Ontology Development 101

Natasha Noy
Stanford University

A large part of this tutorial is based on
by Natalya F. Noy and Deborah L. McGuinness
Outline

- What is an ontology?
  - definition
  - terminology
- Why develop an ontology?
- Step-By-Step: Developing an ontology
- Underwater ??
- What to look out for
Which wine should I serve with seafood today?

A shared ontology of wine and food

French wines and wine regions

California wines and wine regions
An ontology is an explicit description of a domain:
- concepts
- properties and attributes of concepts
- constraints on properties and attributes
- individuals

An ontology defines
- a common vocabulary
- a shared understanding
Ontology examples

- **Taxonomies on the Web**
  - Yahoo! categories

- **Catalogs for on-line shopping**
  - Amazon product catalog

- **Domain-specific standard terminology**
  - Unified Medical Language System (UMLS)
  - UNSPSC - terminology for products and services
Why develop an ontology?

- To share common understanding of the structure of information
  - among people
  - among software agents
- To enable reuse of domain knowledge
  - to avoid “re-inventing the wheel”
  - to introduce standards
More reasons

- 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)
An ontology is often just the beginning

Ontologies

Declare structure

Databases

Knowledge bases

Provide domain description

Software agents

Problem-solving methods

Domain-independent applications
Outline

- What is an ontology?
- Why develop an ontology?
- **Step-By-Step: Developing an ontology**
- Underwater ??
- What to look out for
What Is “Ontology Development”? 

- Defining terms in the domain and relations among them
  - Defining concepts in the domain (classes)
  - Arranging the concepts in a hierarchy (subclass-superclass hierarchy)
  - Defining which attributes and properties (slots) classes can have and constraints on their values
  - Defining individuals and filling in slot values (instances)
Wines and wineries

Diagram:
- Pauillac
- Château Lafite Rothschild
- Winery
- Bordeaux
- Château Lafite Rothschild Pauillac

Relationships:
- Château Lafite Rothschild produces Château Lafite Rothschild Pauillac
- Pauillac is associated with Château Lafite Rothschild
- Bordeaux is associated with Winery

Note: The diagram illustrates the production and origin of Château Lafite Rothschild Pauillac in relation to Pauillac and Bordeaux.
Ontology-development process

In reality - an iterative process:

1. Determine scope
2. Consider reuse
3. Enumerate terms
4. Define classes
5. Define properties
6. Define constraints
7. Create instances
8. Consider reuse
9. Define properties
10. Define classes
11. Define properties
12. Define constraints
13. Create instances
Ontology development versus Object-oriented modeling

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

**An OO Structure**
- reflects the structure of the data and code
- is usually about behavior (methods)
- describes the physical representation of data (long int, char, etc.)
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?
- Who will use and maintain the ontology?

Answers to these questions may change during the ontology lifecycle
Competency question for the Wine ontology

- 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?
Consider reuse

Why reuse other ontologies?

- to save the effort
- to interact with the tools that use other ontologies
- to use ontologies that have been validated through use in applications
What to reuse?

- **Ontology libraries**
  - Protégé ontology library (protege.stanford.edu)
  - Ontolingua ontology library (www.ksl.stanford.edu/software/ontolingua/)

- **Upper ontologies**
  - IEEE Standard Upper Ontology (suo.ieee.org)
  - Cyc (www.cyc.com)

- **Domain-specific ontologies**
  - UMLS Semantic Net
  - GO (Gene Ontology) (www.geneontology.org)
  - OBO (Open Biological Ontologies) (obo.sourceforge.net)
Enumerate important terms

- What are the terms we need to talk about?
- What are the properties of these terms?
- What do we want to say about the terms?
Enumerating terms: The Wine ontology

- wine, grape, winery, location,
- wine color, wine body, wine flavor, sugar content
- white wine, red wine, Bordeaux wine
- food, seafood, fish, meat, vegetables, cheese
Define classes and the class hierarchy

- A class is a concept in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you’ll have for lunch
Class inheritance

- **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
- **If you think of a class as a set of elements, a subclass is a subset**
Class inheritance: Examples

- **Apple is a subclass of Fruit**
  - Every apple is a fruit

- **Red wines is a subclass of Wine**
  - Every red wine is a wine

- **Chianti wine is a subclass of red wine**
  - Every Chianti wine is a red wine
Define properties of classes: Slots

- Slots in a class definition describe attributes of instances of the class
  - each wine will have color, sugar content, producer, etc.
Types of properties

- "intrinsic" properties: flavor and color of wine
- "extrinsic" properties: name and price of wine
- parts: ingredients in a dish
- relations to other objects: producer of wine (winery)

Simple and complex properties

- simple properties (attributes): contain primitive values (strings, numbers)
- complex properties: contain other objects (e.g., a winery instance)
Slots for the class Wine

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL,MEDIUM,LIGHT}</td>
</tr>
<tr>
<td>color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED,ROŠÉ,WHITE}</td>
</tr>
<tr>
<td>flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE,MODERATE,STRONG}</td>
</tr>
<tr>
<td>grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td>maker</td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY,SWEET,OFF-DRY}</td>
</tr>
</tbody>
</table>
Slot and class inheritance

- A subclass **inherits** all the slots from the superclass
  - If a wine has a name and flavor, a red wine also has a name and flavor
- If a class has **multiple superclasses**, it inherits slots from all of them
  - Port is both a dessert wine and a red wine. It inherits “sugar content: high” from the former and “color:red” from the latter
Property constraints (facets) describe or limit the set of possible values for a slot

- the name of a wine is a string
- the wine producer is an instance of Winery
- a winery has exactly one location
## Facets for slots at the Wine class

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL, MEDIUM, LIGHT}</td>
</tr>
<tr>
<td>color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED, ROSÉ, WHITE}</td>
</tr>
<tr>
<td>flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE, MODERATE, STRONG}</td>
</tr>
<tr>
<td>grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td><strong>maker</strong></td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY, SWEET, OFF-DRY}</td>
</tr>
</tbody>
</table>
Common facets: Cardinality

- **Slot cardinality** – the number of values a slot can or must have
  - **Minimum cardinality**
    - Minimum cardinality 1 means that the slot must have a value (required)
    - Minimum cardinality 0 means that the slot value is optional
  - **Maximum cardinality**
    - Maximum cardinality 1 means that the slot can have at most one value *(single-valued slot)*
    - Maximum cardinality greater than 1 means that the slot can have only one value *(multiple-valued slot)*
Common facets: Value Type

Slot value type – what values can the slot have

- **String**: a string of characters ("Château Lafite")
- **Number**: an integer or a float (15, 4.5)
- **Boolean**: a true/false flag
- **Enumerated type**: a list of allowed values (red, white, rosé)
- **Complex type**: an instance of another class or a class itself

Specify the class to which the instances belong

For example, the **Wine** class is the value type for the **produces** slot at the **Winery** class
Defining facets: Example

- **Name**: maker
- **Value Type**: Instance
- **Documentation**: The maker of a wine (a Winery). This slot has an inverse - the slot produces at the Winery class.
- **Allowed Classes**: Winery
- **Cardinality**: required at least 1, multiple at most 1
- **Minimum** and **Maximum**: Empty
- **Inverse Slot**: produces
Facets and class inheritance

- A subclass **inherits** all the slots from the superclass
- A subclass can **override** the facets to “narrow” the list of allowed values
  - Make the cardinality range smaller
  - Replace a class in the range with a subclass
Create instances

- Create an instance of a class
  - The class becomes a *direct type* of the instance
  - Any superclass of the direct type is a *type* of the instance
- Assign slot values for the instance frame
  - Slot values should conform to the facet constraints
  - Knowledge-acquisition tools often check that
Creating an instance: Example
Outline

- What is an ontology?
- Why develop an ontology?
- Step-By-Step: Developing an ontology
- Underwater ??
- What to look out for
Going deeper

determine scope
consider reuse
enumerate terms
define classes
define properties
define constraints
create instances

determine scope
consider reuse
enumerate terms
define classes
define properties
define constraints
create instances
Defining classes and a class hierarchy

- **The question to ask:**
  - “Is each instance of the subclass an instance of its superclass?”

- **The things to remember:**
  - There is no single correct class hierarchy
  - But there are some guidelines
Multiple inheritance

- A class can have more than one superclass
- The subclass inherits slots and facet restrictions from all the parents
- Different systems resolve conflicts differently
Avoiding class cycles

Danger of multiple inheritance: cycles in the class hierarchy

*Classes A, B, and C have equivalent sets of instances*

*By many definitions, A, B, and C are thus equivalent*
Disjoint classes

- Classes are disjoint if they cannot have common instances.
- Disjoint classes cannot have any common subclasses either.
  - Red wine, White wine, Rosé wine are disjoint.
  - Dessert wine and Red wine are not disjoint.
Levels in the class hierarchy

Different modes of the development

- **top-down** - define the most general concepts first and then specialize them
- **bottom-up** - define the most specific concepts and then organize them in more general classes
- **combination**
Levels in the class hierarchy

Bottom level

Middle level

Top level
Siblings in the class hierarchy

- All the siblings in the class hierarchy must be at the same level of generality
- Compare to section and subsections in a book
The perfect family size

- If a class has only one child, there may be a modeling problem
- If the only Red Burgundy we have is Côtes d’Or, why introduce the subhierarchy?
- Compare to bullets in a bulleted list
The perfect family size (II)

- If a class has more than a dozen children, additional subcategories may be necessary.
- However, if no natural classification exists, the long list may be more natural.
A completed hierarchy of wines
A “wine” is not a kind-of “wines”
A wine is an instance of the class Wines
Class names should be either
- all singular
- all plural
Classes and their names

- Classes represent concepts in the domain, not their names.
- The class name can change, but it will still refer to the same concept.
- **Synonym** names for the same concept are not different classes.
- Many systems allow listing synonyms as part of the class definition.
When to introduce a new class?

- **Subclasses of a class usually have**
  - Additional *properties*
  - Additional slot *restrictions*
  - Participate in different *relationships*

- **Subclasses of a class have**
  - New slots
  - New facet values
But

In terminological hierarchies, new classes do not have to introduce new properties
A new class or a property value?

- Do concepts with different slot values become restrictions for different slots?
- How important is the distinction for the domain?
- A class of an instance should not change often
A class or an instance?

- Individual instances are the most specific objects in an ontology
- If concepts form a natural hierarchy, represent them as classes
Metaclasses: Templates for class definitions

- Metaclasses enable us to add attributes to class definitions
- *By default, we have:*
  - Class name
  - Documentation
  - Slots
  - …
Metaclasses (II)

Additional attributes:
- Synonyms
- UMLS CUI
- Latin name
- Other class-level properties
Best Wineries

Red Bordeaux

Best Wineries:
- Chateâu Lafite Rothschild
- Chateâu Margaux
- Chateâu Latour
- Chateâu Haut-Brion
- Chateâu Mouton-Rothschild

Template Slots:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>tannin level</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={LOW, MODER}...</td>
</tr>
<tr>
<td>maker</td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED, ROS}...</td>
</tr>
<tr>
<td>grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td>body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL, MEDIUM,...}</td>
</tr>
<tr>
<td>flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE, MILD}</td>
</tr>
<tr>
<td>sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY, SWEET,...}</td>
</tr>
</tbody>
</table>
Defining a metaclass

Wine template *(type= STANDARD-CLASS)*

**Name**: Wine template
**Documentation**: Metaclass for wines, defining best wineries
**Role**: Concrete

**Template Slots**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>best wineries</td>
<td>Instance</td>
<td>multiple</td>
<td>classes= {Winery}</td>
</tr>
<tr>
<td>ROLE</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values = {Abstract, Concrete}</td>
</tr>
<tr>
<td>DOCUMENTATION</td>
<td>String</td>
<td>multiple</td>
<td></td>
</tr>
<tr>
<td>SLOT-CONSTRAINT</td>
<td>Instance</td>
<td>multiple</td>
<td>classes= {CONSTRAINT}</td>
</tr>
<tr>
<td>DIRECT-INSTANCE</td>
<td>Instance</td>
<td>multiple</td>
<td>classes= {THING}</td>
</tr>
<tr>
<td>DIRECT-SUPERCLASS</td>
<td>Class</td>
<td>multiple</td>
<td>parents= {THING}</td>
</tr>
<tr>
<td>DIRECT-CLASS</td>
<td>Class</td>
<td>multiple</td>
<td>parents= {THING}</td>
</tr>
</tbody>
</table>

**Supertypes**: STANDARD-CLASS
Domain and range of slot

- **Domain** of a slot – the class (or classes) that have the slot
  - More precisely: class (or classes) instances of which can have the slot

- **Range** of a slot – the class (or classes) to which slot values belong
When defining a domain or range for a slot, find the most general class or classes.

Consider the produces slot for a Winery:
- Range: Red wine, White wine, Rosé wine
- Range: Wine

Consider the flavor slot:
- Domain: Red wine, White wine, Rosé wine
- Domain: Wine
Defining domain and range

- A class and a superclass – replace with the superclass
- All subclasses of a class – replace with the superclass
- Most subclasses of a class – consider replacing with the superclass
Inverse slots

- **Maker and Producer** are *inverse slots*
Inverse slots contain *redundant information*, but:
- Allow acquisition of the information in either direction
- Enable additional verification
- Allow presentation of information in both directions

**The actual implementation differs from system to system**
- Are both values stored?
- When are the inverse values filled in?
- What happens if we change the link to an inverse slot?
Default values

- **Default value** – a value the slot gets when an instance is created

- A default value can be changed

- The default value is a *common* value for the slot, but is *not a required value*

- For example, the default value for wine *body* can be *FULL*
What’s in a name?

- Define a naming convention for classes and slots and adhere to it
- Features of an ontology tool to consider:
  - Can classes and slots have the same names?
  - Is the system case-sensitive?
  - What delimiters are allowed?
What’s in a name? (II)

• Capitalization and delimiters
  ▪ Use spaces: Meal course
  ▪ Run words together: MealCourse
  ▪ Use underscore or dash: Meal_Course

• Singular or plural
  ▪ Be consistent

• Prefix and suffix conventions
  ▪ Common for slots: has-maker, has-winery
  ▪ Wine rather than Wine class
  ▪ Consistency: if Red wine, then White wine
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
Limiting the scope (II)

- Ontology of wine, food, and their pairings probably will not include:
  - Bottle size
  - Label color
  - My favorite food and wine

- An ontology of biological experiments will contain:
  - Biological organism
  - Experimenter

- Is the class Experimenter a subclass of Biological organism?