Modular Ontology Modeling

Pascal Hitzler

Data Semantics Laboratory (DaSe Lab)
Center for Artificial Intelligence and Data Science (CAIDS)
Kansas State University
http://www.pascal-hitzler.de
Premise

Many ontologies are hard to understand and to re-use.

Some reasons:

- Poor (ad-hoc) modeling.
- Large, monolithic ontologies.
- Different use-case requirements on granularity (some parts too fine-grained, others too coarse).
- Different requirements on data representation for parts of the ontology (e.g., how spatial information is encoded).
Approach: Two main components

1. Modules
   - Rather than thinking of an ontology primarily as an enhanced taxonomy, we think of it as a set of interrelated (and possibly overlapping) modules.
   - Each module is essentially a part of an ontology representing a complex concept in a way which “makes sense” for a human expert. E.g., “oceanographic cruise”.

2. Use of ontology design patterns (ODPs)
   - An ODP is a solution template for a recurring ontology modeling problem.
   - ODPs are instantiated (and modified) to become modules. E.g., a general “Trajectory” ODP may be used as a template to create an “ocean science cruise trajectory” module.
Modeling Teamwork

The modeling team ideally has:
• domain experts
• data experts
• ontology engineers

Divide and Conquer
• First decide on the set of modules to be modeled, then draft modules one at a time.

Joint modeling
• Work mainly through schema diagrams and natural language with the domain and data experts.
• Ontology engineers spell out model details between meetings, and cycle back to the experts for feedback.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
A Few Pattern Examples
Joining patterns

**Generic AgentRole pattern**

owl:Thing \(\xrightarrow{\text{providesAgentRole}}\) AgentRole \(\xrightarrow{\text{performedBy}}\) Agent

AgentRole \(\xrightarrow{\text{startsAtTime, endsAtTime}}\) TimeInstant

**Generic NameStub pattern**

owl:Thing \(\xrightarrow{\text{hasName}}\) xsd:string

**Joined:**

owl:Thing \(\xrightarrow{\text{providesAgentRole}}\) AgentRole \(\xrightarrow{\text{performedBy}}\) Agent

Agent \(\xrightarrow{\text{hasName}}\) xsd:string
Patterns as templates

Joined AgentRole and NameStub patterns:

![Diagram of AgentRole and NameStub patterns](image)

Used as a template for a concrete modeling problem:

![Diagram of concrete modeling problem](image)
The Role Patterns

\[ T \sqsubseteq \forall \text{providesRole}. \text{Role} \]
\[ \exists \text{roleProvidedBy}. T \sqsubseteq \text{Role} \]
\[ \text{providesRole} \equiv \text{roleProvidedBy}^- \]
\[ T \sqsubseteq \forall \text{performsRole}. \text{Role} \]
\[ \exists \text{rolePerformedBy}. T \sqsubseteq \text{Role} \]
\[ \text{rolePerformedBy} \equiv \text{performsRole}^- \]
\[ \text{Role} \sqsubseteq \exists \text{hasTemporalExtent}. \text{TemporalExtent} \]
\[ \forall \text{hasTemporalExtent}. \text{TemporalExtent} \]
\[ \sqcap (\leq 1 \text{roleProvidedBy}. T) \]
\[ \sqcap (\leq 1 \text{rolePerformedBy}. T) \]
\[ \text{Role} \sqsubseteq \exists \text{roleProvidedBy}. T \sqcap \exists \text{rolePerformedBy}. T \]
\[ \text{DisjointClasses}(\text{Role}, \text{TemporalExtent}) \]

range
domain
inverse
range
domain
inverse
existential
scoped range
range cardinality
range cardinality
existentials
disjointness
The Agent Role Pattern

Axioms: all previous plus the following.

\[ \text{AgentRole} \sqsubseteq \text{Role} \]
\[ \exists \text{rolePerformedBy} . \text{Agent} \sqsubseteq \text{AgentRole} \]
\[ \text{AgentRole} \sqsubseteq \forall \text{rolePerformedBy} . \text{Agent} \]
Quantities and Units

Borrowed from the QUDT ontology
Provenance

Borrowed from PROV-O
The Stub Metapattern

Bottom: The CookingEquipmentStub derived from it.
Pascal Hitzler, Adila Krisnadhi
A Tutorial on Modular Ontology Modeling with Ontology Design Patterns: The Cooking Recipes Ontology.
Technical Report, DaSe Lab, Department of Computer Science and Engineering, Wright State University, Dayton, OH, August 2018. 22 pages

http://daselab.cs.wright.edu/pub2/mom-recipes-example.pdf
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Design an ontology which can be used as part of a “recipe discovery” website. The ontology shall be set up such that content from existing recipe websites can in principle be mapped to it (i.e., the ontology gets populated with data from the recipe websites). On the discovery website, detailed graph queries (using the ontology) shall produce links to recipes from different recipe websites as results. The ontology should be extendable towards incorporation of additional external data, e.g., nutritional information about ingredients or detailed information about cooking equipment.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Competency Questions

• From available data and from application use cases, devise competency questions, i.e. questions which should be convertible into queries, which in turn should be answerable using the data.

Gluten-free low-calorie desserts.
How do I make a low-carb pot roast?
How do I make a Chili without beans?
Sweet breakfast under 100 calories.
Breakfast dishes which can be prepared quickly with 2 potatoes, an egg, and some our.
How do I prepare Chicken thighs in a slow cooker?
A simple recipe with pork shoulder and spring onions.
A side prepared using Brussels sprouts, bacon, and chestnuts.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.

3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.

4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.

5. Add axioms for each module, informed by the pattern axioms.

6. Put the modules together and add axioms which involve several modules.

7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.

8. Create OWL files.
Key notions

- Use the competency questions.
- Possibly also query domain experts as to the main notions for the application domain.
- E.g. for the recipes scenario, these would include
  - Recipe
  - Food
  - Time
  - Equipment
  - Classification of food (e.g., as a side)
  - Difficulty level
  - Nutritional information
  - Provenance
Key notions

- Then prioritize which notions to model first. In this case, e.g.
  - recipe
  - food
  - equipment
  - classification
  - difficulty level
  - time
  - nutritional information
  - provenance
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Identifying suitable patterns

• Understand the nature of the things you are modeling.

Food: A concrete piece of food? An abstract quantity of food?
Equipment: Do we want a complex model at this stage? No. Stub
Classification: Do we want a complex model at this stage? No. Stub
Difficulty level: Do we want a complex model at this stage? No. Stub
Time: Probably already incorporated in plan?
Nutritional information: model along some existing standard?
Provenance: just that!
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Recipe

A plan, a description.

Pattern:
Plan as a Description

Recipe

Recipe Instructions

hasRequiredTime

TimeInterval
Food

An abstract **quantity** of food.

Pattern:
**QuantityOfStuff** (with **Quantity** sub-pattern)

(derived from **QUDT**)

IOF TOB online presentation, January 2020
Equipment

No complex model desired at this stage. We just want to use strings, i.e., use our stub meta-pattern.

Figure 2.10: Top, the Stub (meta)pattern. Bottom, its instantiation for equipment.
Classification (e.g., entrée)

No complex model desired at this stage. We just want to use strings, i.e., use our **stub meta-pattern**.
No complex model desired at this stage. We just want to use strings, i.e., use our **stub meta-pattern**.
Already incorporated in plan!
Use an ontology design pattern based on PROV-O.

PROV-O derived Provenance pattern:

We’ll use only this:
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Nutritional information

Model along some existing standard.

Let’s use the U.S. FDA Nutritional Facts label standard.
Nutritional information

Model along some existing standard.

Figure 2.13: Nutritional Information module. The box indicates a modified instance of the QuantityOfStuff pattern.
Adequacy check

- Triplify sample data using the ontology.  Does it work?
- Check if competency questions can be answered.
- Add axioms as appropriate (the graph is only for intuition, the OWL axioms are the actual ontology).
- (there are more post-hoc details to be taken care of, but let’s leave it at that)
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Figure 2.17: Generic node-edge-node schema diagram for explaining systematic axiomatization

1. $A \cap B \subseteq \perp$
2. $\exists R.T \subseteq A$
3. $\exists R.B \subseteq A$
4. $T \subseteq \forall R.B$
5. $A \subseteq \forall R.B$
6. $A \subseteq R.B$
7. $B \subseteq R^{-}.A$
8. $T \subseteq \leq 1R.T$
9. $T \subseteq \leq 1R.B$
10. $A \subseteq \leq 1R.T$
11. $A \subseteq \leq 1R.B$
12. $T \subseteq \leq 1R^{-}.T$
13. $T \subseteq \leq 1R^{-}.A$
14. $B \subseteq \leq 1R^{-}.T$
15. $B \subseteq \leq 1R^{-}.A$

Figure 2.18: Most common axioms which could be produced from a single edge $R$ between nodes $A$ and $B$ in a schema diagram: description logic notation.
Figure 2.19: Most common axioms which could be produced from a single edge $R$ between nodes $A$ and $B$ in a schema diagram: Manchester syntax.
Example Axiomatization

ofFoodType, ofQuantity: scoped range, existential
hasQuantityKind, hasQuantityValue: scoped domain, scoped range, existential, inverse existential, scoped qualified functionality
hasUnit: scoped range, existential, scoped qualified functionality
hasNumericValue: scoped range, existential, functionality
Mutually disjoint: QuantityOfFood, FoodType, QuantityKind, Quantity, QuantityValue, Unit
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. **Put the modules together and add axioms which involve several modules.**
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
• We are currently developing a set of compatible tools, as Protégé plug-ins.
  – See http://comodide.com/

• We are also developing ODP libraries.
  – See https://daselab.cs.ksu.edu/content/modl-modular-ontology-design-library
Thanks!
References


References


References


References


References


