# COMP3220:
# Document Processing and Semantic Technologies
# Ontology Engineering

Rolf Schwitter

Rolf.Schwitter@mq.edu.au

# Today's Agenda

- OWL 2 DL for Knowledge Representation
- Ontology Engineering
- A Sketch of a Pizza Ontology
- Ontology Editor: Protégé
- Creating the Pizza Ontology
- HermiT OWL Reasoner

# OWL 2 DL for Knowledge Representation

- OWL 2 DL allows us to represent domain knowledge.
- The basic notions are:
  - Axioms:

    the basic statement that an OWL ontology expresses
  - Entities:

    elements (classes, individuals, properties) to describe real-world objects
  - Expressions:

    combinations of entities to form complex descriptions from basic ones using constructors.

# Ontology Engineering

- There is no one correct way to model a domain.
- The best solution always depends on the application.
- Ontology development is an iterative process.
- Concepts in the ontology should be close to objects and relationships in the application domain.
- Good naming convention is important.
- Nouns usually describe objects.
- Verbs usually describe relationships.

# An Knowledge-Engineering Methodology

1. Determine the domain and scope of the ontology.
2. Consider reusing existing ontologies.
3. Concepts and relationships should reflect the domain.
4. Enumerate important terms for the ontology.
5. Stick to a naming convention.
6. Define classes and the class hierarchy.
7. Define properties of classes.
8. Define cardinality constraints.
9. Define domain and range restrictions.
10. Create instances.

# A Sketch of a Pizza Ontology

- Main classes:
  - **`Food`**
    - **`IceCream`**
    - **`Pizza`**
    - **`PizzaBase`**
    - **`PizzaTopping`**

# A Sketch of a Pizza Ontology

- Initial class hierarchy:
  - **`Pizza`**
    - **`NamedPizza`**
      - **`Margherita`**
  - **`PizzaBase`**
    - **`DeepPanBase`**
    - **`ThinAndCrispyBase`**
  - **`PizzaTopping`**
    - **`CheeseTopping`**
    - **`….`**

# A Sketch of a Pizza Ontology

- A list of properties and their domain and range:
  - **hasCountryOfOrigin**
  - **hasIngredient** **domain: Food** **range: Food**
    - **hasBase** **domain: Pizza** **range: PizzaBase**
    - **hasTopping** **domain: Pizza** **range: PizzaTopping**
  - **hasSpiciness** **range: Spiciness**
  - **...**

# A Definition in Functional-Style Syntax

Definition of `VegetarianPizza` in functional-style syntax:

```
EquivalentClasses(:VegetarianPizza
  ObjectIntersectionOf(:Pizza
    ObjectComplementOf(ObjectSomeValuesFrom(
        :hasTopping :SeafoodTopping))
    ObjectComplementOf(ObjectSomeValuesFrom(
        :hasTopping :MeatTopping))))
```

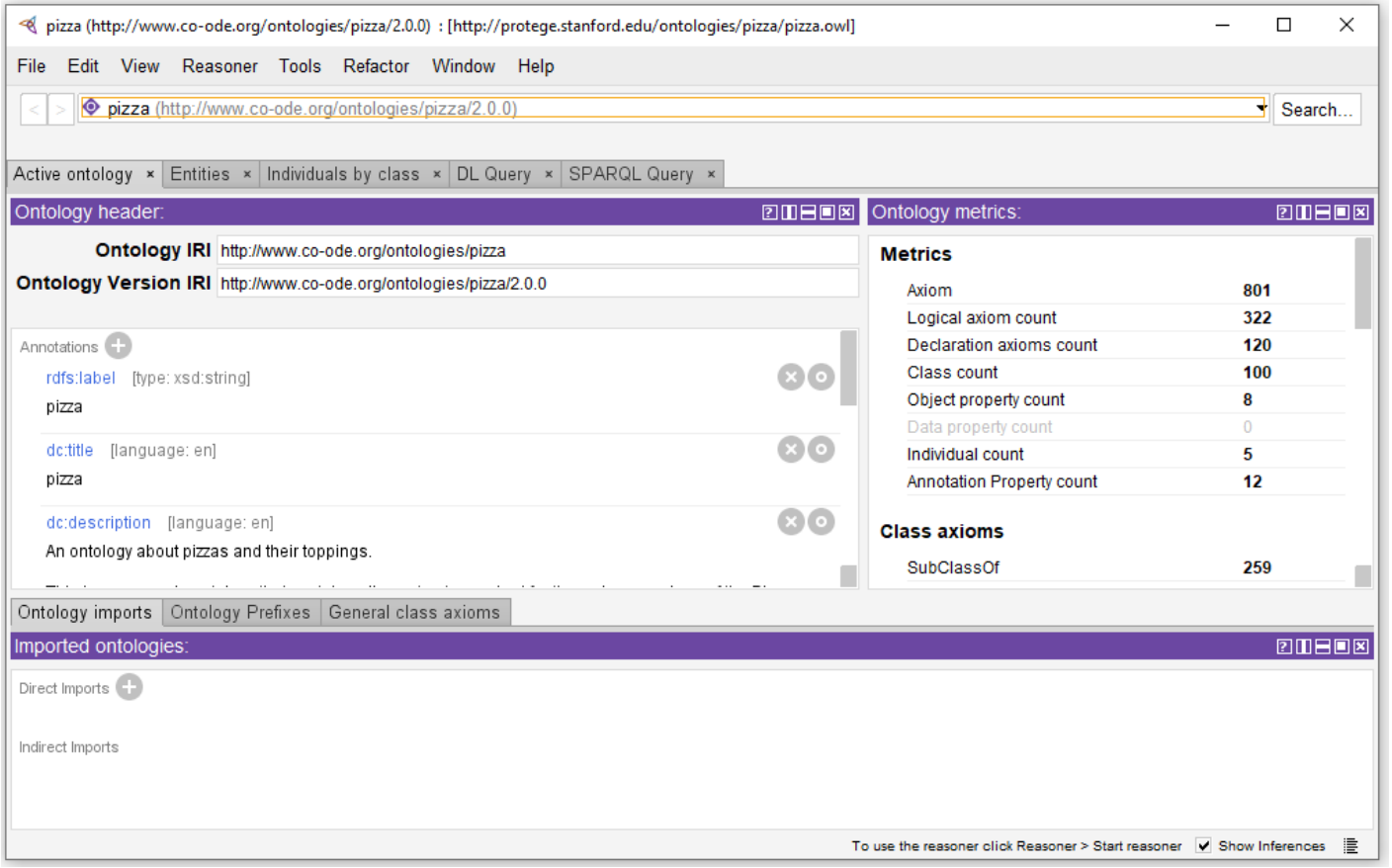
- A simpler representation in Manchester syntax:

```
Class: VegetarianPizza
EquivalentTo: Pizza
                and(not(hasTopping some SeafoodTopping))
                and(not(hasTopping some MeatTopping))
```

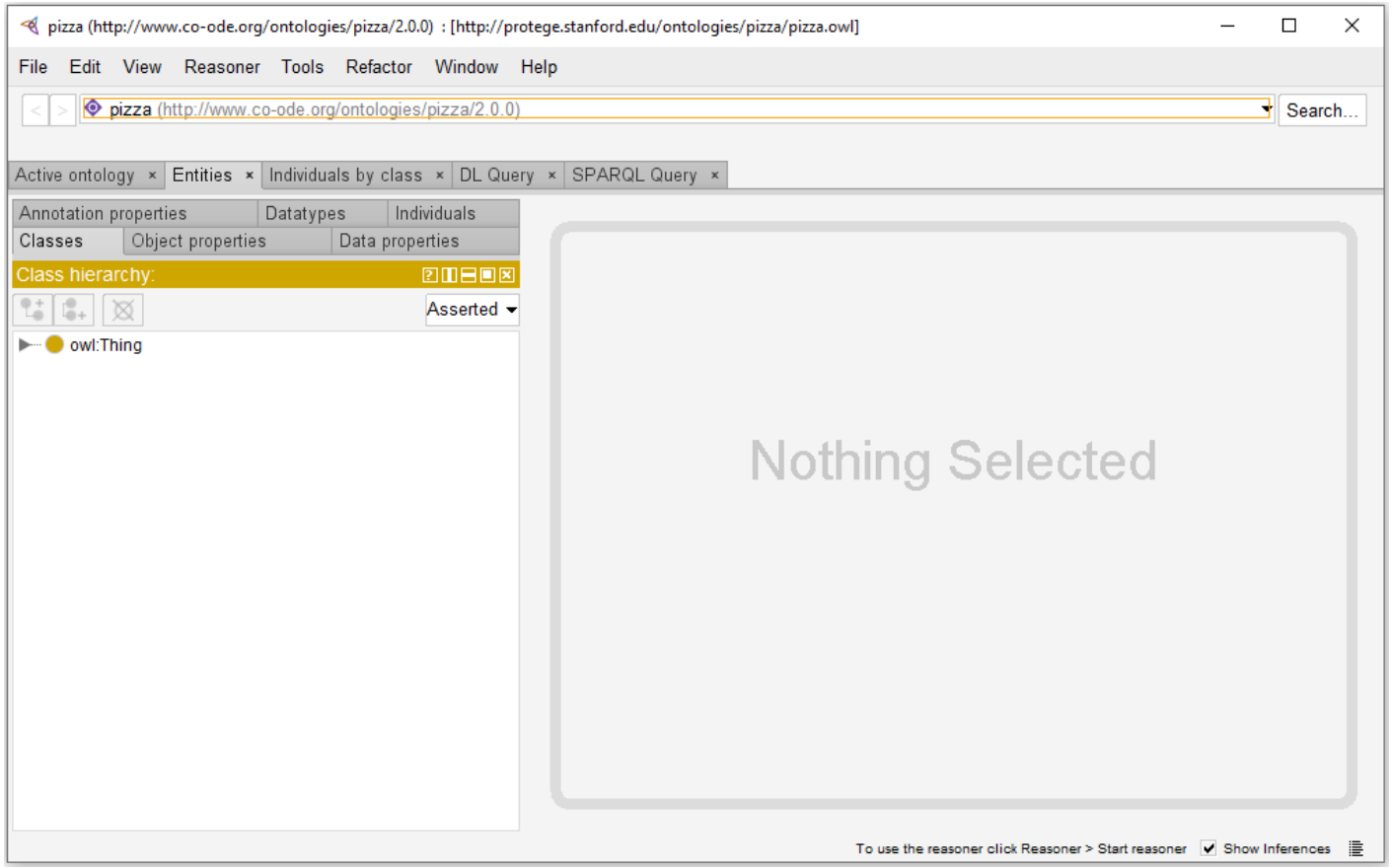
# Protégé

- Protégé is a free open source ontology editor:
  - `http://protege.stanford.edu/`
- Protégé fully supports OWL 2.
- Protégé ontologies can be exported in RDF/XML and many other formats.
- Protégé is implemented in Java.
- The editor provides a plug-and-play environment.
- There exists also a web-based version: WebProtégé.
- Protégé has about 366,000 users.

# Start Protégé



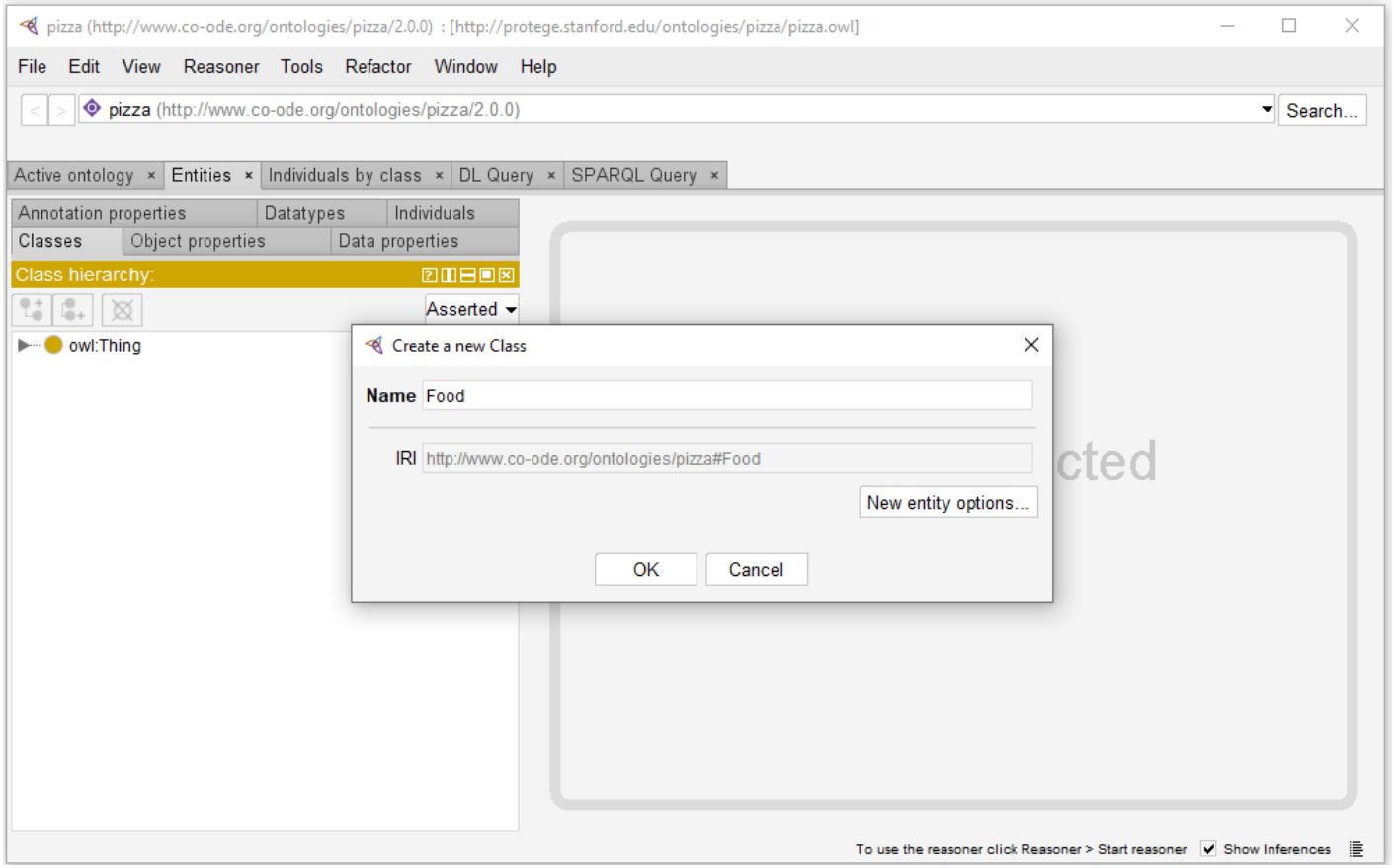
# Class Hierarchy



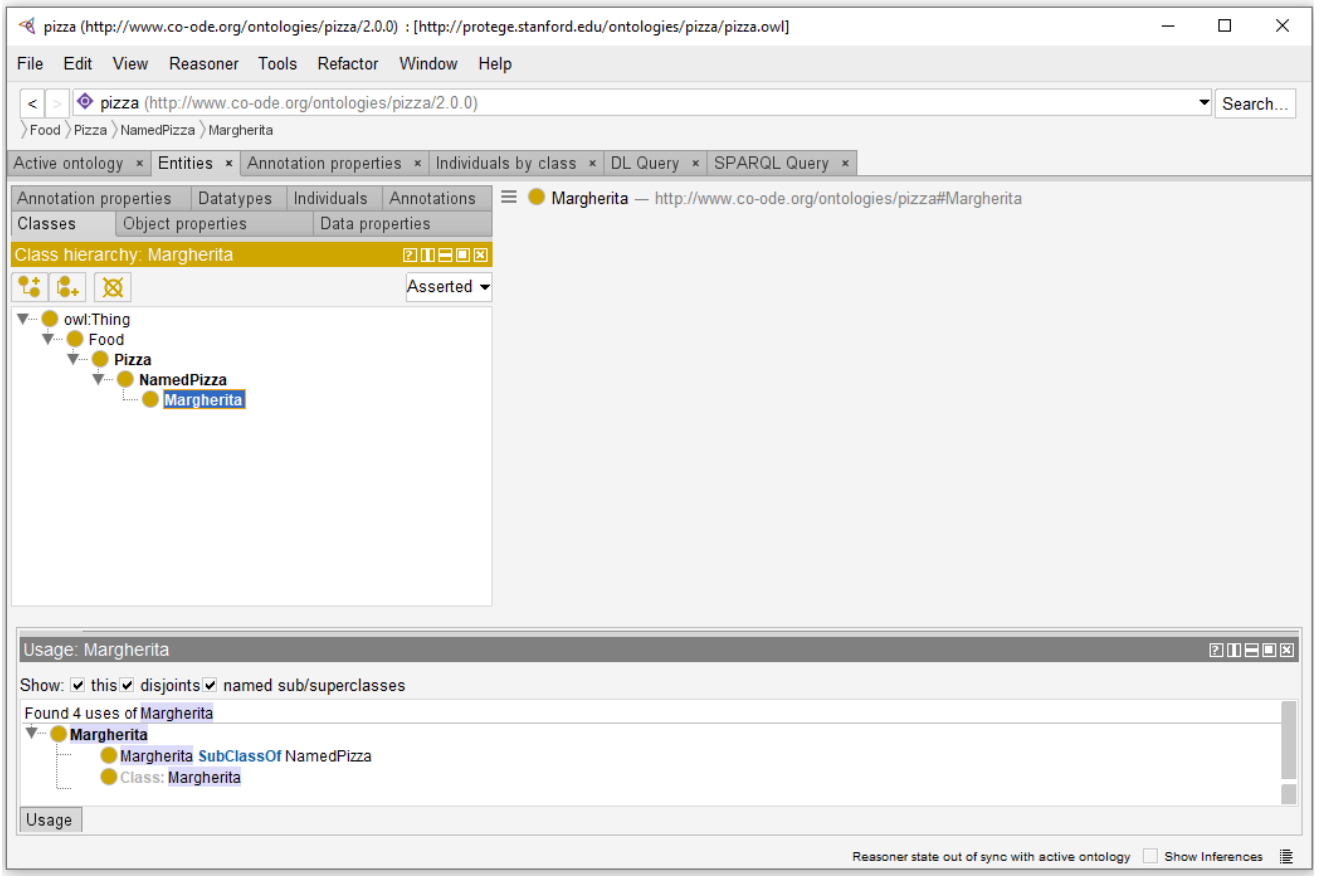
# OWL Pizza Ontology

- The OWL Pizza Ontology is an ontology that has been built for teaching and demo purposes:
  http://protege.stanford.edu/ontologies/pizza/pizza.owl
- The pizza ontology is available in RDF/XML and can be loaded directly into Protégé.
- The pizza ontology can be exported, for example in:
  - OWL Functional Syntax
  - Manchester OWL Syntax
  - Turtle
  - JSON-LD

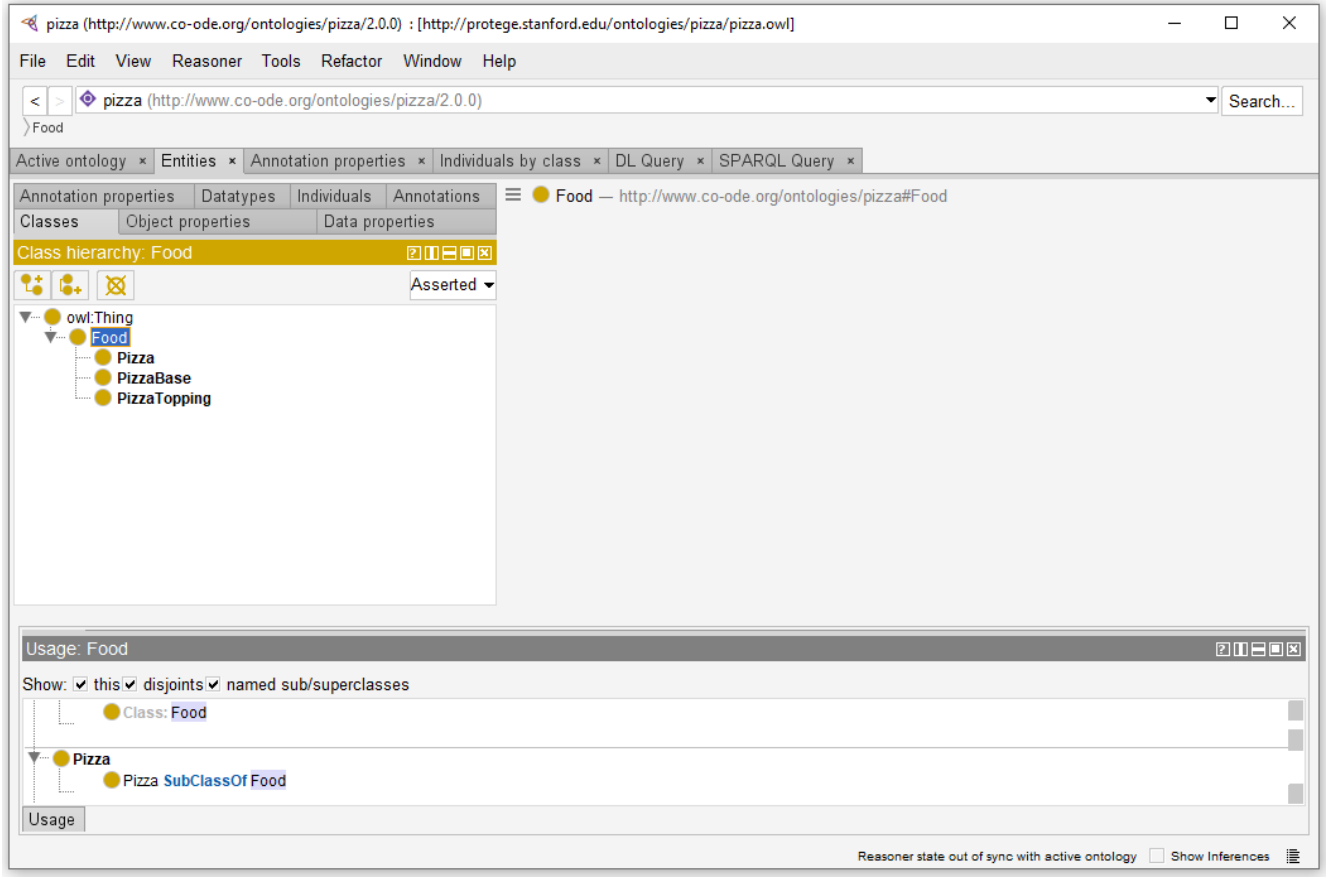
# Creating the Top Hierarchy



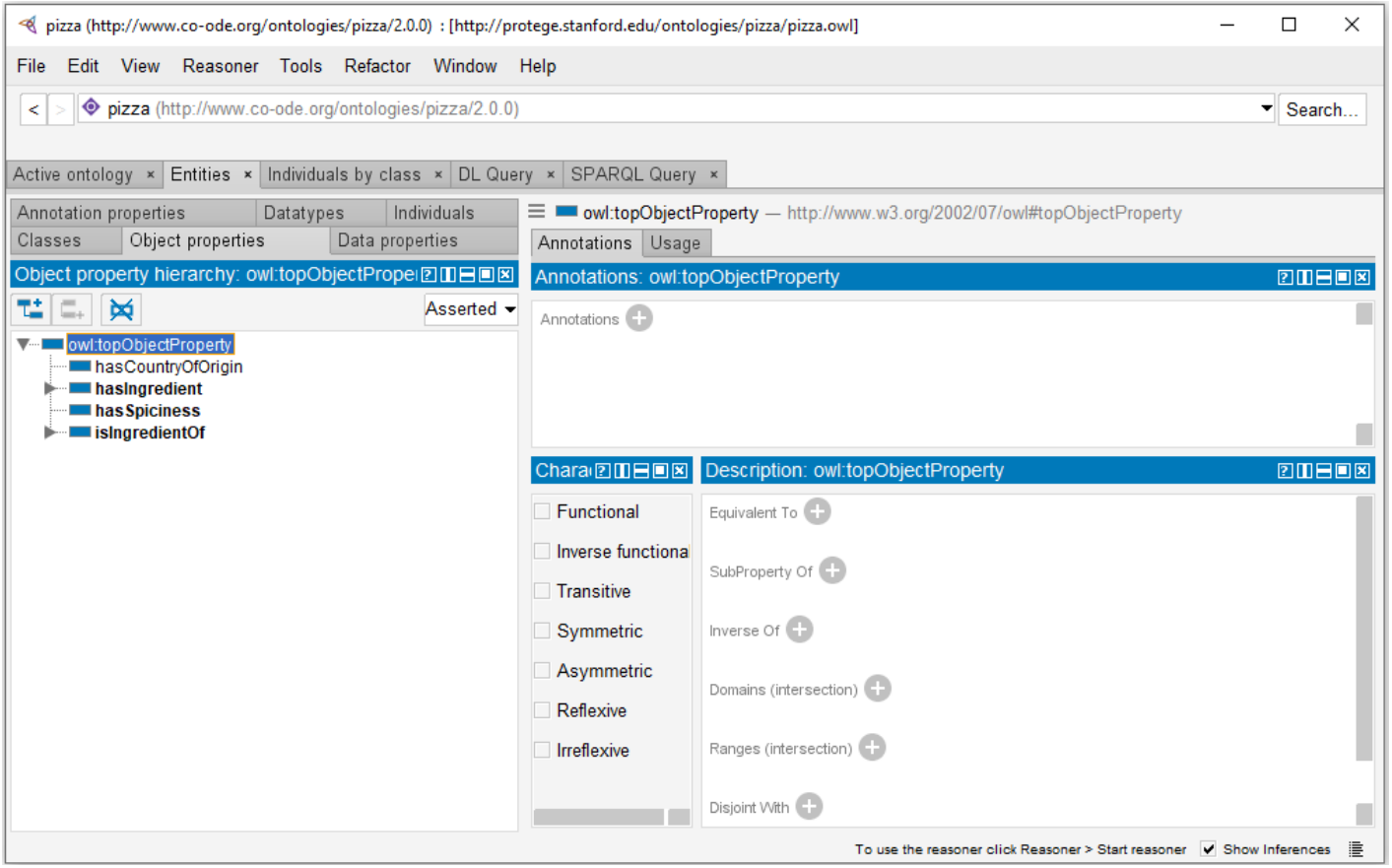
# Creating each Branch of the Taxonomy



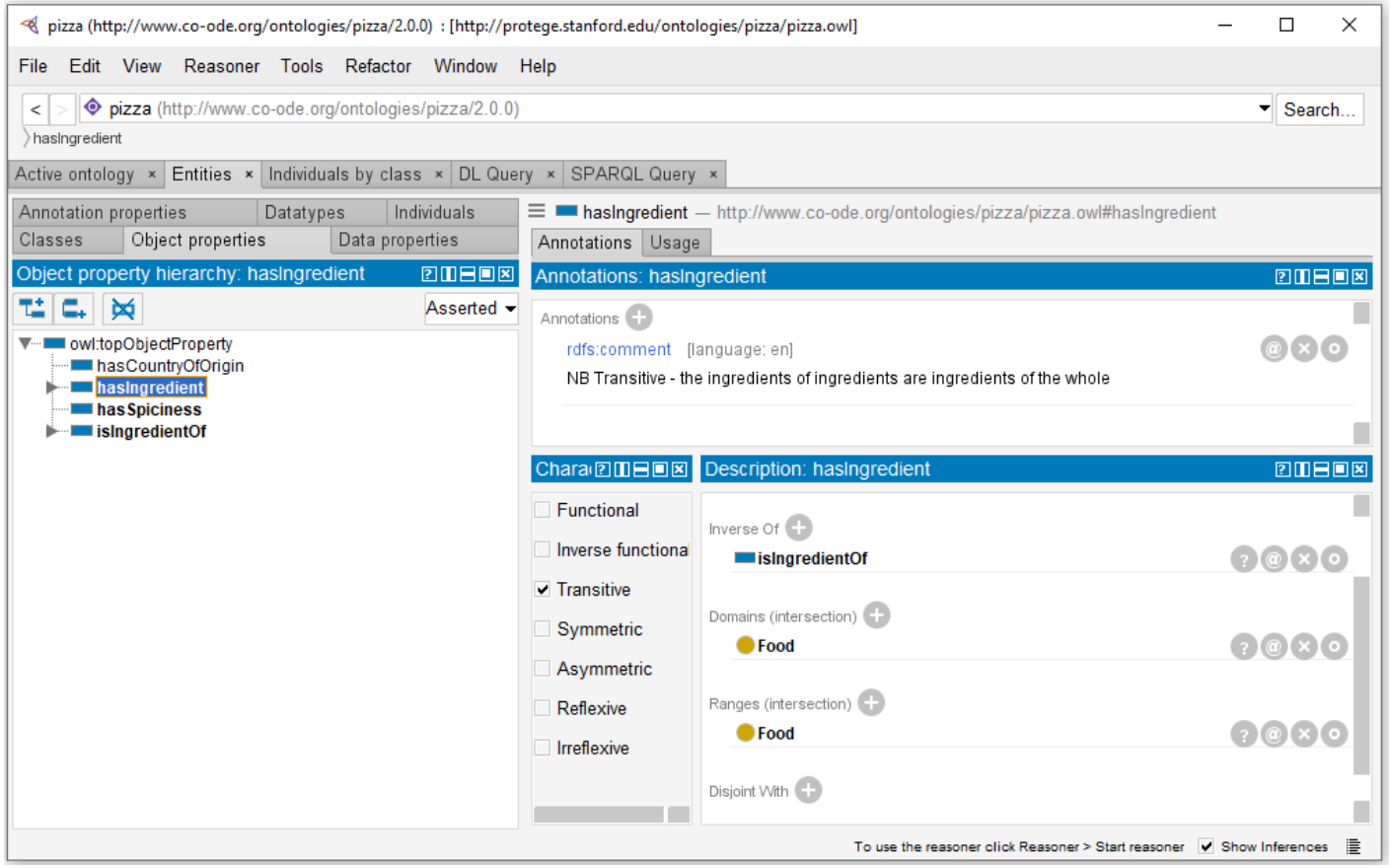
# Creating each Branch of the Taxonomy



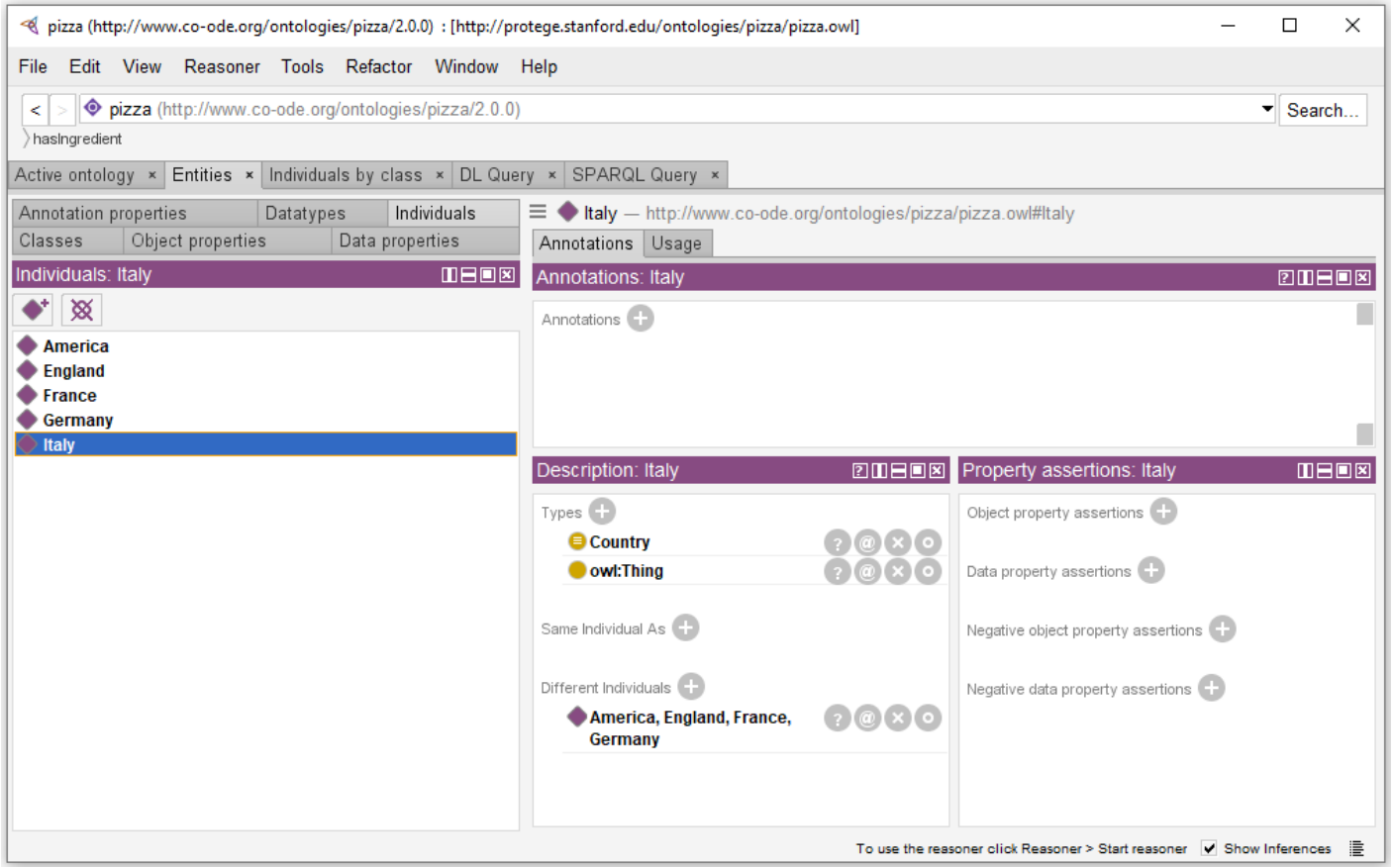
# Creating Object Properties



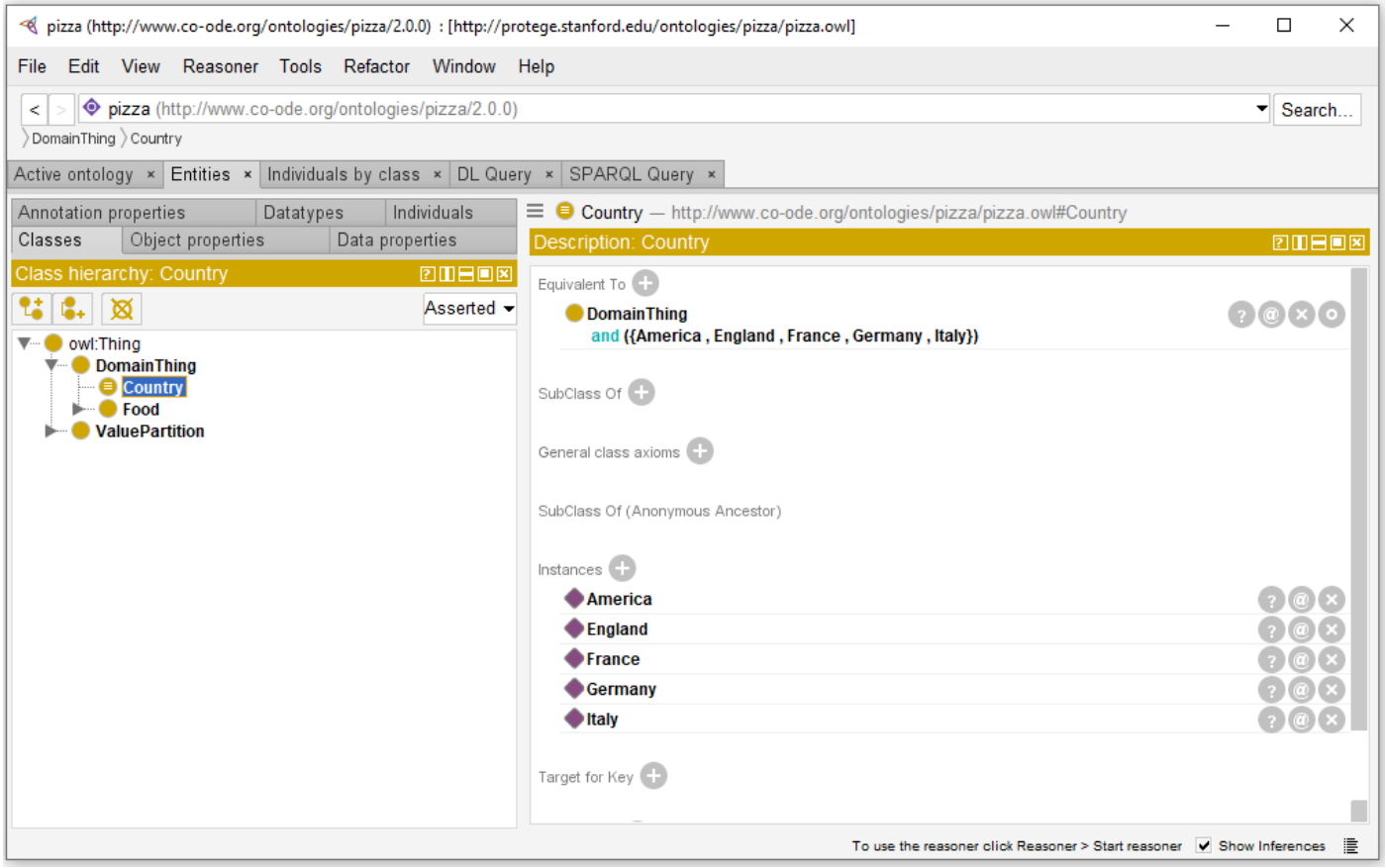
# Adding Property Characteristics



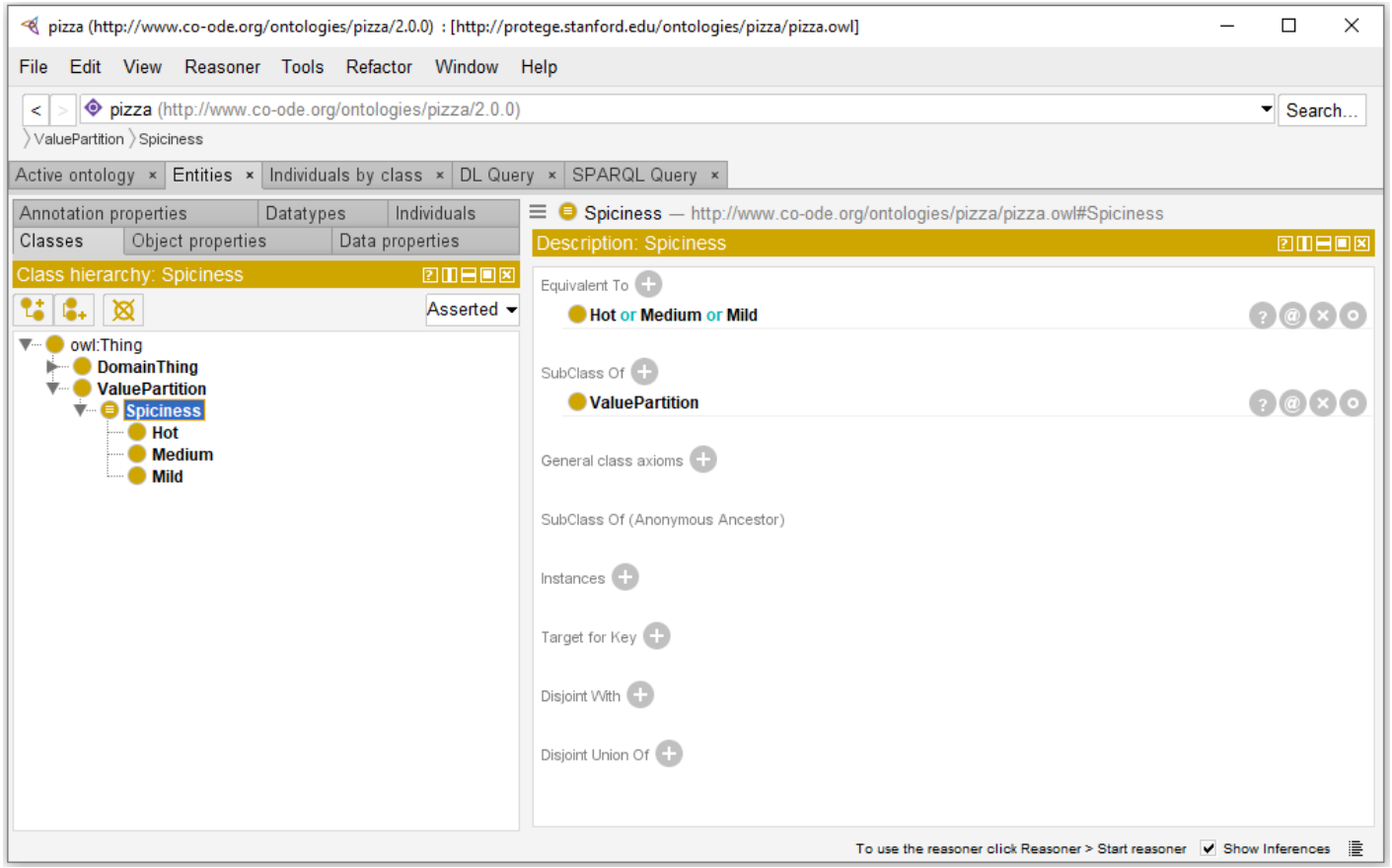
# Adding Different Individuals



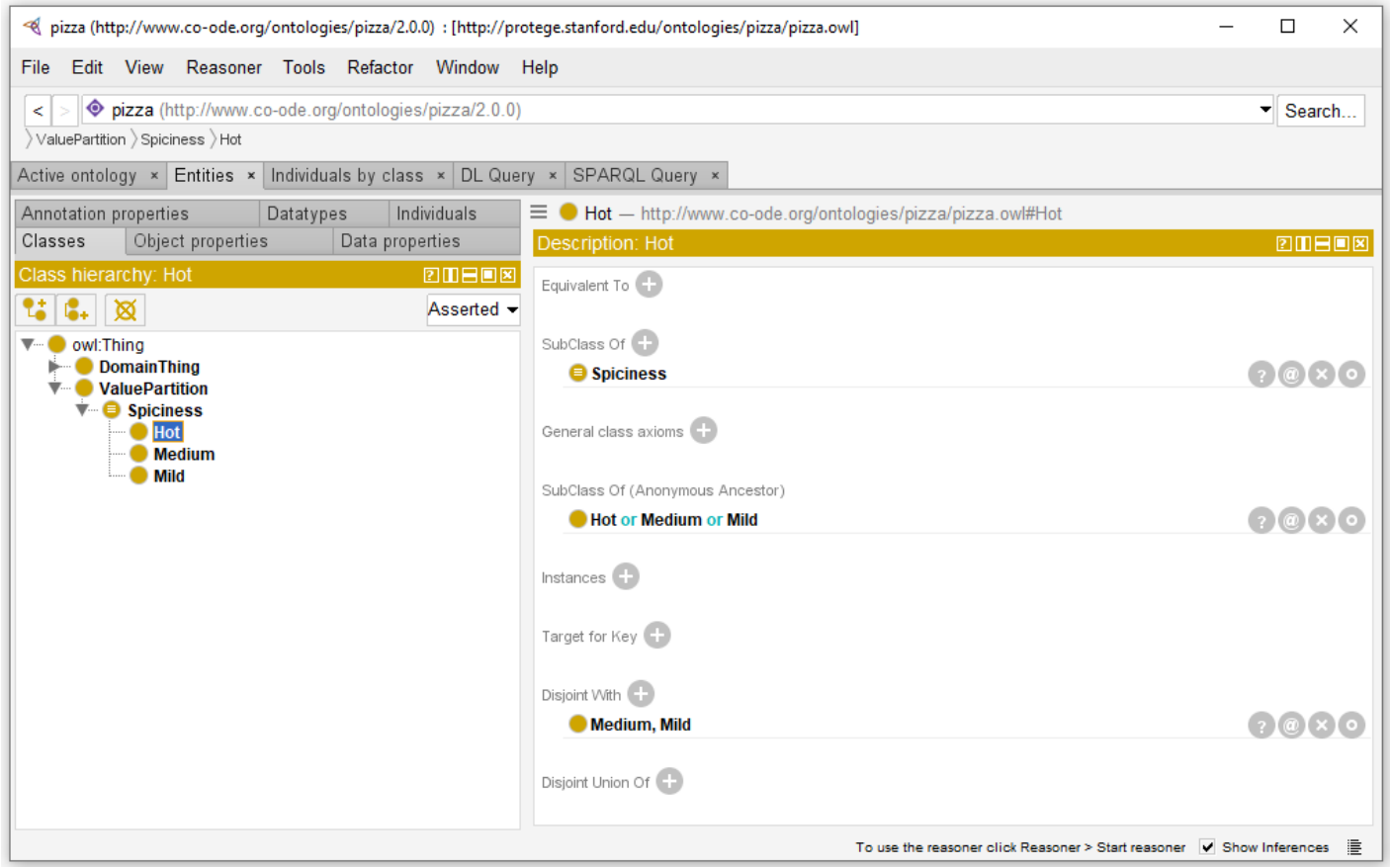
# Adding `DomainThing` and Defining `Country`



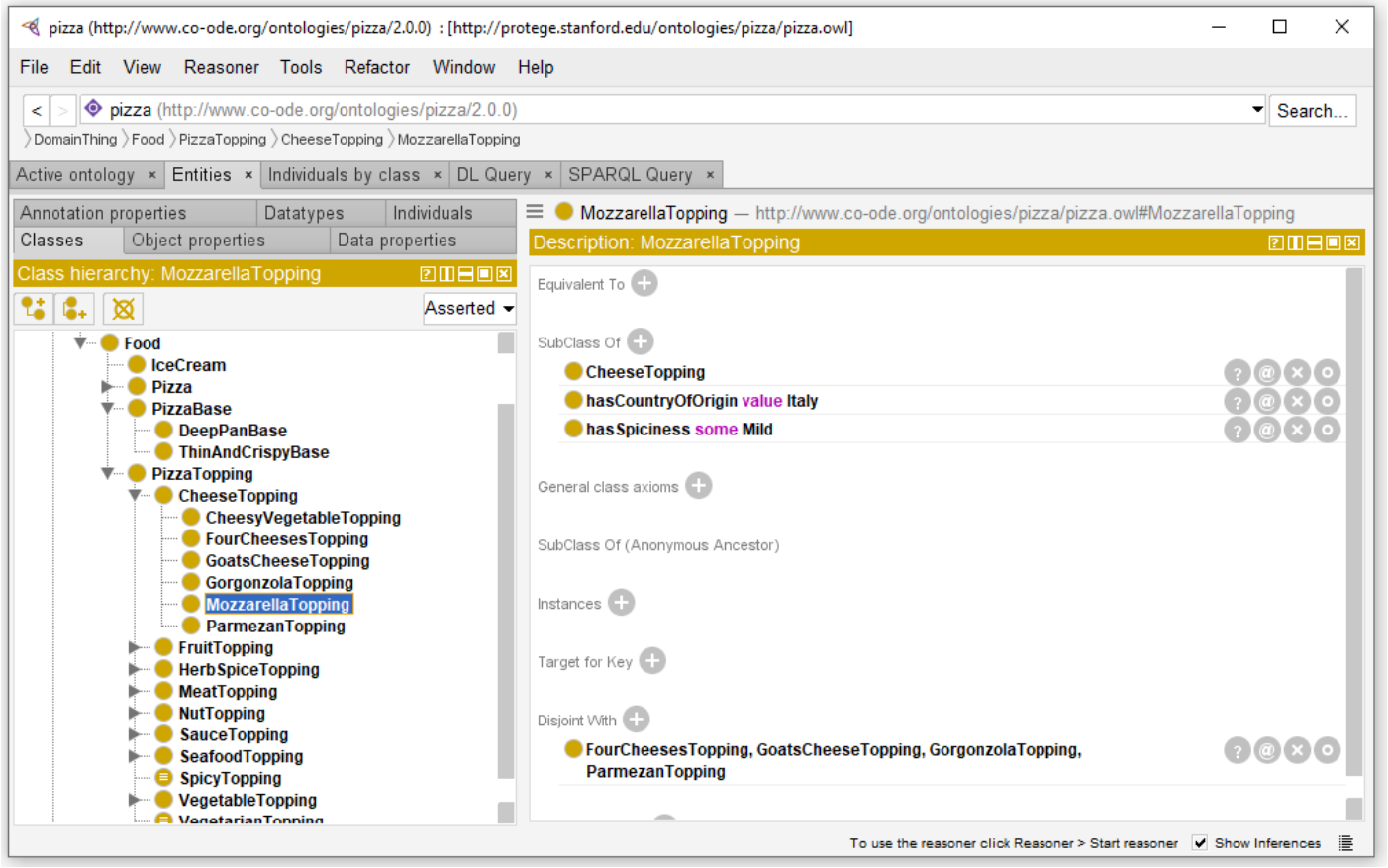
# Adding `ValuePartition` and Defining `Spiciness`



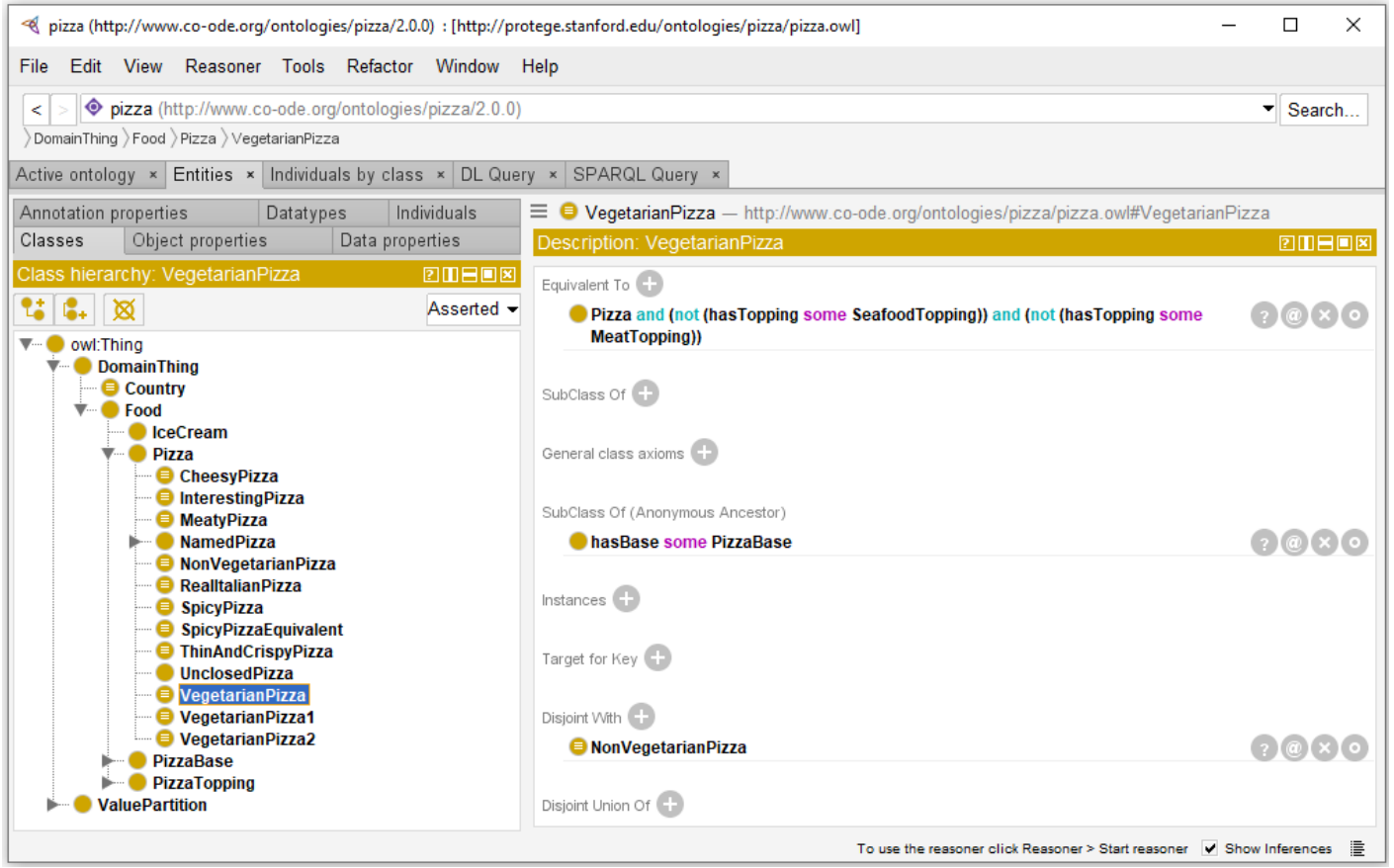
# Specifying `Hot`



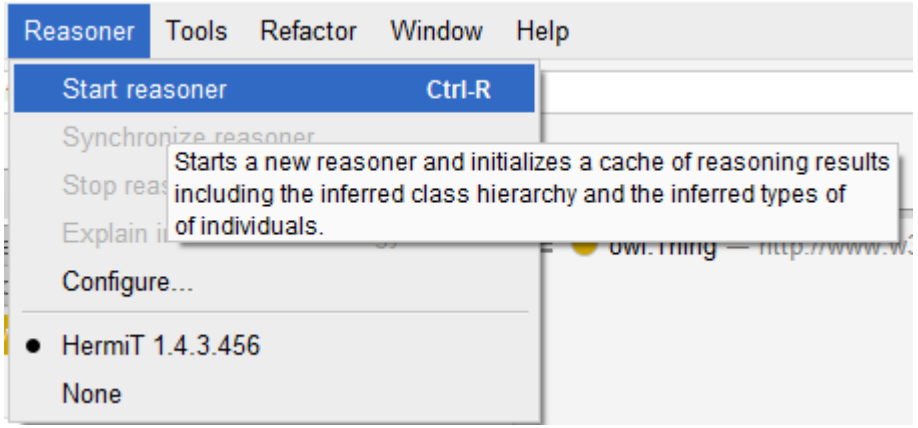
# Describing `MozzarellaTopping`



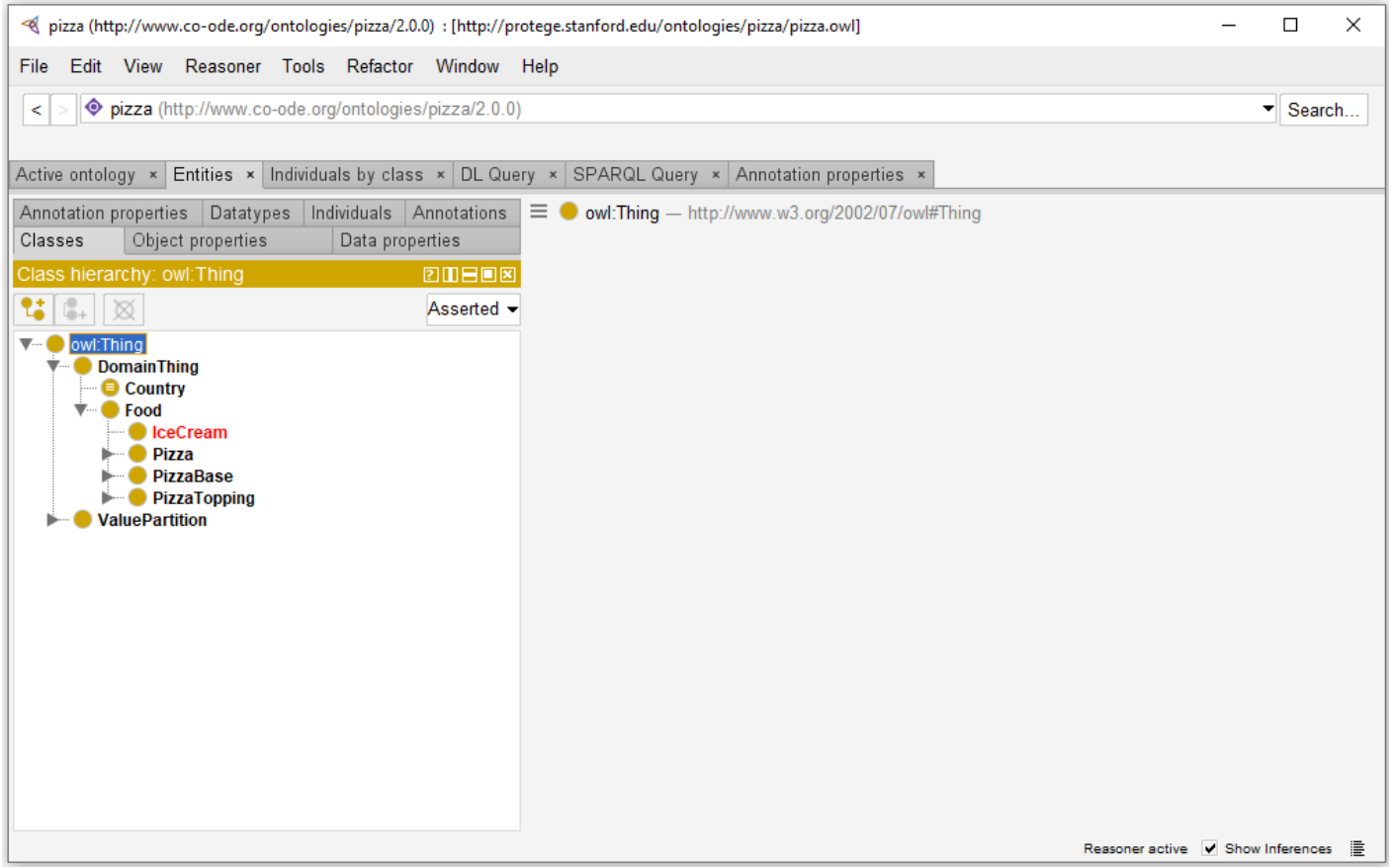
# Defining `VegetarianPizza`



# Start Reasoner



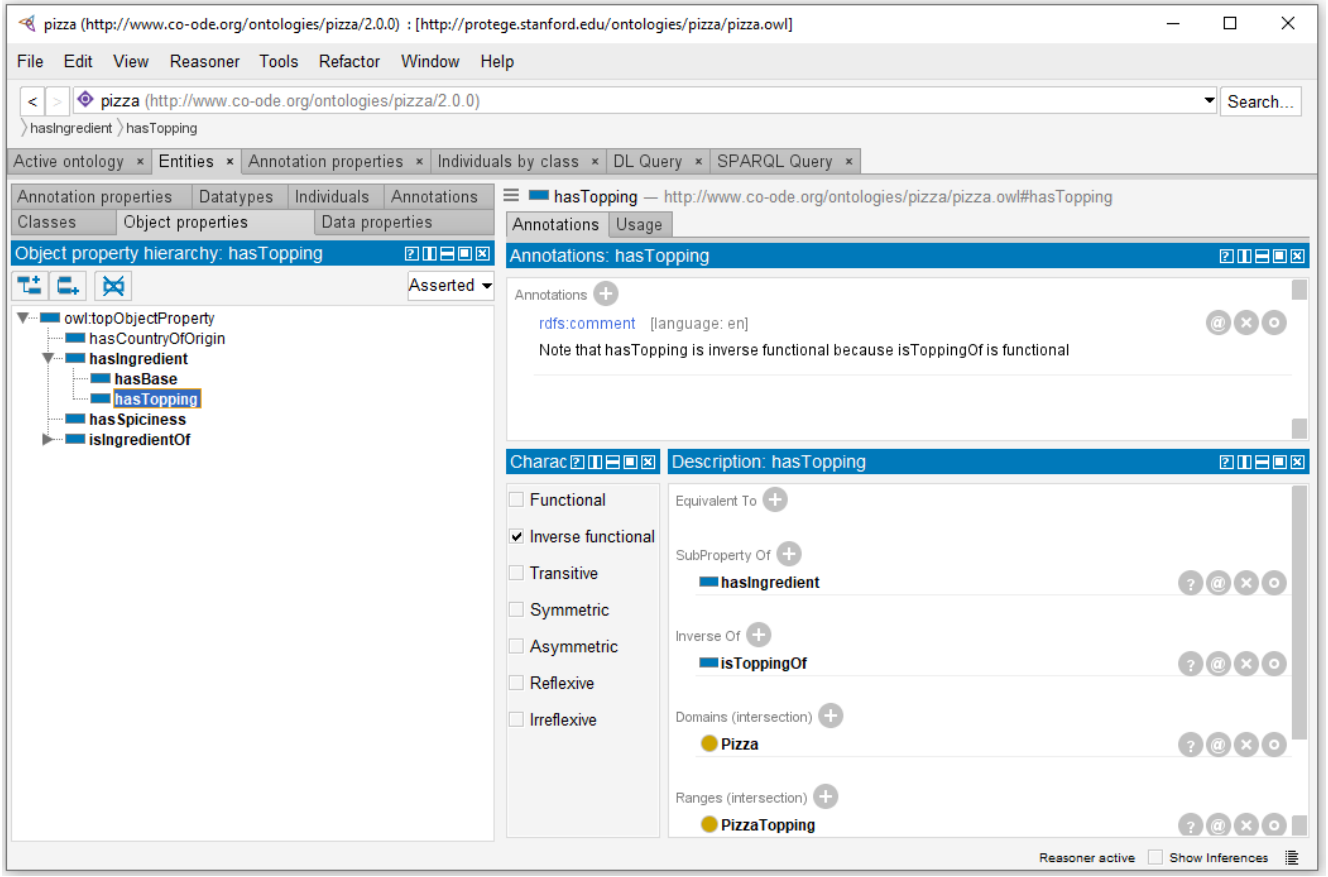
# Mistake: IceCream



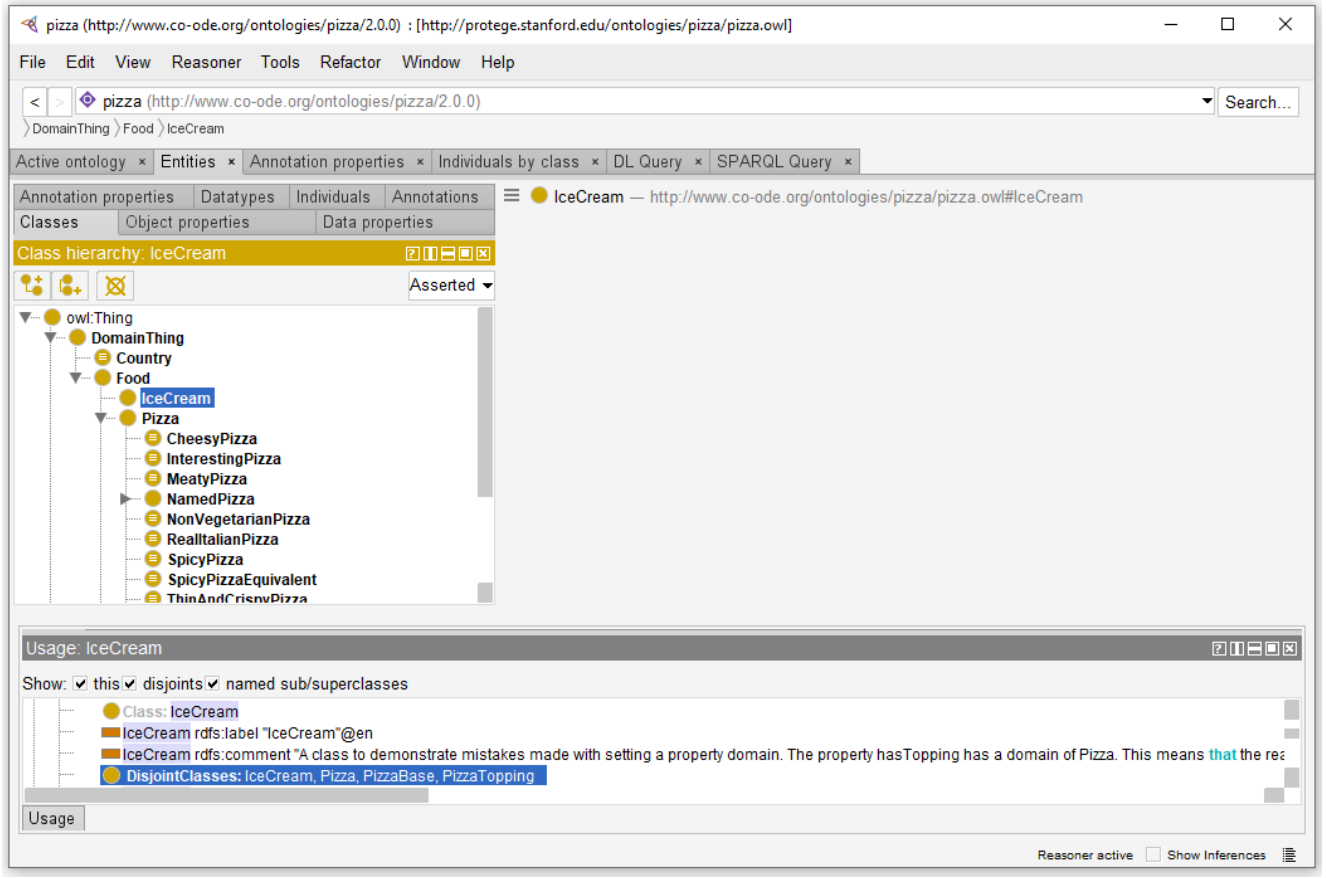
# Explanation

- IceCream rdfs:comment "A class to demonstrate mistakes made with setting a property domain. The property hasTopping has a domain of Pizza. This means that the reasoner can infer that all individuals using the hasTopping property must be of type Pizza. Because of the restriction on this class, all members of IceCream must use the hasTopping property, and therefore must also be members of Pizza. However, Pizza and IceCream are disjoint, so this causes an inconsistency. If they were not disjoint, IceCream would be inferred to be a subclass of Pizza."@en

# Explanation



# Explanation



# HermiT OWL Reasoner

- Protégé uses the HermiT OWL 2 reasoner as default:
  - **`http://hermit-reasoner.com/`**
- HermiT is available as a Java library.
- HermiT includes both a Java API and a command-line interface.
- HermiT reduces all reasoning tasks to ontology satisfiability testing.
- HermiT provides support for SPARQL 1.1.
- There exist other OWL reasoners: FaCT++, Pellet, ELK.

# HermiT OWL Reasoner

```
Actions:
  -l, --load                        parse and preprocess ontologies (default
                                    action)
  -c, --classify                    classify the classes of the ontology,
                                    optionally writing taxonomy to a file if -o
                                    (--output) is used
  -O, --classifyOPs                 classify the object properties of the
                                    ontology, optionally writing taxonomy to a
                                    file if -o (--output) is used
  -D, --classifyDPs                 classify the data properties of the
                                    ontology, optionally writing taxonomy to a
                                    file if -o (--output) is used
  -P, --prettyPrint                 when writing the classified hierarchy to a
                                    file, create a proper ontology and nicely
                                    indent the axioms according to their leven
                                    in the hierarchy
  -k, --consistency[=CLASS]         check satisfiability of CLASS (default owl:
                                    Thing)
  -d, --direct                      restrict next subs/supers call to only
                                    direct sub/superclasses
  -s, --subs=CLASS                  output classes subsumed by CLASS (or only
                                    direct subs if following --direct)
  -S, --supers=CLASS                output classes subsuming CLASS (or only
                                    direct supers if following --direct)
  -e, --equivalents=CLASS           output classes equivalent to CLASS
  -U, --unsatisfiable               output unsatisfiable classes (equivalent to
                                    --equivalents=owl:Nothing)
      --print-prefixes              output prefix names available for use in
                                    identifiers
  -E, --checkEntailment             check whether the premise (option premise)
                                    ontology entails the conclusion ontology
                                    (option conclusion)
```

# Direct Sub-classes

```
E:\HermiT>java -jar Hermit.jar -dsowl:Thing
   https://raw.githubusercontent.com/owlcs/pizza-
   ontology/master/pizza.owl

Direct sub-classes of 'owl:Thing':
:DomainConcept
:ValuePartition
```

# Classification

```
F:\HermiT>java -jar HermiT.jar --classify
   --prettyPrint --output=classification.txt
   https://raw.githubusercontent.com/owlcs/pizza-
   ontology/master/pizza.owl
```

# Classification

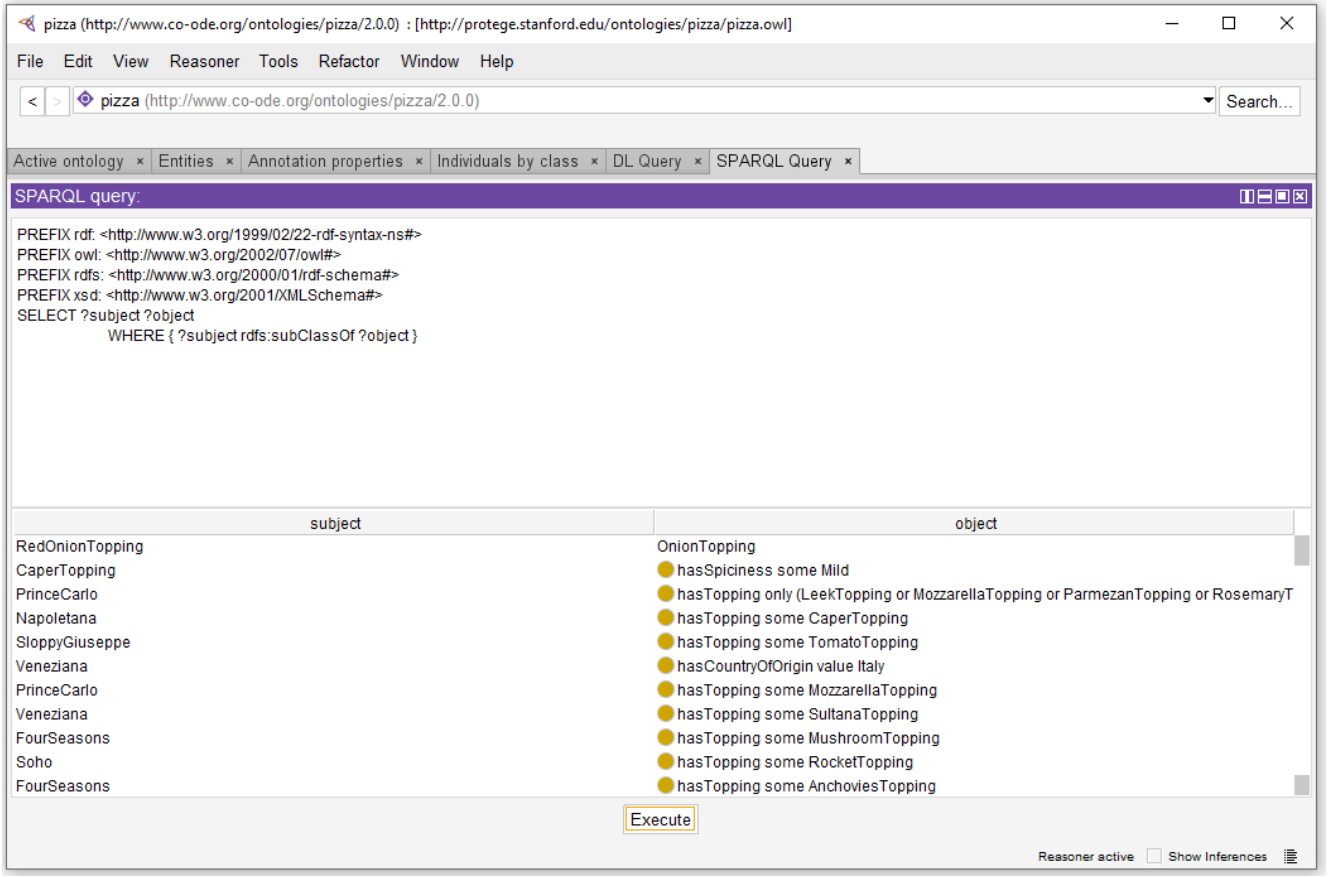
```
. . .
Declaration( Class( :MeatyPizza ) )
             SubClassOf( :American :MeatyPizza )
             SubClassOf( :AmericanHot :MeatyPizza )
             SubClassOf( :Capricciosa :MeatyPizza )
             SubClassOf( :FourSeasons :MeatyPizza )
             SubClassOf( :LaReine :MeatyPizza )
             SubClassOf( :Parmense :MeatyPizza )
             SubClassOf( :PolloAdAstra :MeatyPizza )
             SubClassOf( :Siciliana :MeatyPizza )
             SubClassOf( :SloppyGiuseppe :MeatyPizza )
. . .
```

# Checking for Consistency

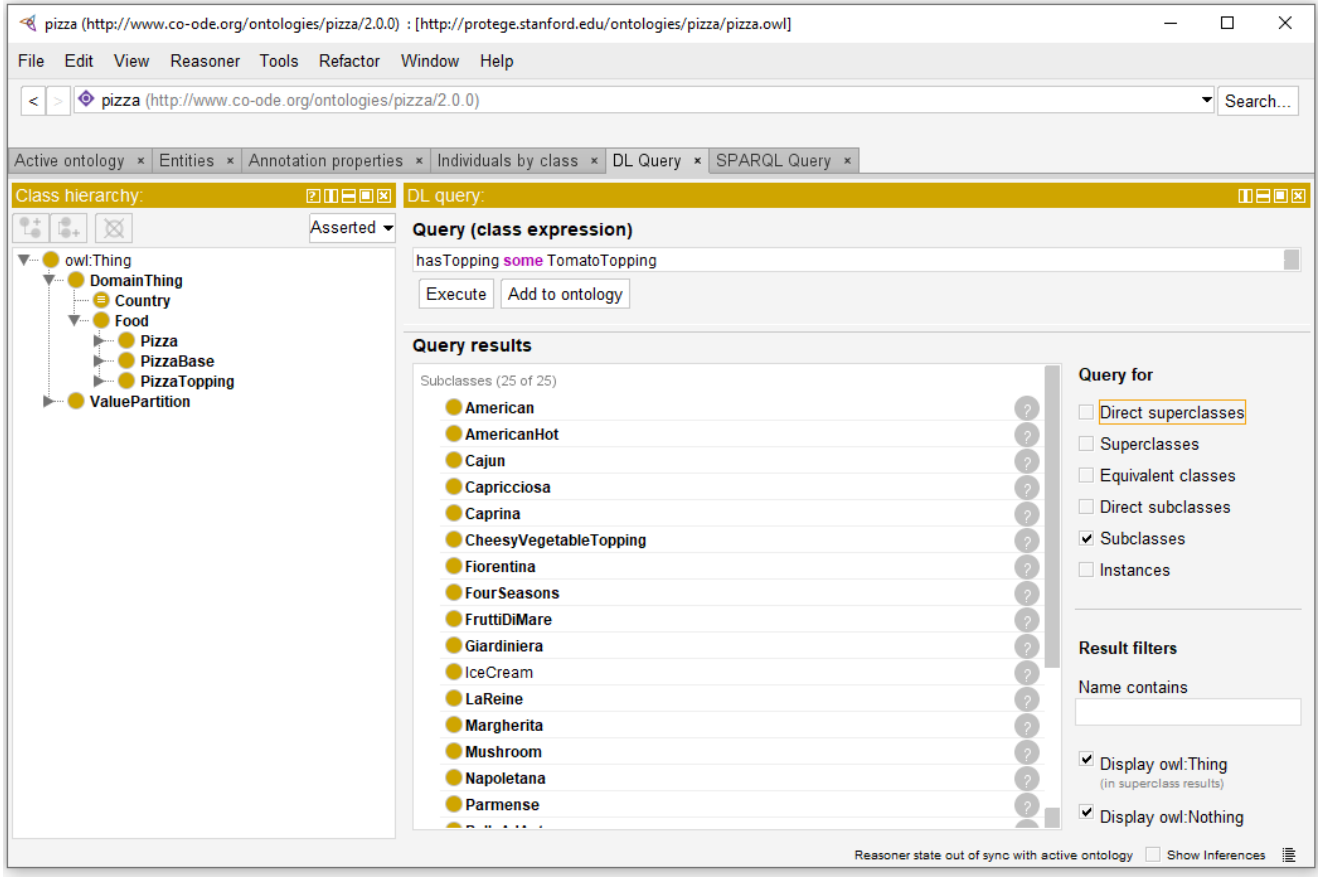
```
F:\HermiT>java -jar HermiT.jar --consistency
   https://raw.githubusercontent.com/owlcs/pizza-
   ontology/master/pizza.owl

http://www.w3.org/2002/07/owl#Thing is satisfiable.
```

# SPARQL Query



# DL Query (Manchester Syntax)



# Take-Home Messages

- There is no one correct way to model a domain.
- Ontology development is an iterative process.
- Choose a good naming convention.
- Protégé is a state-of-the-art ontology editor.
- Protégé uses HermiT as default reasoning service.
- HermiT supports OWL 2.
- HermiT reduces all reasoning tasks to satisfiability testing.