



Ontology Engineering

Knut Hinkelmann

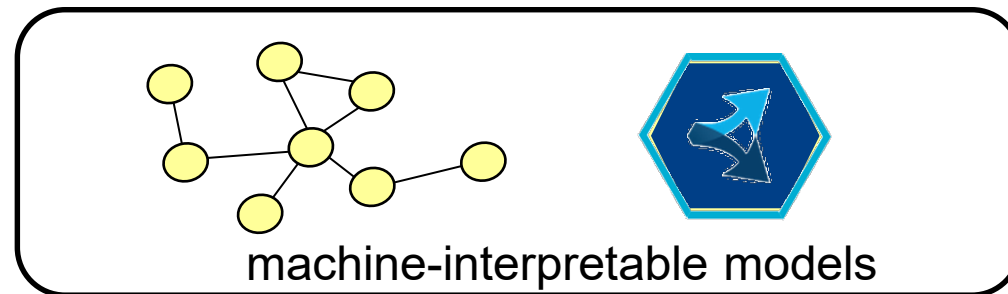


Knowledge-Representation and Reasoning

Reasoning/Inference



Knowledge Base



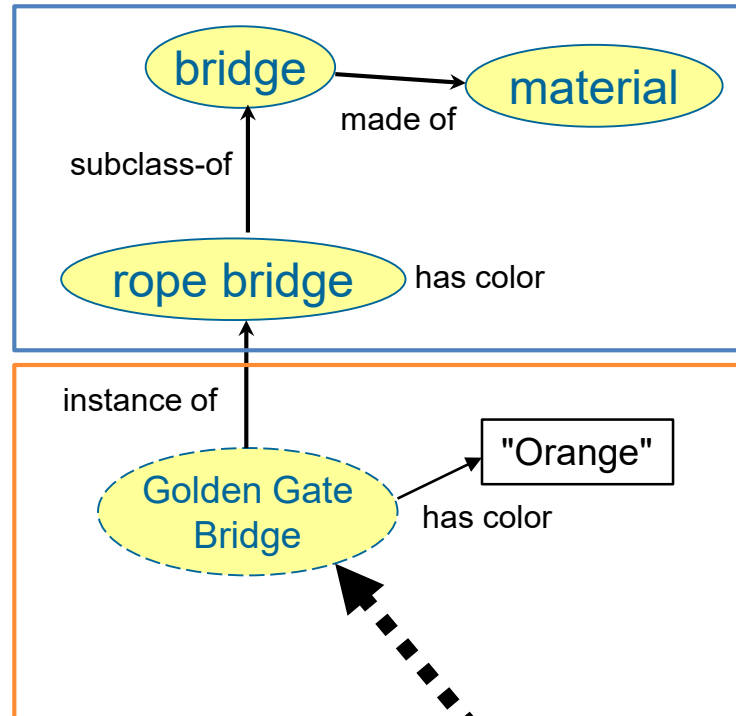
Reality



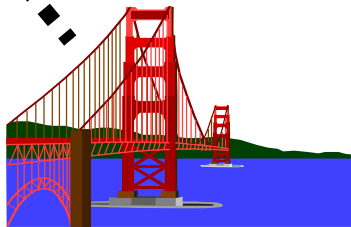
An Ontology – very informal

An ontology is a formal explicit description of concepts in a domain of discourse

- An **ontology** consists of
 - ◆ Concepts (Classes),
 - ◆ Relationships (Object Properties) between concepts
 - ◆ Attribute (Data Properties) of concepts
 - ◆ Constraints that hold between/for the concepts,
 as a representation of a particular domain
- An ontology together with a set of individual instances constitutes a **knowledge base**

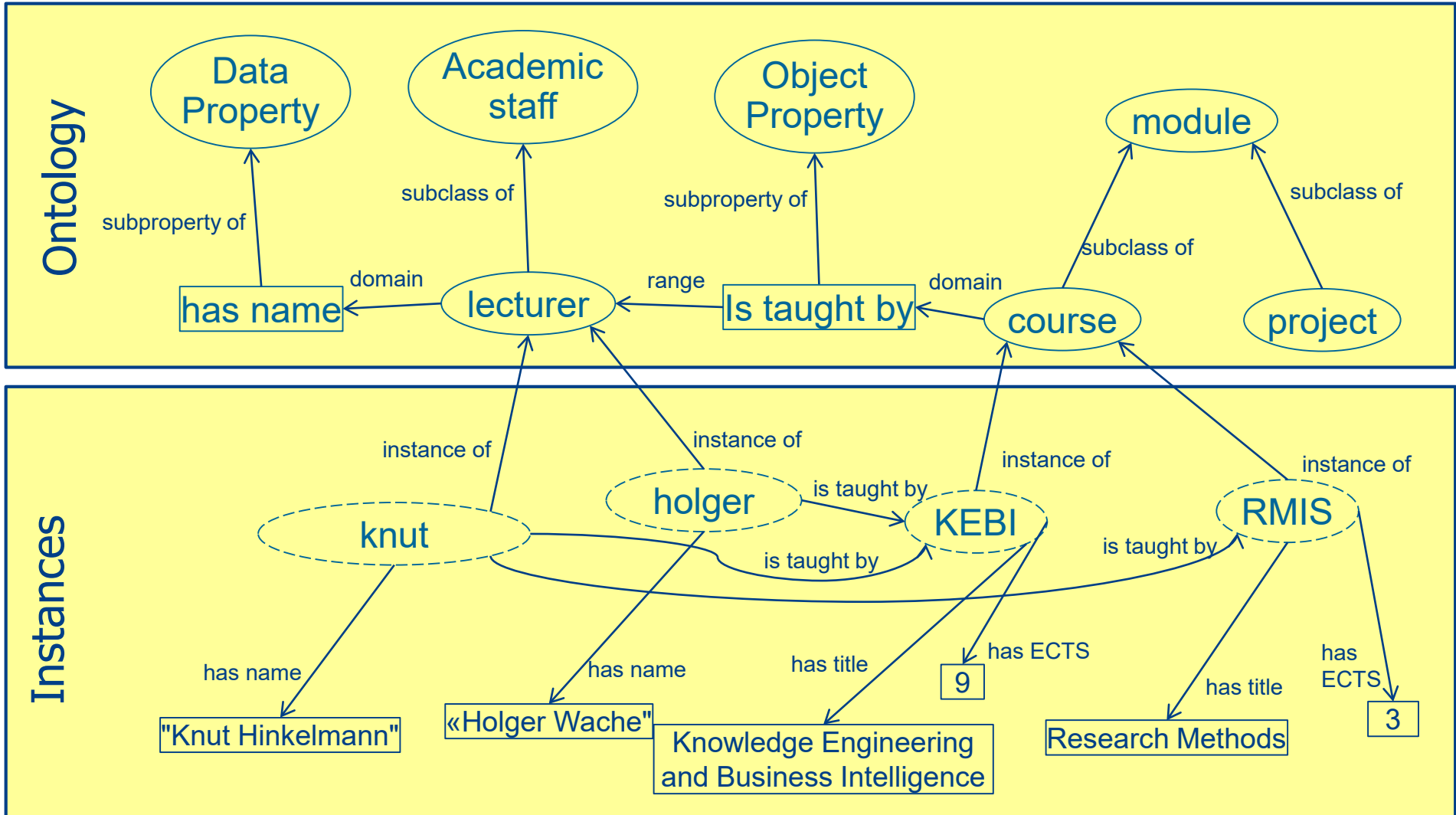


real object



ontology engineering
is
knowledge engineering

Example of an Ontology



Ontology Representation Formalisms

■ Representations of Ontologies

◆ ***RDF(S)***

Our focus

◆ OWL

◆ Neo4J

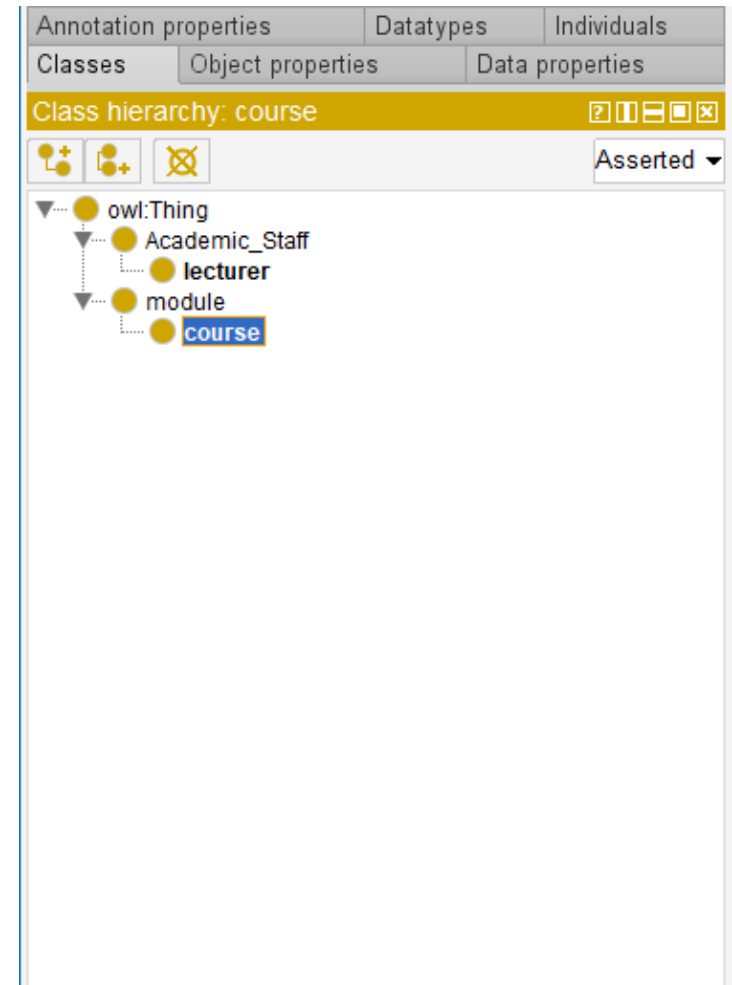
◆ ...

Strategy

- Defining classes in the ontology
- Arranging the classes in a taxonomic (subclass-superclass) hierarchy
- Defining properties and describing allowed values for the properties
- Creating instances and filling the values for properties

Define Classes and Class Hierarchy

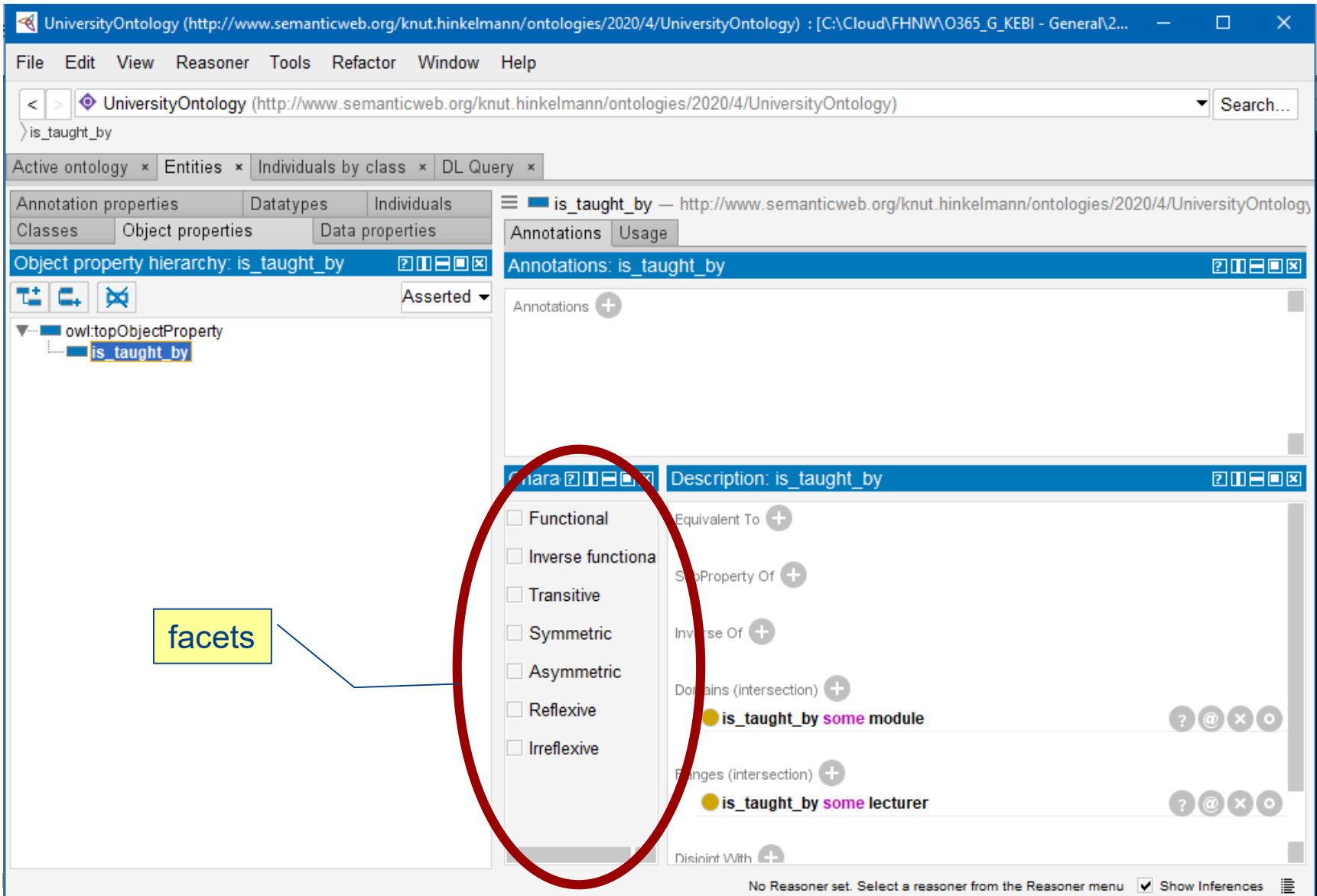
- There are several approaches
 - ◆ Top-down: Start with the most general concept, and work your way down
 - ◆ Bottom-up: Start with the most specific, and work your way up
 - ◆ Combination



Define Properties of Classes

- Describe the internal structure of concepts
 - ◆ Data Properties: Attributes
 - Range are data types like String, Integer, ...
 - ◆ Object Properties: Relations to other concepts
 - Range are Classes
- Describe facets: Characteristics of Properties
- Inheritance to Subclasses

Object Property



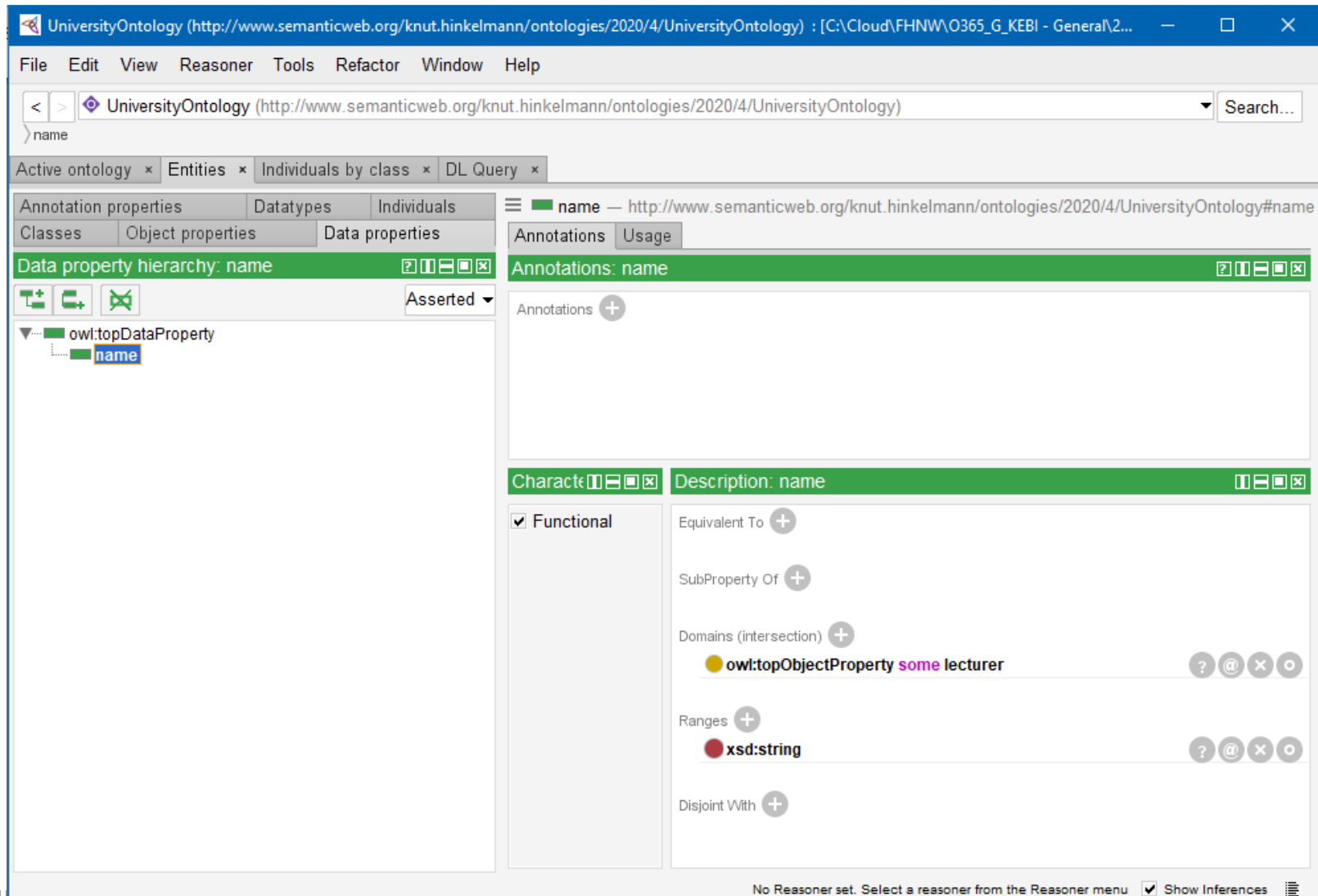
The screenshot shows the Protege interface for the 'UniversityOntology'. The 'is_taught_by' property is selected, and its configuration is displayed in the 'Description: is_taught_by' panel. The 'facets' menu is highlighted with a red circle, and a yellow box labeled 'facets' points to it. The facets listed are:

- Functional
- Inverse functiona
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

The 'Description: is_taught_by' panel also shows the following options:

- Equivalent To +
- SubProperty Of +
- Inverse Of +
- Domains (intersection) +
 - is_taught_by some module
- Ranges (intersection) +
 - is_taught_by some lecturer
- Disjoint With +

Data Property

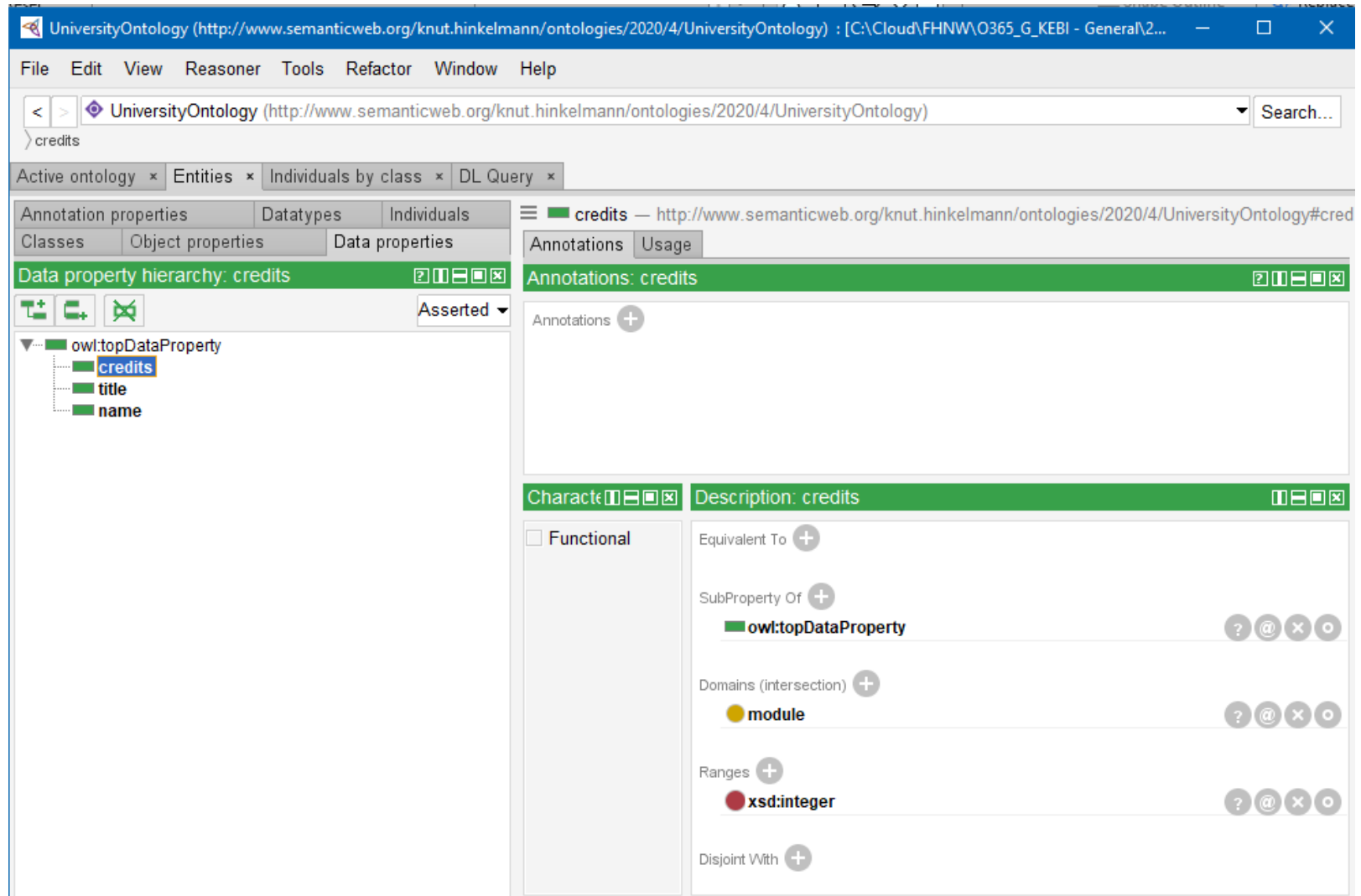


The screenshot shows the Protégé interface for editing the 'name' data property in the 'UniversityOntology'.

- Left Panel:** Shows the 'Data property hierarchy: name' tree with 'name' selected under 'owl:topDataProperty'.
- Top Panel:** Shows the 'name' property URI: `http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology#name`.
- Annotations:** An empty list for annotations.
- Description:** Shows the property is **Functional**. It lists:
 - Equivalent To: (empty)
 - SubProperty Of: (empty)
 - Domains (intersection): `owl:topObjectProperty some lecturer`
 - Ranges: `xsd:string`
 - Disjoint With: (empty)

At the bottom, the status bar indicates: "No Reasoner set. Select a reasoner from the Reasoner menu" and "Show Inferences" is checked.

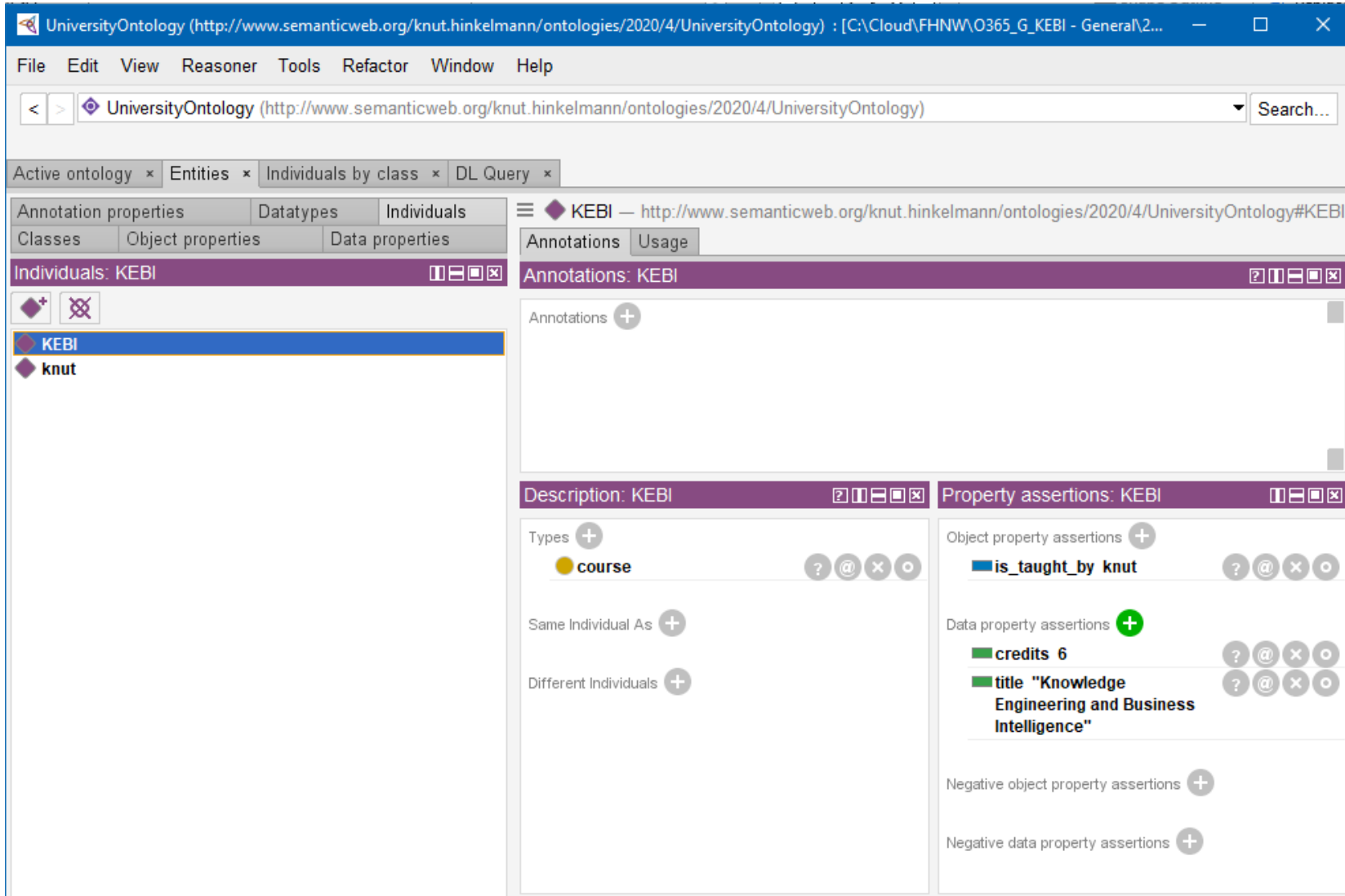
Data Property



The screenshot shows the UniversityOntology web interface. The main content area displays the configuration for the 'credits' data property. On the left, a tree view shows the hierarchy: owl:topDataProperty > credits > title > name. The 'credits' property is selected. The right pane shows the 'Description: credits' section, which includes the following configuration:

- Functional
- Equivalent To: +
- SubProperty Of: + owl:topDataProperty
- Domains (intersection): + module
- Ranges: + xsd:integer
- Disjoint With: +

Instances



The screenshot shows a web browser window displaying the 'UniversityOntology' (http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology). The interface includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help) and a search bar. The main content area is divided into several panels:

- Active ontology:** UniversityOntology
- Entities:** Individuals by class
- Annotation properties, Datatypes, Individuals, Classes, Object properties, Data properties:** Navigation tabs for the ontology view.
- Individuals: KEBI:** A list of instances for the 'KEBI' class, including 'KEBI' and 'knut'.
- Annotations: KEBI:** A panel for viewing annotations on the selected instance.
- Description: KEBI:** A panel showing the class description, including the type 'course' and various logical constraints (Same Individual As, Different Individuals).
- Property assertions: KEBI:** A panel showing property assertions for the instance 'knut', including 'is_taught_by knut', 'credits 6', and 'title "Knowledge Engineering and Business Intelligence"'. It also includes sections for negative object and data property assertions.

Querying

- Query Language: SPARQL

- ◆ Variables: ?x

- Elements are denoted as URI

- ◆ Prefixes for Abbreviations

- Example: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

- Sample query: Select all lecturers:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX uo: <http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology#>
SELECT ?subject
      WHERE { ?subject rdf:type uo:lecturer }
```

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?instance
      WHERE { ?instance rdf:type lecturer }
```

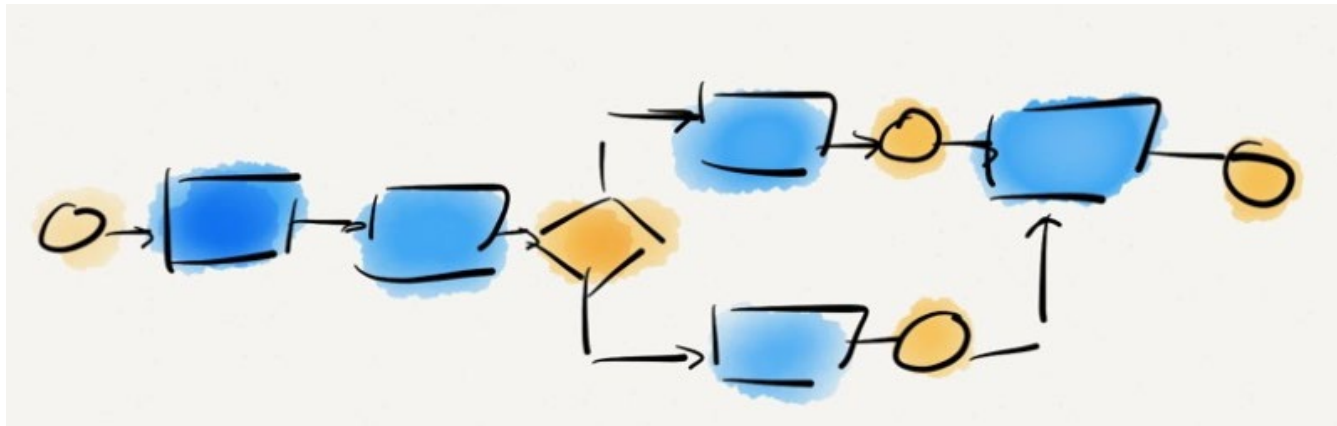
Exercise

- Add new classes: project
 - ◆ A project has a supervisor
 - ◆ Master Thesis is a project
 - ◆ Supervisor is a lecturer
 - ◆ A project is performed by a student
- Add new instances

Exercise: Modeling Process Knowledge in an Ontology

Exercise: Modeling Process Knowledge in an Ontology

- We create a knowledge base for process knowledge
 - ◆ Define the ontology
 - ◆ Represent knowledge of a process



Ontology Development 101

- 1 Determine the domain and scope of the ontology
- 2
 - Consider reusing existing ontologies
- 3
 - Enumerate important terms
- 4
 - Define classes and class hierarchy
- 5
 - Define the data and object properties of classes
- 6
 - Define the facets of properties
- 7
 - Create instances

Determine the domain and scope of the ontology

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers? → Competency questions
- Who will use and maintain the ontology?

Competency Questions

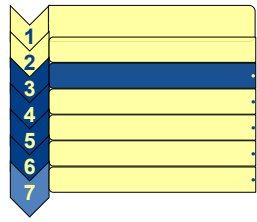
- One of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer (Gruninger and Fox 1995)
 - ◆ Does the ontology contain enough information to answer these types of questions?
 - ◆ Do the answers require a particular level of detail or representation of a particular area?

- Exercise: We want to represent knowledge about
 - ◆ the process flow
 - ◆ Responsibilities for tasks
- Competency Questions:
 - Who executes task X?
 - Which task is executed after task X?
 - When can task X start?
- Sample process:

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Consider reusing existing ontologies

- It is always worth considering what others have done, and check if their work can be refined and extended for our particular domain and task
- Mandatory if the system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies



Enumerate important terms in the ontology

- What are the terms we would like to talk about?
- What are their properties?
- What would we like to say about those terms?

Define Classes and Class Hierarchy

- Several possible approaches in developing a class hierarchy:
 - ◆ Top-down: General to specific concepts
 - ◆ Bottom-up: Specific to general concepts
 - ◆ Combination: Salient to general and specific concepts
- Classes for
 - ◆ Modeling Objects
 - ◆ Relations

Define the properties of classes

- Describe the internal structure of concepts
 - ◆ Data Properties: Attributes
 - Range are data types like String, Integer, ...
 - ◆ Object Properties: Relations to other concepts
 - Range are Classes
- Inheritance to Subclasses

Create Instances

- Model a business process in an ontology

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Conclusion

Modeling Business Processes as ontologies is not adequate for business people

→ Graphical Models

