Course: Intelligent Systems

Unit 3: Ontology Engineering

Using Protégé (Desktop)

Mari Carmen Suárez de Figueroa Baonza

Technical University of Madrid

Index

- Brief Introduction
- Main Modelling Components
- How to use it

Introduction (I): What is Protégé?

- A free, open-source **ontology editor** and **knowledge-base framework** for building intelligent systems
- It is based on Java, is extensible, and provides a plug-and-play environment
  - **Plug-ins** for extending the core functionality
    - Examples are OWLViz and Graphviz
- It is supported by a strong community of developers and academic, government and corporate users
- It is used to build knowledge-based solutions in areas as diverse as biomedicine, e-commerce, and organizational modelling

protégé

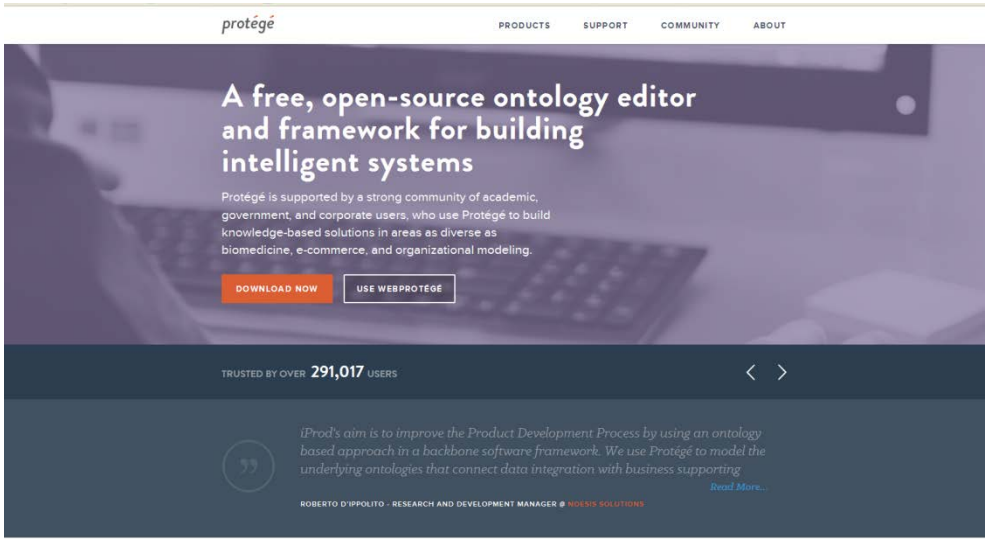
Introduction (II): What can you do with Protégé?

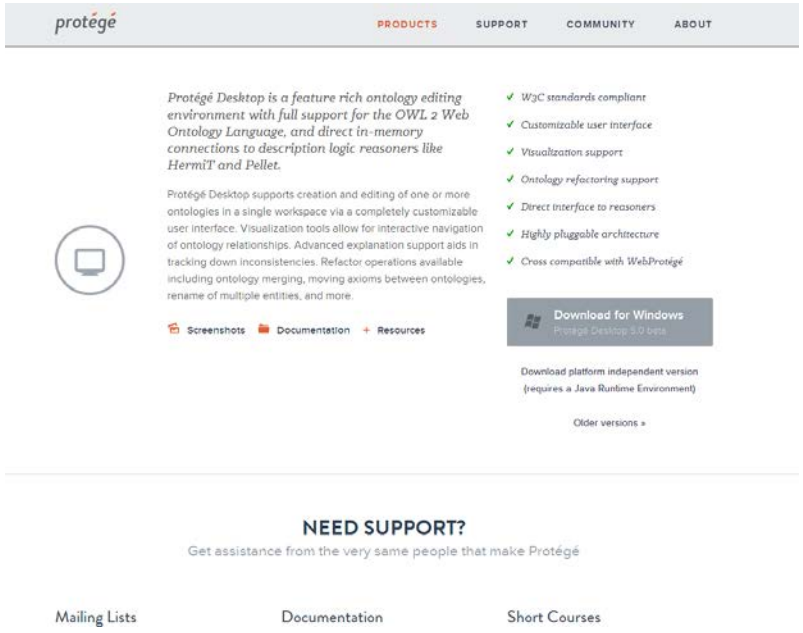
- To create a new **ontology (model) from scratch**
- To download and **extend an existing ontology (model)**
- To **export ontologies (models)** in a variety of formats
  - OWL/XML
  - RDF/XML

protégé

Introduction (III): Where you can find Protégé?

- **Homepage**: *http://protege.stanford.edu/*
- It is available for **downloading** at
  - *https://protege.stanford.edu/products.php#desktop-protege*





Introduction (IV): How does Protégé works?

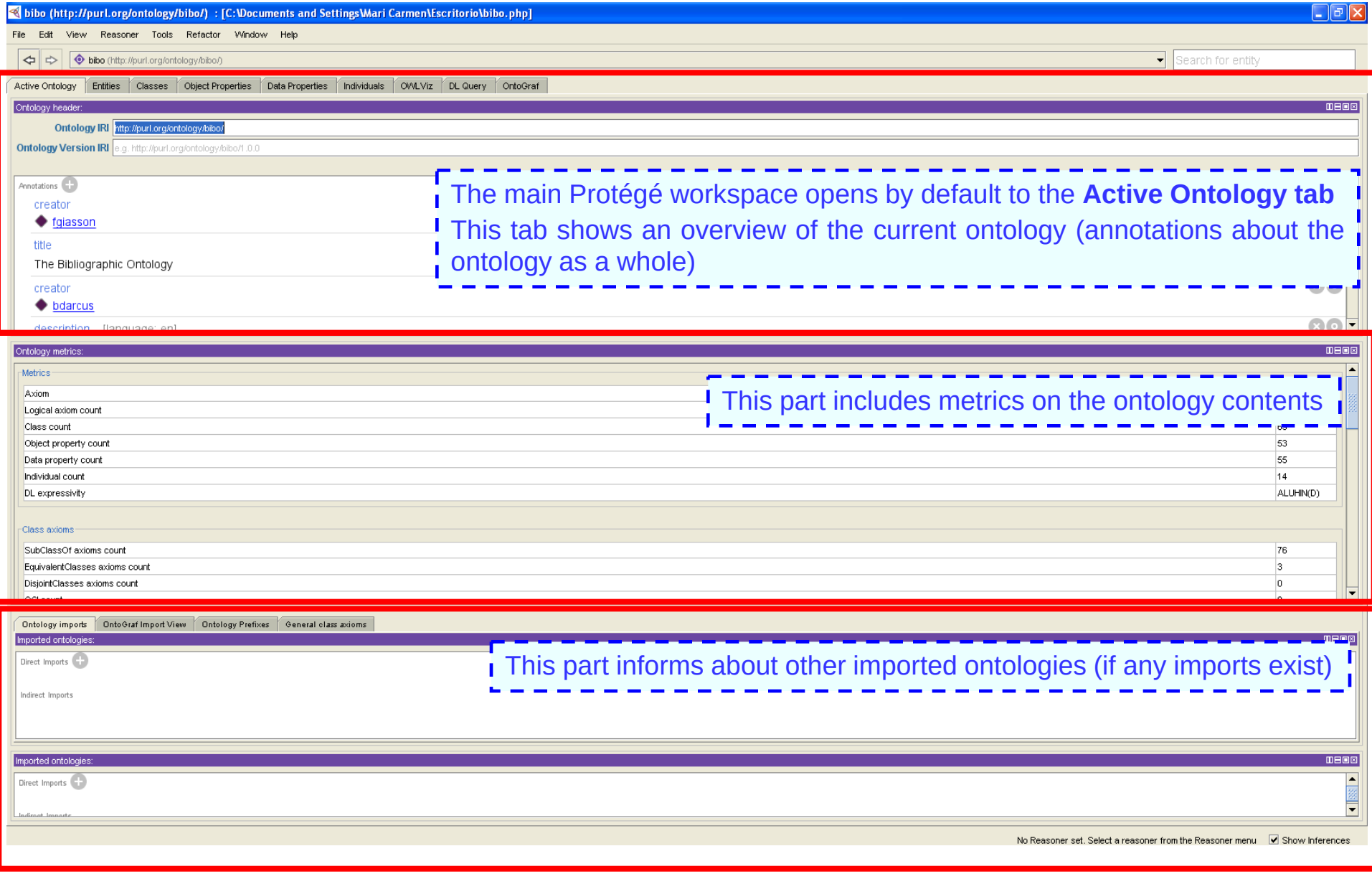
- Objects in the domain are expressed through a series of interrelated **classes**
- **Class hierarchy** is similar to that used by object-oriented languages such as Java
  - Superclasses
  - Subclasses
  - Sibling Classes
- Heavy reliance on inheritance
  - Protégé supports **multiple inheritance**

protégé

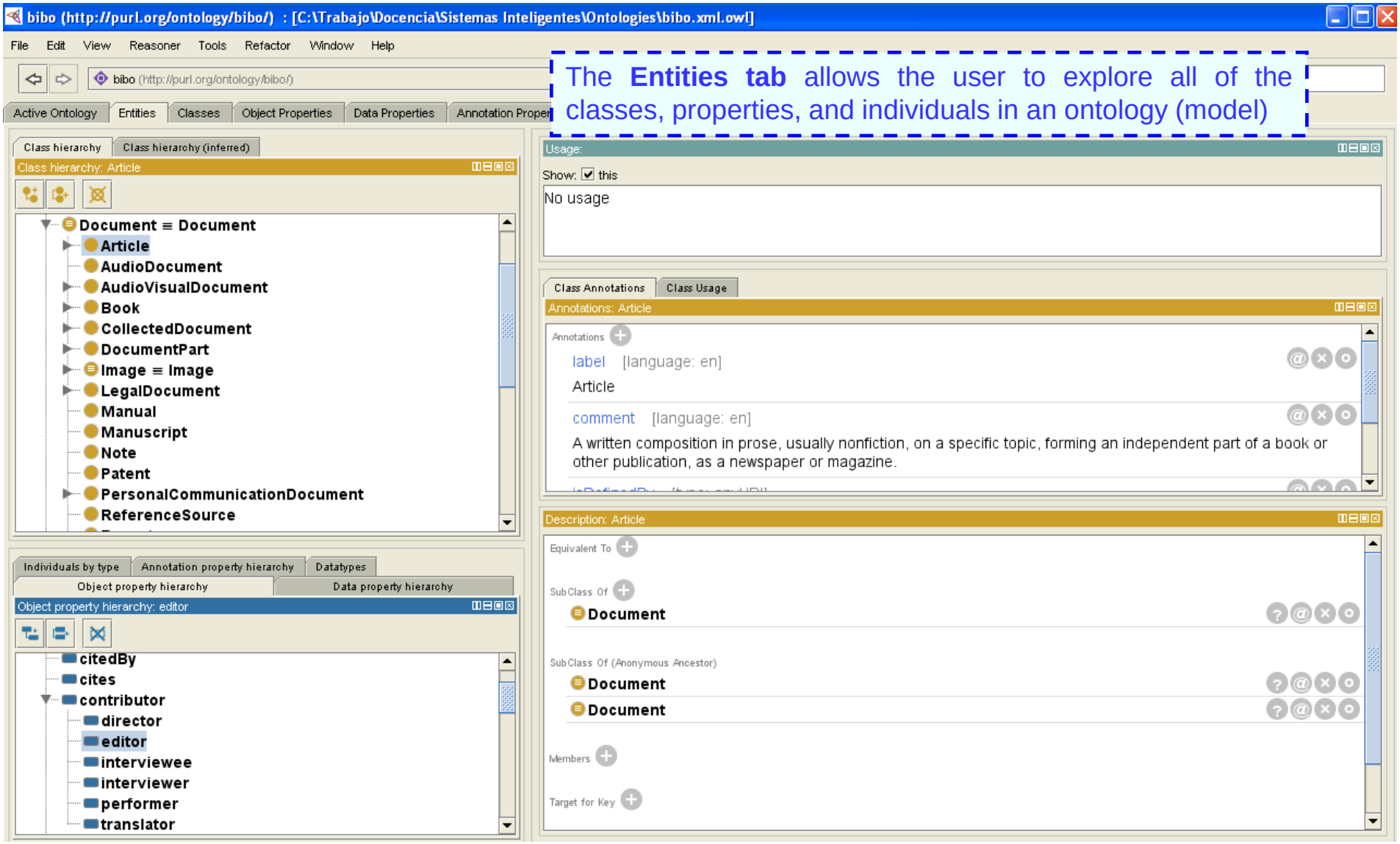
Introduction (V): Notes

- Screenshots provided in this presentation were made using Protégé 4.3
  - This means layout and options could be different from other versions of Protégé

Introduction (VI): General View



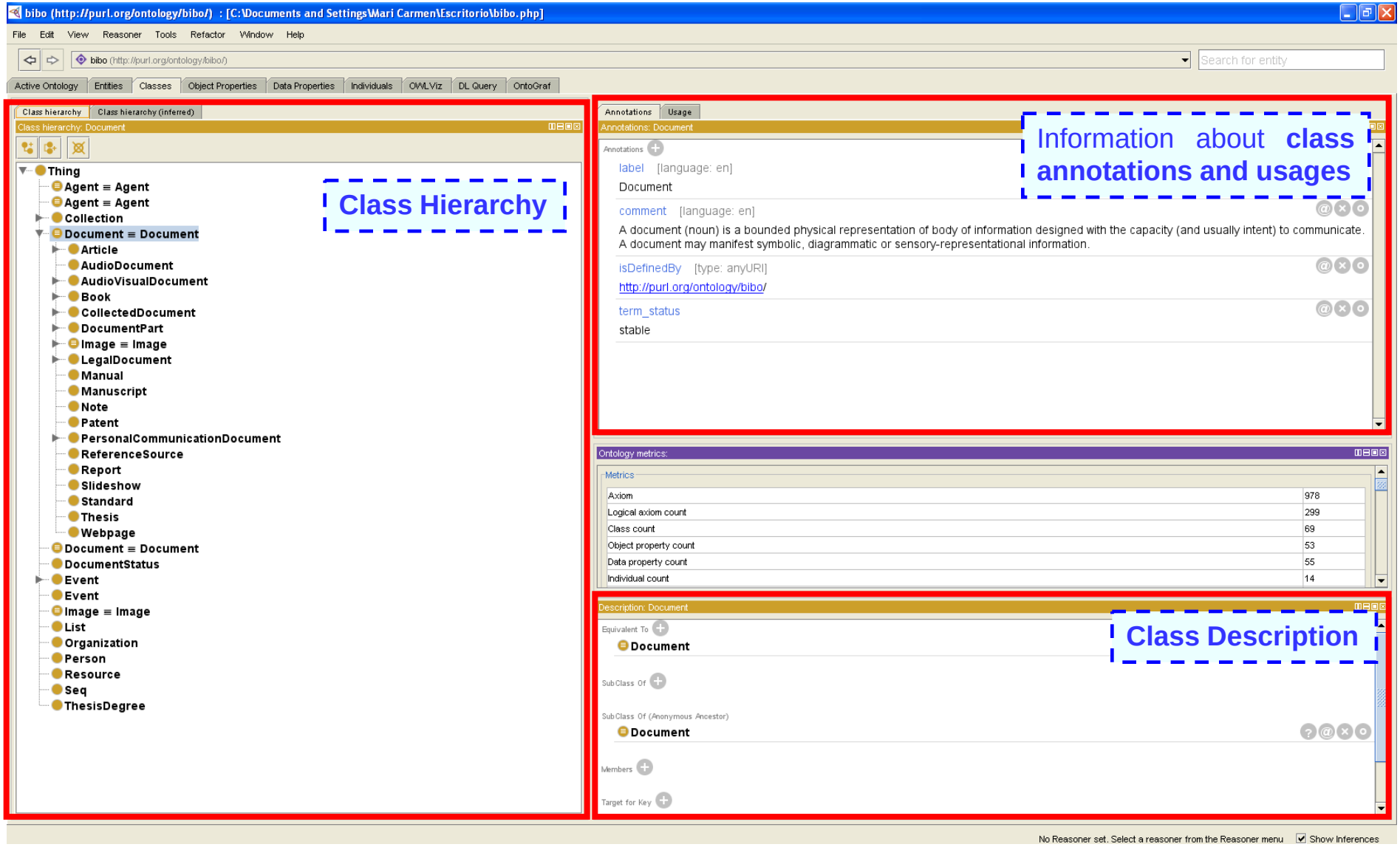
Introduction (VII): General View



Modelling Components (I)

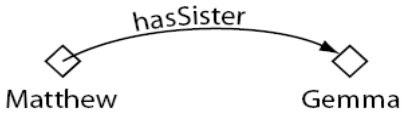
- **Classes** are sets that contain individuals
  - *Examples*: Table, Chair, Person, Building, etc.
- **Thing** is a class representing the set containing all individuals
  - All classes are subclasses of Thing
- Classes are assumed to 'overlap'
  - Individuals of a class A can also be individuals of class B
  - To 'separate' a group of classes
    - One must make them **disjoint** from one another
    - If A is disjoint from B, then an individual of class A cannot also be an individual of class B
      - *Example*: Man is disjoint with Woman; Even is disjoint with Odd

Modelling Components (I): Classes Tab



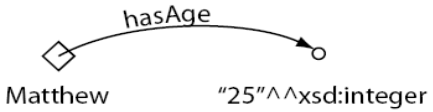
Modelling Components (II)

- **Properties** are binary relations (represent relationships between two objects)
  - between two individuals (**Object Property**)
    - Correspond to relationships in UML
    - *Examples*:
      - Man is subclass of Person
      - Person works in Building



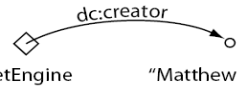
An object property linking the individual Matthew to the individual Gemma

  - between one individual and a data values (**Datatype Property**)
    - Correspond to attributes in UML
    - *Examples*:
      - Person has age
      - Book has title



A datatype property linking the individual Matthew to the data literal '25', which has a type of an xml:integer.

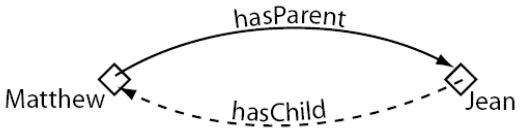
  - **Annotation properties**: to be used to add annotation information to the entire ontology (model), classes, individuals, and properties



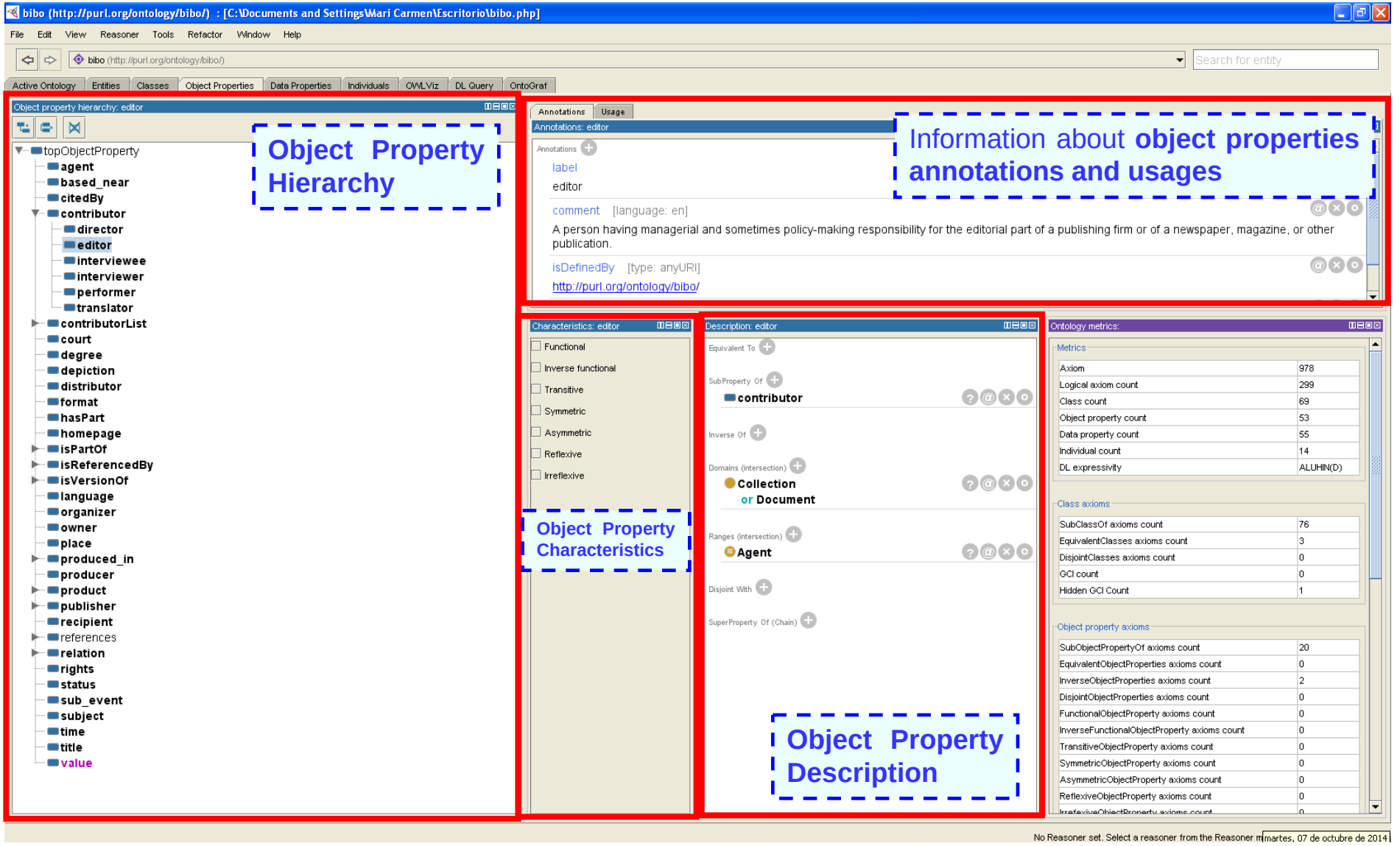
An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

Modelling Components (II)

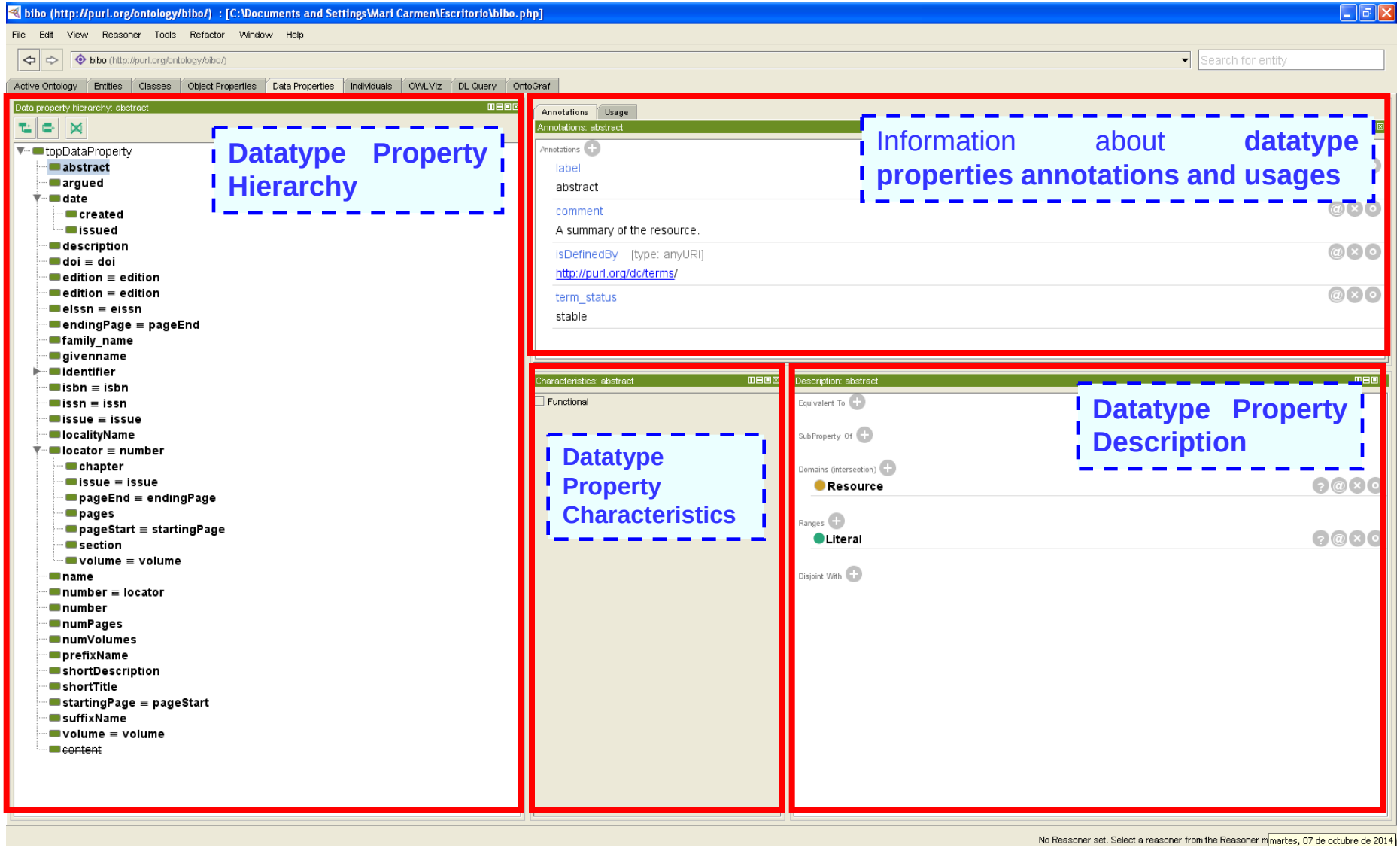
- **Properties** are binary relations (represent relationships between two objects)
  - Domain and Range
    - Properties link individuals from the **domain** to individuals or datatypes from the **range**
  - Super Properties
    - Properties can be further refined as **sub-properties** inheriting the domain, range, characteristics and restrictions
  - **Inverted properties in object properties**
    - If some property links individual A to individual B then its inverse property will link individual B to individual A
    - *Example*: hasIngredient versus isIngredientOf



Modelling Components (II): Object Properties Tab



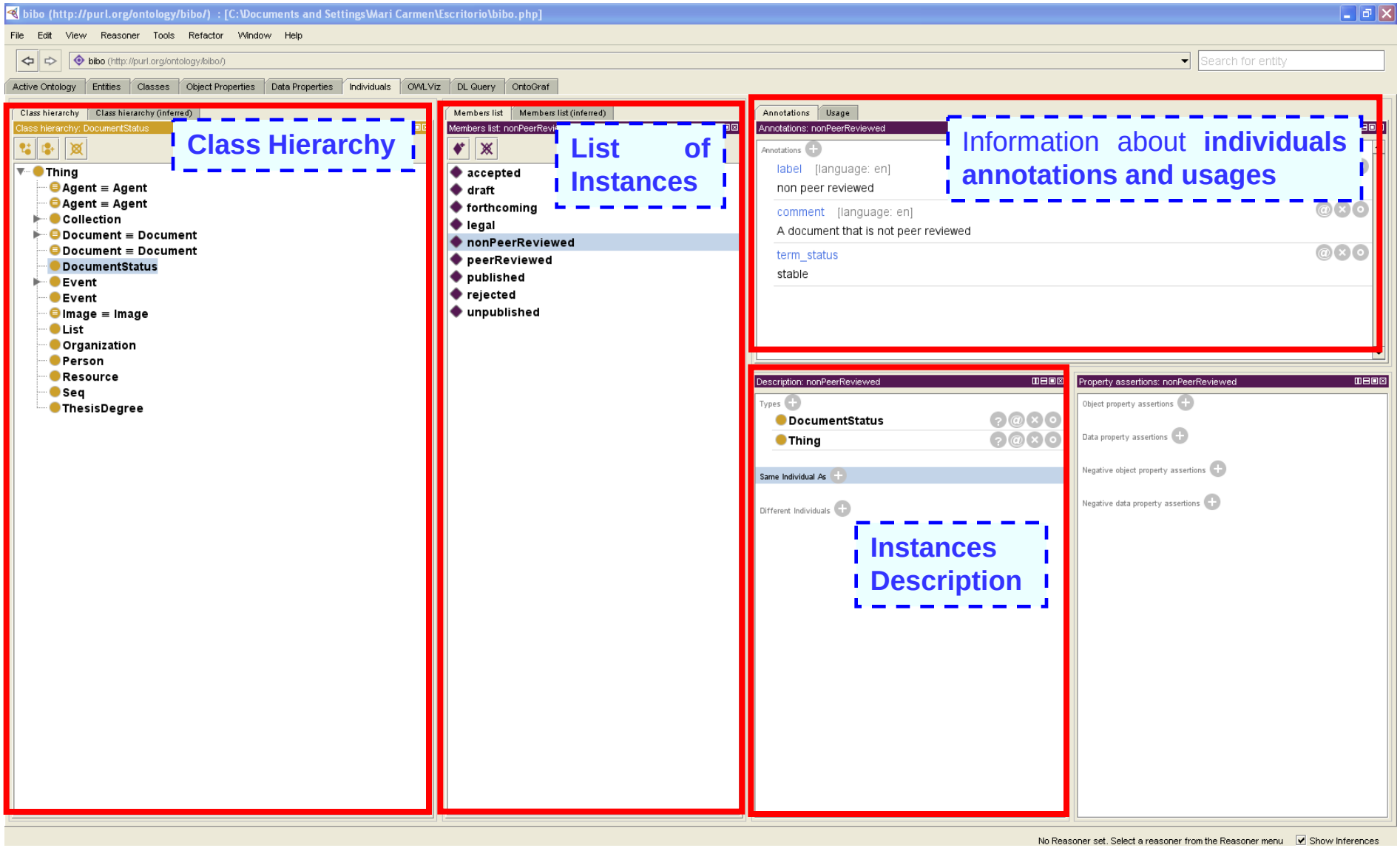
Modelling Components (II): Data Properties Tab



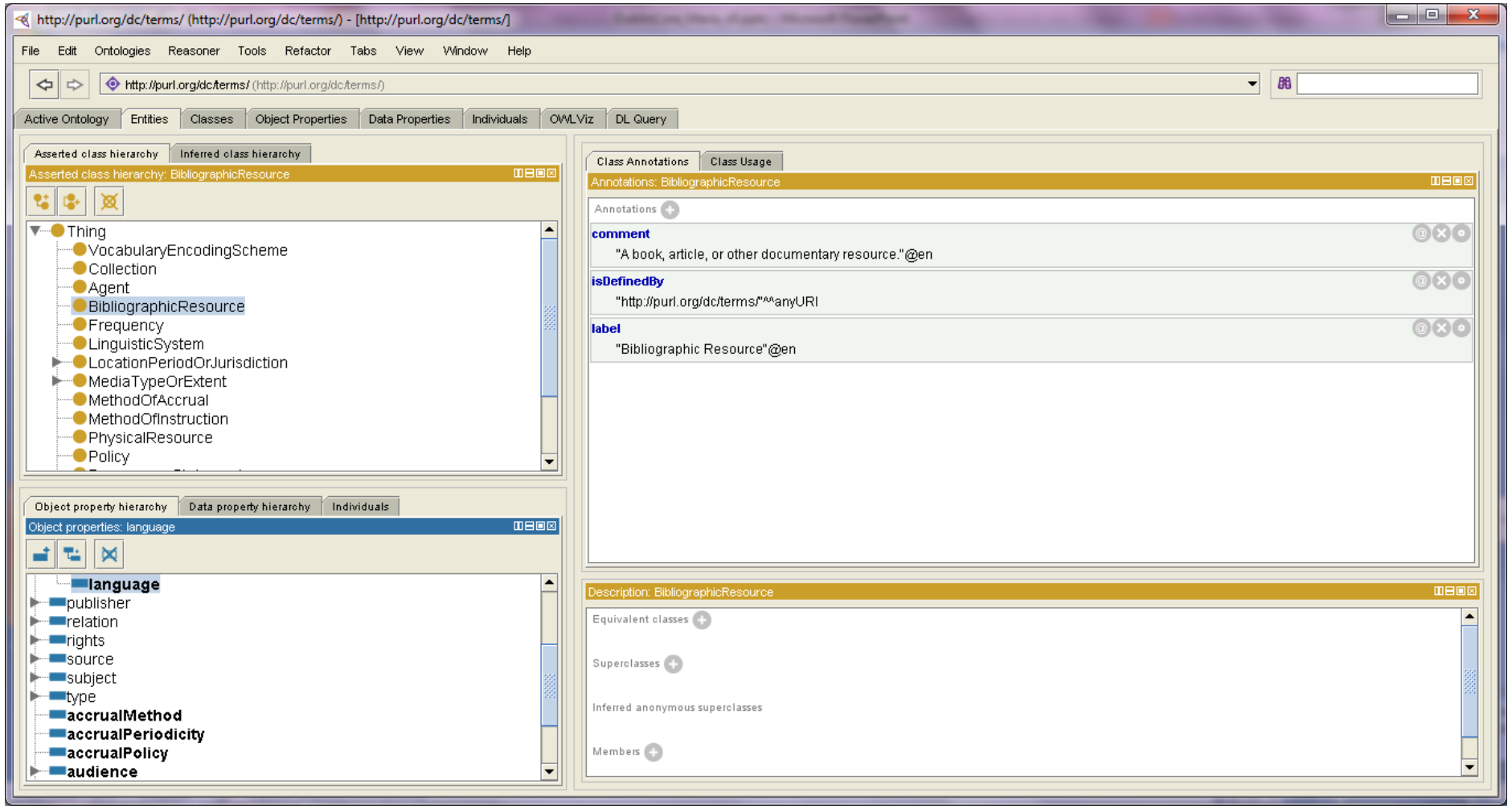
Modelling Components (III)

- **Individuals** represent objects within the model (members of classes)
  - *Examples*: María, Torre Espacio, UPM, etc.

Modelling Components (III): Individuals Tab



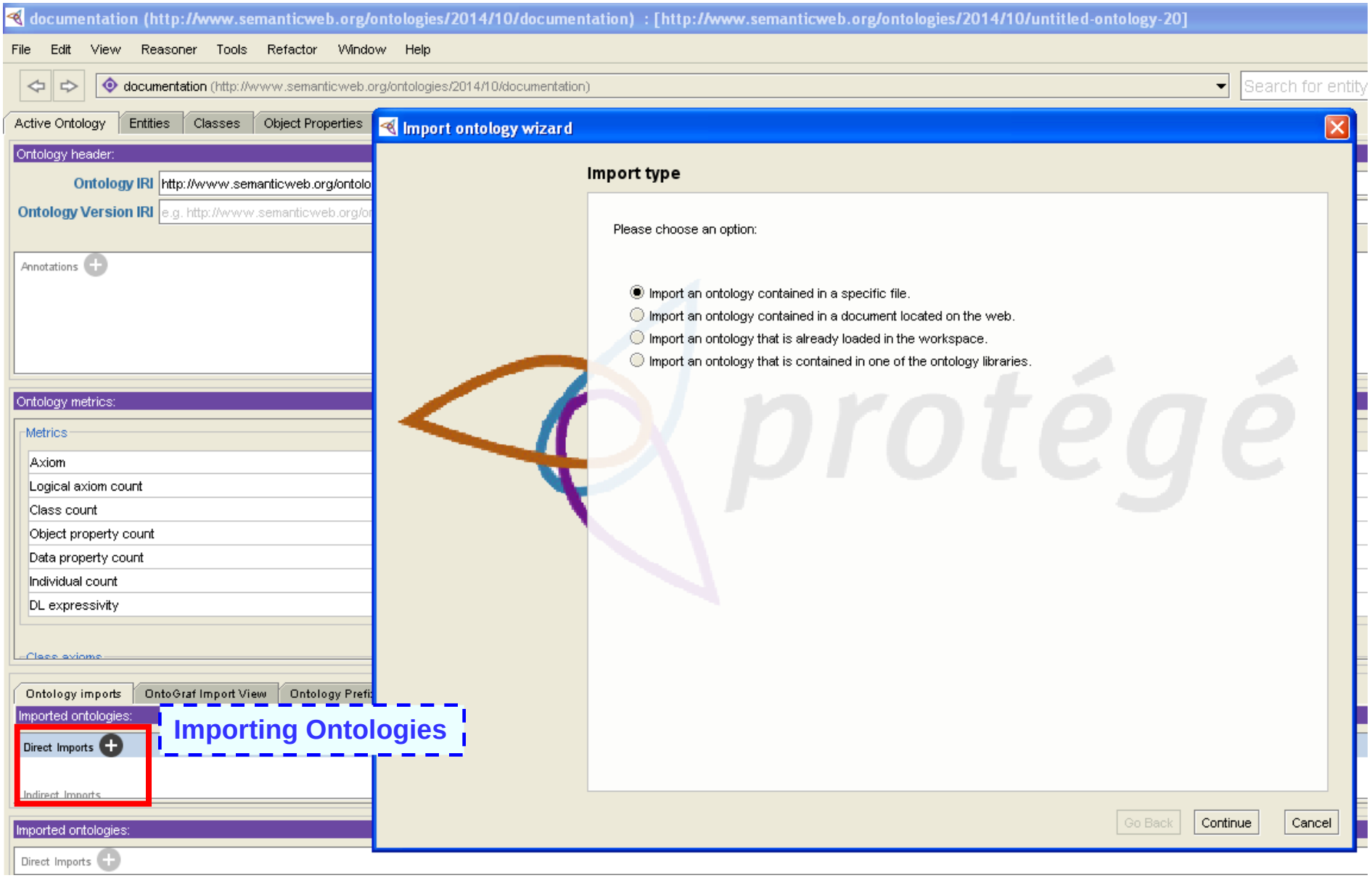
Example: DC in Protégé



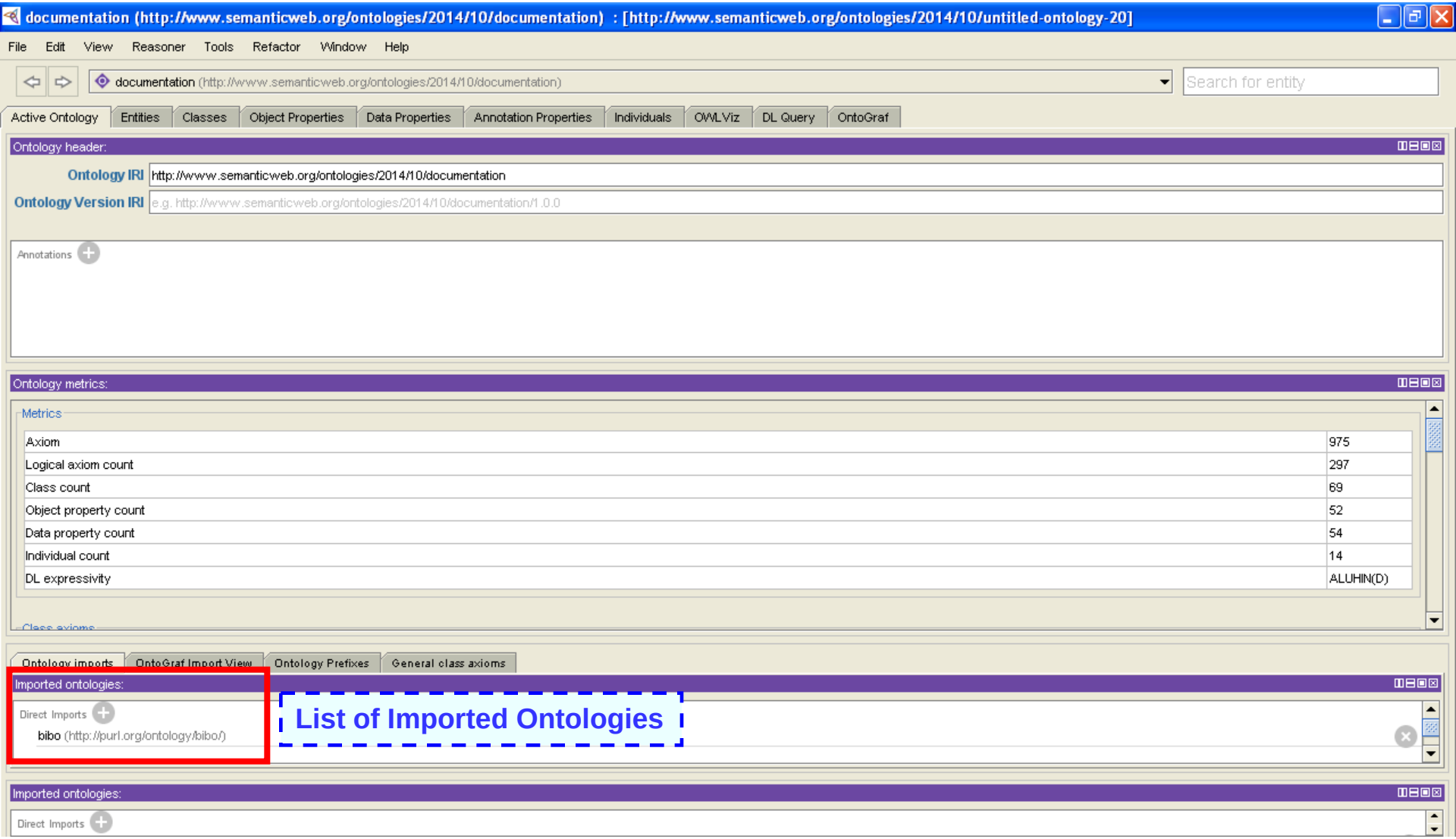
Index

- Brief Introduction
- Main Modelling Components
- **How to use it**
  - Importing and Extending Ontologies
  - Including Equivalences
  - Importing Ontology Design Patterns
  - Referencing Elements from other Ontologies

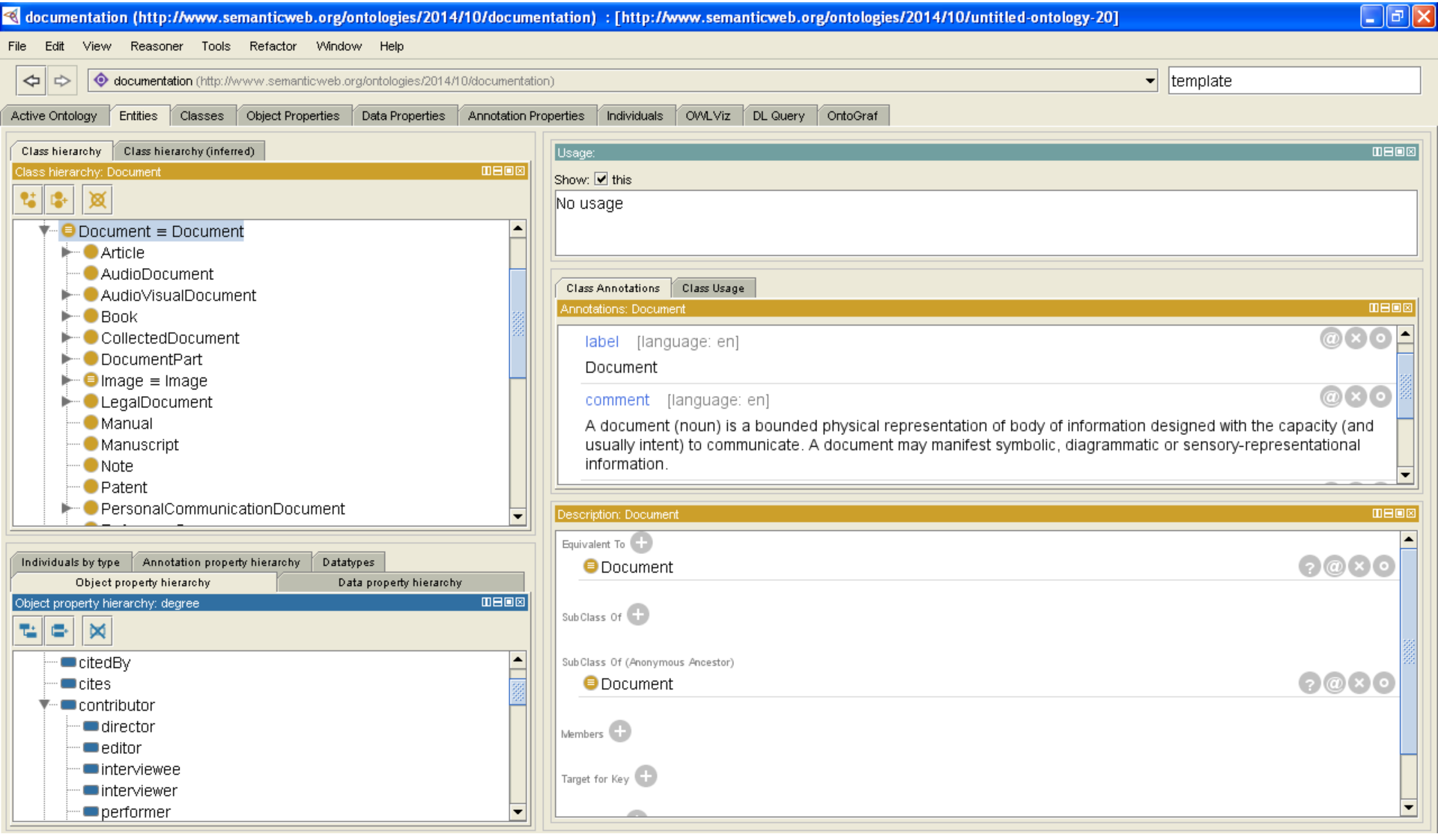
Importing Ontologies (I)



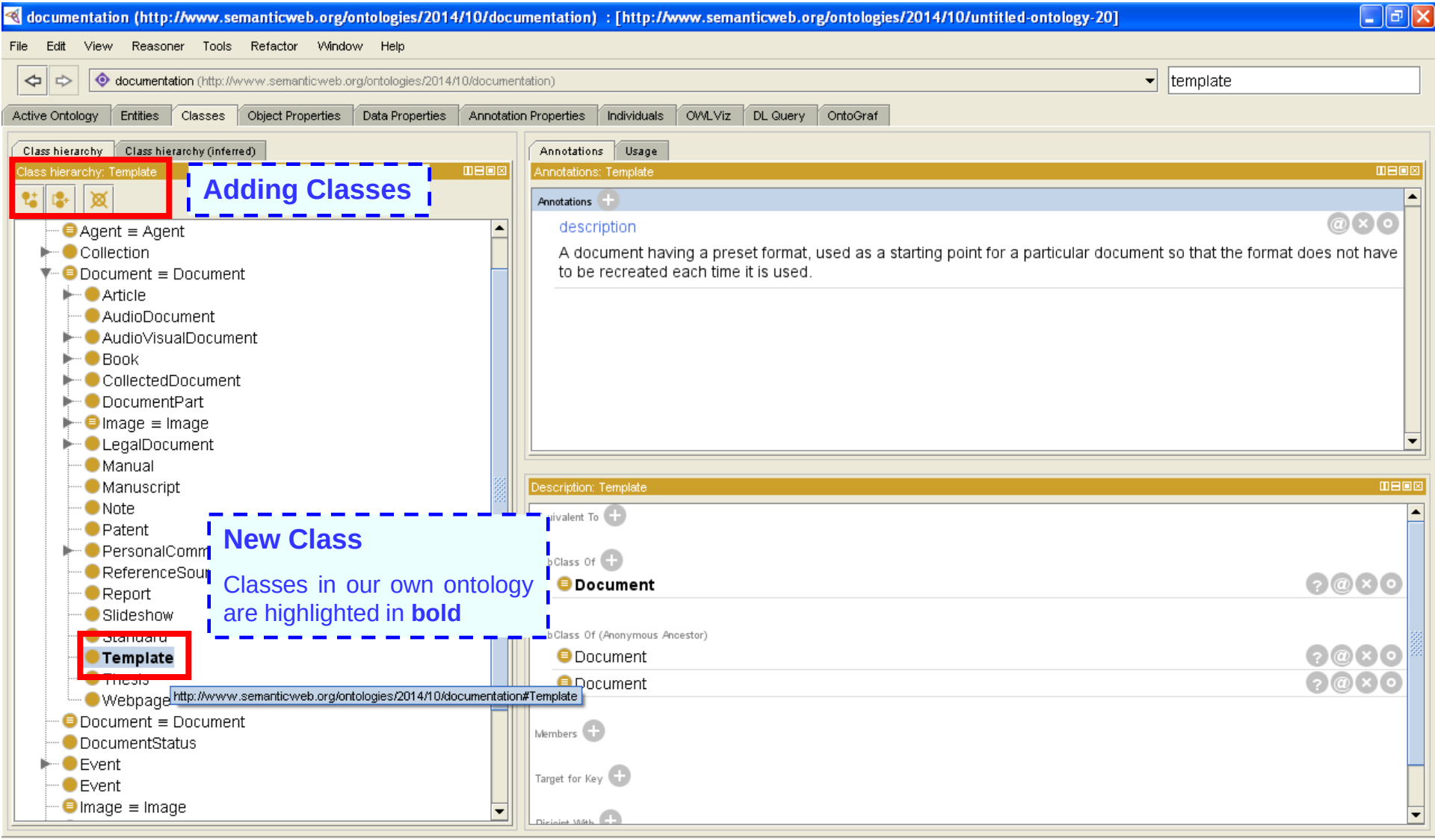
Importing Ontologies (II)



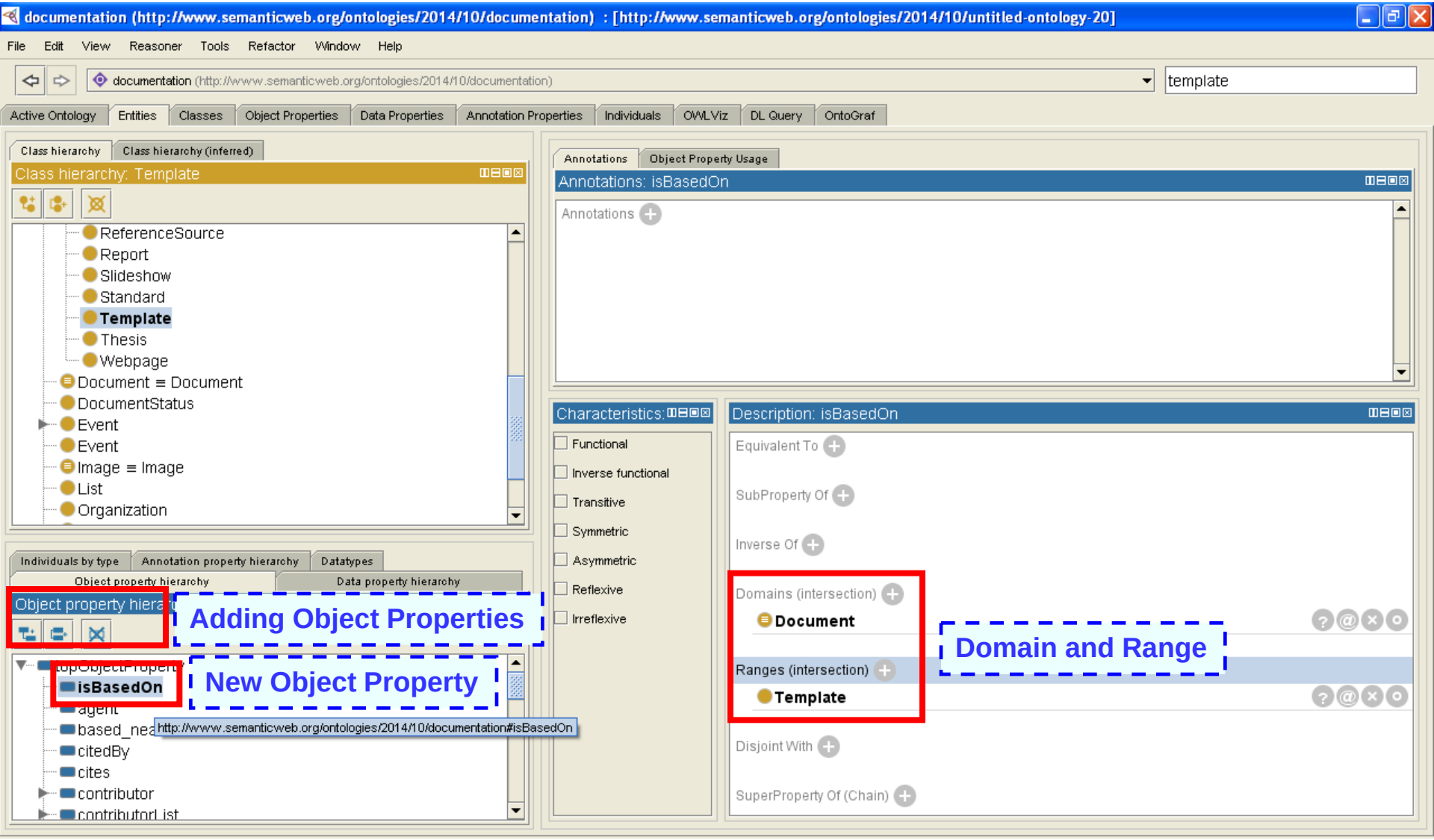
Importing Ontologies (III): Overview of bibo ontology



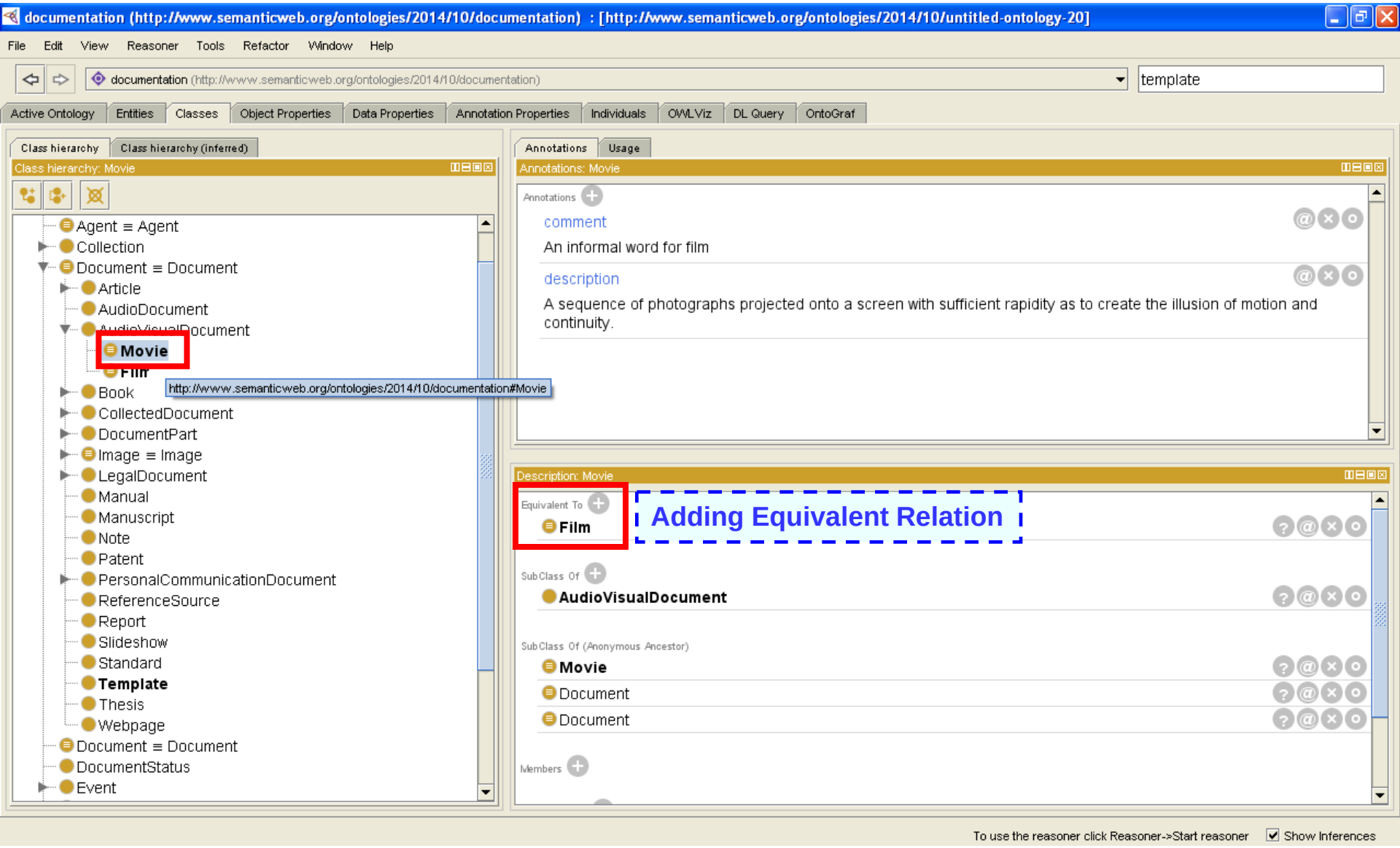
Extending Ontologies (I): Specializing the Document taxonomy



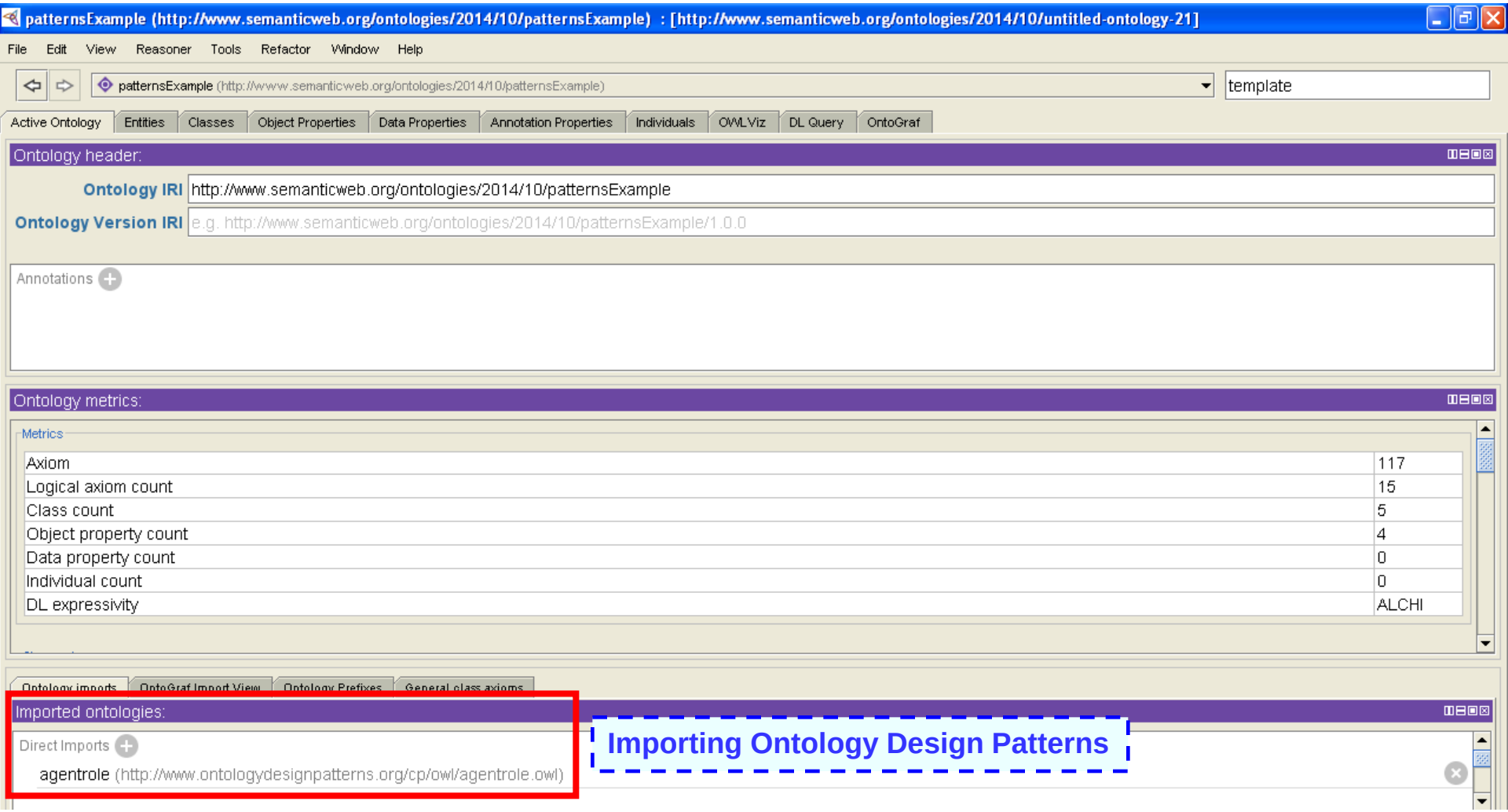
Extending Ontologies (II): Including a new Object Property



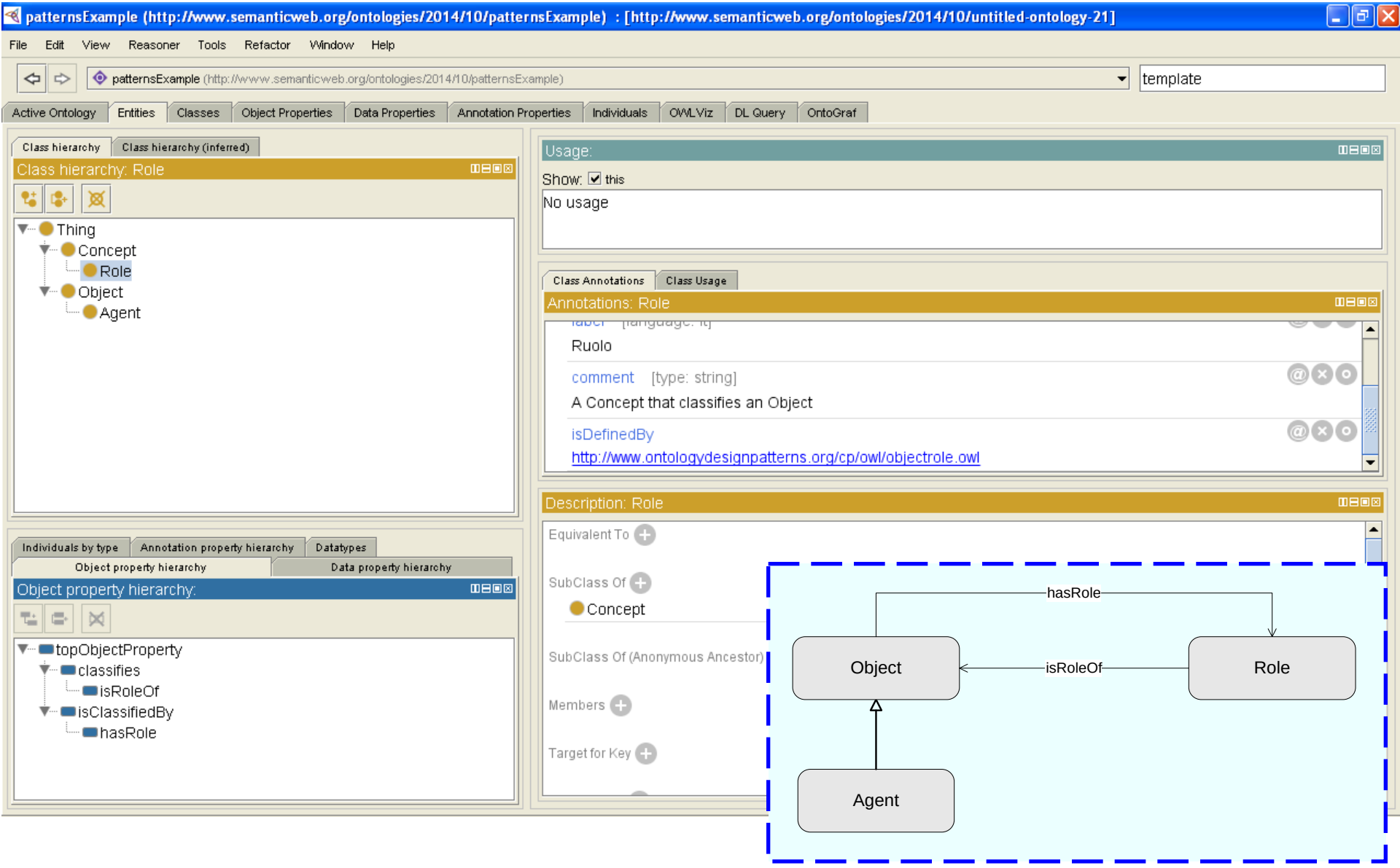
Including Equivalences



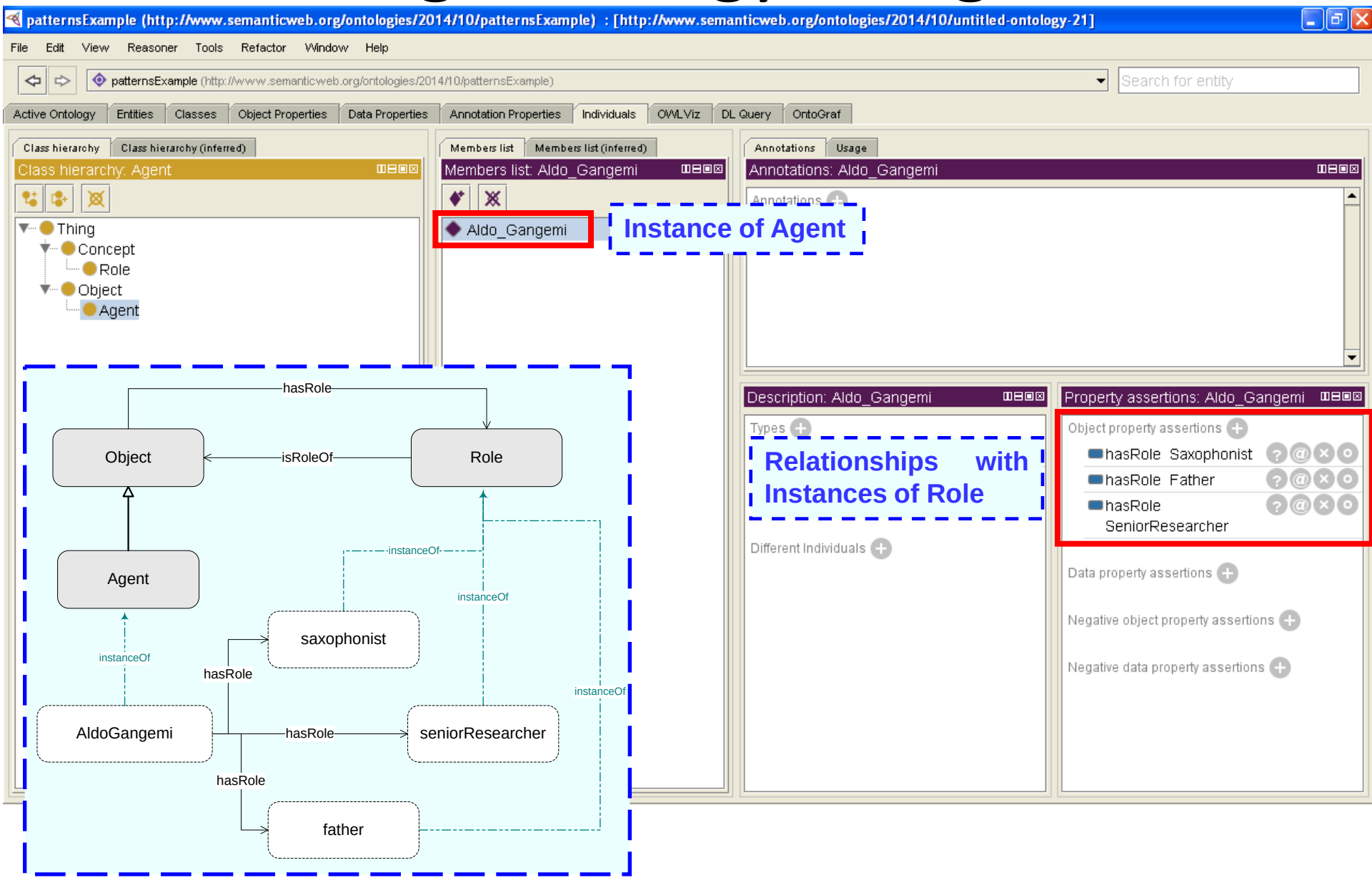
Importing Ontology Design Patterns (I)



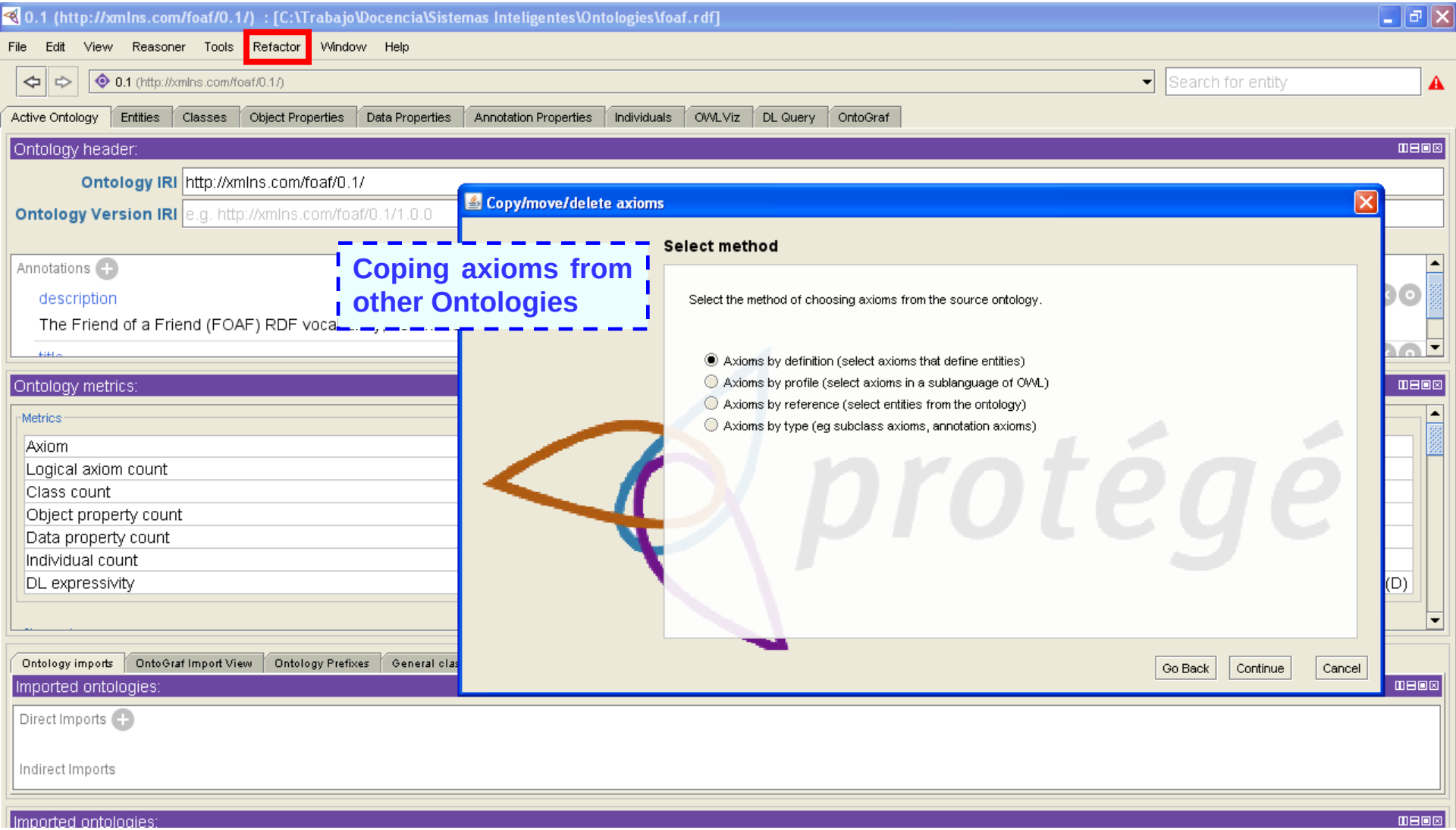
Importing Ontology Design Patterns (II): Overview of the AgentRole Pattern



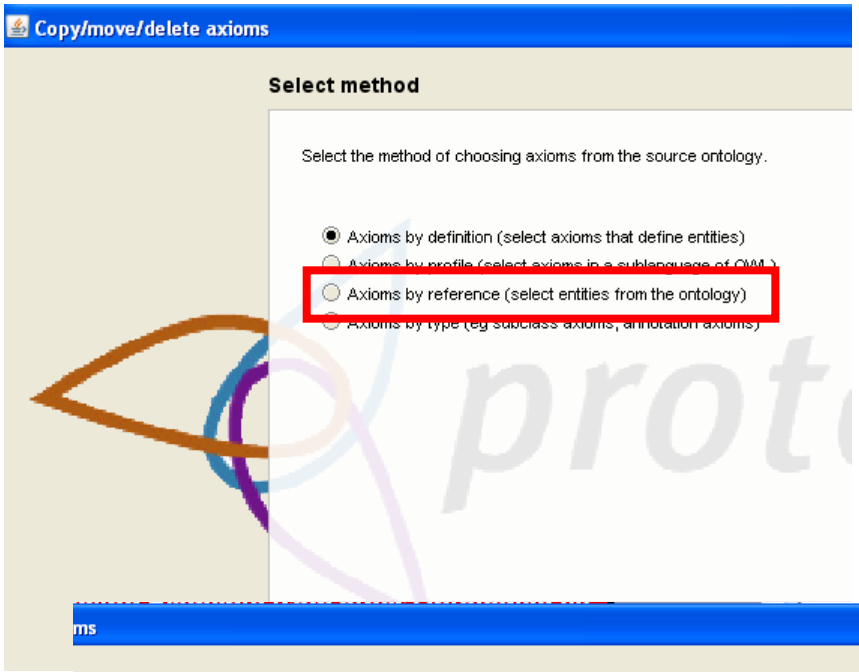
Instanciating Ontology Design Patterns

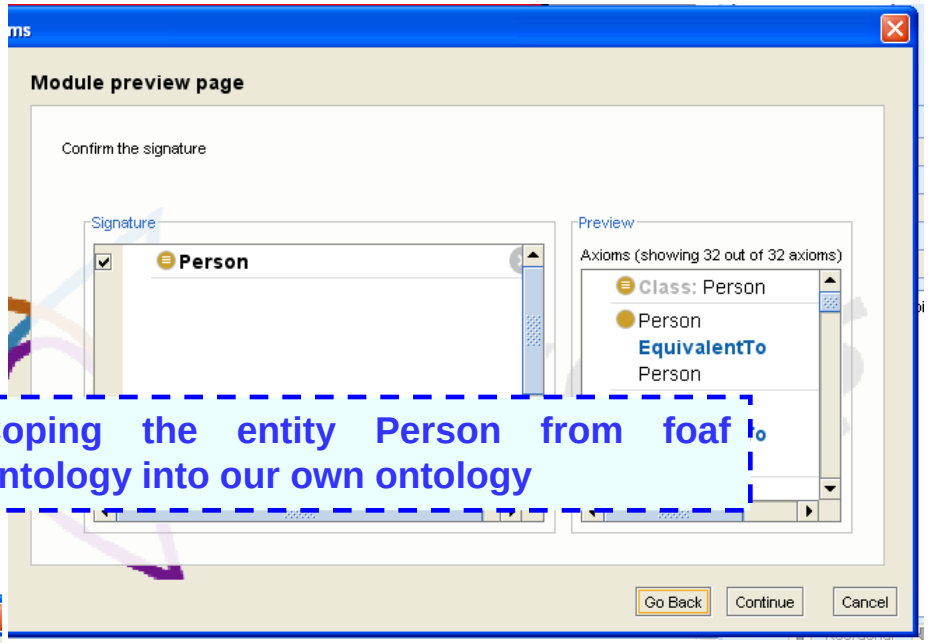


Referencing Elements from other Ontologies (I)

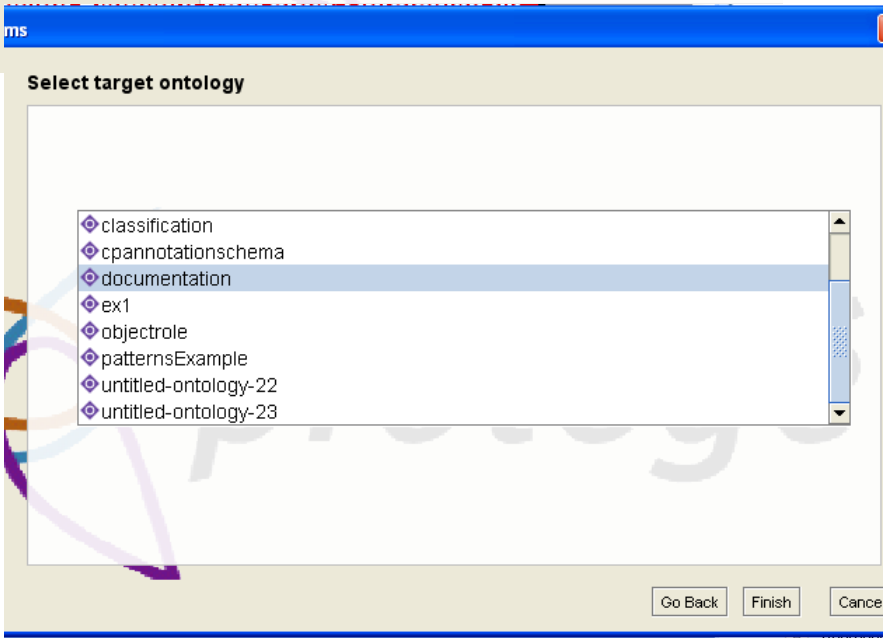


Referencing Elements from other Ontologies (II)



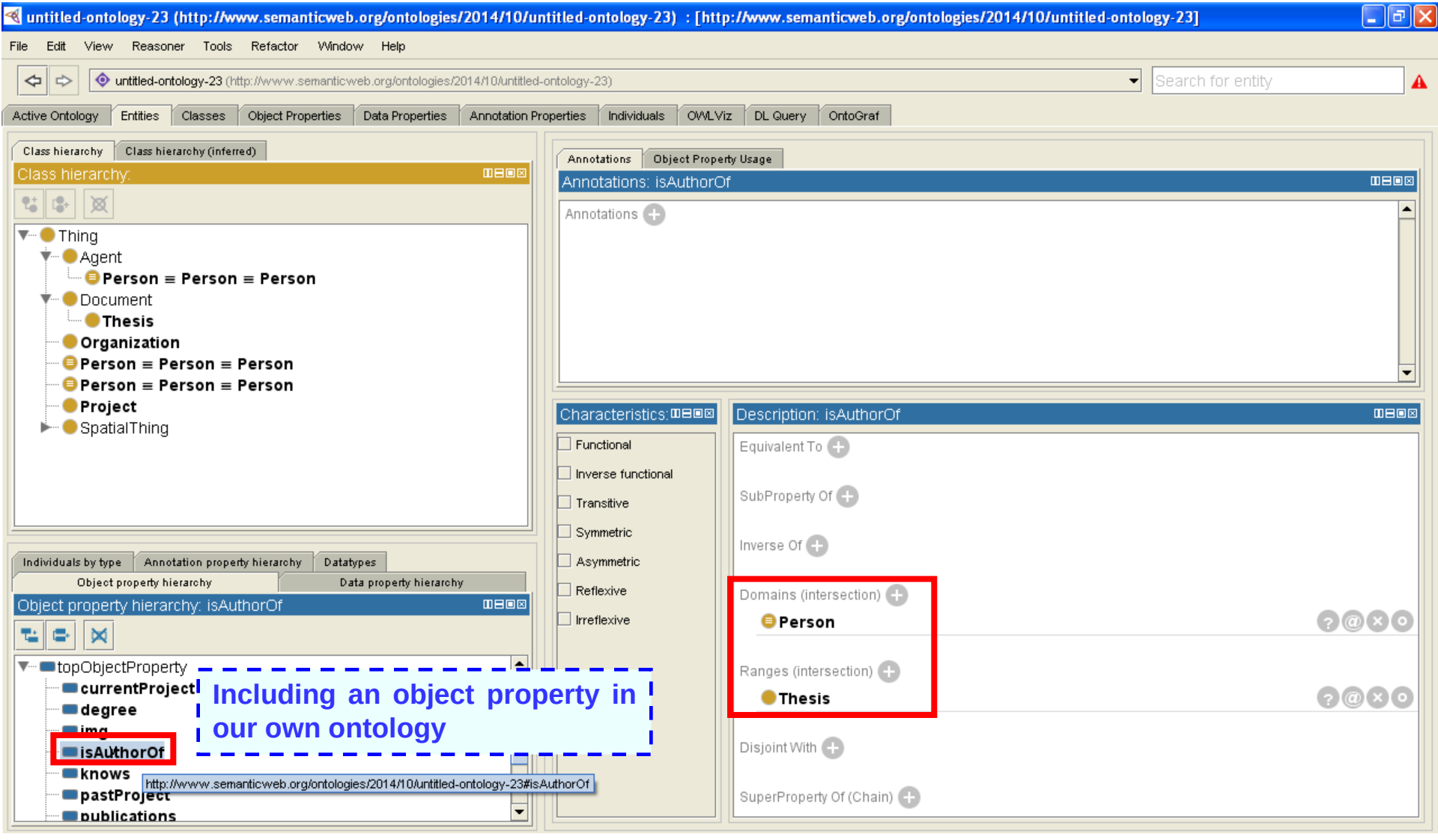


Coping the entity Person from foaf ontology into our own ontology



The same process for coping the entity Thesis from bibo ontology into our own ontology

Referencing Elements from other Ontologies (III)



Course: Intelligent Systems

Unit 3: Ontology Engineering

Using Protégé (Desktop)

Mari Carmen Suárez de Figueroa Baonza

Technical University of Madrid