Ontology 101

Design principles

Sergej Sizov

Ontology design principles

- Based on paper by

- Natasha Noy & Deborah McGuinness

  "Ontology Development 101: A Guide to Creating Your First Ontology"

  Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

Why use ontologies?

- To share common understanding of the structure of information among people or software agents
  - One common knowledge and data model (computer readable and understandable!)
- To enable reuse of domain knowledge
  - Define once, use in many applications in the same field
- To make domain assumptions explicit
  - Define classes, relationships and instances
- To separate domain knowledge from the operational knowledge
  - What is schema (meta information) and instances (real world)
- To analyze domain knowledge
  - What is the meaning of associations between objects

Building ontology – the basics

- In practical terms, developing an ontology includes:
  - defining **classes** in the ontology,
  - arranging the classes in a **taxonomic** (subclass–superclass) hierarchy,
  - defining **slots** (*relationships*) and describing allowed values for these slots,
  - filling in the values for slots for **instances**.

Fundamental thumb rules

There is no one correct way to model a domain

there are always viable alternatives.

The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.

Complexity of your ontology should reflect your particular interest in specific area – model what you need, not all that you can.

Fundamental thumb rules

Ontology development is **necessarily an iterative process**.

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.

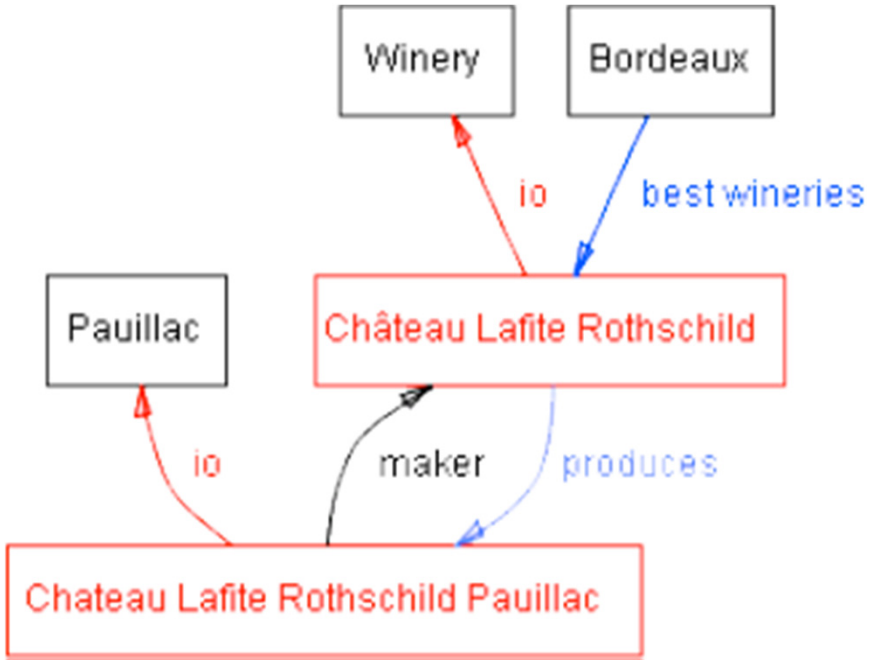
Step 1. Domain and Scope

- 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?
- Who will **use** and **maintain** the ontology?

Step 1 Example

The Wine Ontology

(very, very simplified …)



- Competency question
  - Which wine characteristics should I consider when choosing a wine?
  - Is Bordeaux a red or white wine?
  - Does Cabernet Sauvignon go well with seafood?
  - What is the best choice of wine for grilled meat?
  - Which characteristics of a wine affect its appropriateness for a dish?
  - Does a bouquet or body of a specific wine change with vintage year?
  - What were good vintages for Napa Zinfandel?

Step 2. Consider reusing existing ontologies

WeST

Reuse

Sure, if they exist!!!

Profit from importing existing ontologies – not only cover the scope you need (to some extent), but they also include additional information, classification, etc.

Check if existing ontologies fit your needs

- Can you use them directly?
- Can you use part of them?
- Maybe only small extension will do?

If it doesn't work or fit you – design your own

Step 3. Enumerate important terms

What are the terms we would like to talk about?

What properties do those terms have?

What would we like to say about those terms?

Example -

- wine, grape, winery, location,
- a wine's color, body, flavor and sugar content;
- different types of food, such as fish and red meat;
- subtypes of wine such as white wine, and so

Step 4. Define the classes and hierarchy

- Methods
  - Top-down
  - Bottom-up
  - Combination

- Start from defining classes
- Creating hierarchy will then be easier …

Step 4. Define the classes and hierarchy

Result

- hierarchical arrangements of concepts
- If a class A is a superclass of class B, then every instance of B is also an instance of A
- This implies “In other words, the class B represents a concept that is a “kind of” A.”

Step 4. Define the classes and hierarchy

- Top-down
  - Food and Wine -- followed by White, Blush and Red
- Bottom-up
  - define specific wine class first and the work your way up
- Combination
  - Start from known classes and fill the gaps

Step 5. Define properties of classes—slots

Types of properties

- "**intrinsic**" properties such as the flavor of a wine;
- "**extrinsic**" properties such as a wine's name, and area it comes from;
- parts, if the object is structured; these can be both physical and abstract "**parts**" (e.g., the courses of a meal)
- **relationships to other individuals** – these are the relationships between individual members of the class and other items (e.g., the maker of a wine, representing a relationship between a wine and a winery, and the grape the wine is made from.)

Step 5. Define properties of classes—slots

- Different naming for the same thing:
  - Relationship
  - Property
  - Slot

- Examples
  - a wine's
    - color,
    - body,
    - flavor
    - sugar content
  - location of a winery.

Step 6. Define the facets of the slots

- **Slot-value type**
  - what types of values can fill in the slot.
  - common value types:
    - **String**
    - **Number**
    - **Boolean**
    - **Enumerated**
    - **Date**

- **Slot cardinality**
  - defines how many values a slot *can* have or you *must* have.

Step 6. Define the facets of the slots

Facets of relationships

means: Role restrictions

Sample facets

- value type
- allowed values
- the number of the values (cardinality – single, multiple …)

- … other features of the values the slot can take.

Step 6 - rules slot Domain and Ranges

When defining a domain or a range for a slot, find the most general classes or class that can be respectively the domain or the range for the slots . On the other hand, do not define a domain and range that is overly general: all the classes in the domain of a slot should be described by the slot and instances of all the classes in the range of a slot should be potential fillers for the slot. Do not choose an overly general class for range (i.e., one would not want to make the range THING) but one would want to choose a class that will cover all fillers

Shortly:

aim well – not too general, not too specific

Step 6 - rules slot Domain and Ranges

If a list of classes defining a range or a domain of a slot includes a class and its subclass, remove the subclass.

→ keep only the most general classes

If a list of classes defining a range or a domain of a slot contains all subclasses of a class A, but not the class A itself, the range should contain only the class A and not the subclasses.

→ if you have all subclasses, use only superclass

If a list of classes defining a range or a domain of a slot contains all but a few subclasses of a class A, consider if the class A would make a more appropriate range definition.

→ if you have all but few subclasses, use superclass

→ or give a second thought to created hierarchy

Step 7 - Creating Instances

- Body: Light
- Color: Red
- Flavor: Delicate
- Tannin level: Low
- Grape: Gamay (instance of the Wine grape class)
- Maker: Chateau-Morgon (instance of the Winery class)
- Region: Beaujolais (instance of the Wine-Region class)
- Sugar: Dry

Are we done?

What about
Consistency
Validity
Sanity
checks???

Step 8 – Def. classes and a class hierarchy

Ensuring that the class hierarchy is correct

- An "**is-a**" relation
  - *A subclass of a class represents a concept that is a "kind of" the concept that the superclass represents.*
- A single wine is *not a subclass* of all wines
- Remember about **transitivity** of the hierarchical relations

Evolution of a class hierarchy

- distinction between **classes** and their **names**
  - hence synonyms of concept name do not represent different classes
- avoid class hierarchy ***cycles***
- *All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality.*

Step 8 – Def. classes and a class hierarchy

How many is too many and how few are too few?

- 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 categories may be necessary. (may not always be possible)

Multiple Inheritance

- Use it to combine properties of both (or many) classes within one

When do you introduce a new class?

- Subclasses of a class usually
  - (1) have **additional properties** that the superclass does not have, or
  - (2) **different restrictions** from those of the superclass, or
  - (3) participate in **different relationships** than the superclasses

Step 8 – Def. classes and a class hierarchy

Counter example to thumb rule in previous slide

- classes in terminological hierarchies do not have to introduce new properties
- *MeSH* at NLM National Institute of Health
- purpose is just to organize for ease of searching/indexing

Step 8 – Def. classes and a class hierarchy

- **A new class or a property value?**
  - Class White Wine or simply property of class Wine that takes value White
  - Depends on how *important* a concept White Wine is in the domain

- **An instance or a class?**
  - starts with deciding what is the lowest level of granularity in the representation,
  - individual instances are the most specific concepts represented in a knowledge base.

Step 8 – Limiting the scope

- The ontology should not contain all the possible information about the domain:
  - you do not need to specialize (or generalize) more than you need for your application (at most one extra level each way).
  - tailor ontology for your needs and applications, but …
  - … think about possible extensibility (how easy, in which direction)

- The ontology should **not contain** all the possible properties of and distinctions among classes in the hierarchy.

Other details of design

- Inverse slots
  - Functional or non-functional properties
  - What do you express with inverse relation?
- Naming conventions
  - Are there available/well-established in general or in your field/area?
  - Stick to one naming convention – be consistent
- Synonyms
  - Just different name or really different objects?
  - Maybe multiple labels for the same object?
- Defaults
  - What values are there in instance if user do not give any?
- Disjoint subclasses
  - What is the reason for introducing additional restrictions?

Conclusion

- There are steps worth following in designing ontology
  - Have rationale for each of your design choices
- Remember:

  there is no single correct ontology for any domain

  "Ontology design is a creative process and no two ontologies designed by different people would be the same."
- There are helpful methodologies and patterns
- Designing ontology is an iterative process
  - ontology can evolve and change while you design it