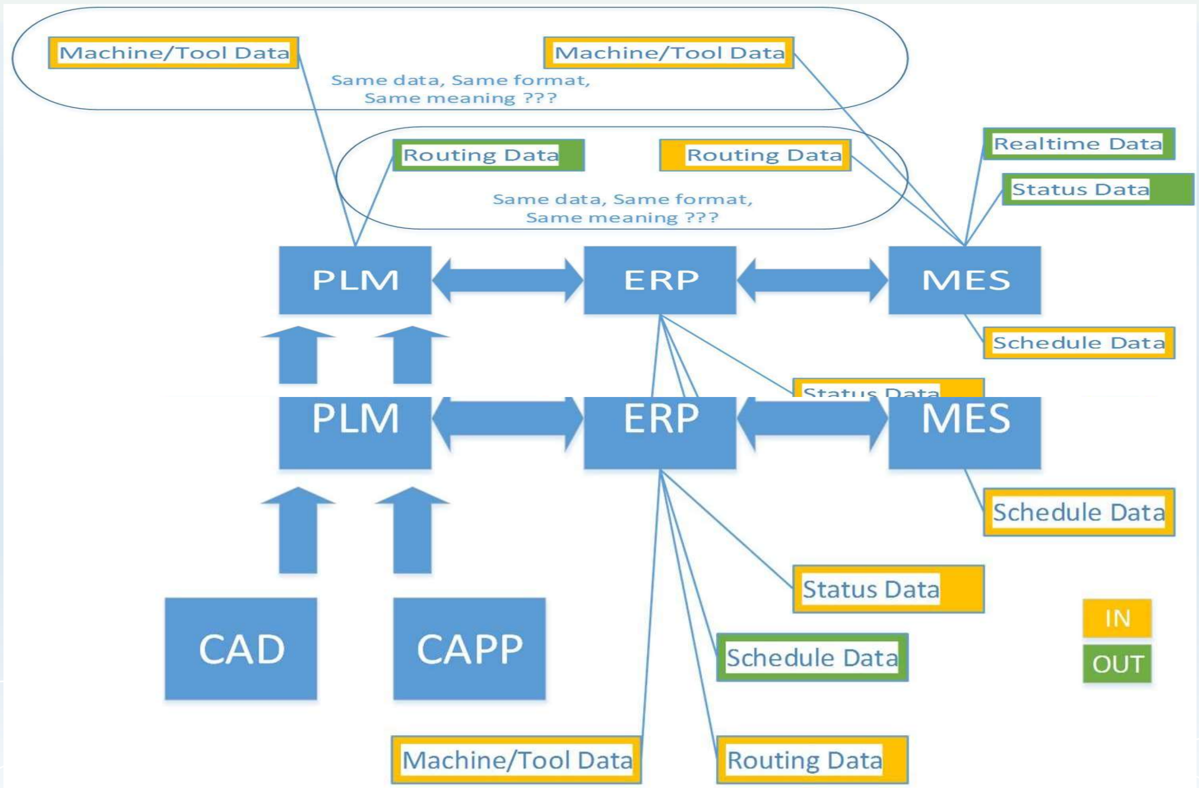
Overview of Digital Thread

Digital Model, Digital Shadow, Digital Twin
IOF and Approaches

Examples of Digital Models (Artifacts)

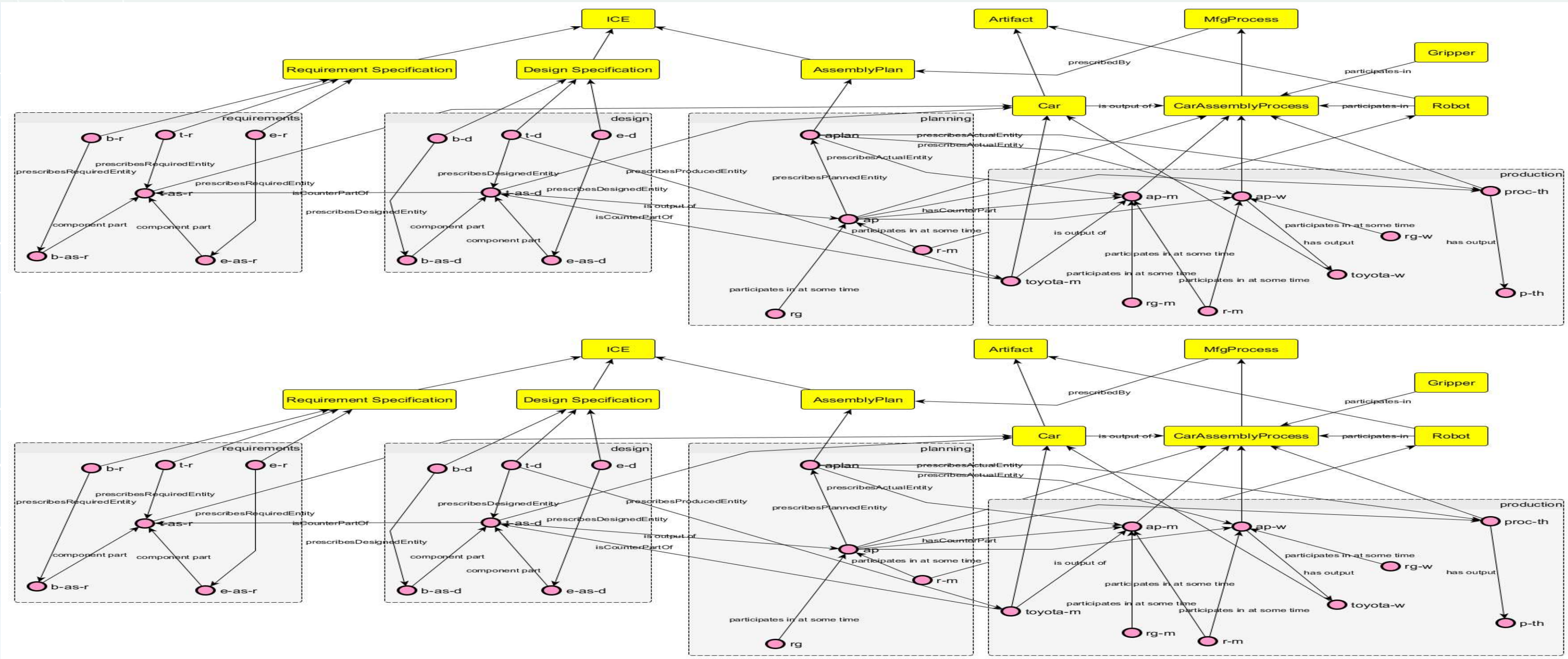


Information Interoperability



Sormaz, D., and Sarkar, A., 2021, “Interoperability between PLM, ERP, and MES Systems Using Formal Ontologies,” Taichung, Taiwan.

Product Life Cycle



APPROACHES

Approaches for Future/Digital Artifacts

ICE

Information Content Entity

Use ICE to represent all information (knowledge, decisions) about the future artifacts

MRO

Modal Relation Ontology

Use MRO approach to represent future artifacts, based on replica of relations (from BFO or any) into Modal relation space

R/S

Representation and Specification

Use R/S approach (given in a paper by Sarkar and Sormaz), specialize ICE to have Representation and Specification as subclasses

CR

Counterpart Relation

Use CR approach, which is motivated by MRO approach but connects relations and provides for new relations between designed/planned entities and actual entities

Sormaz, D. , Kulvatunyou, B. , Drobnjakovic, M. , Seeharit, S. and Sarkar, A. (2023), Comparison of Ontological Representations of Relations between Digital and Physical Artifacts In Manufacturing Domain, Proceedings of the ASME 2023 IDETC Conference, Boston, MA, US,

Approaches Used in This Tutorial

ICE

Information Content Entity

CR

Counterpart Relation

Use Cases in the Tutorial

Jet Engine and Biomanufacturing Process

Jet Engine Design and Requirement Verification

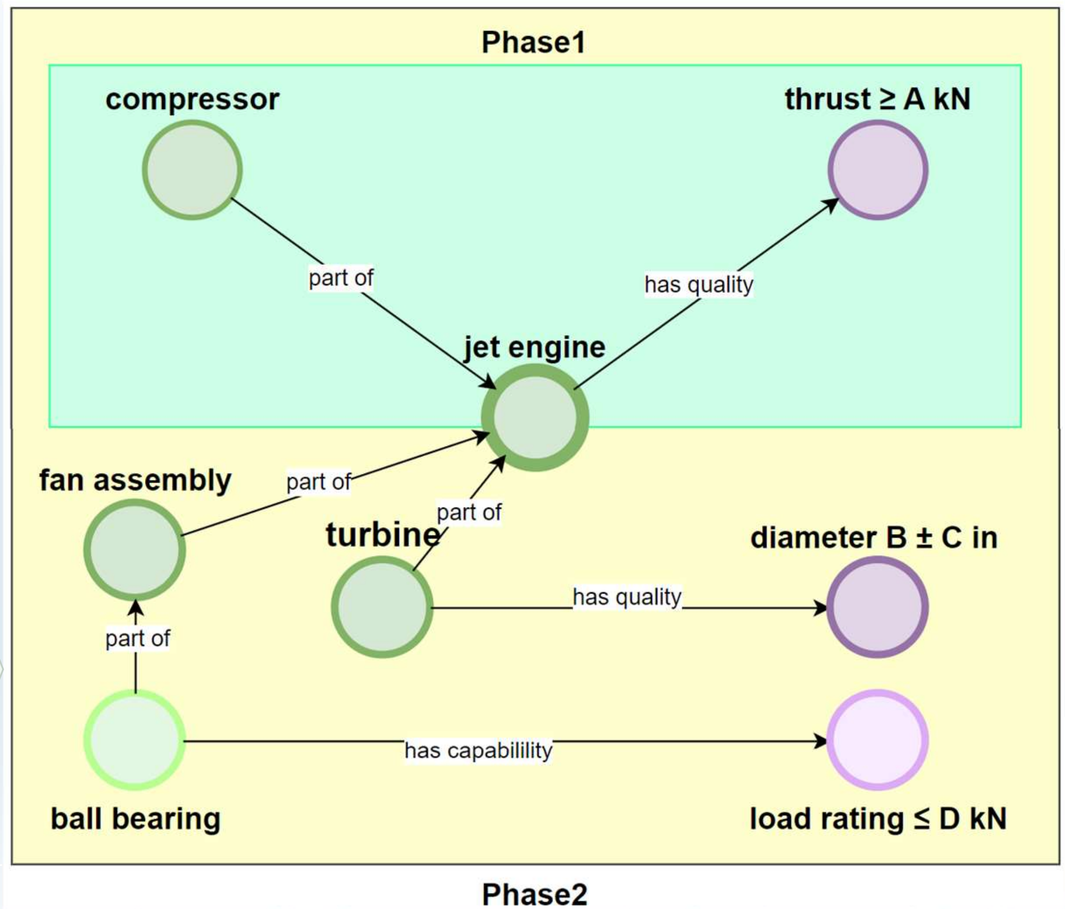
An engineering task:

"There is a need to design and produce a jet engine that will have a compressor as its part, and it will be able to produce a minimal thrust of 700 kN".

This simple example provides sufficient elements to compare the approaches.

Jet Engine Phase 2

- Add Components and Qualities
  - Turbine, Fan Assembly, Ball Bearing
  - Turbine diameter, Ball bearing load
- Run the same competency questions



Jet Engine Competency Questions

Does a real engine have at least the same thrust as its design?

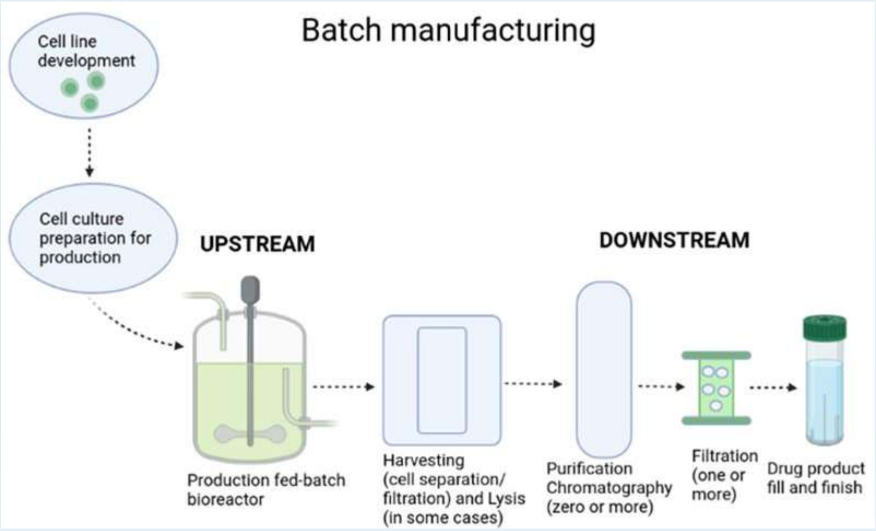
Does a turbine diameter of real engine have value as specified in the design?

Which (real) engines failed design specification?

Bio Process

The fed-batch production bioreactor unit operation

- The fed-batch production bioreactor unit operation consists of two phases: growth phase and production phase. The growth phase precedes the production phase. In both phases the pH MUST be kept at 7 ± 0.1. To achieve this a pH controller is required as a process participant which has the capability to control pH up to a precision of 0.1. The process specification prescribes to use either bioreactor instance1 or instance2. In the run instance 1 was used. Both bioreactors are identical w.r.t. volume which is 3L. The unit operation duration is 21 days. However, due to some in-process complications the run only lasted 18days (we do not have to capture the in-process complications) just the duration difference.



Bio Process Competency Questions

Which equipment are used in a (real) process?

What is the difference of duration between actual and planned process?

Tools for Ontology Development, Reasoning, and Testing

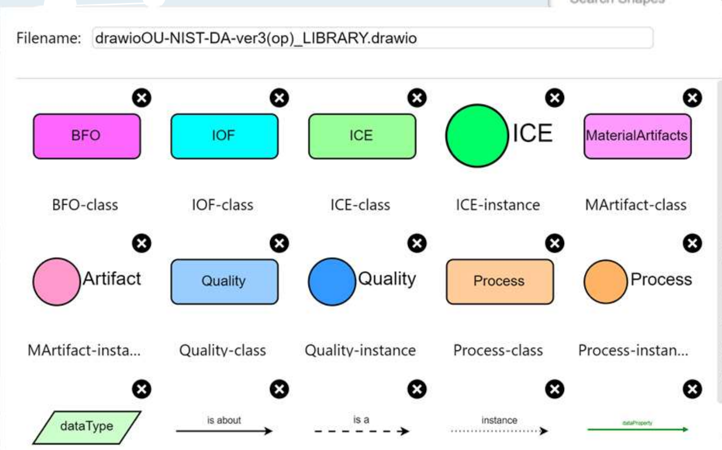
Protégé, GraphDB, SPARQL

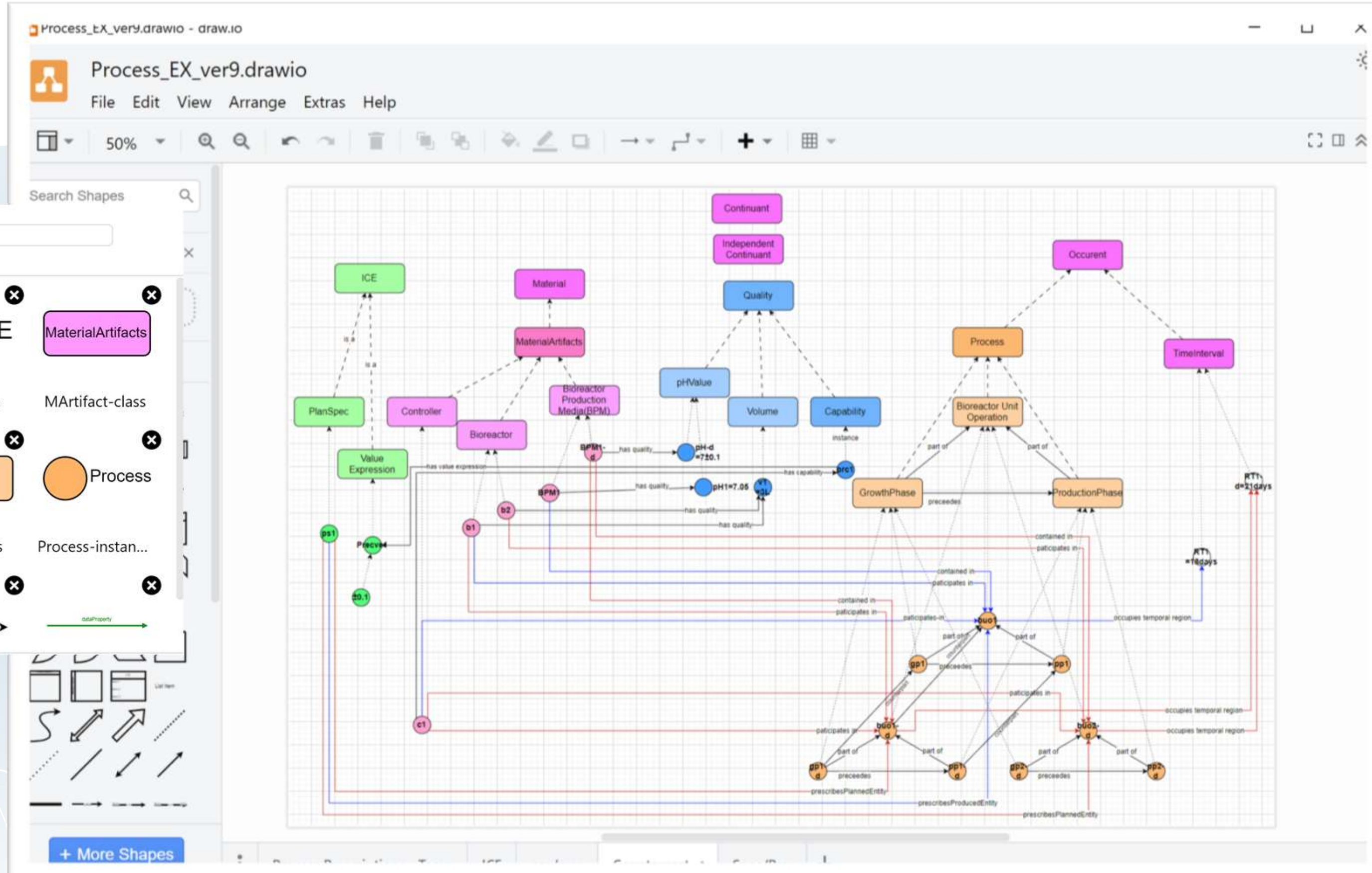
List of Tools

- Drawio editor
- yEd (Graphml) editor

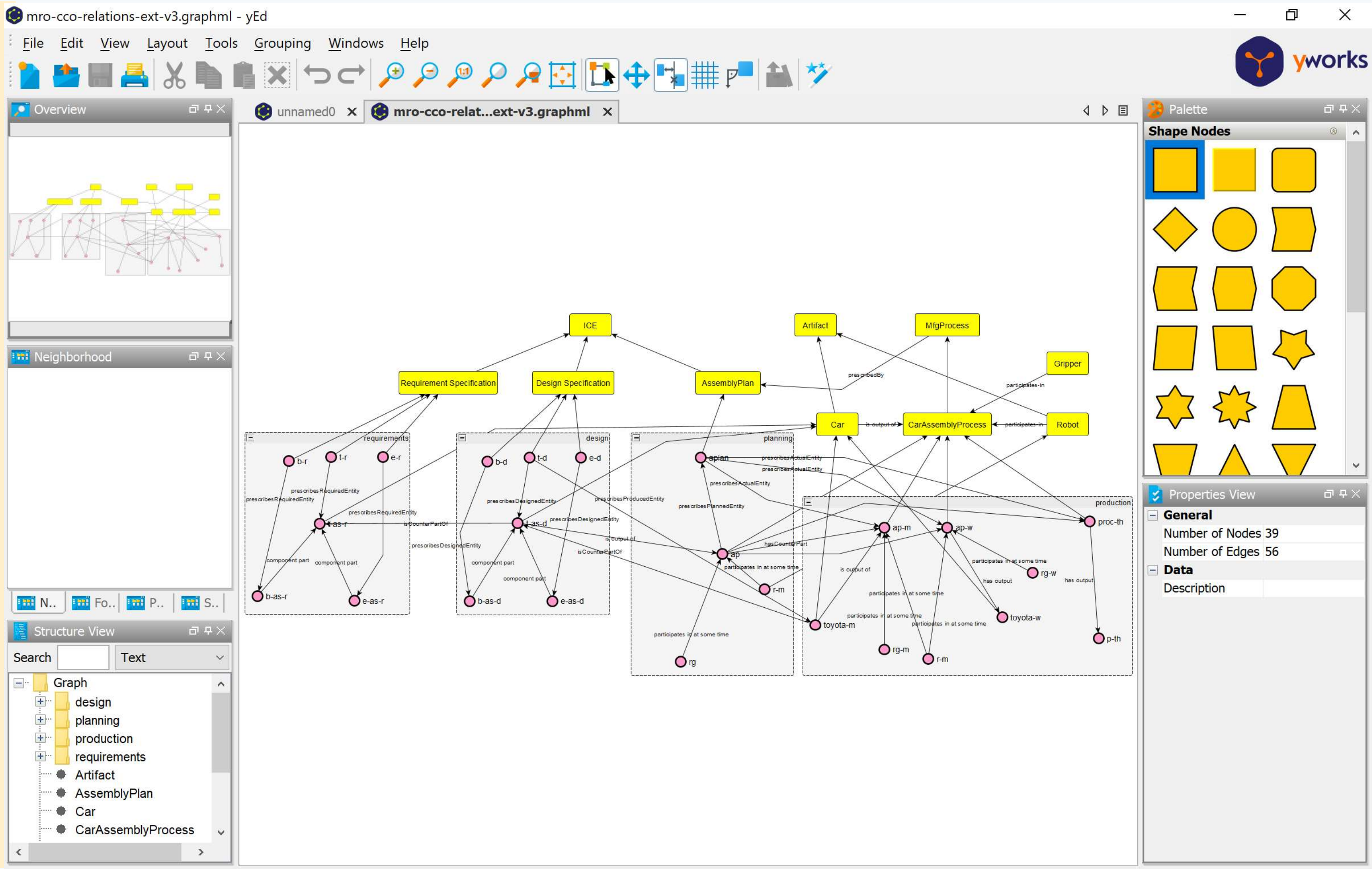
- Protégé
- GraphDB
  - Repository
  - Knowledge graph exploration
  - SPARQL queries

Drawio Editor





yEd (graphml) Editor



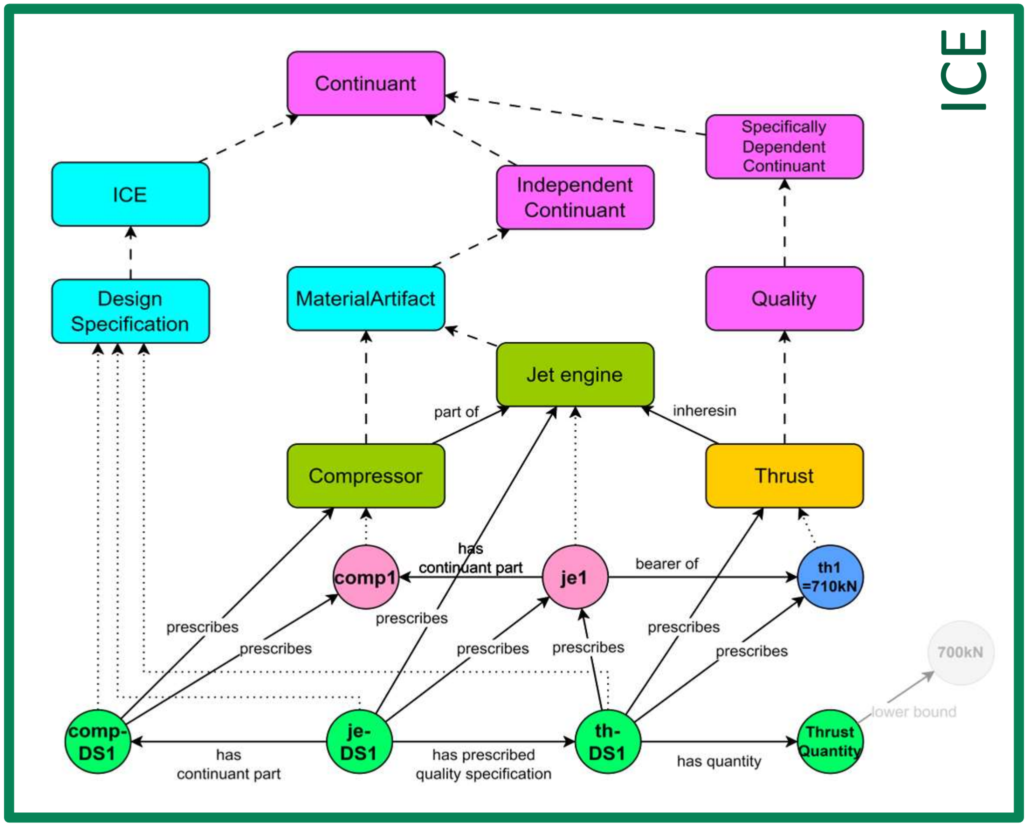
Ontologies

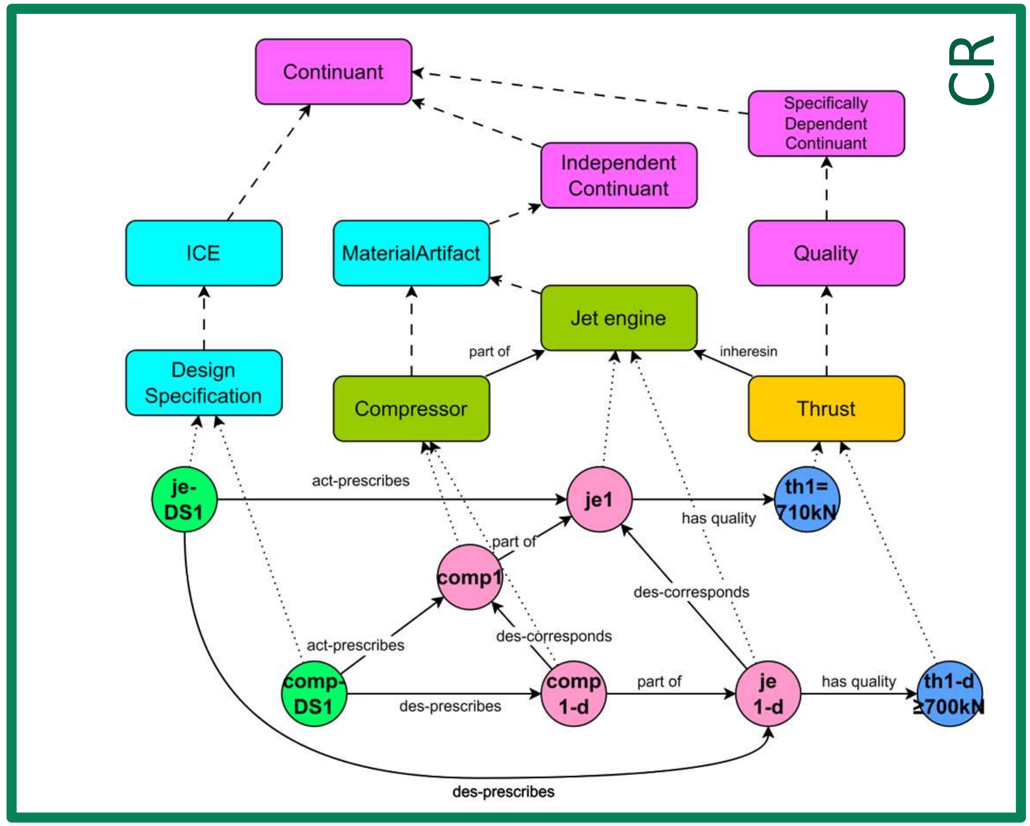
- BFO
- IOF Core
- Mfg-Planning (not used in the practice)
- Jet engine, Bio Manufacturing
- Example Data (A-Box)

Practical Work

Hands-on Experience

Jet Engine Model Approaches

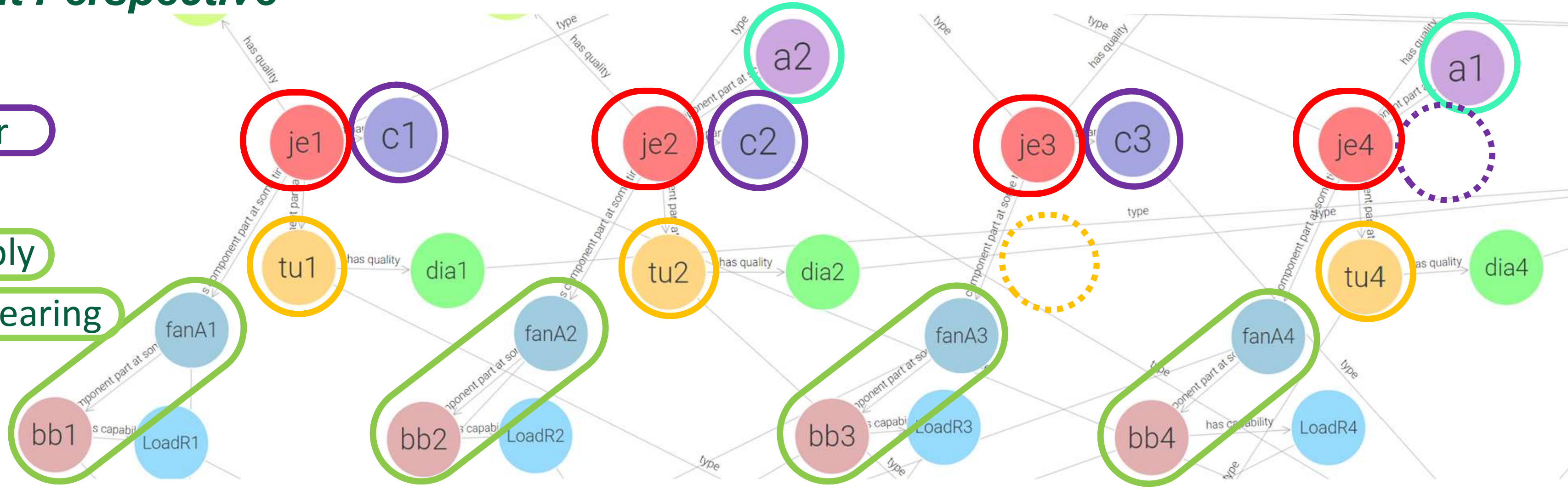




Use Case Description

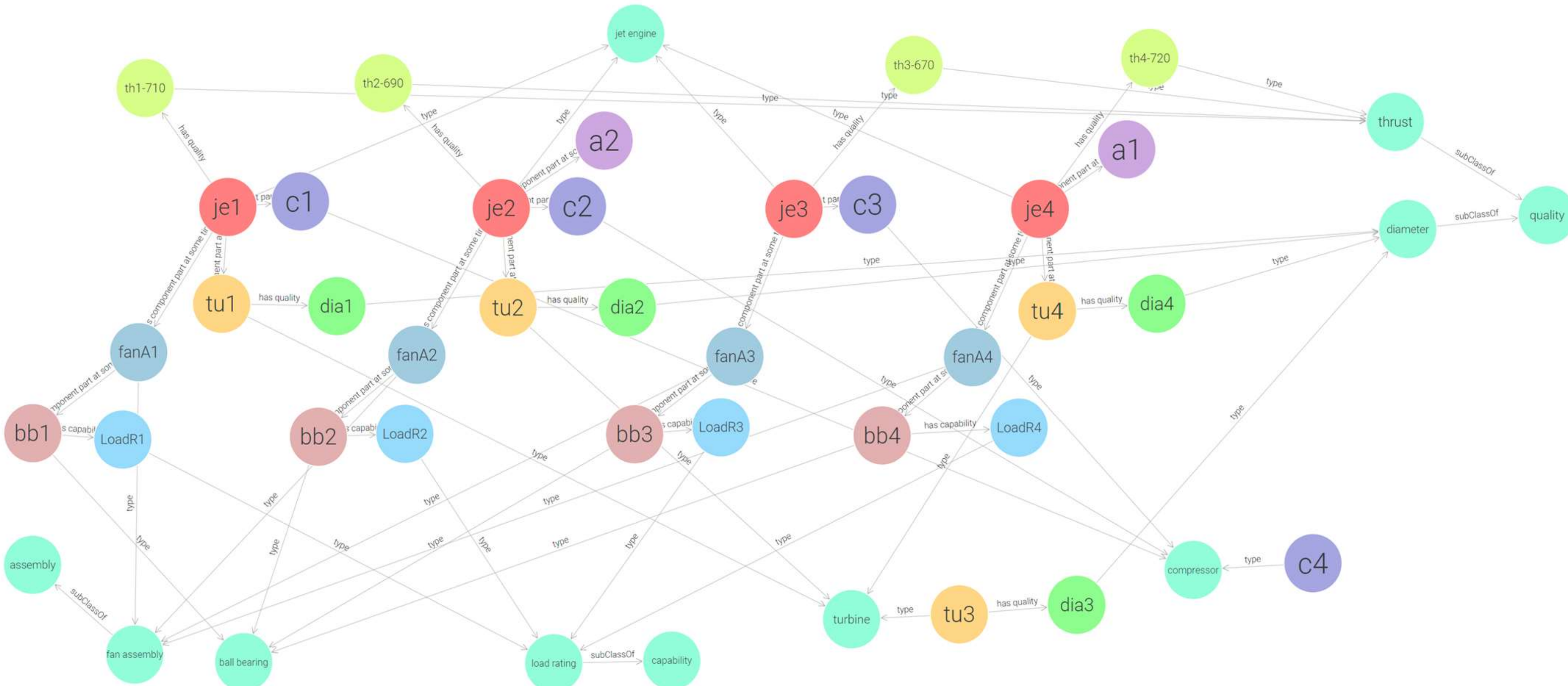
Component Perspective

Jet Engine
- ❑ Compressor
- ❑ Turbine
- ❑ Fan Assembly
  - ❑ Ball Bearing



Condition\Jet Engine	Jet Engine 1	Jet Engine 2	Jet Engine 3	Jet Engine 4	[countif ROW value = TRUE]
(A) Has ALL design component	TRUE	TRUE	FALSE (tur)	FALSE (com)	2
(B) No Extra component	TRUE	FALSE (a2)	TRUE	FALSE (a1)	2
Component test	TRUE	FALSE	FALSE	FALSE	1

Classes and instances [Jet Engine Phase 2 case study] showing in GraphDB



SPARQL Query b – CR approach

Does a real engine have at least the same thrust as designed? (minimum of 700 kN)

```
SELECT ?engine ?thrust ?realthrustValue ?spec ?engdesign ?thrustdesign ?thrustDesignValueExpr ?isSatisfactory
  WHERE {
  ?engine rdf:type jeb:JetEngine.                          # find engine and its thrust
  ?spec cr:prescribesActualEntity ?engine.
  ?engine core:hasQuality ?thrust.

  ?thrust core:hasValueExpressionAtAllTimes ?thrustValueExpr.          # find the thrust value
  ?thrustValueExpr core:hasSimpleExpressionValue ?realthrustValue.

  ?spec cr:prescribesDesignedEntity ?engdesign.            # find the engine design from associated spec
  ?engdesign core:hasQuality ?thrustdesign.

  ?thrustdesign core:hasValueExpressionAtAllTimes ?thrustDesignValueExpr.   # find design thrust value
  ?thrustDesignValueExpr jeb:hasLowerBoundValue ?designThrustValue.

  BIND((?realthrustValue >= ?designThrustValue) as ?isSatisfactory)      }   # compare them
```

SPARQL Query b – ICE approach

Does a real engine have at least the same thrust as designed? (minimum of 700 kN)

```
SELECT DISTINCT ?jetEngineDesignSpec ?jetEngine ?designedThrustValue ?realThrustValue ?isSatisfactory
    WHERE {

    ?jetEngine rdf:type jet:JetEngine.                                   #retrieve all individuals that are jet engines
    ?jetEngine core:hasQuality ?thrust.

    ?thrust core:hasValueExpressionAtAllTimes ?thrustValueExpr.          #retrieve thrust
    ?thrustValueExpr core:hasSimpleExpressionValue ?realThrustValue.

    ?jetEngineDesignSpec rdf:type jet-ice:JetEngineSpec.                 #retrieve all designs for the jet engine
    ?jetEngineDesignSpec core:prescribes ?jetEngine.
    ?jetEngineDesignSpec core:prescribes ?thrustSpec.

    ?thrustSpec bfo:BFO_0000110 ?thrustDesignValueExpr.                  #retrieve thrust specification value
    ?thrustDesignValueExpr jet-ice:hasLowerBoundValue ?designedThrustValue.

    BIND((?realthrustValue >= ?designedthrustValue) as ?isSatisfactory) }   #compare them
```

https://github.com/ohio-ontology/IOF-DigitalThread-Tutorial/

protégé

Setup Protégé

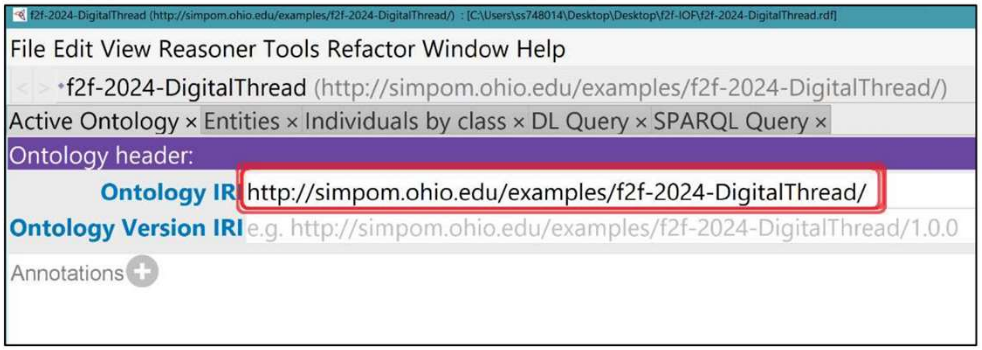


Figure 1: Setting Up an IRI for Your Ontology

Figure 4: Choosing a Location to Save Your File.

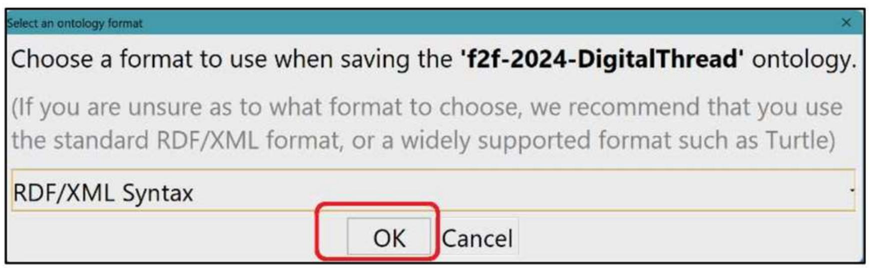


Figure 3: Selecting an Ontology Format

Import Ontology File

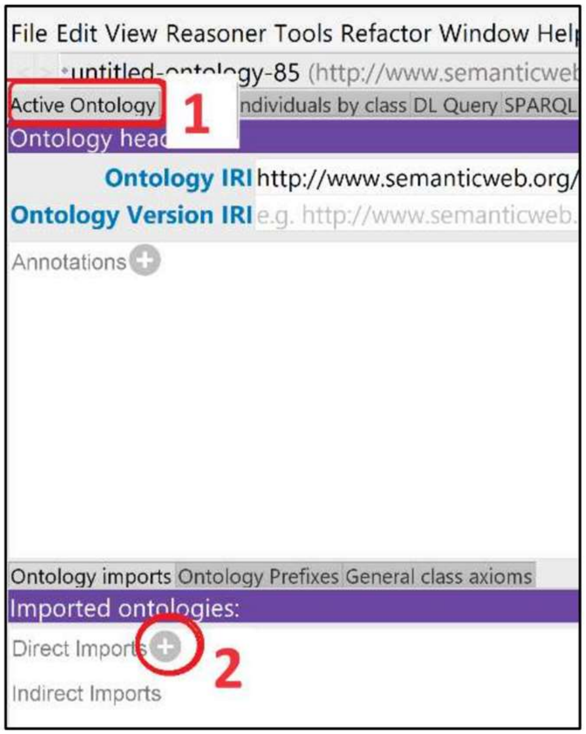


Figure 5: Importing Ontology File.

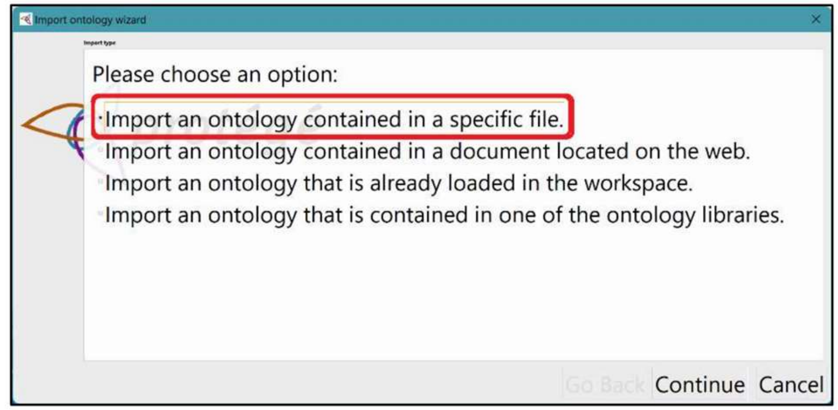


Figure 6: Import Ontology Wizard.

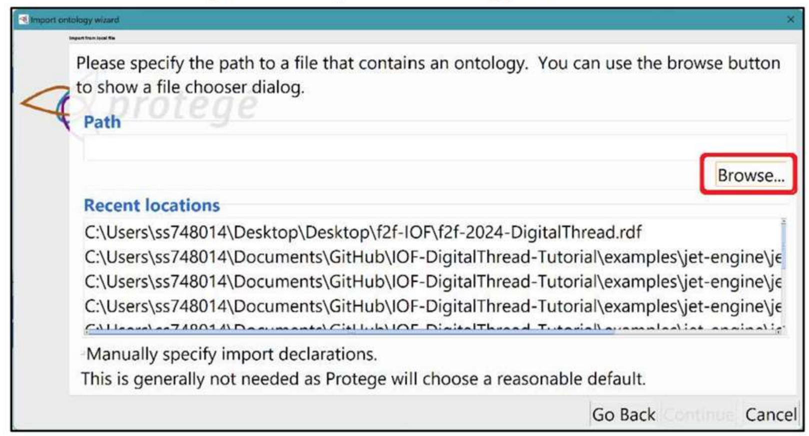


Figure 7: Browsing File to Import.

Import Ontology File

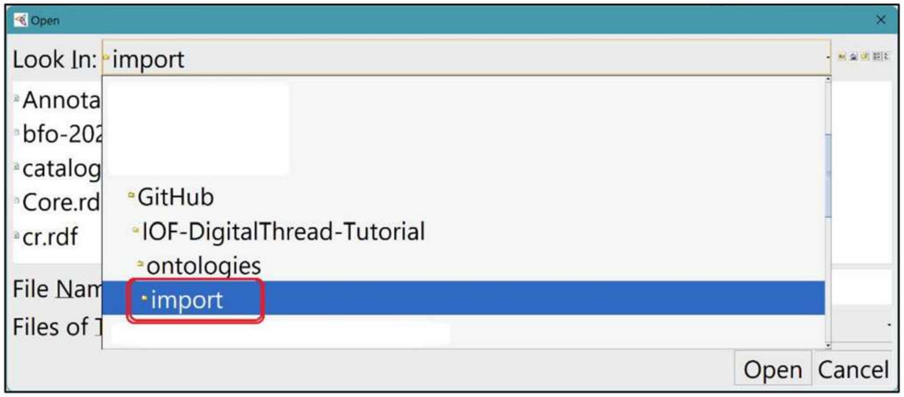


Figure 8: Locating Ontology File for Importing.

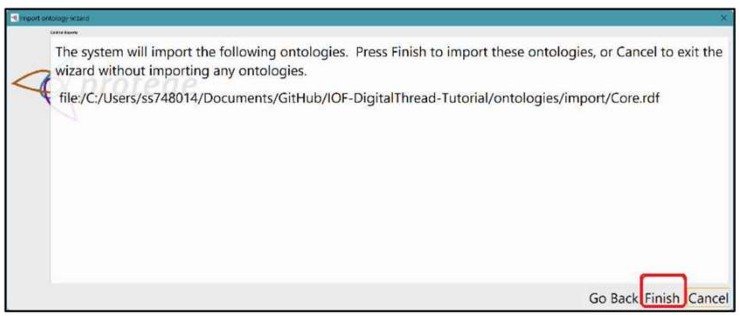


Figure 10: Import Ontology Wizard (3).

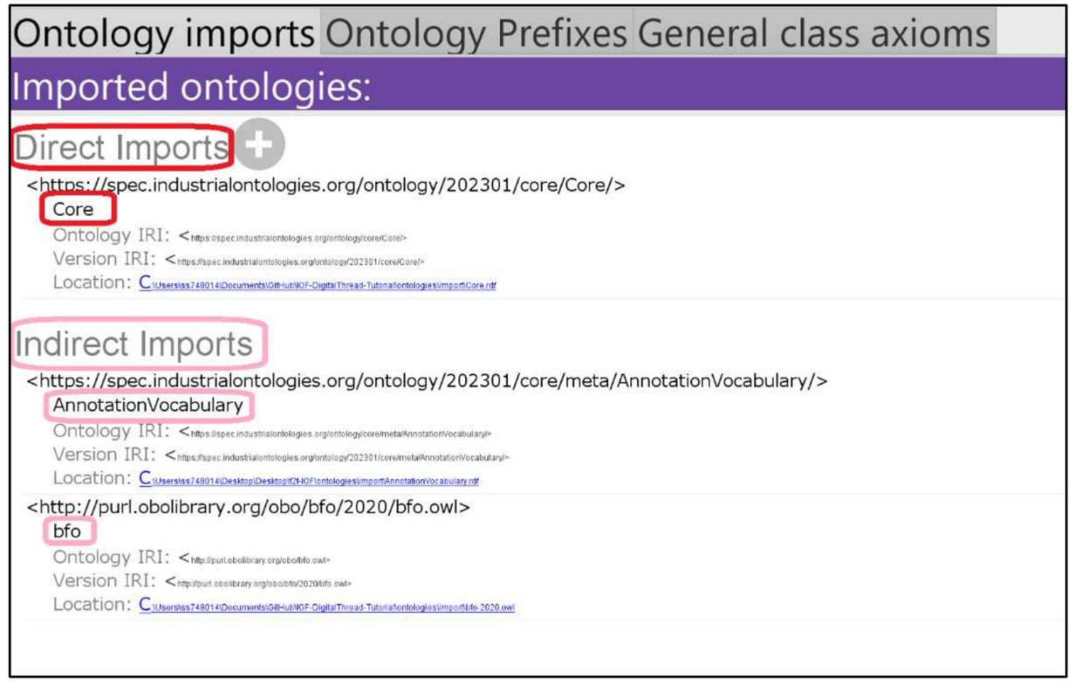


Figure 11: Imported Ontology View (Direct and Indirect Imports).

Create Classes

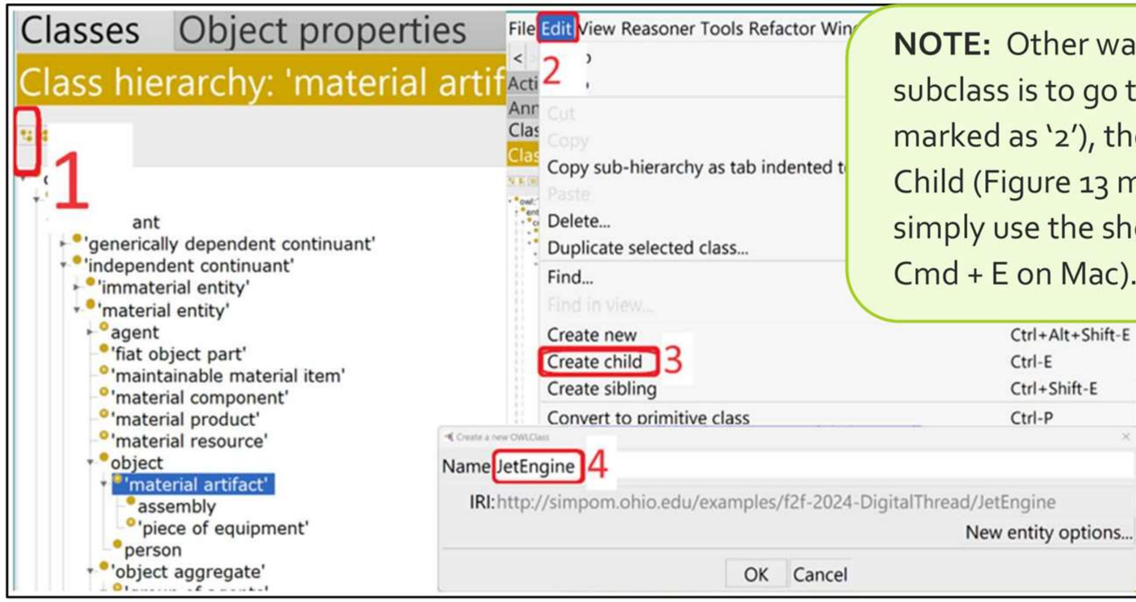


Figure 13: Adding Class.

NOTE: Other way to add a subclass is to go to Edit (Figure 13 marked as '2'), then select Create Child (Figure 13 marked as '3'), or simply use the shortcut Ctrl + E (or Cmd + E on Mac).

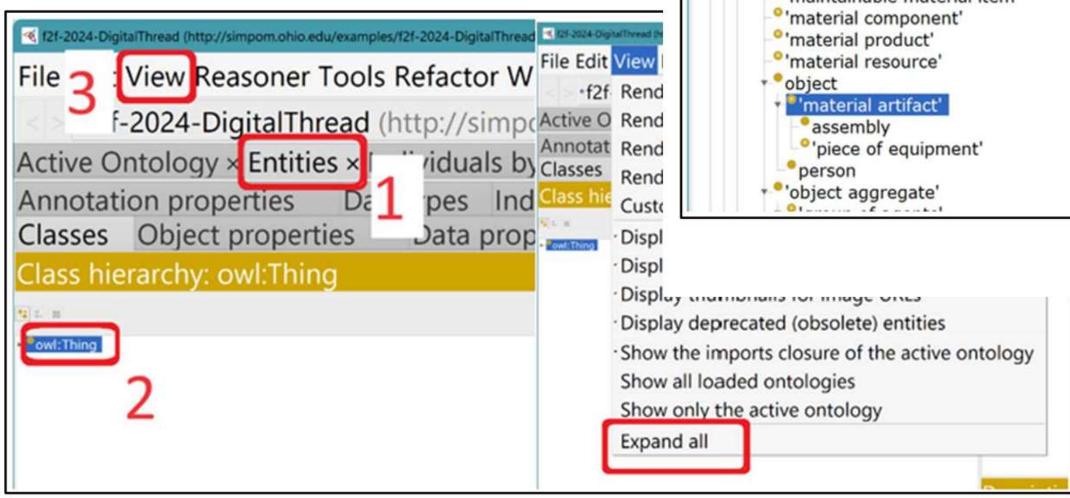


Figure 12: View All Imported Classes.

Create Individuals

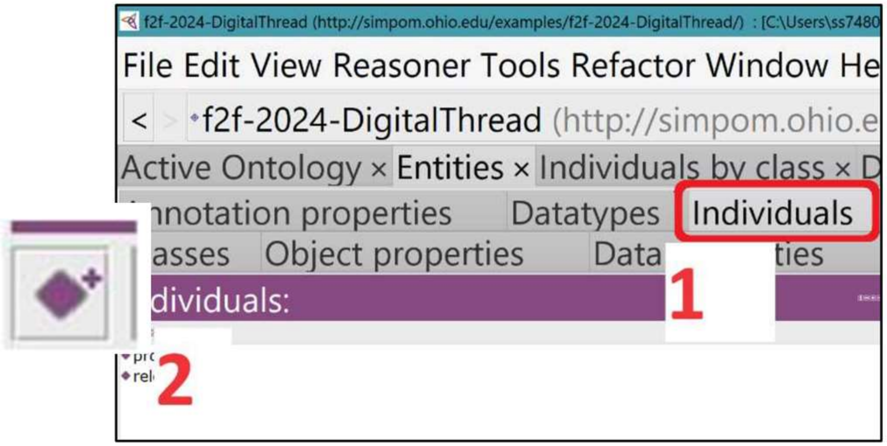


Figure 14: Create an Individual (Instance).

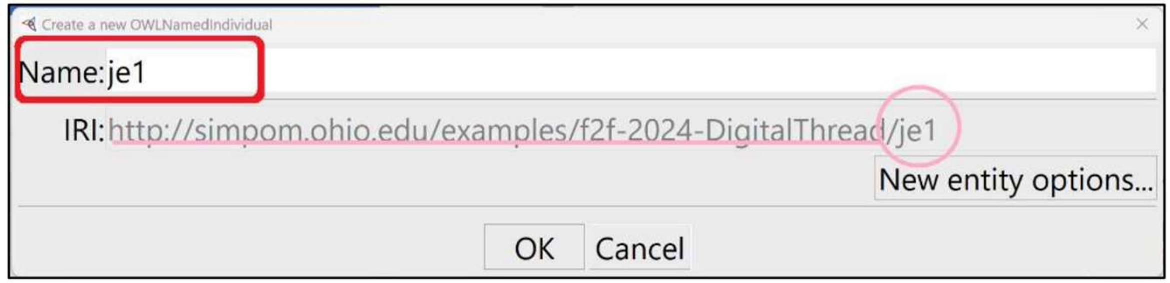


Figure 15: Create a new OWLNamedIndividual.

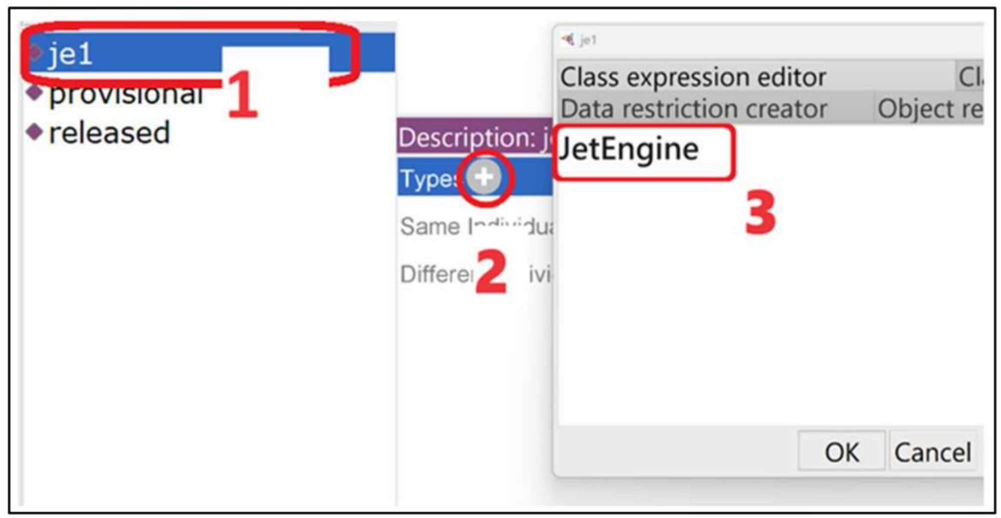


Figure 16: Class Assignment for an Individual.

Object Property Assertions

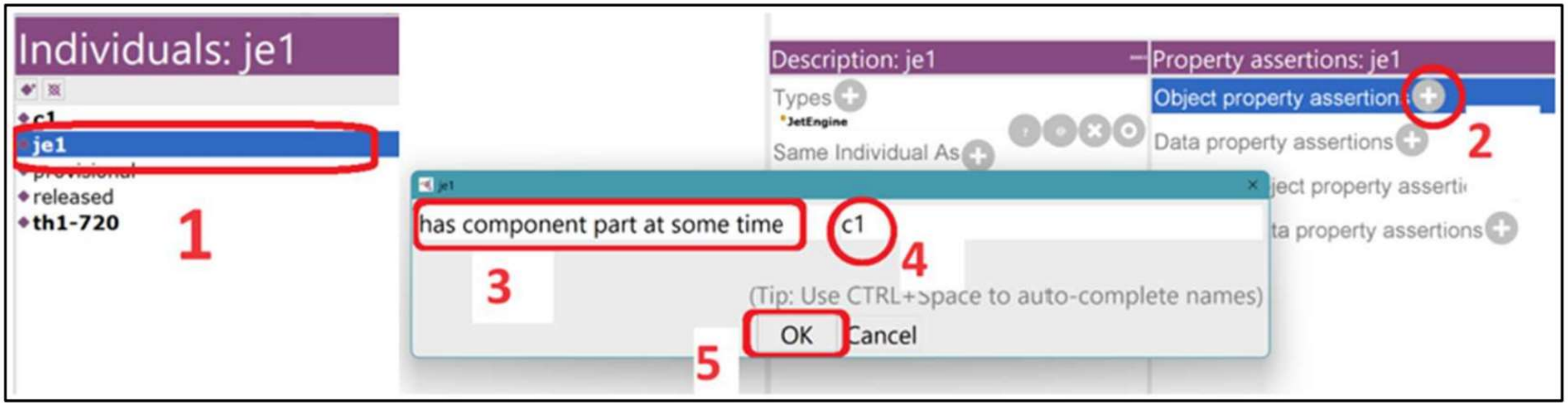


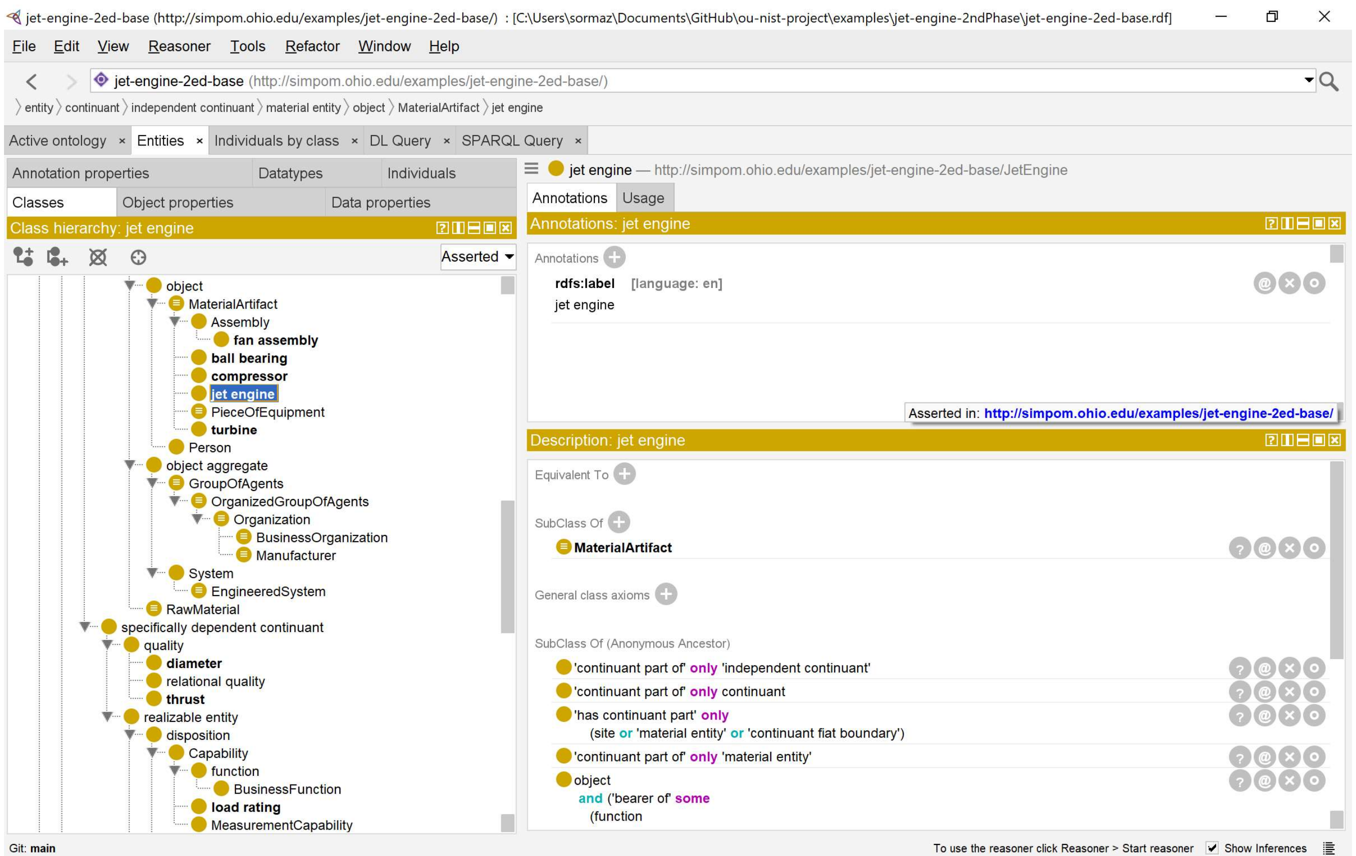
Figure 17: Object Property Assertions.

Practice on Protégé! (test?)

1. While in Protégé, navigate to File (located at the top left corner of the application) and select Open... Note** If there is a dialog box called "Open in current window" appear, select "No" for this scenario.
2. Navigate to the tutorial folder and locate the file named **"jet-engine-CR-forPractice.rdf"**
   a) For this example, it is found in → IOF-DigitalThread-Tutorial-main (this is the folder extracted from the ZIP file downloaded from GitHub) → IOF-DigitalThread-Tutorial-main → examples → jet-engine
   b) You can either double-click on the file "jet-engine-CR-forPractice.rdf" or select the file, then click "Open".
3. Assignments (refer back to sections 1.4 and 1.5 if needed for guidance):
   a) Add an individual named "c-DS1" (for compressor's design specification)
   b) Assign the Types in the Description View to c-DS1 as DesignSpecification, which is a class from the IOF Core ontology. Note** c-DS1 is an acronym for compressor's design specification number 1.
   c) Add multiple Object property assertions:
      i. prescribesProducedEntity c1
      ii. prescribesProducedEntity c2
      iii. prescribesDesignedEntity c-d1

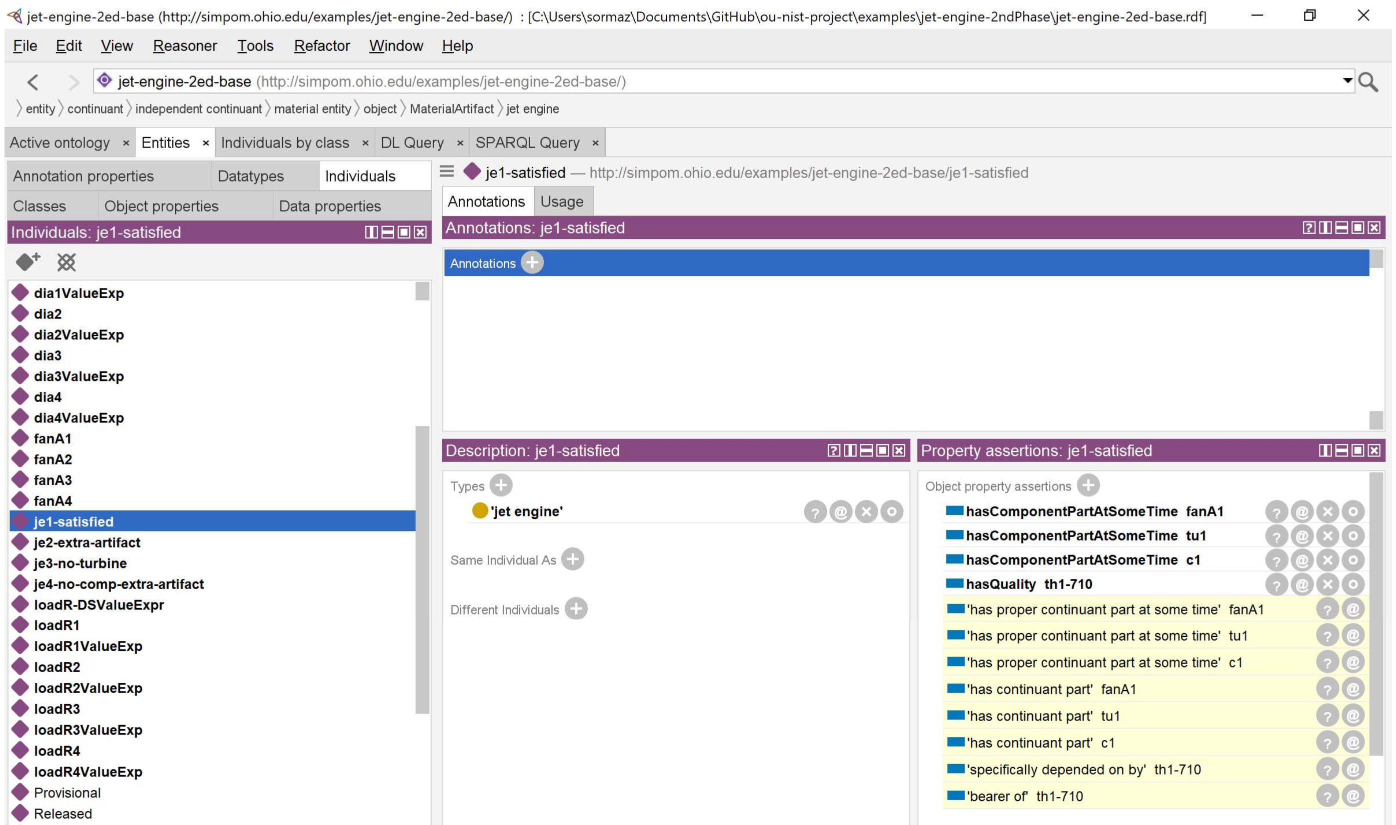
Note** the properties CR:prescribesProducedEntity and CR:prescribesDesignedEntity are part of new Object properties currently under proposal.

Build Ontology in Protégé (Ex. Jet Engine phase 2)



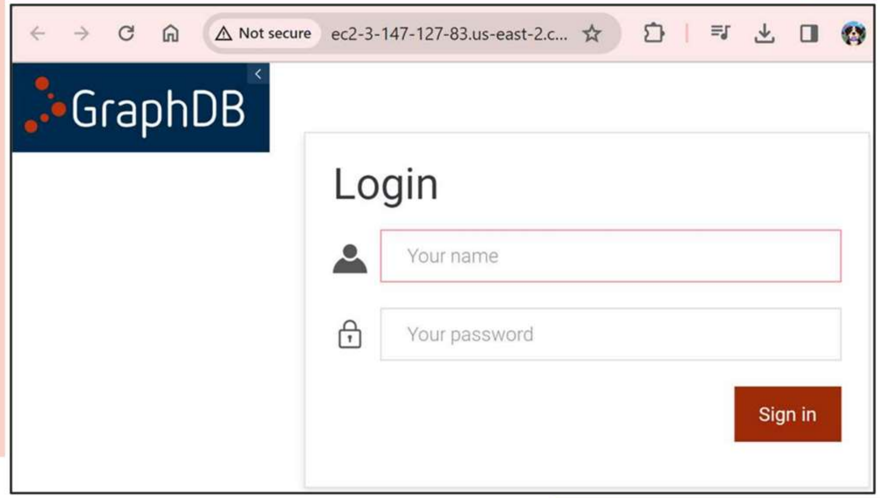
Populate Instance Data in Protégé (Phase 2)

Using Reasoner



GraphDB

Log In to the GraphDB



2.2 *Log In to the GraphDB*

1. To access the remote location provided by OntoText, connect using the following details:
   a. **URL**: http://ec2-3-147-127-83.us-east-2.com@rPY8Hi-kMN9A70*Fpute.amazonaws.com/
   b. **Shortened URL:** https://bit.ly/iof2402
   c. This GraphDB port availability is thru 2024-07-09.
   d. Username: **<last name>** (all in lower cases)
   e. Password: **<first name>** (all in lower cases)
2. Ensure to verify the URLs and credentials provided for accuracy and security compliance before attempting to connect (Figure 19).

Setup a Repo & Import Data

2.3 Setup a Repository and Import Data

1. Login to GraphDB (session 2.2 procedure 1 Log In to the GraphDB).
2. In the Main Menu as shown in Figure 20, click on **"Setup"** (labeled as 'E'), then choose **"Repositories"**.
3. From the main interface select the option to **"Create new repository"** 

4. Choose **"GraphDB Repository"** as the type of repository for this tutorial. 

5. Enter a name for your repository. It's recommended to use a *combination of your initials or unique letters and/or numbers,* followed by **"-DigitalThread-CR",** to make it easily identifiable.
6. When setting up Inference and Validation under the Ruleset section, select **"OWL2-QL (Optimized)"** which is recommended for the purposes of this tutorial.
7. Then at the bottom of the page click **'Create'** 

. This will finalize the creation of your new repository configured specifically for this tutorial.

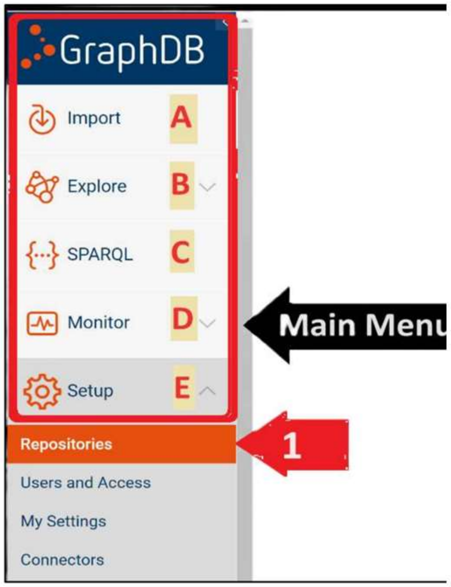


Figure 20: GraphDB Main Menu.

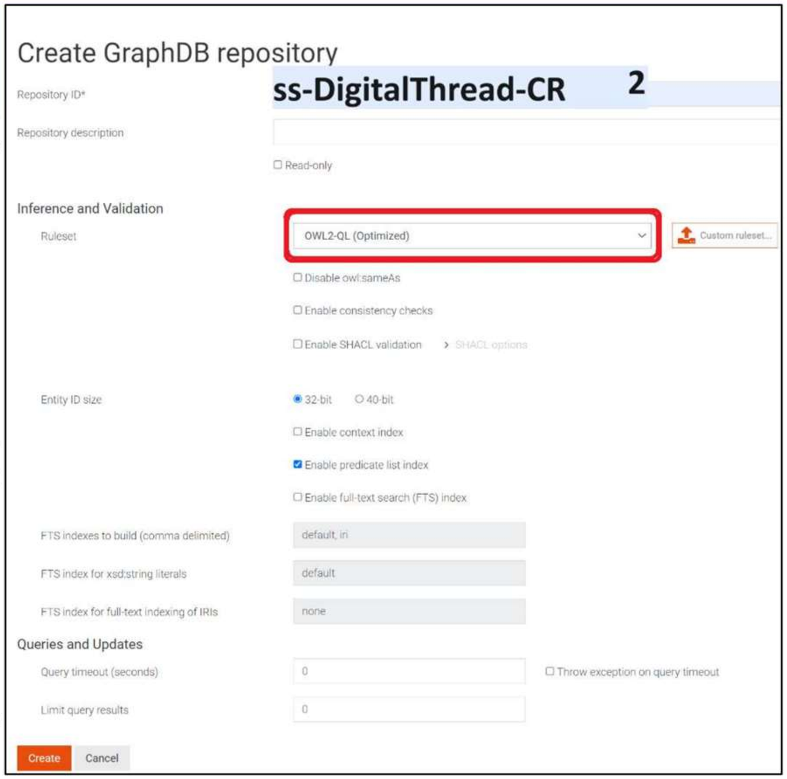


Figure 21: Create GraphDB Repository.

Setup a Repo & Import Data (2)

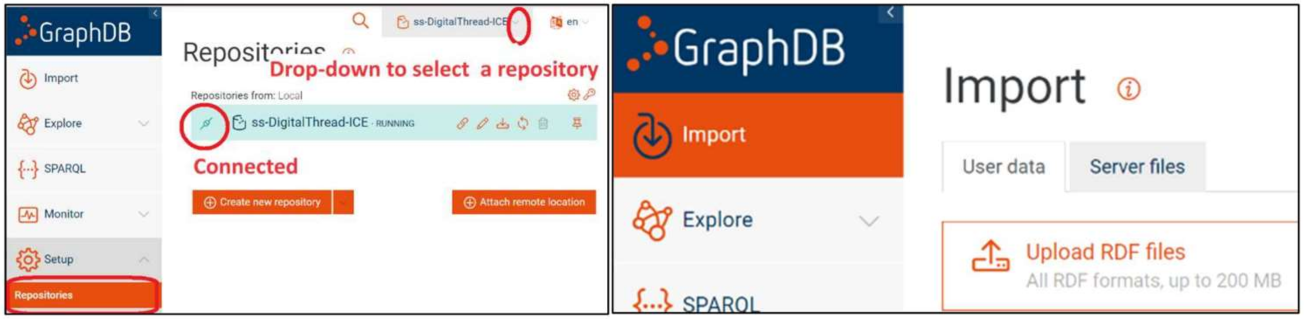


Figure 22: Repositories View (left) and Import View (right).

a. Path: *IOF-DigitalThread-Tutorial-main > ontologies > **import***
b. Import **ALL** the files in the import folder, including:
   i. Core.rdf
   ii. bfo-2020.owl
   iii. AnnotationVocabulary.rdf
   iv. cr.rdf.

a. Jet-engine-Base.rdf
b. jet-engine-cr.rdf

Figure 23: Import Ontology Files.

Explore Knowledge Graphs

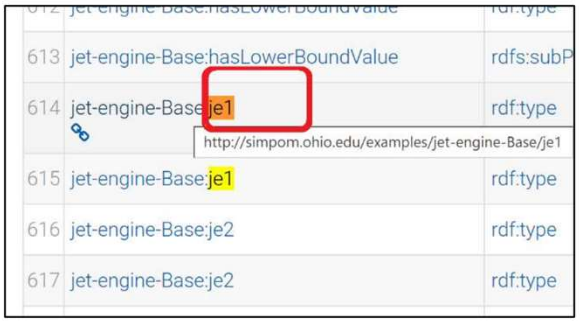


Figure 24: Locate a Specific Class or Individual.

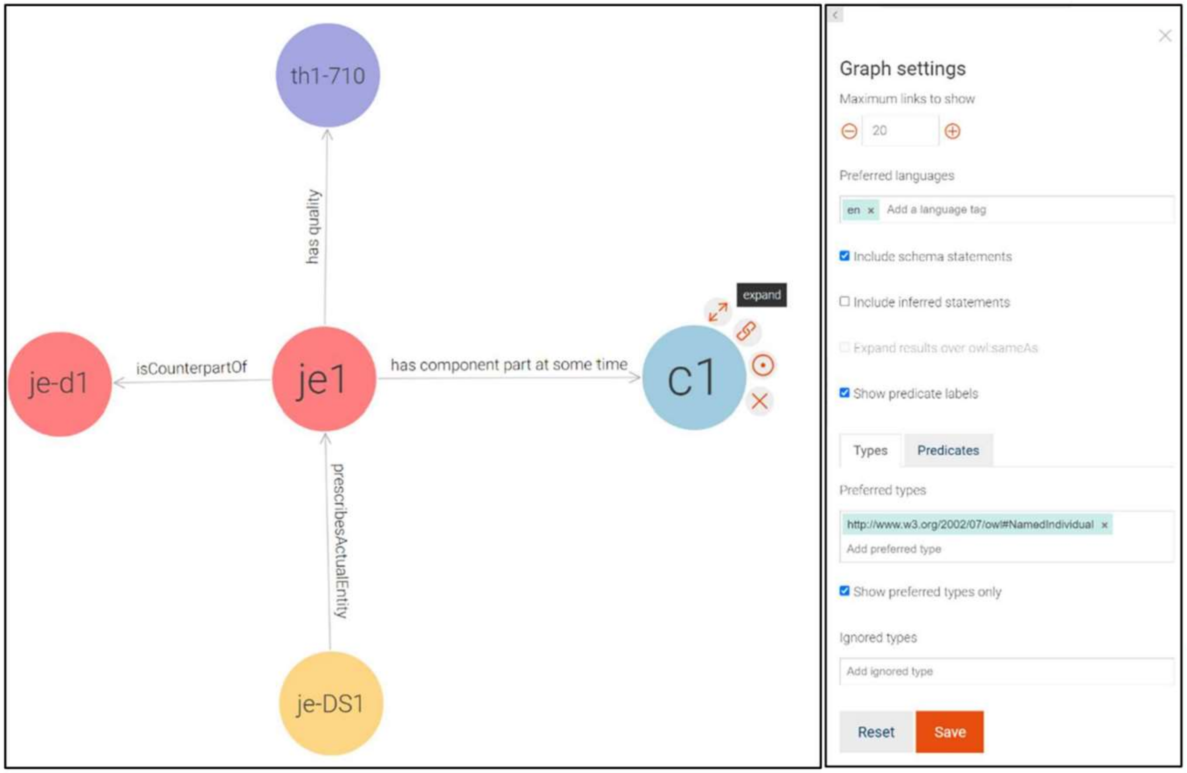


Figure 25: Adjusting Graph Settings.

SPARQL Query

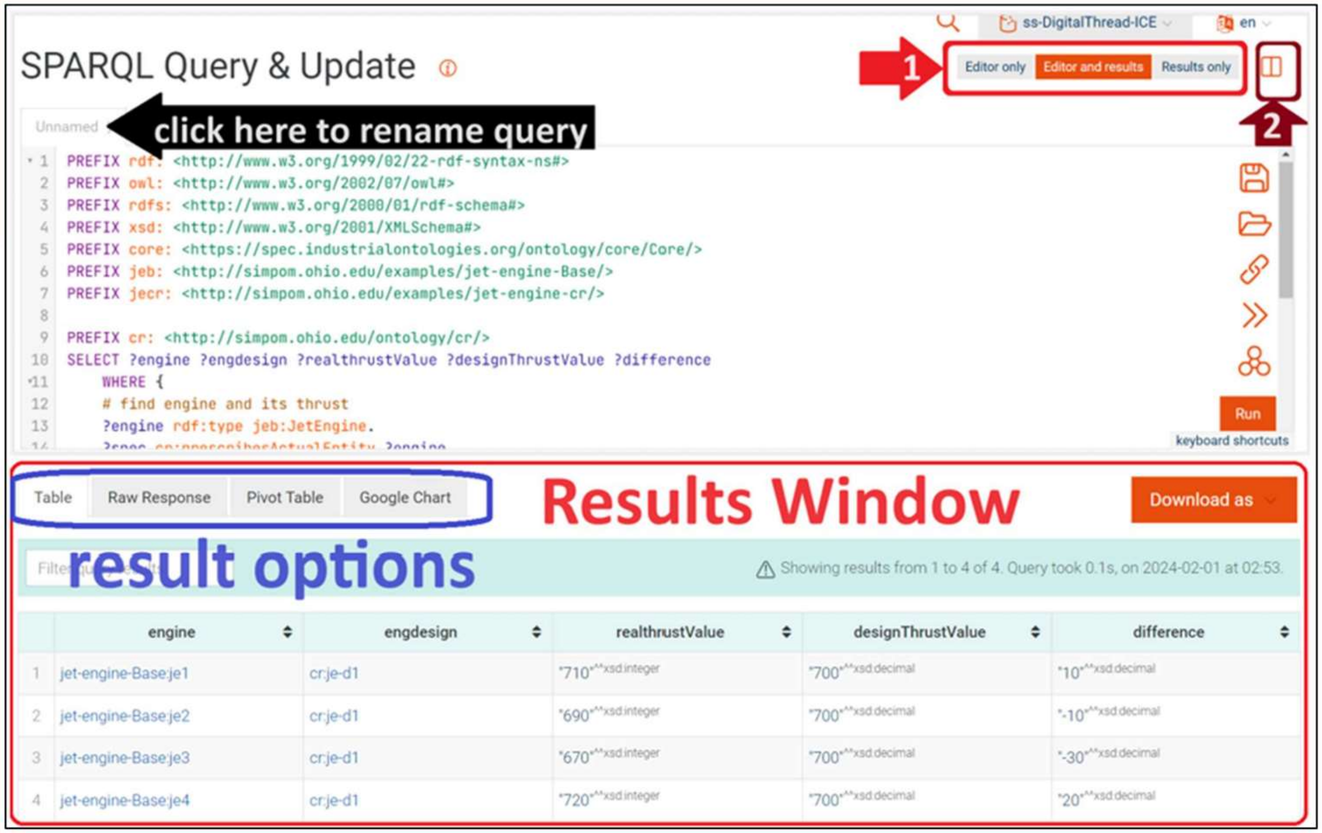


Figure 26: SPARQL Window View.

SPARQL with Pivot Table Results

Figure 27: Pivot Table (Available Variables).

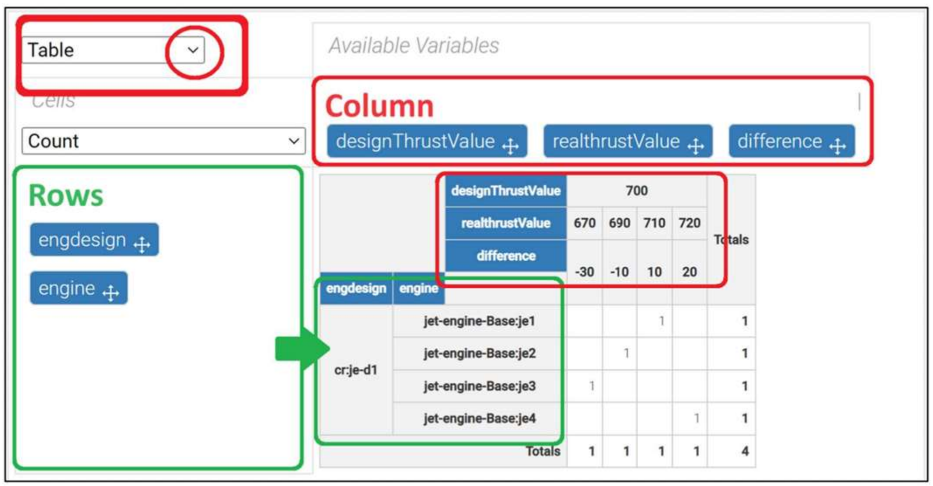


Figure 28: Pivot Table Details.

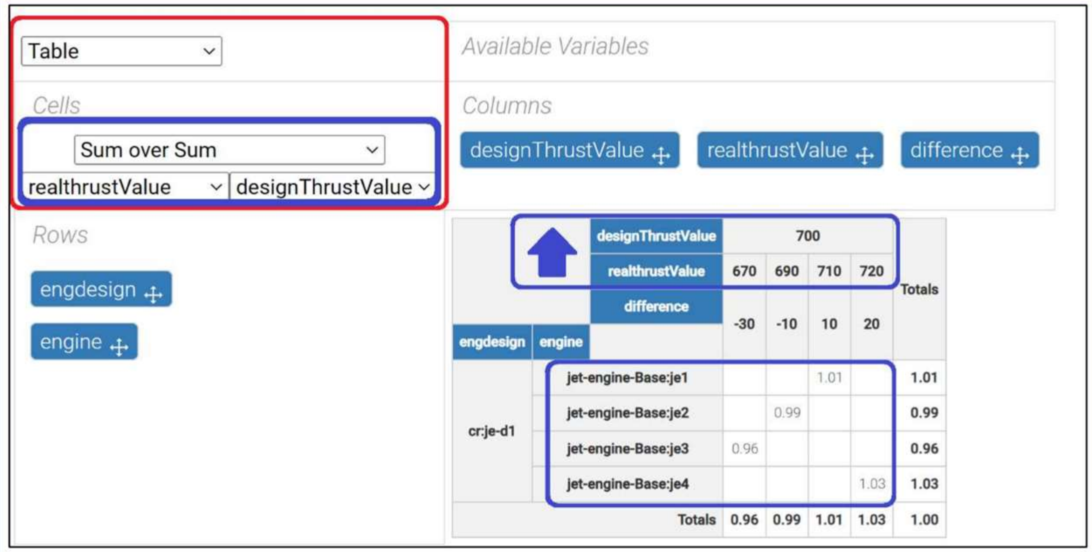


Figure 29: Using Pivot Table for Simple Analyses.

SPARQL with Google Chart Results

Figure 30: Google Chart Configurations.

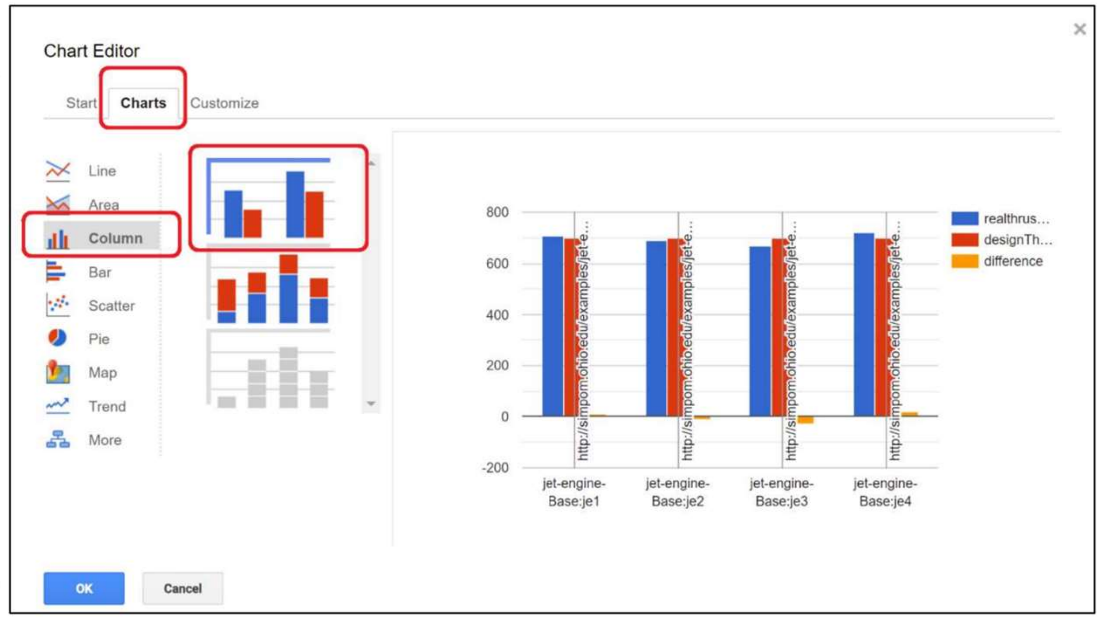


Figure 31: Google Chart Configurations (2).

GraphDB

1. **Create a New Repository:** Start by creating a new repository. Name it something relevant to ICE (Information Content Entity) to reflect the focus of this exercise.

2. **Import Files:** Import all necessary files into your newly created repository. Instead of importing jet-engine-CR.rdf, import jet-engine-ICE.rdf. This change targets a different aspect of the ontology, focusing on Information Content Entities.

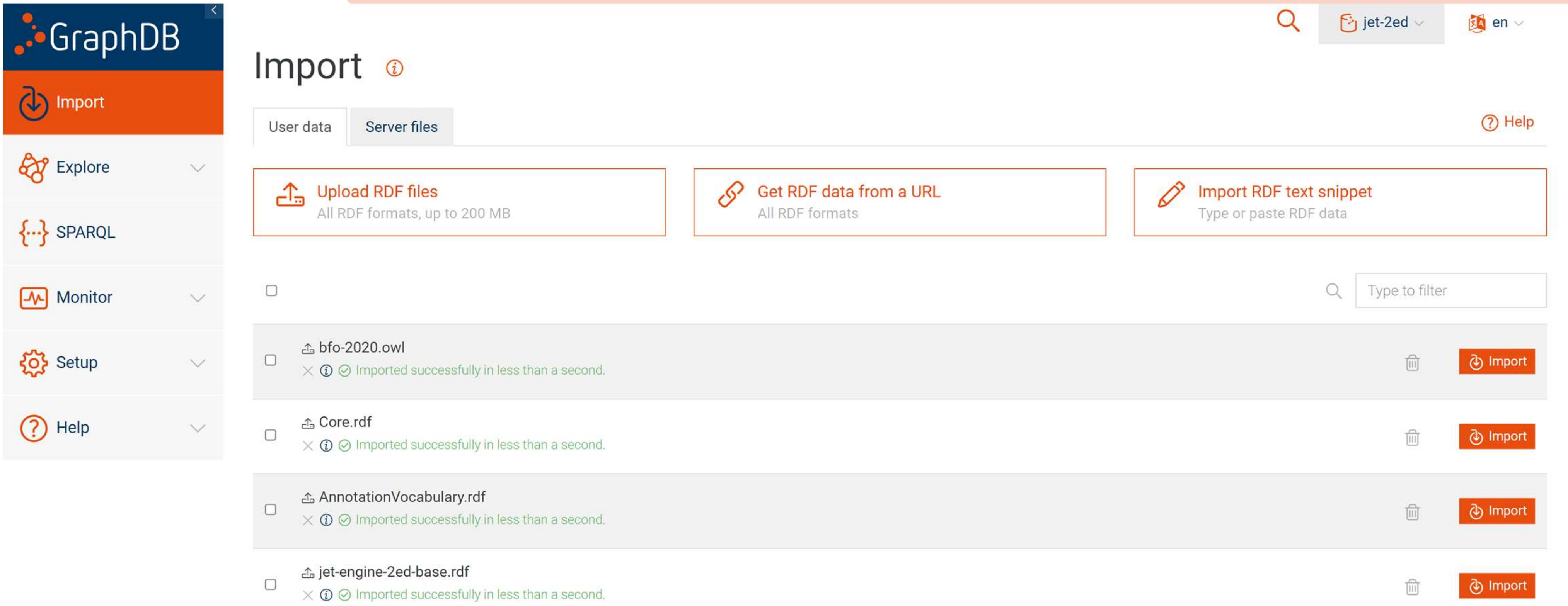
3. **Knowledge Graph Differences:** Once you've imported the files, explore the Knowledge Graph. Observe the differences compared to the CR (Counterpart Relation) model. Consider how Information Content Entities are represented and related within this new context.

4. **Edit SPARQL Queries:** Adjust your SPARQL queries to align with the new ontology. This may involve changing PREFIX definitions and other necessary components to ensure your queries are compatible with the ICE-focused ontology.

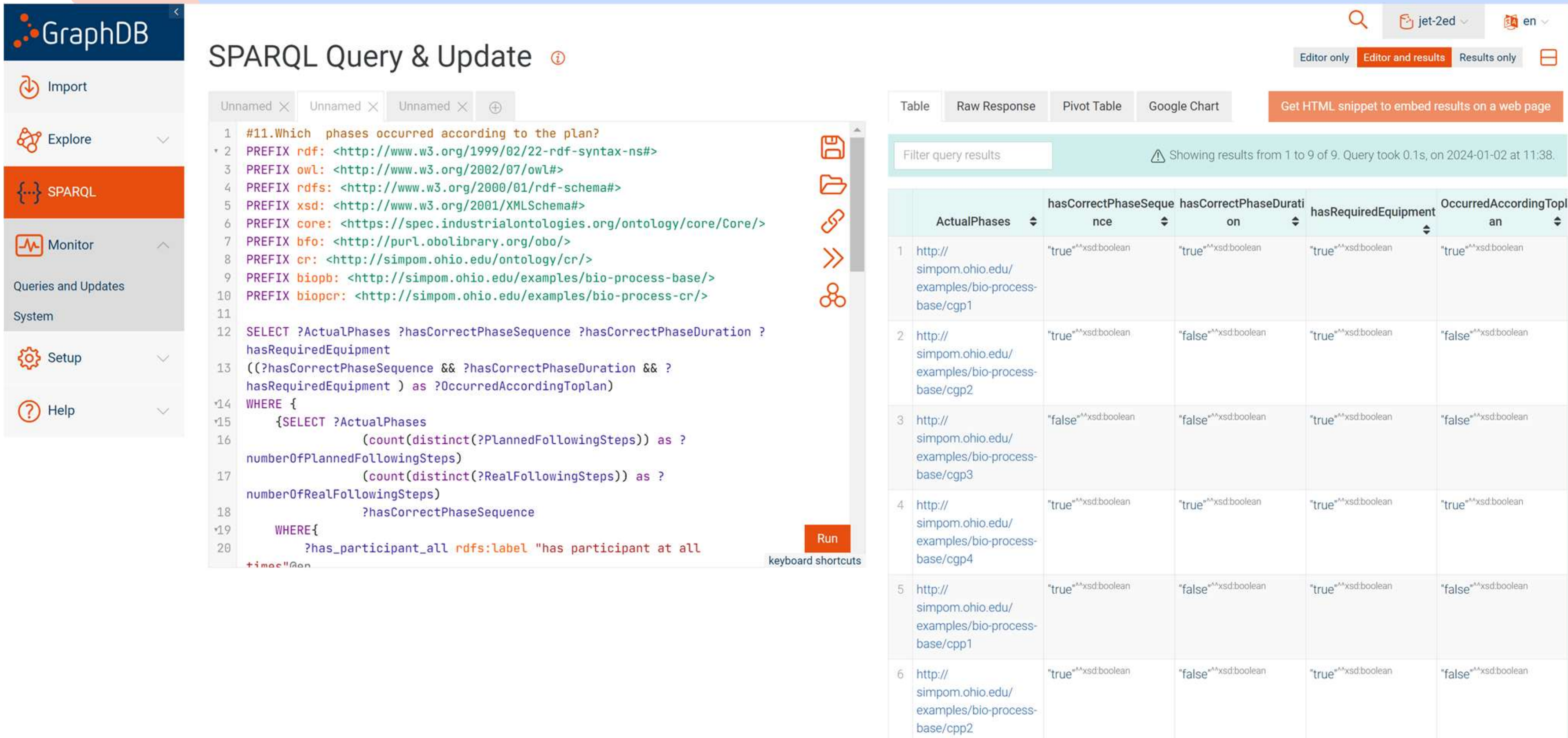
5. **Explore Result Functions:** With your updated SPARQL query, run and explore the different result functions available in GraphDB. Pay attention to how the results differ from those related to the CR ontology and what insights they might offer regarding Information Content Entities.

GraphDB

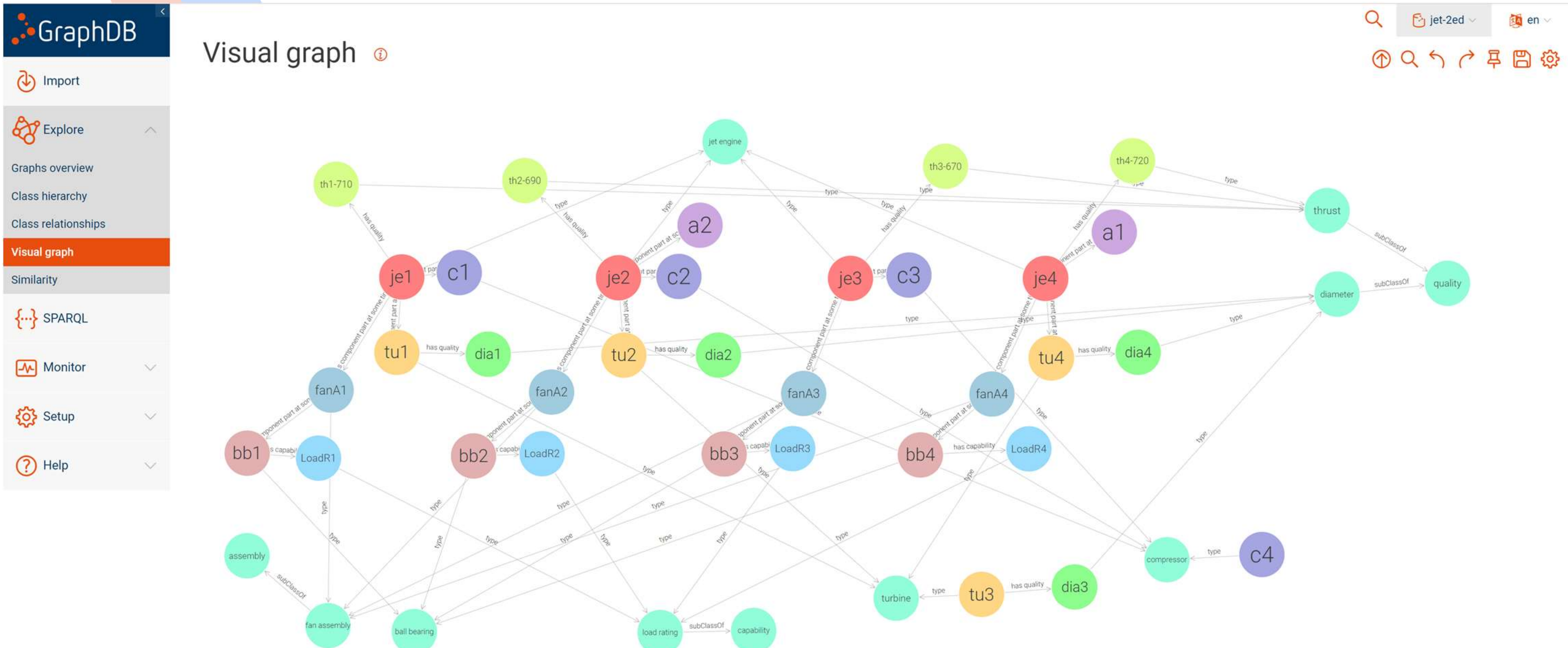
GraphDB Repository Setup



GraphDB SPARQL Editor and Results



GraphDB Explore Repository



Importing Data into GraphDB

- We can import the data in CSV format
- Ontotext refine is tool to import data
- Procedure
  - Inspect CSV file and underlying ontology
  - Design the mapping from CSV columns into ontology
  - Define mapping visual editor
  - Refine mapping SPAQRL in text editor
  - Execute the SPARQL query
  - Export the RDF file

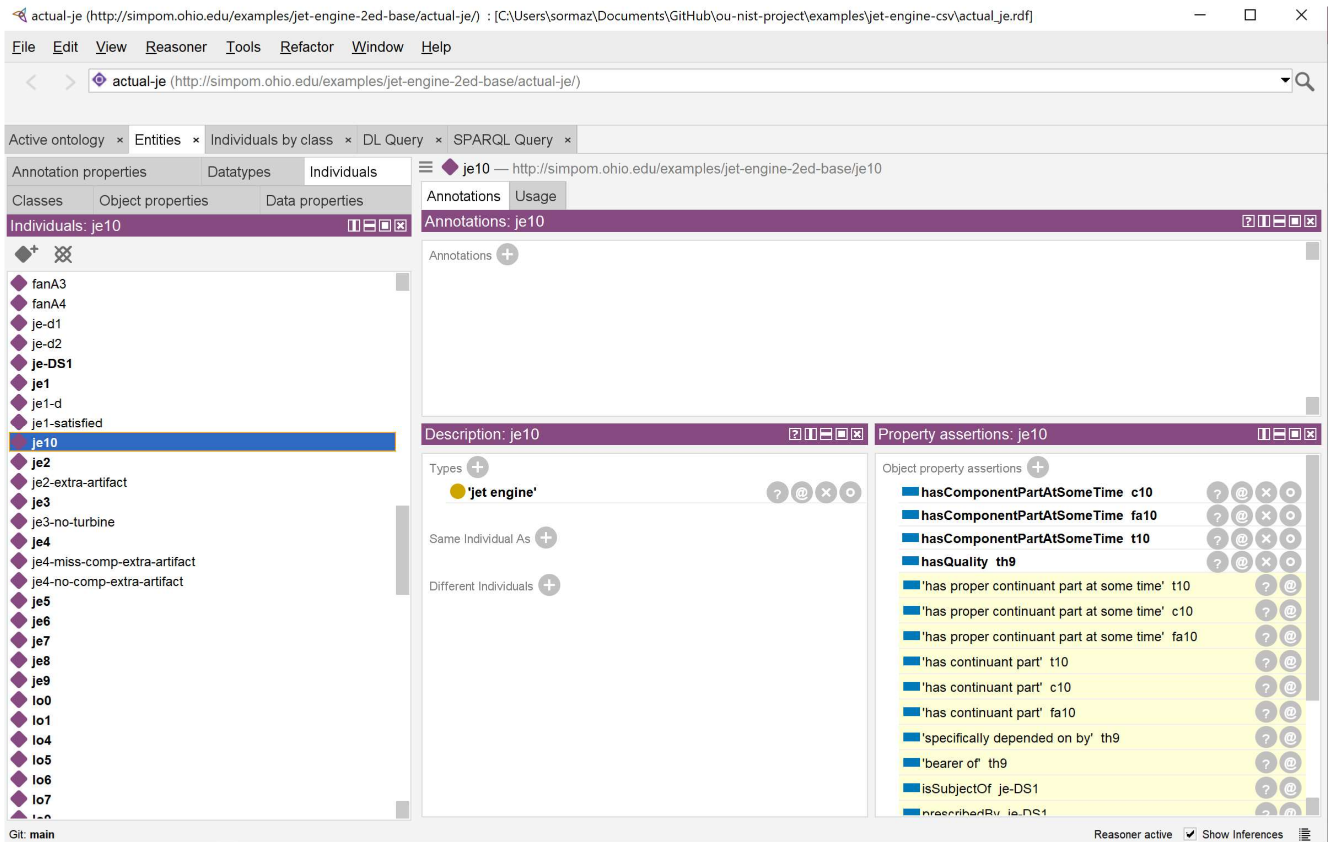
Jet Engine Sample Data

Simulated data about 10 engines

- The example has data with missing values

	A	B	C	D	E	F	G	H
1	engine	compresso	turbine	fan Assy	Ball bearin	load	diameter	thrust
2	je1	c1	t1	fa1	bb1	0.915	47.9285	695.5
3	je2	c2	t2	fa2	bb2	0.903	48.0813	697.7
4	je3		t3	fa3			47.9494	696.0
5	je4	c4	t4				48.0423	702.3
6	je5	c5		fa5	bb5	0.915		692.8
7	je6	c6	t6	fa6	bb6	0.904	47.9745	707.9
8	je7	c7	t7	fa7	bb7	0.916	48.0051	708.1
9	je8			fa8	bb8	0.911		699.5
10	je9	c9	t9	fa9			48.0730	705.9
11	je10	c10	t10	fa10	bb10	0.914	48.0339	700.2

Mapped Example Data in Protege



Acknowledgements

Funded by NIST through Ohio University cooperation agreement #70NANB22H155

NIST | NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Thank you