Semantic Web

6. Ontology Engineering

PD Dr. Matthias Thimm

thimm@uni-koblenz.de

Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

Ontology engineering

- An ontology (or any other form of knowledge base)
  - can be used for reasoning, answering queries
  - should be reusable
- Process of building an ontology is called *ontology engineering*
  1. Define language
     - terms
     - concepts,
     - relations, etc.
  2. Define knowledge
     - What are equivalent concepts?
     - Are there subset relations?
     - Which constraints have to be imposed? etc.
  3. Use ontology for reasoning

Outline

1. Ontology Design Principles

2. Some philosophical issues

3. Summary

Outline

1. Ontology Design Principles

2. Some philosophical issues

3. Summary

Building ontologies - the basics

In practical terms, developing an ontology includes:

- defining classes (concepts) 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 (the 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.
- Ontology development is necessarily an iterative process.

A simple knowledge-engineering methodology

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies (or parts of them)
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of the classes (slots)
6. Define the facets of the slots
7. Create instances (of classes)

[Noy, McGuiness. Ontology Development 101: A Guide to Creating Your First Ontology]

Step 1: Determine the 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?
  → comprehensible modeling/design decisions

The answers to these questions may change during the ontology design process.

In general, these answers help to limit the scope of the model.

Step 1: Example

The Wine Ontology (http://www.w3.org/TR/owl-guide.rdf)

Competency questions:

- 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?

→ An ontology should contain enough information to answer these questions.

Step 2: Consider reusing existing ontologies

- Reuse existing ontologies as much as possible
- Profit from importing existing ontologies — not only cover the scope you need (to some extent), but they also include additional information, classification, axiomatization, 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?
- There are libraries of reusable ontologies.

If it does not work or fit you — design your own!

Step 3: Enumerate important terms

- What are the terms we would like to talk about?
- What are the properties that connect those terms?
- What would we like to say about those terms?
- Example for wine related terms:
  - Wine, grape, winery, location, etc.
  - 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, etc.

→ initially, it is important to get a comprehensive list of terms without worrying about overlapping

Step 4: Define the classes and hierarchy

- Methods
  - Top-down: Starts with the definition of the most general terms and subsequently specialize concepts
  - Bottom-up: Starts with the definition of the most specific terms
  - combination
- Start from defining classes
- Creating hierarchy will then be easier ...

Step 5: Define properties of classes — slots

Types of properties

- “Intrinsic” properties (essential 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 — 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 6: Define the facets of the slots

- Facets of relationships
  - means: Role restrictions
- Sample facets
  - Value type (e.g., value type of name is string)
  - Allowed values
  - Number of the values (cardinality - single, multiple ...)
  - ... other features of the values the slot can take
- When defining a domain or range ...
  - Find the most general classes or class that can be respectively the domain or the range for the slots
- 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.

Step 7: Create Instances

- Body: light-1
- Color: red-1
- Flavor: delicat-1
- Tannin level: low-1
- Grape: gamay-1 (instance of the Wine grape class)
- Maker: chateau-morgon-1 (instance of the Winery class)
- Region: beaujolais-1 (instance of the Wine-Region class)
- Sugar: dry-1

Further guidelines

Ontology engineering is an iterative process. The outcome should be checked and the process repeated.

- Ensure that the class hierarchy is correct
  - "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

  → this typically occur if singular and plural names are used, e.g., Wine is a subclass of Wines.
- Keep in mind **transitivity** of hierarchical relations
  - For instance: define a class White wine as a subclass of Wine. Then we define a class Chardonnay as a subclass of White wine. Transitivity of the subclass relationship means that the class Chardonnay is also a subclass of Wine. Chardonnay is a direct subclass (i.e., the closest subclass) of White wine and is not a **direct** subclass of Wine.

Further guidelines

- *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
- Siblings in class hierarchy: how many are 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 incomplete
    - If there are *more than a dozen subclasses* for a given class then *additional intermediate categories* may be necessary. (may not always be possible!)

Further guidelines

- 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

Further guidelines

- Counterexample to the rule for "When do you introduce a new class?"

  > *An ontology underlying an electronic medical-record system may include a classification of various diseases. This classification may be just a hierarchy of terms, without properties (or with the same set of properties).*

- Classes in terminological hierarchies do not have to introduce new properties

→ In that case, it is still useful to organize the terms in a hierarchy rather than a list.

Reasons:

- easier to explore and navigate
- easier/better selection with respect to concept granularity

Further guidelines

- 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 DOGMA approach (Developing Ontology-Grounded Methods and Applications):
  - Domain axiomatization vs. application axiomatization
  - Partitioning your ontology in these two promotes reusability *and* usability

Further guidelines

- Inverse slots
  - Functional or non-functional properties
  - What do you express with an inverse relation?
- Naming conventions
  - Are there available/well-established in general or in your field/area?
  - Stick to one naming convention - be consistent
  - Use existing vocabularies
- Synonyms
  - Just different name or really different objects?
  - Maybe multiple labels for the same object?
- Defaults
  - What values are there in case the user does not give any?
- Capitalization and delimiters
  - Some systems allow spaces in concept names
- Disjoint subclasses
  - What is the reason for introducing additional restrictions?

Outline

1. Ontology Design Principles
2. Some philosophical issues
3. Summary

Some philosophical issues 1/2

- Ontology engineering originates from philosophy
- Not only technical aspects (representation language, expressivity, . . .) have to be considered when representing knowledge

Example: `part-of`

- `ex:wheel1 ex:part-of ex:bike1`
- `ex:john ex:part-of ex:johnandevesmarriage`
- `ex:tree1 ex:part-of ex:forest1`
- `ex:piepiece1 ex:part-of ex:pie1`

→ is this the same `part-of`?

→ what about transitivity?

Some philosophical issues 2/2

Example: same-as

- ex:john owl:sameAs ex:johnsmith
- ex:universitaetsstr1 owl:sameAs ex:unikoblenz
- Many same-as links in linked open data are *wrong*

Example: Subclasses, properties, individuals

- RedWine is a subclass of Wine or is hasColor a property of Wine?
- Is Alice in Wonderland an instance of Book? What about my physical copy of Alice in Wonderland?
- Temporal aspects: Is Human a subclass of LivingBeing? What happens if someone dies?
- Intrinsic vs. extrinsic properties

Outline

1. Ontology Design Principles
2. Some philosophical issues
3. Summary

Summary

- There are steps worth following in designing ontology
- Have rationale for each of your design choices!
- Remember: there is no single correct ontology for modeling any domain
  - "Ontology design is a creative process and no two ontologies designed by different people would be the same."
- Designing ontology is an iterative process
- Ontology can evolve and change while you design it
- There are helpful methodologies and patterns:
  - http://ontologydesignpatterns.org/

Pointers to further reading

- Natasha Noy and Deborah McGuiness. 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.
- Riichiro Mizoguchi: Tutorial on Ontological Engineering: Part 3: Advanced Course of Ontological Engineering. New Generation Comput. 22(2): 193-220 (2004).
- Elena Simperl, Markus Luczak-Rösch. Collaborative ontology engineering: a survey. The Knowledge Engineering Review, Vol. 29:1, 101-131. 2013