

IOF for Digital Thread Tutorial

Organizer: Dušan Šormaz¹

Presenters: Arkopaul Sarkar² and Saruda Seeharit¹

¹Industrial and Systems Engineering Department, Ohio University, Athens, OH, USA

²The National Engineering School of Tarbes (ENIT), Tarbes, France

Prepared by IOF Production Planning and Scheduling Working Group



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

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

Practical skills for developing relations between digital artifacts and produced artifacts

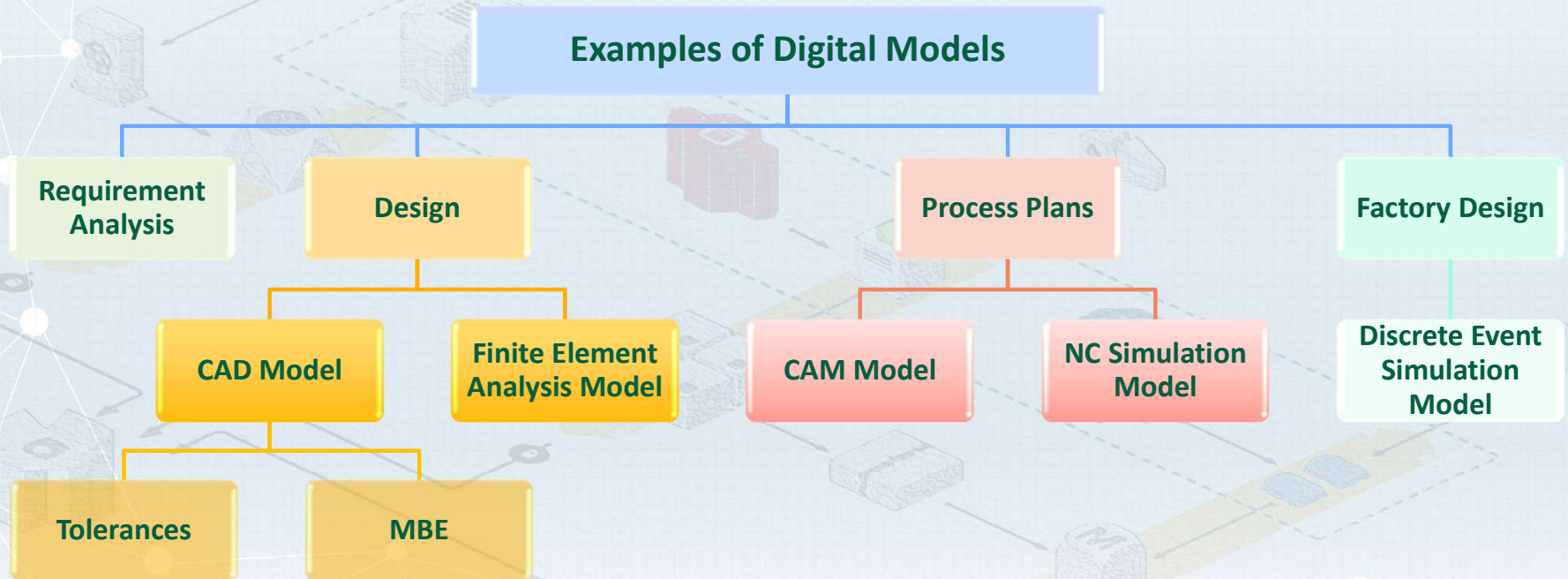
What are the expected outcomes?

Overview of Digital Thread

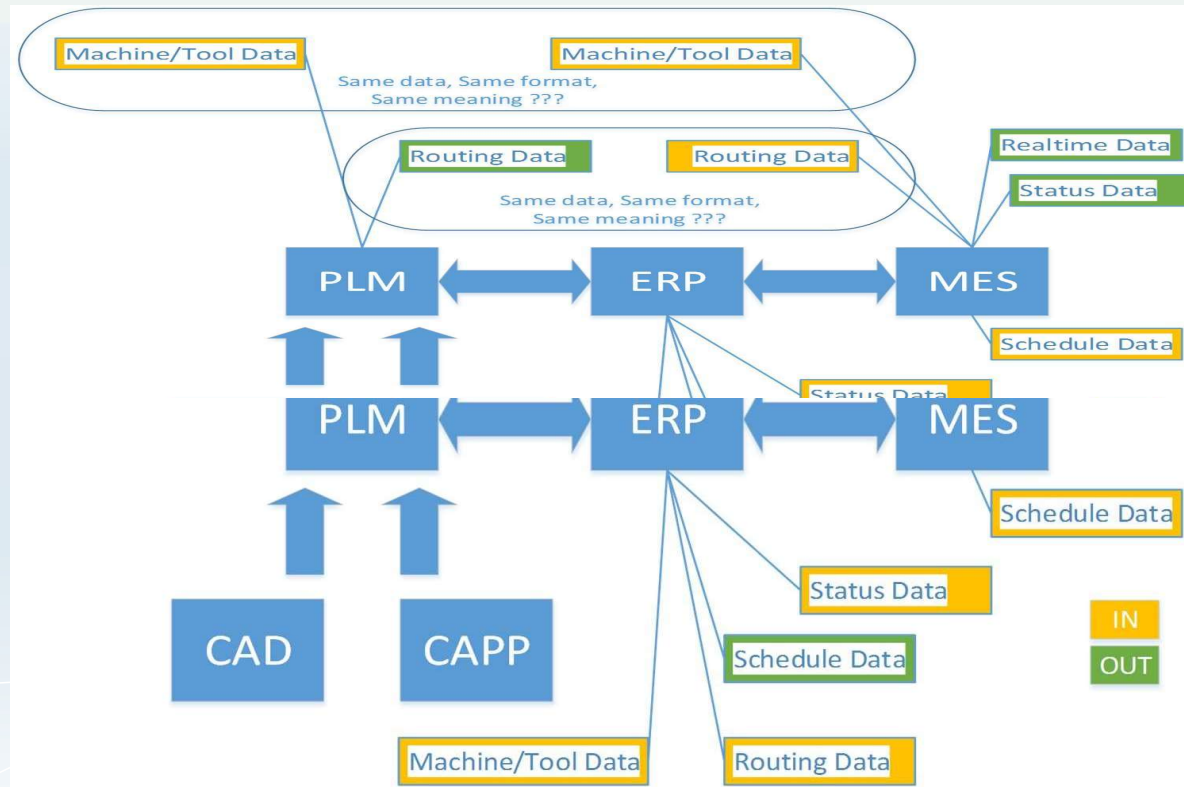
Digital Model, Digital Shadow, Digital Twin
IOF and Approaches



Examples of Digital Models (Artifacts)

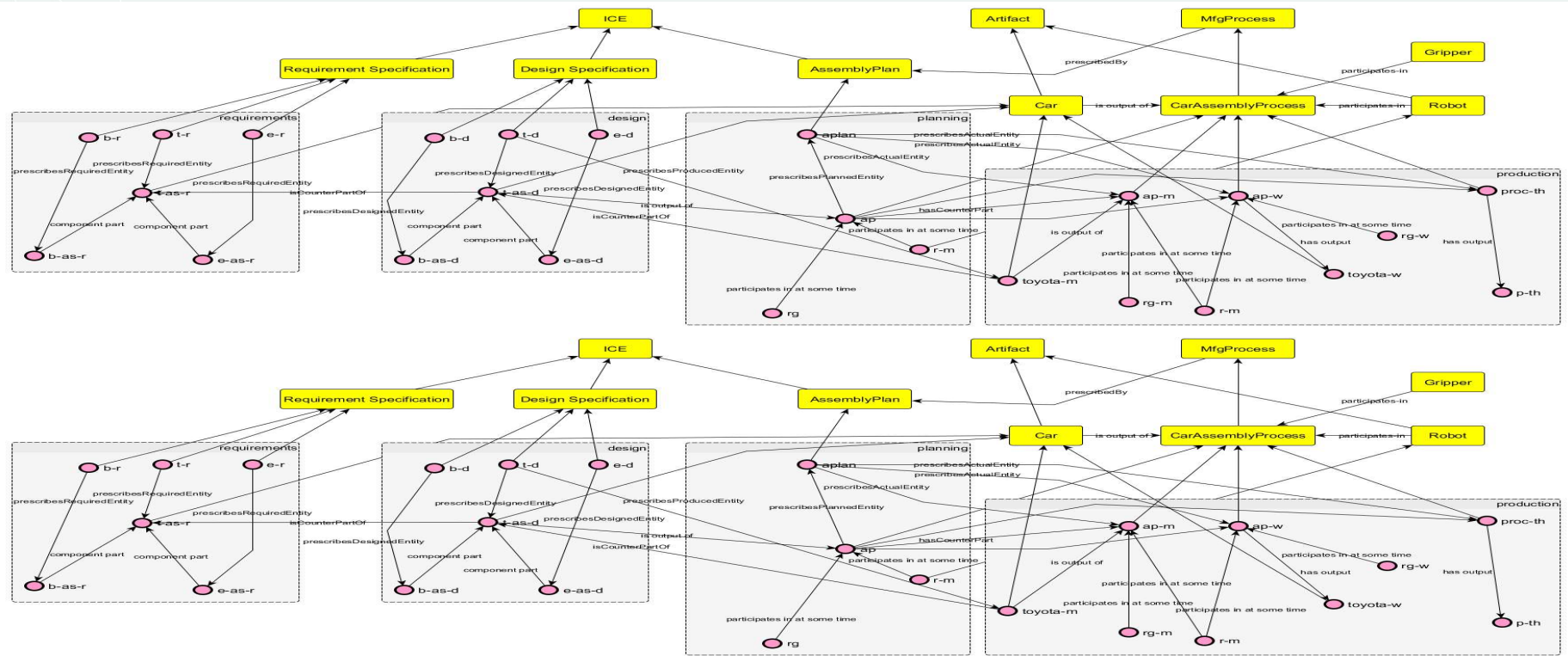


Information Interoperability



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

Information
Content Entity

ICE

CR

Counterpart
Relation

Use Cases in the Tutorial

Jet Engine and Biomanufacturing Process



Jet Engine Design and Requirement Verification

An engineering task:

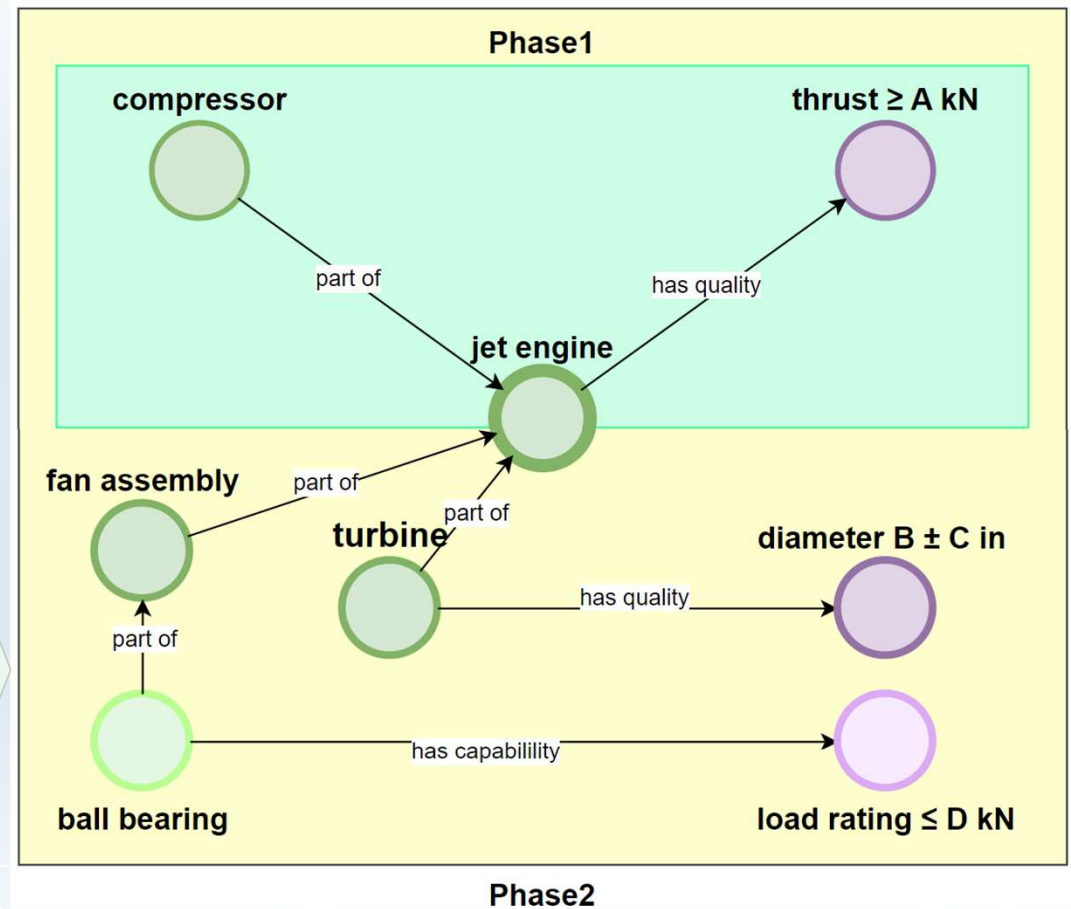
“There is a need to design and produce a jet engine that will have a compressor as its part, and it will be able to produce a minimal thrust of 700 kN”.

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?

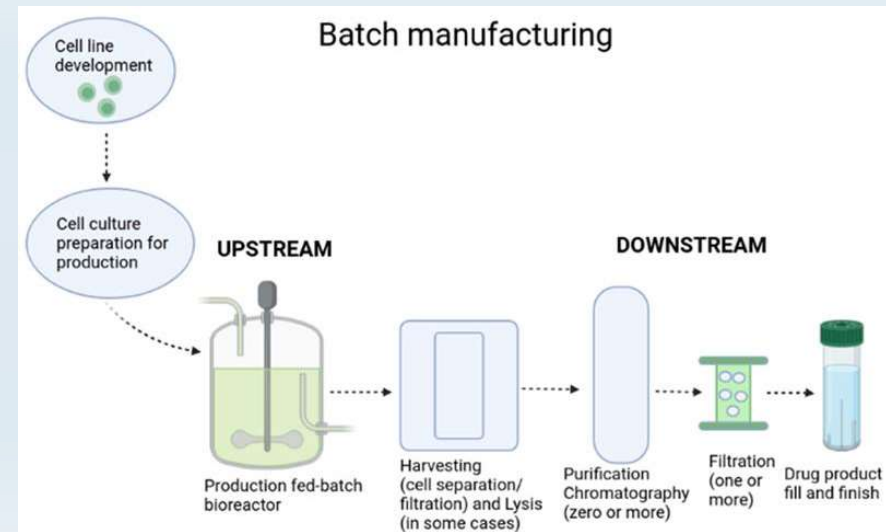
Does a turbine diameter of real engine have value as specified in the design?

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?

A. Which equipment are used in a (real) process?

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

C. What is the concentration of a drug substance (or unwanted substance) after a unit operation O1 compared to the planned values?

D. What is the concentration of a drug substance (or unwanted substance) after a unit operation O1 compared to the planned values within the control strategy? Is the difference within the plan?

E. What is the concentration of a drug substance (or unwanted substance) after a unit operation O1 compared to the planned values within the control strategy? Is the difference within the plan?

F. What is the concentration of a drug substance (or unwanted substance) after a unit operation O1 compared to the planned values within the control strategy? Is the difference within the plan?

G. Was the pH within limits for the entire process?

F. What is the concentration of a drug substance (or unwanted substance) after a unit operation O1 compared to the planned values within the control strategy? Is the difference within the plan?

G. Was the pH within limits for the entire process?

G. Was the pH within limits for the entire 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



draw.io

Drawio Editor

The screenshot shows the Draw.io editor window titled "Process_EX_ver9.drawio". The interface includes a menu bar (File, Edit, View, Arrange, Extras, Help), a toolbar with various drawing tools, and a search bar for shapes. The main canvas displays a complex process diagram with nodes such as ICE, Material, Quality, Process, Bioreactor, and Bioreactor Unit Operation, connected by various relationships like "has quality", "part of", and "precedes".

Filename: drawioOU-NIST-DA-ver3(op)_LIBRARY.drawio

The shape library panel displays various shapes and their corresponding labels and relationships. The shapes include:

- BFO (pink rectangle)
- IOF (cyan rectangle)
- ICE (green rectangle)
- ICE (green circle)
- MaterialArtifacts (purple rectangle)
- BFO-class (pink circle)
- IOF-class (cyan circle)
- ICE-class (green circle)
- ICE-instance (green circle)
- MArtifact-class (purple circle)
- Artifact (pink circle)
- Quality (blue rectangle)
- Quality (blue circle)
- Process (orange rectangle)
- Process (orange circle)
- MArtifact-Insta... (purple circle)
- Quality-class (blue circle)
- Quality-instance (blue circle)
- Process-class (orange circle)
- Process-Instan... (orange circle)
- dataType (green parallelogram)

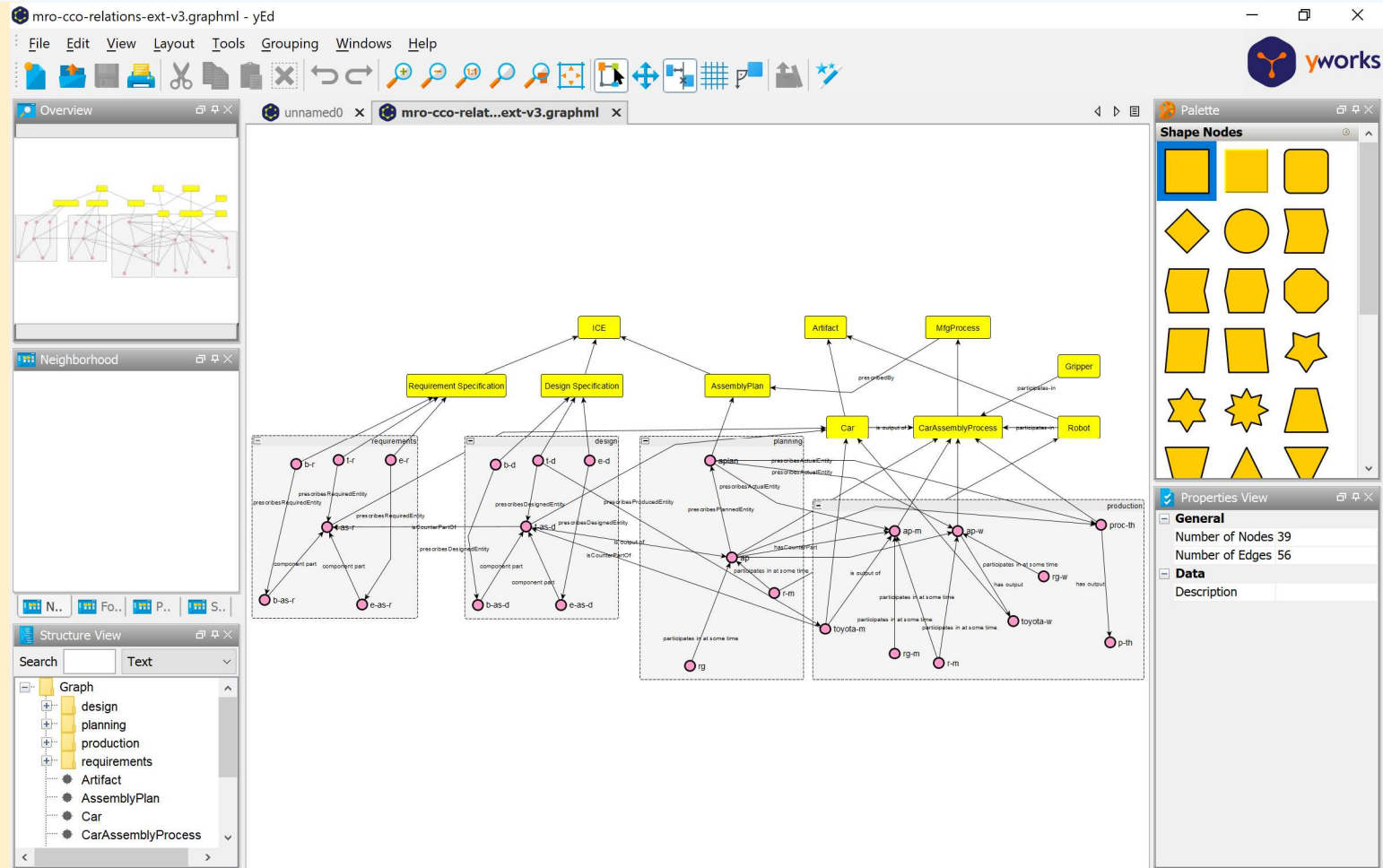
Relationships shown include:

- is about (dashed arrow)
- is a (solid arrow)
- instance (dotted arrow)
- relationship (green arrow)

+ More Shapes



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



Use Case Description

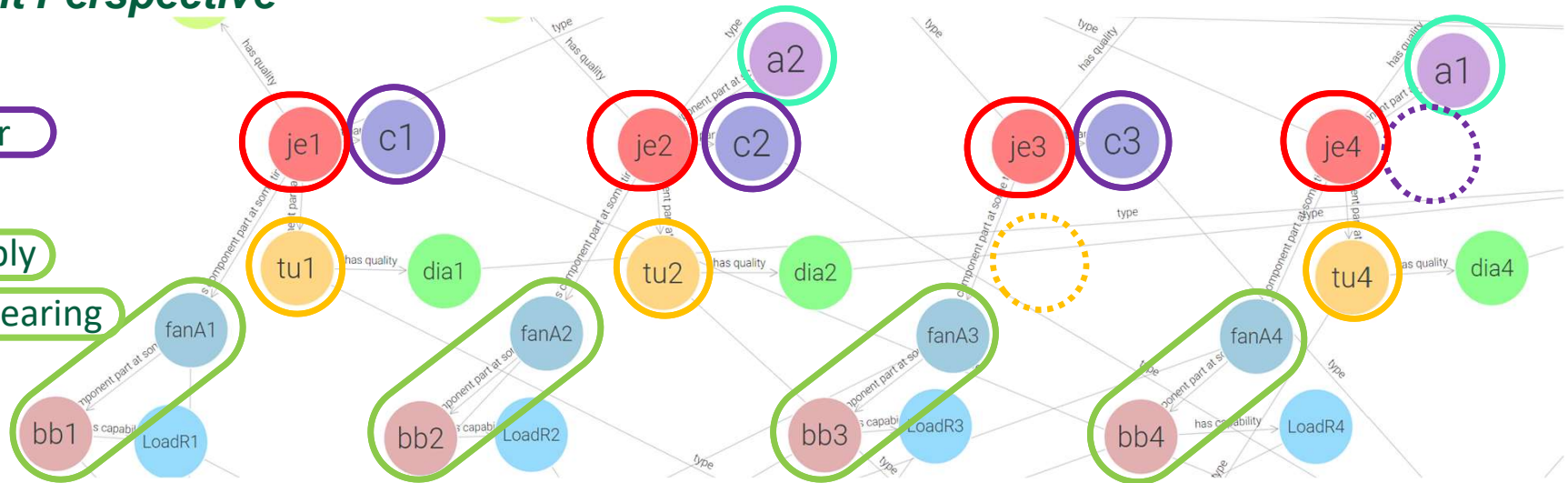
Component Perspective

Jet Engine

 Compressor

 Turbine

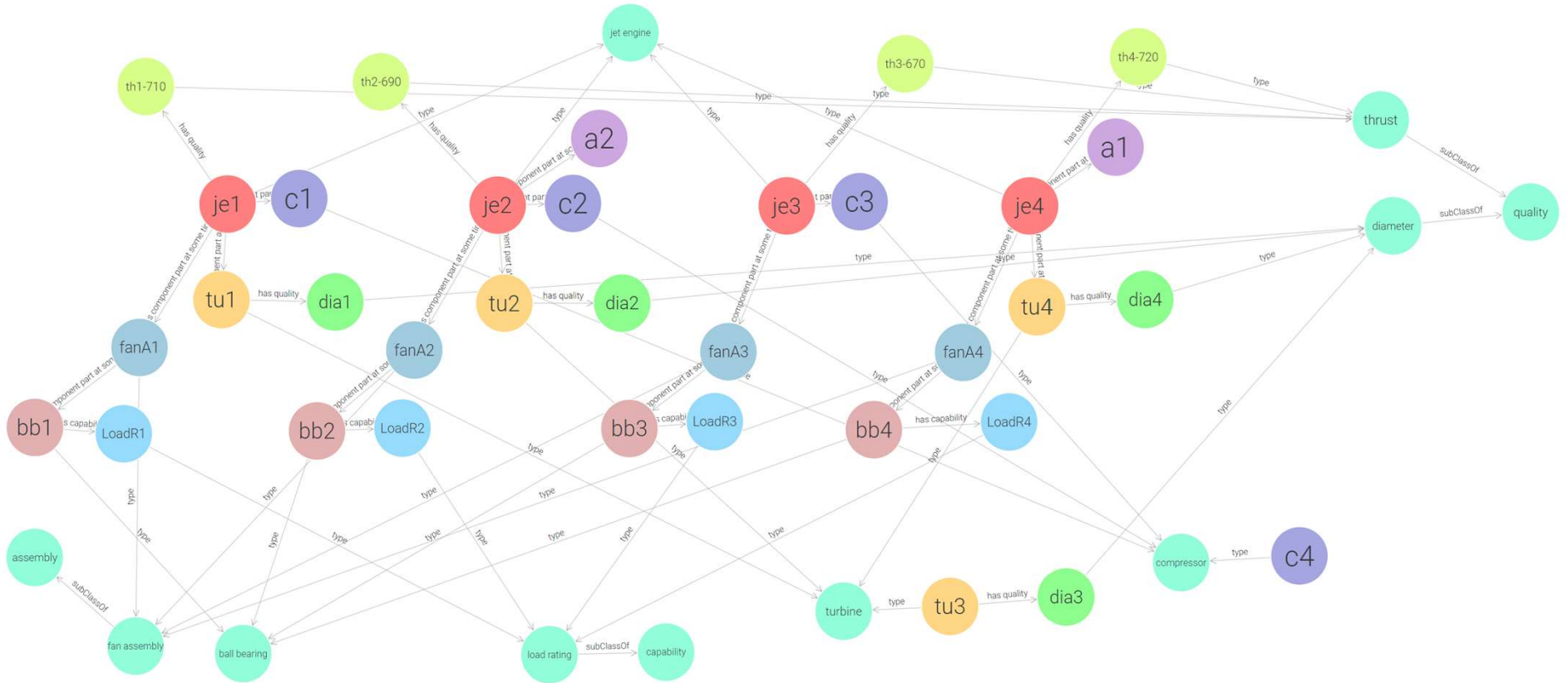
 Fan Assembly

 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.
```

```
?jetEngine core:hasQuality ?thrust.
```

#retrieve all individuals that are jet engines

```
?thrust core:hasValueExpressionAtAllTimes ?thrustValueExpr.
```

```
?thrustValueExpr core:hasSimpleExpressionValue ?realThrustValue.
```

#retrieve thrust

```
?jetEngineDesignSpec rdf:type jet-ice:JetEngineSpec.
```

```
?jetEngineDesignSpec core:prescribes ?jetEngine.
```

```
?jetEngineDesignSpec core:prescribes ?thrustSpec.
```

#retrieve all designs for the jet engine

```
?thrustSpec bfo:BFO_0000110 ?thrustDesignValueExpr.
```

```
?thrustDesignValueExpr jet-ice:hasLowerBoundValue ?designedThrustValue.
```

#retrieve thrust specification value

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

#compare them



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





Setup Protégé

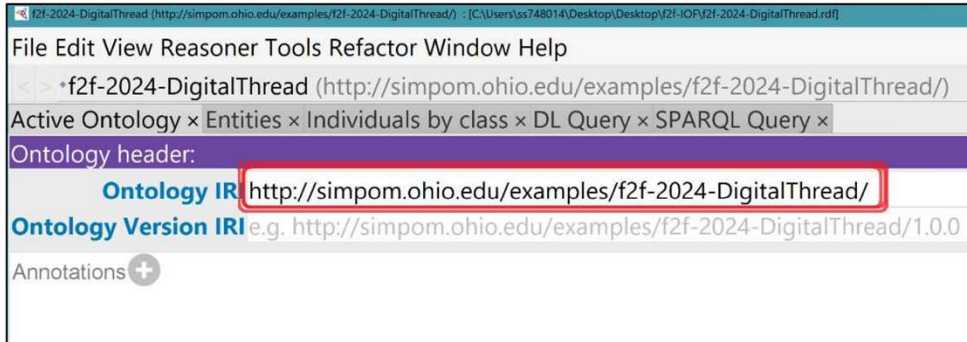


Figure 1: Setting Up an IRI for Your Ontology



Figure 4: Choosing a Location to Save Your File.

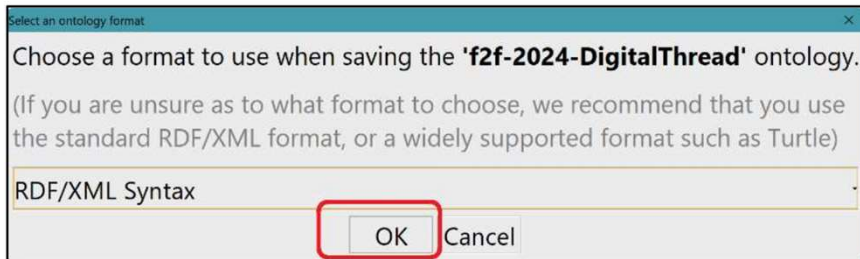


Figure 3: Selecting an Ontology Format



Import Ontology File

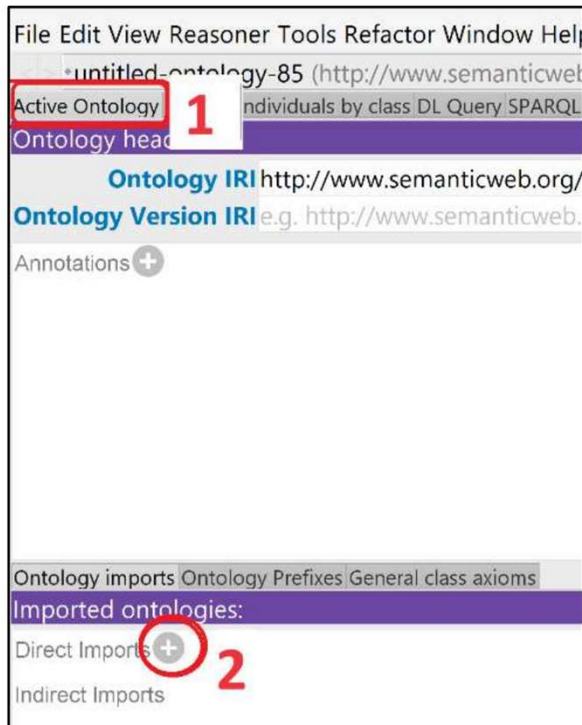


Figure 5: Importing Ontology File.

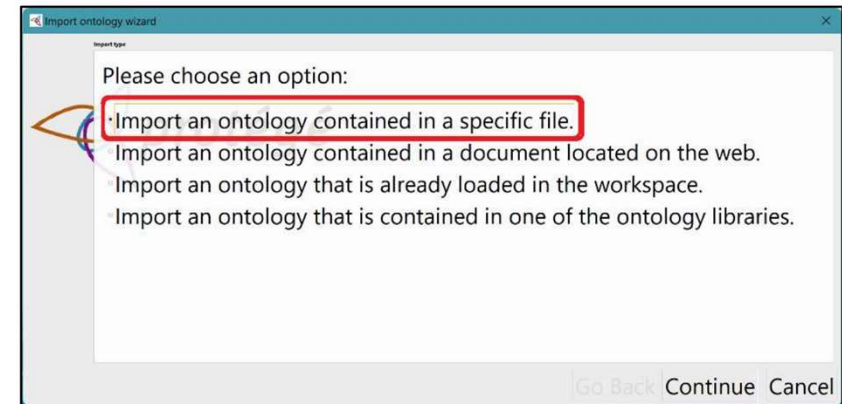


Figure 6: Import Ontology Wizard.

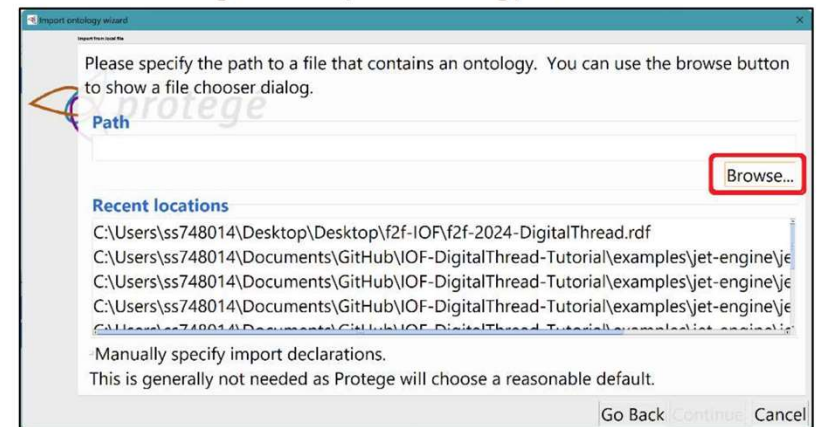


Figure 7: Browsing File to Import.



Import Ontology File

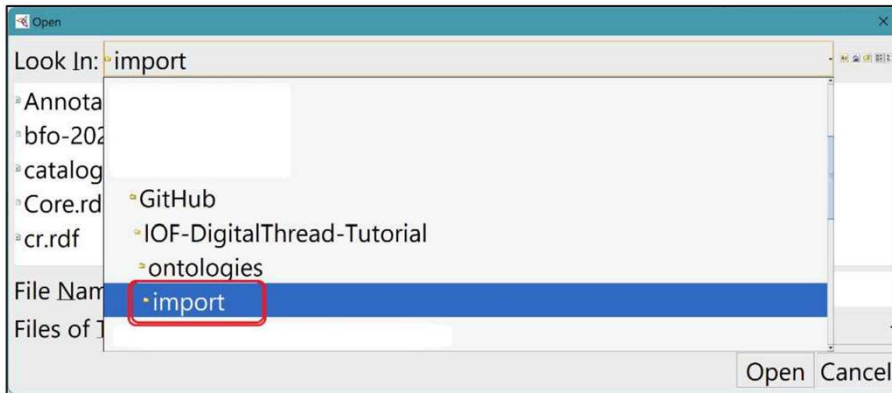


Figure 8: Locating Ontology File for Importing.

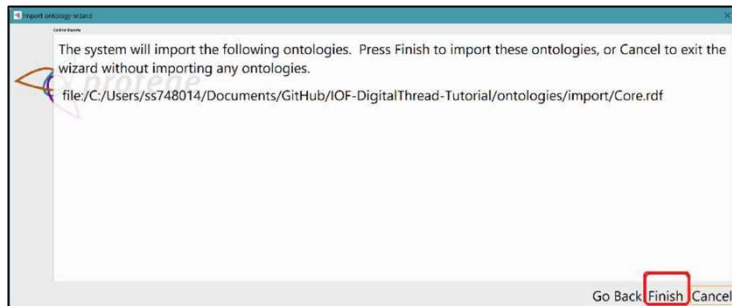


Figure 10: Import Ontology Wizard (3).

Ontology imports Ontology Prefixes General class axioms

Imported ontologies:

Direct Imports +

<https://spec.industrialontologies.org/ontology/202301/core/Core/>
Core
 Ontology IRI: <https://spec.industrialontologies.org/ontology/core/Core/>
 Version IRI: <https://spec.industrialontologies.org/ontology/202301/core/Core/>
 Location: C:\Users\ss748014\Documents\GitHub\IOF-DigitalThread-Tutorial\ontologies\import\Core.rdf

Indirect Imports

<https://spec.industrialontologies.org/ontology/202301/core/meta/AnnotationVocabulary/>
AnnotationVocabulary
 Ontology IRI: <https://spec.industrialontologies.org/ontology/core/meta/AnnotationVocabulary/>
 Version IRI: <https://spec.industrialontologies.org/ontology/202301/core/meta/AnnotationVocabulary/>
 Location: C:\Users\ss748014\Desktop\Desktop\IOF-DigitalThread-Tutorial\ontologies\import\AnnotationVocabulary.rdf

<http://purl.obolibrary.org/obo/bfo/2020/bfo.owl>
bfo
 Ontology IRI: <http://purl.obolibrary.org/obo/bfo.owl>
 Version IRI: <http://purl.obolibrary.org/obo/bfo/2020/bfo.owl>
 Location: C:\Users\ss748014\Documents\GitHub\IOF-DigitalThread-Tutorial\ontologies\import\bfo_2020.owl

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



Create Classes

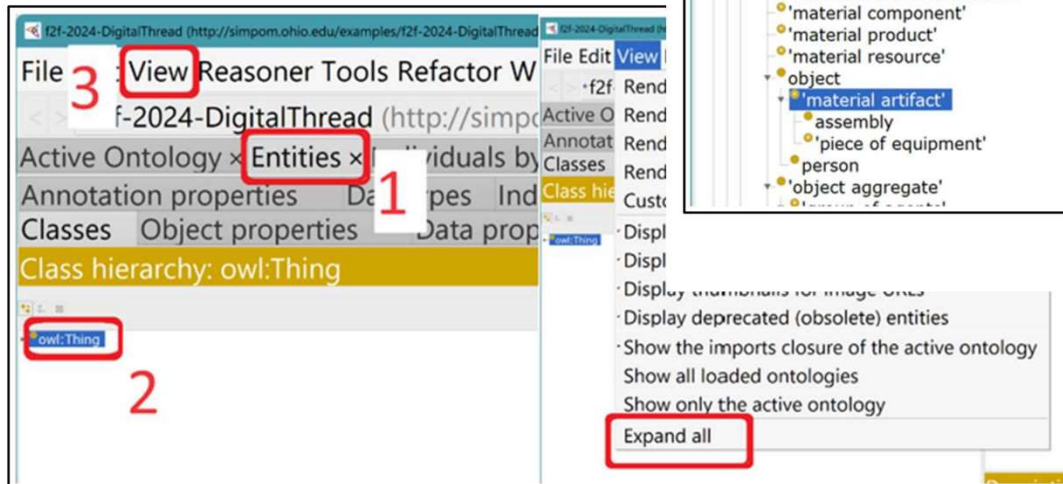


Figure 12: View All Imported Classes.

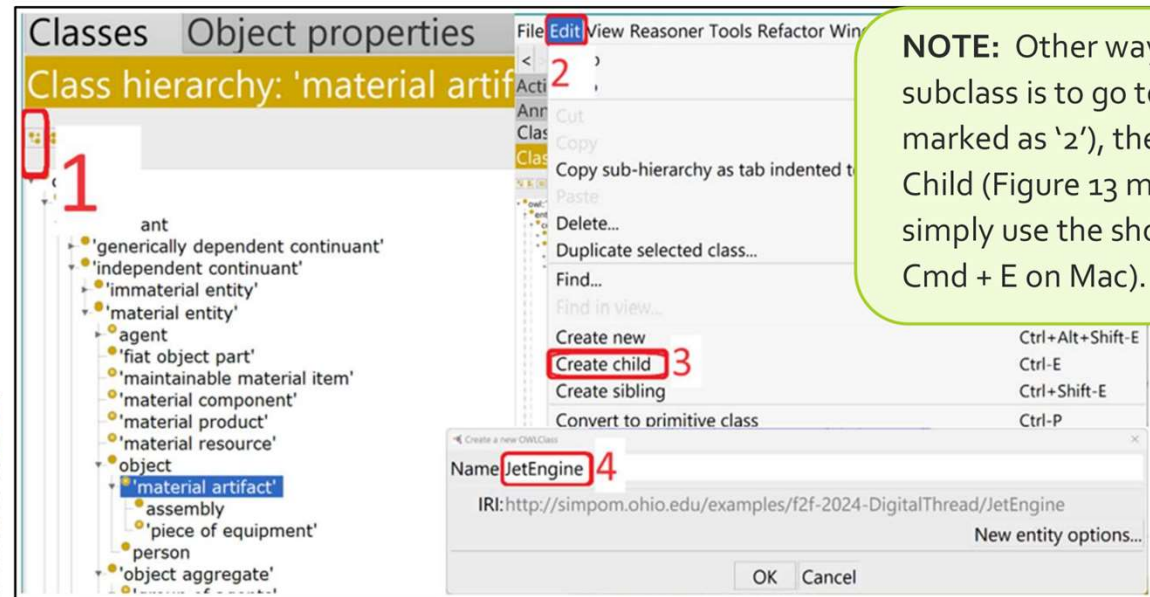


Figure 13: Adding Class.

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).

Create Individuals

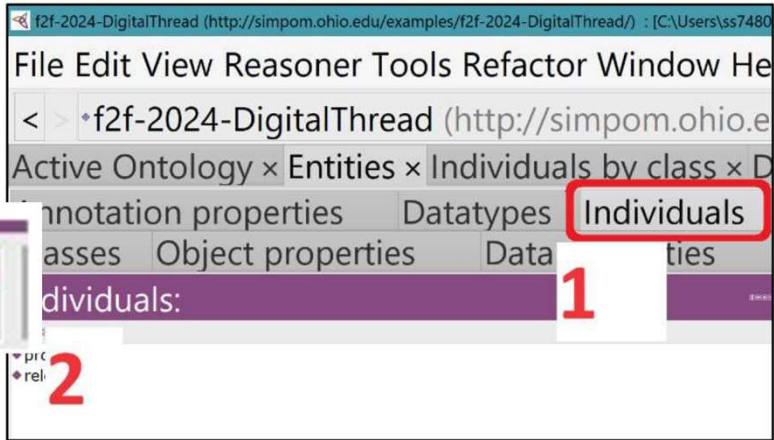


Figure 14: Create an Individual (Instance).

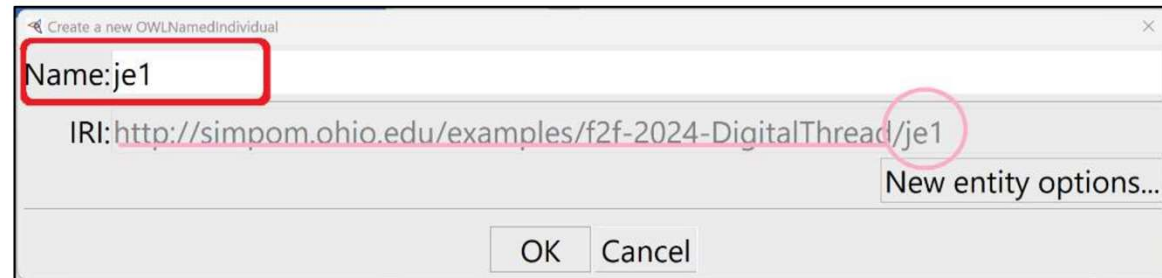


Figure 15: Create a new OWLNamedIndividual.

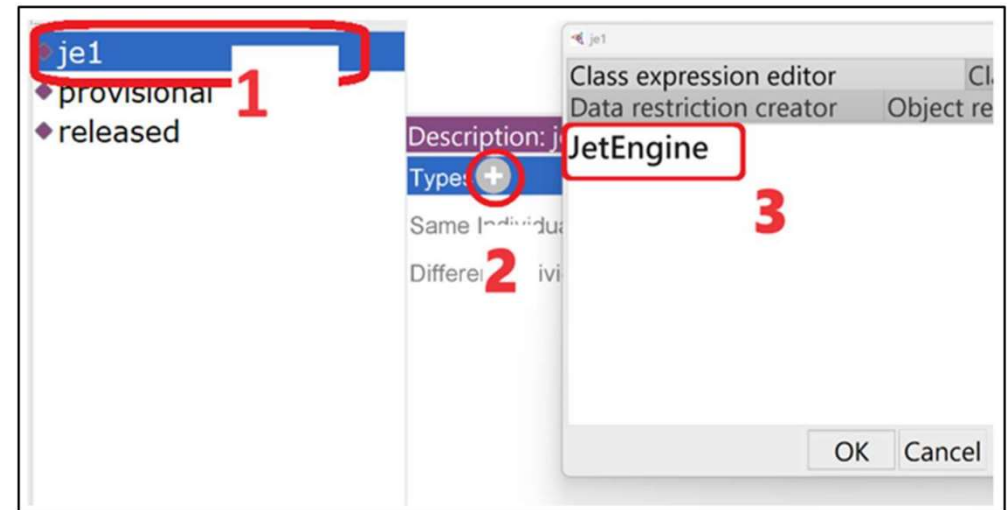


Figure 16: Class Assignment for an Individual.

Object Property Assertions

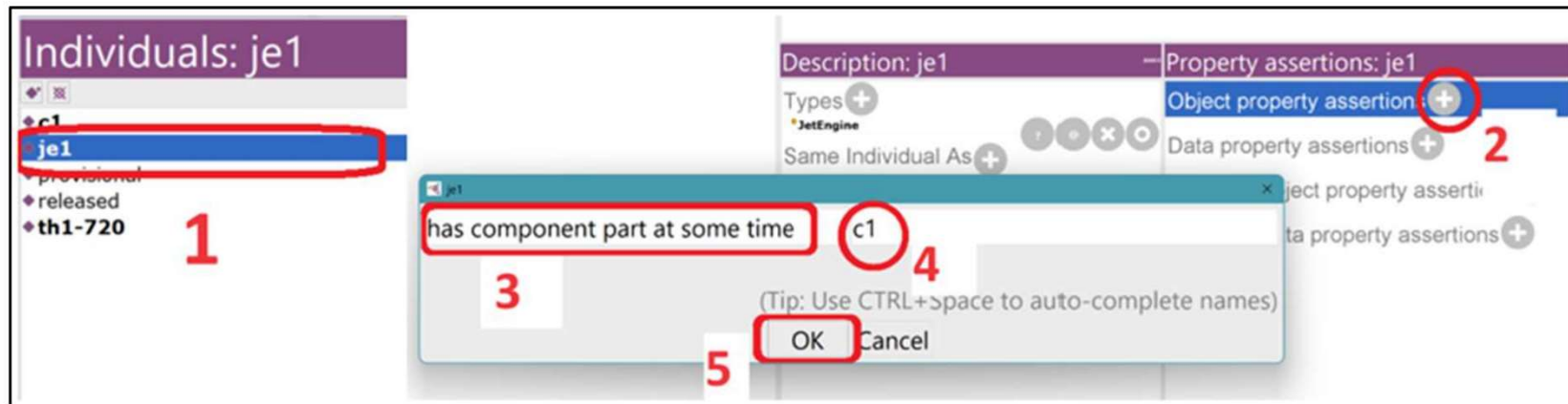


Figure 17: Object Property Assertions.

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)

jet-engine-2ed-base (http://simpom.ohio.edu/examples/jet-engine-2ed-base/) : [C:\Users\sormaz\Documents\GitHub\ou-nist-project\examples\jet-engine-2ndPhase\jet-engine-2ed-base.rdf]

File Edit View Reasoner Tools Refactor Window Help

< > jet-engine-2ed-base (http://simpom.ohio.edu/examples/jet-engine-2ed-base/)

> entity > continuant > independent continuant > material entity > object > MaterialArtifact > jet engine

Active ontology x Entities x Individuals by class x DL Query x SPARQL Query x

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Class hierarchy: jet engine

- object
 - MaterialArtifact
 - Assembly
 - fan assembly
 - ball bearing
 - compressor
 - jet engine
 - PieceOfEquipment
 - turbine
 - Person
 - object aggregate
 - GroupOfAgents
 - OrganizedGroupOfAgents
 - Organization
 - BusinessOrganization
 - Manufacturer
 - System
 - EngineeredSystem
 - RawMaterial
 - specifically dependent continuant
 - quality
 - diameter
 - relational quality
 - thrust
 - realizable entity
 - disposition
 - Capability
 - function
 - BusinessFunction
 - load rating
 - MeasurementCapability

Annotations Usage

Annotations: jet engine

Annotations +

rdfs:label [language: en]
jet engine

Asserted in: <http://simpom.ohio.edu/examples/jet-engine-2ed-base/>

Description: jet engine

Equivalent To +

SubClass Of +

- MaterialArtifact

General class axioms +

SubClass Of (Anonymous Ancestor)

- 'continuant part of' only 'independent continuant'
- 'continuant part of' only continuant
- 'has continuant part' only
(site or 'material entity' or 'continuant flat boundary')
- 'continuant part of' only 'material entity'
- object
and ('bearer of' some
(function

Git: main To use the reasoner click Reasoner > Start reasoner Show Inferences



Populate Instance Data in Protégé (Phase 2)

Using Reasoner

The screenshot shows the Protégé interface with the 'je1-satisfied' class selected in the 'Individuals' list on the left. The right-hand pane displays the 'Property assertions' for this class, including:

- hasComponentPartAtSomeTime fanA1
- hasComponentPartAtSomeTime tu1
- hasComponentPartAtSomeTime c1
- hasQuality th1-710
- 'has proper continuant part at some time' fanA1
- 'has proper continuant part at some time' tu1
- 'has proper continuant part at some time' c1
- 'has continuant part' fanA1
- 'has continuant part' tu1
- 'has continuant part' c1
- 'specifically depended on by' th1-710
- 'bearer of' th1-710

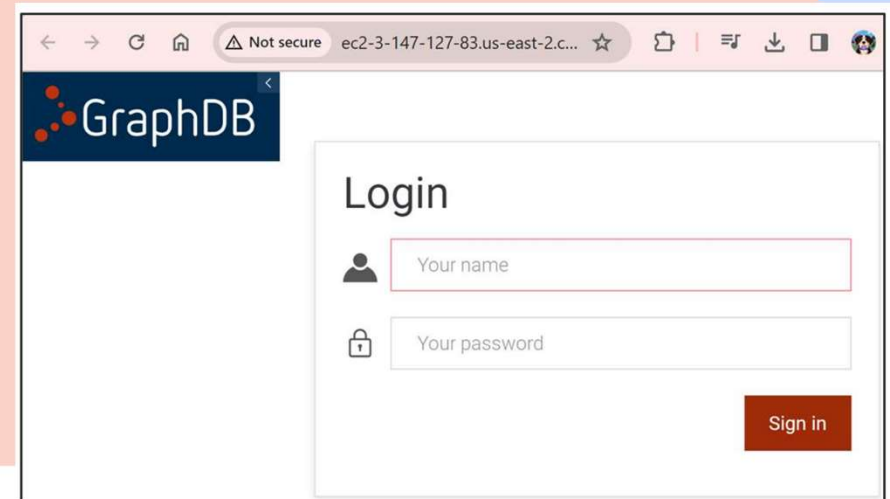




GraphDB



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-kMN9A7o*Fpute.amazonaws.com/
 - b. Shortened URL: <https://bit.ly/iof24o2>
 - 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"

+ Create new repository
4. Choose "GraphDB Repository" as the type of repository for this tutorial.

 GraphDB Repository
GraphDB repositories store data, answer queries and execute updates.
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.

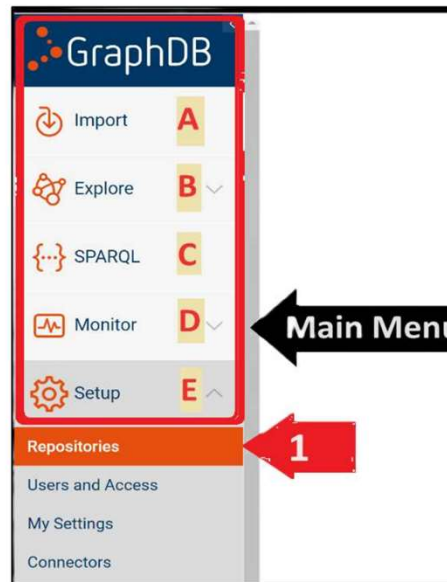


Figure 20: GraphDB Main Menu.

Create GraphDB repository

Repository ID* **ss-DigitalThread-CR** 2

Repository description

Read-only

Inference and Validation

Ruleset OWL2-QL (Optimized) Custom ruleset...

Disable owl:sameAs

Enable consistency checks

Enable SHACL validation SHACL options

Entity ID size: 32-bit 40-bit

Enable context index

Enable predicate list index

Enable full-text search (FTS) index

FTS indexes to build (comma delimited): default, iri

FTS index for xsd:string literals: default

FTS index for full-text indexing of IRIs: none

Queries and Updates

Query timeout (seconds): Throw exception on query timeout

Limit query results:

Create Cancel

Figure 21: Create GraphDB Repository.

7. Then at the bottom of the page click 'Create' Create. This will finalize the creation of your new repository configured specifically for this tutorial.

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**.



Figure 23: Import Ontology Files.

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

Explore Knowledge Graphs

612	jet-engine-Base:hasLowerBoundValue	rdf:type
613	jet-engine-Base:hasLowerBoundValue	rdfs:subP
614	jet-engine-Base:je1	rdf:type
	http://simpom.ohio.edu/examples/jet-engine-Base/je1	
615	jet-engine-Base:je1	rdf:type
616	jet-engine-Base:je2	rdf:type
617	jet-engine-Base:je2	rdf:type

Figure 24: Locate a Specific Class or Individual.

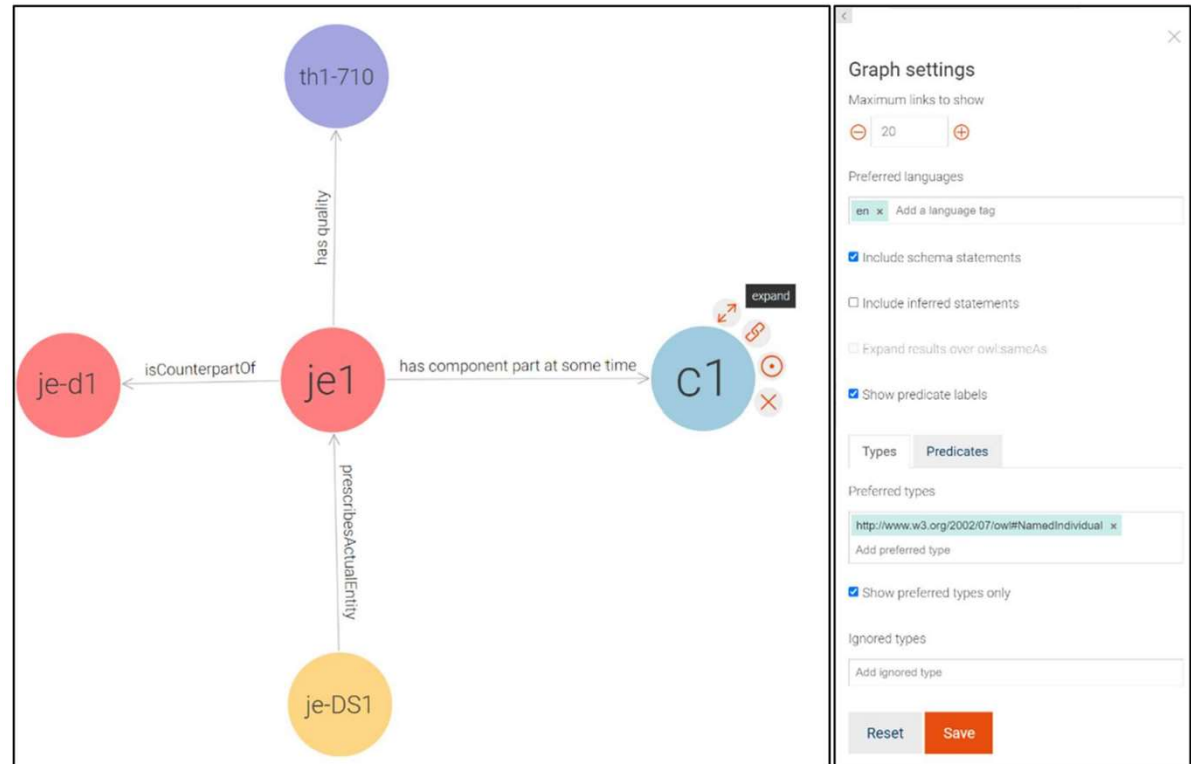


Figure 25: Adjusting Graph Settings.

SPARQL Query

SPARQL Query & Update

Unnamed **click here to rename query**

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5 PREFIX core: <https://spec.industrialontologies.org/ontology/core/Core/>
6 PREFIX jeb: <http://simpom.ohio.edu/examples/jet-engine-Base/>
7 PREFIX jecr: <http://simpom.ohio.edu/examples/jet-engine-cr/>
8
9 PREFIX cr: <http://simpom.ohio.edu/ontology/cr/>
10 SELECT ?engine ?engdesign ?realthrustValue ?designThrustValue ?difference
11 WHERE {
12   # find engine and its thrust
13   ?engine rdf:type jeb:JetEngine.
14   ?engine cr:engineHasActualEntity ?engdesi

```

Results Window

result options

Showing results from 1 to 4 of 4. Query took 0.1s, on 2024-02-01 at 02:53.

	engine	engdesign	realthrustValue	designThrustValue	difference
1	jet-engine-Baseje1	crje-d1	*710**xsd integer	*700**xsd decimal	*10**xsd decimal
2	jet-engine-Baseje2	crje-d1	*690**xsd integer	*700**xsd decimal	*-10**xsd decimal
3	jet-engine-Baseje3	crje-d1	*670**xsd integer	*700**xsd decimal	*-30**xsd decimal
4	jet-engine-Baseje4	crje-d1	*720**xsd integer	*700**xsd decimal	*20**xsd decimal

Figure 26: SPARQL Window View.

SPARQL with Pivot Table Results

Table Raw Response Pivot Table Google Chart [Get HTML snippet to embed results on a web page](#)

Showing results from 1 to 4 of 4. Query took 0.1s, on 2024-02-01 at 02:53.

Available Variables: engine, engdesign, realthrustValue, designThrustValue, difference

Cells: Count

Rows: Totals 4

Figure 27: Pivot Table (Available Variables).

Table

Available Variables

Column: designThrustValue, realthrustValue, difference

Rows: engdesign, engine

		designThrustValue	700				Totals
		realthrustValue	670	690	710	720	
		difference	-30	-10	10	20	
engdesign	engine						
	jet-engine-Base:je1				1		1
	jet-engine-Base:je2			1			1
	jet-engine-Base:je3	1					1
	jet-engine-Base:je4					1	1
Totals		1	1	1	1	1	4

Figure 28: Pivot Table Details.

Table

Available Variables

Cells: Sum over Sum, realthrustValue, designThrustValue

Columns: designThrustValue, realthrustValue, difference

Rows: engdesign, engine

		designThrustValue	700				Totals
		realthrustValue	670	690	710	720	
		difference	-30	-10	10	20	
engdesign	engine						
	jet-engine-Base:je1				1.01		1.01
	jet-engine-Base:je2			0.99			0.99
	jet-engine-Base:je3	0.96					0.96
	jet-engine-Base:je4					1.03	1.03
Totals		0.96	0.99	1.01	1.03	1.00	

Figure 29: Using Pivot Table for Simple Analyses.



SPARQL with Google Chart Results



Figure 30: Google Chart Configurations.

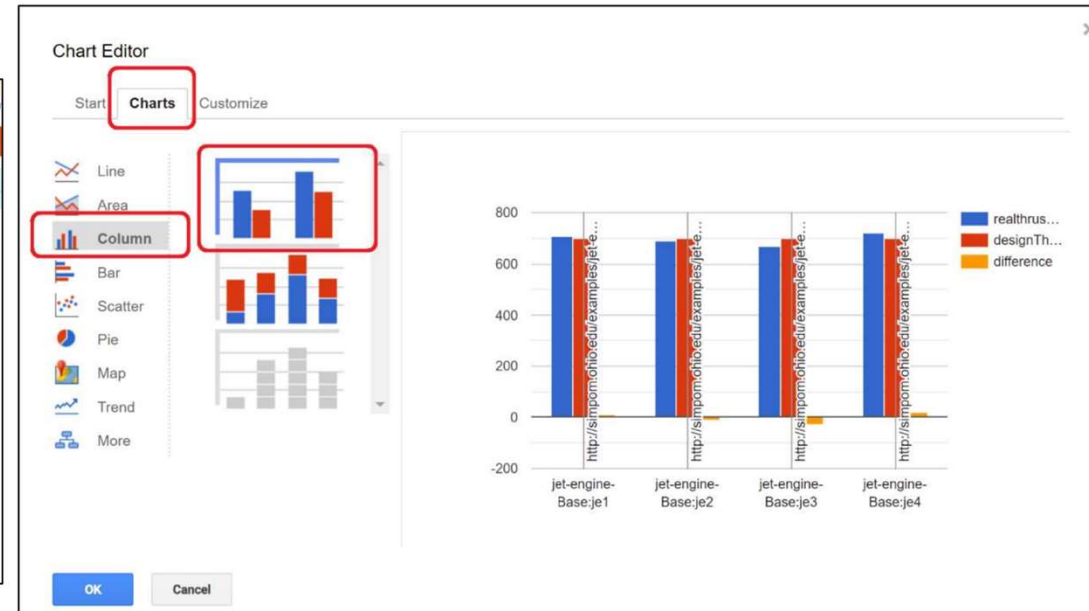


Figure 31: Google Chart Configurations (2).



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

- Import
- Explore
- SPARQL
- Monitor
- Setup
- Help

Import ?



jet-2ed

en

User data

Server files

Help



Upload RDF files

All RDF formats, up to 200 MB



Get RDF data from a URL

All RDF formats



Import RDF text snippet

Type or paste RDF data



Type to filter



bfo-2020.owl

× ? ✓ Imported successfully in less than a second.


Import



Core.rdf

× ? ✓ Imported successfully in less than a second.


Import



AnnotationVocabulary.rdf

× ? ✓ Imported successfully in less than a second.


Import




jet-engine-2ed-base.rdf

× ? ✓ Imported successfully in less than a second.


Import



GraphDB SPARQL Editor and Results



- Import
- Explore
- SPARQL
- Monitor
- Queries and Updates
- System
- Setup
- Help

SPARQL Query & Update

Unnamed × Unnamed × Unnamed × +

```

1 #11.Which phases occurred according to the plan?
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6 PREFIX core: <https://spec.industrialontologies.org/ontology/core/Core/>
7 PREFIX bfo: <http://purl.obolibrary.org/obo/>
8 PREFIX cr: <http://simpom.ohio.edu/ontology/cr/>
9 PREFIX biopb: <http://simpom.ohio.edu/examples/bio-process-base/>
10 PREFIX biopcr: <http://simpom.ohio.edu/examples/bio-process-cr/>
11
12 SELECT ?ActualPhases ?hasCorrectPhaseSequence ?hasCorrectPhaseDuration ?
hasRequiredEquipment
13 ((?hasCorrectPhaseSequence && ?hasCorrectPhaseDuration && ?
hasRequiredEquipment ) as ?OccurredAccordingToplan)
14 WHERE {
15 {SELECT ?ActualPhases
16 (count(distinct(?PlannedFollowingSteps)) as ?
numberOfPlannedFollowingSteps)
17 (count(distinct(?RealFollowingSteps)) as ?
numberOfRealFollowingSteps)
18 ?hasCorrectPhaseSequence
19 WHERE{
20 ?has_participant_all rdfs:label "has participant at all
times"@en

```

Run keyboard shortcuts

jet-2ed
en

Editor only
Editor and results
Results only

Table
Raw Response
Pivot Table
Google Chart
Get HTML snippet to embed results on a web page

Filter query results Showing results from 1 to 9 of 9. Query took 0.1s, on 2024-01-02 at 11:38.

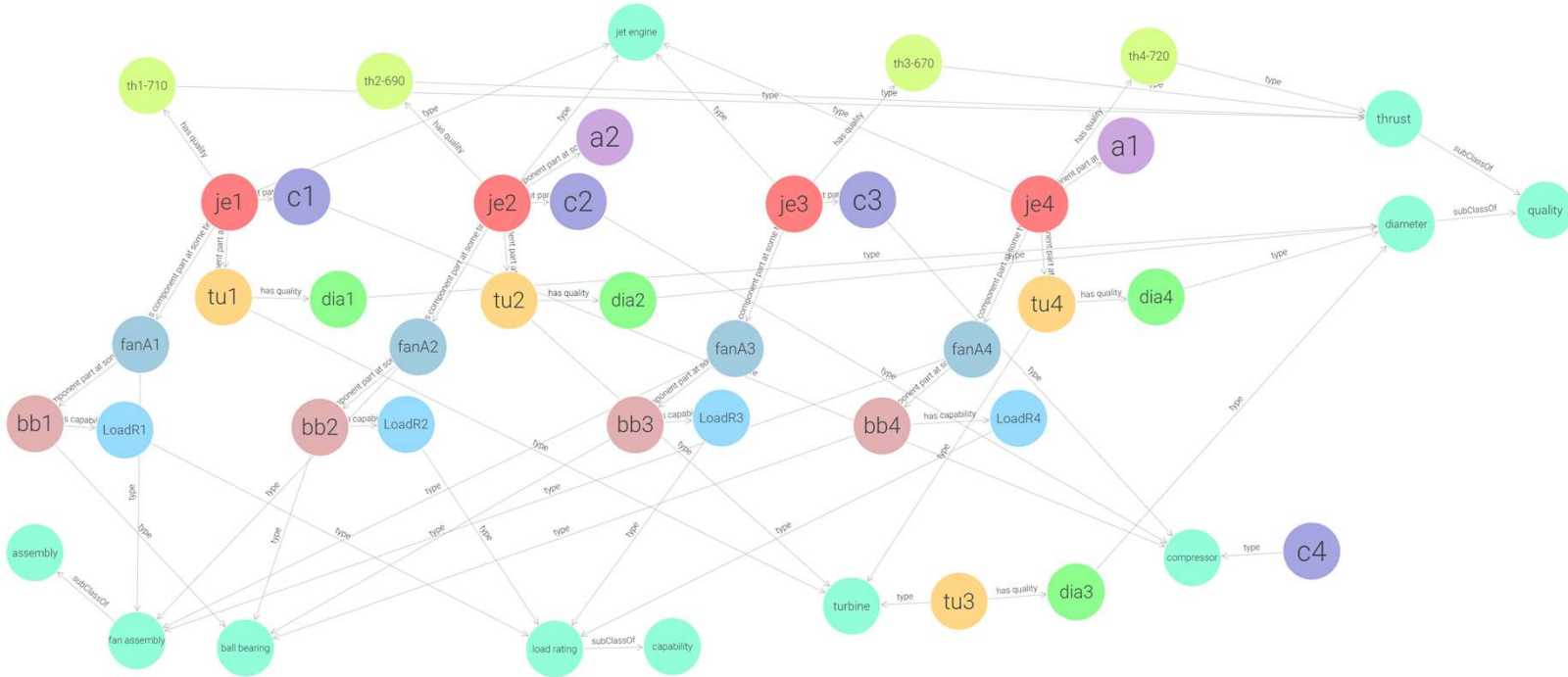
	ActualPhases	hasCorrectPhaseSequence	hasCorrectPhaseDuration	hasRequiredEquipment	OccurredAccordingToplan
1	http://simpom.ohio.edu/examples/bio-process-base/cgp1	*true**xsd:boolean	*true**xsd:boolean	*true**xsd:boolean	*true**xsd:boolean
2	http://simpom.ohio.edu/examples/bio-process-base/cgp2	*true**xsd:boolean	*false**xsd:boolean	*true**xsd:boolean	*false**xsd:boolean
3	http://simpom.ohio.edu/examples/bio-process-base/cgp3	*false**xsd:boolean	*false**xsd:boolean	*true**xsd:boolean	*false**xsd:boolean
4	http://simpom.ohio.edu/examples/bio-process-base/cgp4	*true**xsd:boolean	*true**xsd:boolean	*true**xsd:boolean	*true**xsd:boolean
5	http://simpom.ohio.edu/examples/bio-process-base/cpp1	*true**xsd:boolean	*false**xsd:boolean	*true**xsd:boolean	*false**xsd:boolean
6	http://simpom.ohio.edu/examples/bio-process-base/cpp2	*true**xsd:boolean	*false**xsd:boolean	*true**xsd:boolean	*false**xsd:boolean



GraphDB Explore Repository

- GraphDB
- Import
- Explore
- Graphs overview
- Class hierarchy
- Class relationships
- Visual graph**
- Similarity
- SPARQL
- Monitor
- Setup
- Help

Visual graph ⓘ



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 SPARQL 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	compressor	turbine	fan Assy	Ball bearing	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

The screenshot displays the Protege interface for an ontology. The main window shows the 'Individuals: je10' view. On the left, a list of individuals is shown, with 'je10' selected. The right pane is divided into two sections: 'Description: je10' and 'Property assertions: je10'. The 'Description' section shows the type 'jet engine'. The 'Property assertions' section lists various properties and their values for 'je10', such as 'hasComponentPartAtSomeTime c10', 'hasQuality th9', and 'isSubjectOf je-DS1'. The interface includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help) and a toolbar. The status bar at the bottom indicates 'Git: main', 'Reasoner active', and 'Show Inferences'.



Acknowledgements

Funded by NIST through Ohio University cooperation agreement #70NANB22H155



Thank you