Insights from an OTTR-centric Ontology Engineering Methodology

Moritz Blum, Basil Ell, Philipp Cimiano
{mblum, bell, cimiano}@techfak.uni-bielefeld.de
Motivation: Ontology Engineering with OTTR

- development of large ontologies = multiple design decisions at the same time
- OTTR is a language for ontology modelling patterns → instances build ontology
  - hide complexity from the domain experts
  - separating what to model from how to model
- our approach
  - ↑ bottom-up: begin with existing data
  - ↓ top-down: OTTR template headers first, and bodies are developed iteratively

💬 enhanced communication with domain experts
⚙ agile engineering process
Reasonable Ontology Templates (OTTR)

A template library

A template definition

\[
\text{pz:Pizza} [ \text{?name} , \text{?label} ] \text{ :: } \{
\text{ottr:Triple( ?name, rdf:type, owl:Class },
\text{ax:SubClassOf( ?name, p:Pizza },
\text{ottr:Triple( ?name, rdfs:label, ?label )}
\}. 
\]
Reasonable Ontology Templates (OTTR)

a template library

a template definition

```prolog
ax:SubClassOf( ?sub, ?super ) :: {
    ottr:Triple( ?sub, rdfs:subClassOf, ?super )
}.
```

a template definition

```prolog
pz:Pizza( ?name, ?label ) :: {
    ottr:Triple( ?name, rdf:type, owl:Class ),
    ax:SubClassOf( ?name, p:Pizza ),
    ottr:Triple( ?name, rdfs:label, ?label )
}.
```
Reasonable Ontology Templates (OTTR)

a template library

a template definition

ax:SubClassOf[ ?sub, ?super ] :: {
    otrr:Triple( ?sub, rdfs:subClassOf, ?super )
}.

a template definition

pz:Pizza[ ?name , ?label ] :: {
    otrr:Triple( ?name, rdf:type, owl:Class ),
    ax:SubClassOf( ?name, p:Pizza ),
    otrr:Triple( ?name, rdfs:label, ?label )
}.

a template instance

pz:Pizza( p:Margherita, "Margherita"@it )

result of instantiation

p:Margherita rdf:type owl:Class .
p:Margherita rdfs:label "Margherita"@it .

Moritz Blum, Basil Ell, Philipp Cimiano
Domain: Material Science

● project DiProMag: model experiments related to magnetocaloric alloys from the production, over the characterization to the prototypical application in an application ontology
● close collaboration between ontology engineers and domain experts
● favors a bottom-up ontology engineering approach

Project Website: https://www.dipromag.de/
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration

Moritz Blum, Basil Ell, Philipp Cimiano
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration
9. 

Moritz Blum, Basil Ell, Philipp Cimiano
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration
Step 2 – Design of Template Headers

Template header: \texttt{ex:Pizza[ottr:IRI ?name, xsd:string ?label]}

- abstraction simplifies the communication with domain experts
- design trade-offs like
  - complexity of templates vs. complexity of template relations
    - vs.
  - multiple similar but specific templates vs. few general templates
    - \texttt{ex:PizzaLarge & ex:PizzaMedium & ex:PizzaSmall}
    - vs.
      - \texttt{ex:PizzaOfDiffSize}
- result = initial collection of template headers (improve iteratively)
Step 2 – Design of Template Headers

Experiences:

- derived template parameters directly from the available data
- one template per process step, i.e., one template per material type, one template per synthesis method, and one template per measurement method
- encountered trade-offs: splitting data across multiple templates concerns the correlations and dependencies between data points
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration
Step 4 – Design of Template Bodies

- straightforward, as a pre-structuring is already done
- use template header documentation
  - benefit from collected domain knowledge about, e.g., domain-specific terms
  - to identify sub-templates (iterative development – step back to template header design)
  - to identify and reuse existing ontologies
- possible difficulties
  - incompatibility to the ontologies selected for reuse
  - modeling redundancy → potential inconsistency: avoid by introducing appropriate sub-templates

Experiences:

- modelling processes according to existing ontologies like EMMO required to revisit “Design of Template Headers” to introduce additional template parameters
- a collection of similar physical experiments shares the same parameters (e.g., about the environment)
  → sub-templates were introduced
Methodology

1. Define Scope & Screen Available Data
2. Design of Template Headers
3. Template Header Design Verification & Documentation
4. Design of Template Bodies
5. Inclusion of Axiomatic Triples
6. Template Body Documentation
7. Template Library Documentation
8. Template Instantiation & Data Integration
Step 7 – Template Library Documentation

- list of all templates
- template inclusion/call hierarchy
- individual template documentation
- instantiation order and naming guidelines, e.g.,
  “BulkSample” + \{2\} initials of the creator + YYMMDD + \{0-9\}\{2\} enumerator

Figure 3: OTTR documentation

Figure 4: Template instantiation workflow

Moritz Blum, Basil Ell, Philipp Cimiano
ODPs are compatible with OTTR

+ benefit of OTTR features, e.g., dealing with optional/default parameters or checking data type constraints
Conclusion

- ontology engineering methodology that relies on OTTR templates
  - key steps: defining the ontology’s scope, …, instantiating templates and integrating data
  - our OTTR-based method should be combined with existing ontology engineering practices
  - compatible with ODPs
- experience from a project (~90 templates)
  - low efforts when we had to revisit design decisions
  - greatly helped for communicating with domain experts
Thank you for your attention!