Ontology 101

GETTING STARTED…

A guide and a process for creating OWL ontologies

POLITECNICO
DI TORINO

e-Lite

Recap: OWL2

- OWL 2 is a knowledge representation language, designed to formulate, exchange and reason with knowledge about a domain of interest
- Basic notions
  - Axioms: the basic statements that an OWL ontology expresses
  - Entities: elements used to refer to real-world objects
  - Expressions: combinations of entities to form complex descriptions from basic ones
- The results of the modeling processes are called *ontologies*
- Knowledge consists of elementary pieces that are often referred to as statements or propositions
- Statements that are made in an ontology are called axioms in OWL 2

Recap: OWL2

- When humans think, they draw consequences from their knowledge
- A statement is a consequence of other statements essentially means that this statement is true whenever the other statements are
- In OWL terms: “a set of statements A entails a statement a if in any state of affairs wherein all statements from A are true, also a is true”
- A set of statements may be
  - Consistent, if there is a possible state of affairs in which all the statements in the set are jointly true
  - Inconsistent, if there is no such state of affairs
- The formal semantics of OWL specifies, in essence, for which possible “states of affairs” a particular set of OWL statements is true

Recap: What is an ontology?

- An ontology is an explicit description of a domain
  - concepts
  - properties and attributes of concepts
  - constraints on properties and attributes
  - individuals (often, but not always)
- An ontology defines
  - a common vocabulary
  - a shared understanding

Recap: What is in a OWL2 ontology?

- Classes
- Instances
- Properties
  - Object Properties
  - DataType Properties
- Restrictions
- Annotations

Recap: Tools

- Editors (http://semanticweb.org/wiki/Editors)
  - Most common editor: **Protégé 5.x**
  - Other tools: TopBraid Composer ($), NeOn toolkit
  - Special purpose apps, esp. for light-weight ontologies (e.g., FOAF editors)
- Reasoners (http://semanticweb.org/wiki/Reasoners)
  - OWL DL: Pellet 2.0*, **HermiT**, FaCT++, RacerPro ($)
  - OWL EL: CEL, SHER, snorocket ($), ELLY
  - OWL RL: OWLIM, Jena, Oracle OWL Reasoner ($)
  - OWL QL: Owlgres, QuOnto, Quill

* The next-gen reasoner (version 3) is part of Stardog, a closed source RDF database

Ontology Engineering

- The process of building and maintaining an ontology
- To define terms in a domain and relations among such terms
  - defining concepts (*classes*) in the domain
  - arranging the concepts in a hierarchy (subclass-superclass hierarchy)
  - defining which *properties* classes can have and constraints on their values (*restrictions*)
  - defining individuals (*instances*) and filling in properties values

Ontology Engineering vs. OOP

Ontology Engineering

An ontology

- reflects the structure of the world
- is often about structure of concepts
- actual physical representation is not an issue

Object-Oriented Modeling

A class

- reflects the structure of the data and code
- is usually about behavior (methods)
- describes the physical representation of data (int, string, etc.)

Why develop an ontology?

A. You hate your life!

B. You need to fill several days and weeks with "something"

C. Other

Why develop an ontology?

- To share **common understanding** of the structure of information
  - among people
  - among software artifacts
- To enable **reuse** of domain knowledge
  - to avoid “re-inventing the wheel”
  - to introduce standards to allow interoperability

Why develop an ontology?

- To make domain assumptions **explicit**
  - easier to change domain assumptions (consider a genetics knowledge base)
  - easier to understand and update legacy data
- To **separate** domain knowledge from the operational knowledge
  - re-use domain and operational knowledge separately (e.g., configuration based on constraints)
- To **analyze** domain knowledge

Create an ontology: the process

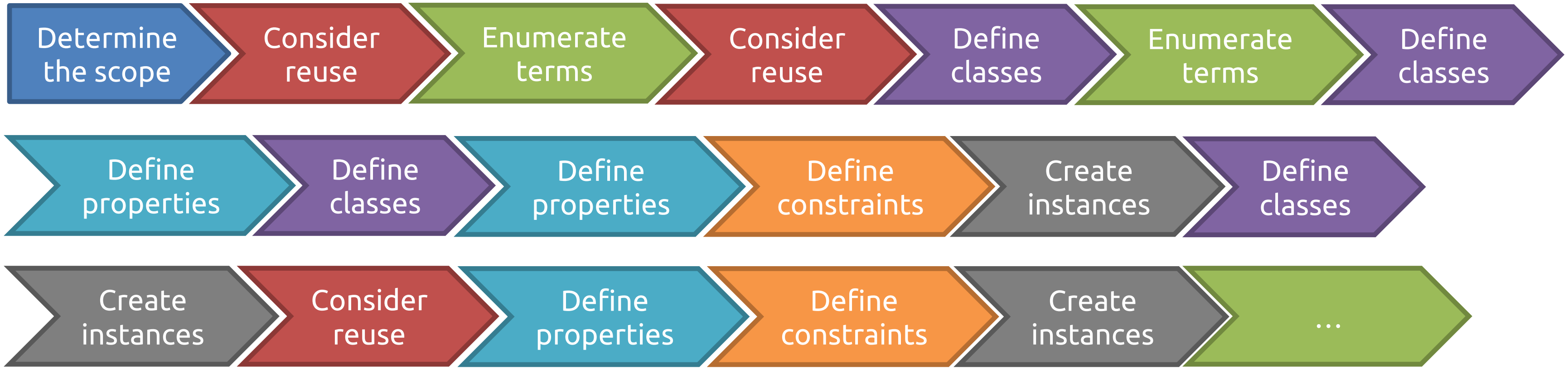
1. Determine the scope
2. Consider reuse
3. Enumerate terms
4. Define classes
5. Define properties
6. Define constraints
7. Create instances

(4–5: closely intertwined)

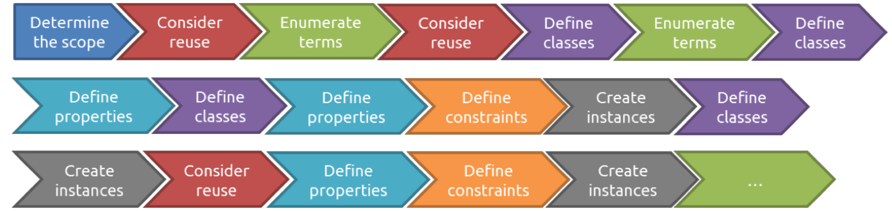
TODAY!

Create an ontology: the process

However, in the real world…



Disclaimer



- There is no one “correct” way or methodology for developing ontologies
- It is a **long**, **hard**, and **precise** activity
- Fundamental rules:
  1. There is **no one correct way** to model a domain— there are always viable alternatives. The best solution depends on the application that you have in mind and the extensions that you anticipate.
  2. Ontology development is necessarily an **iterative** process.
  3. Concepts in the ontology should be **close** to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

1. Determine domain and scope

- 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?

Answers to these questions may change during the lifecycle of an ontology

Competency Questions

- One way to determine the scope of the ontology
- Write down a list of questions that a knowledge base built upon the ontology should be able to answer
- These questions may serve also later, for a preliminary evaluation and for end the process
- Questions **do not** need to be exhaustive, just a "sketch"

Learning by Example

- We will apply the process presented here up to the first iteration

- The chosen domain is: **University**
  - From which perspective?
  - What are our competency questions?

2. Consider reuse

- In some cases, it is a good idea to reuse existing ontologies
- Why?
  - to save the effort
  - to interact with the tools used by other ontologies
  - to employ ontologies that have been already validated through use in applications
- What to reuse?

What to reuse?

- Upper/general ontologies
  - Cyc, http://www.cyc.com
  - DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), http://www.loa.istc.cnr.it/old/DOLCE.html
  - WordNet, http://www.cogsci.princeton.edu/~wn/
- Domain-specific ontologies
  - GO (Gene Ontology), http://ww.geneontology.org
  - DogOnt, http://elite.polito.it/ontologies/dogont
  - MUO, http://idi.fundacionctic.org/muo/

3. Enumerate important terms

- Write down a list of all terms we would like to
  - make a statement about
  - explain to a user
- Questions
  - What are the terms we need to talk about?
  - What are the properties of these terms?
  - What would we like to say about the terms?
- Initially, it is important to get a **comprehensive list** of terms
  - without worrying about overlap between concepts, relations, properties, hierarchy, etc.

4. Define classes and their hierarchy

- A class is
  - a concept in the domain, not the word that denote the concept
  - a collection of elements with similar properties
- Instances of classes
  - specific (named) elements that pertain to that collection
- Classes usually constitute a taxonomic hierarchy
  - a subclass-superclass hierarchy
- A class hierarchy is usually an IS-A hierarchy
  - an instance of a subclass **is an** instance of a superclass

Modes of development

1. top-down
   – define the most general concepts first and then specialize them
2. bottom-up
   – define the most specific concepts and then organize them in more general classes
3. combination
   – define the more salient concepts first and then generalize and specialize them

None of them is inherently better than the others

5. Define properties

- Attributes of instances of the class and relations to other instances
- Types of properties
  - “intrinsic”, e.g., color of an object
  - “extrinsic”, e.g., price of an object
  - relations to other instances, e.g., producer of an object
- Simple and complex properties
  - simple properties (data properties): contain primitive values like strings and numbers
  - complex properties (object properties): contain or point to other objects, e.g., a manufacturer instance

6. Define constraints

- Property describe or limit the set of possible values for a slot
  - The name of an object is a string
  - Politecnico di Torino is an instance of University
  - A university has exactly one location (main campus)
- We mainly refer to properties restrictions
  - cardinality, domain, range, etc.
  - see the previous set of slides for further details

Properties and Classes Inheritance

- A subclass inherits all the properties from the superclass
- If a class has multiple superclasses, it inherits properties from all of them
- A subclass can **override** the restrictions to "narrow" the list of allowed values
  - make the cardinality range smaller
  - replace a class in the range with a subclass

7. Create instances

- The last step (hopefully!)
- Defining an individual require to
  - choose the desired class
  - create the desired instance of the class
  - fill in the properties values
- Some ontologies may not have instances
- In most cases, classes and instances are in two separate OWL files

Create an ontology: the process

1. Determine the scope
2. Consider reuse
3. Enumerate terms
4. Define classes
5. Define properties
6. Define constraints
7. Create instances

COMMON PROBLEMS…

… and their solutions

Multiple inheritance

- A class can have more than one superclass
- A subclass inherits properties and restrictions from all the parents

Disjoint classes

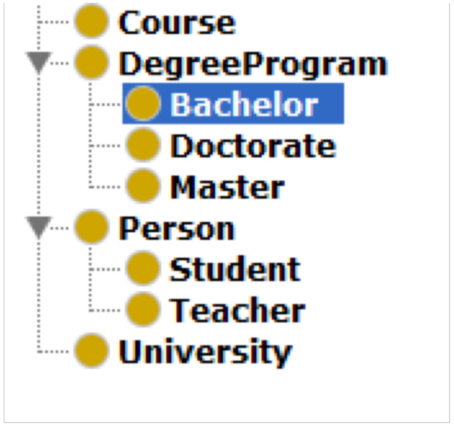
- Classes are disjoint if they **cannot** have common instances
- Disjoint classes **cannot** have any common subclasses either

Avoid class cycles

- Danger of multiple inheritance: cycles in the class hierarchy
- If class A has a subclass B and, at the same time, B is a superclass of A, the classes are **equivalent!**

Siblings in the hierarchy

- All the siblings in the class hierarchy must be at the same level of generality
  - Think about section and subsections in a book



How many class is too many? How few classes are too few?

- Rule of thumb
- If a class has only one direct subclass, there may be a modeling problem
  - or the ontology is not complete
- If there are more than a dozen subclasses for a given class, then additional intermediate may be necessary

Naming, domain and range

- Single and plural class names
  - Singular <u>or</u> plural
  - It is up to you but you need to choose
- Domain and range
  - When defining a domain or range, find the most general class or classes

Limiting the scope

- An ontology should not contain **all** the possible information about the domain
- No need to specialize or generalize more than the application requires
- No need to include all possible properties of a class
  - Only the most salient properties
  - Only the properties that the applications require

References

- Natalya F. Noy and Deborah L. McGuinness, "*Ontology Development 101: A Guide to Creating Your First Ontology*", Knowledge Systems Laboratory, Stanford University, March 2001
- Asunción Gómez-Pérez, Mariano Fernandez-Lopez and Oscar Corcho, "Ontological Engineering", Springer-Verlag London, 2004

Questions?

01RRDIU Semantic Web

Luigi De Russis
luigi.derussis@polito.it

License