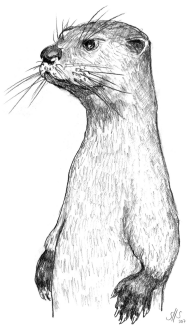
UiO : Department of Informatics
University of Oslo

# Reasonable Ontology Templates

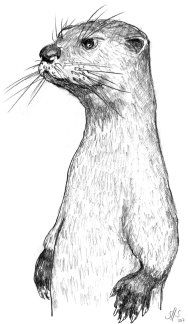
**Martin G. Skjæveland** Henrik Forssell
Johan W. Klüwer Daniel Lupp
Evgenij Thorstensen Arild Waaler

WOP2017
October 21, 2017



OTTR

- Tools and methodology
- A better way of constructing ontologies
- OWL macros expressed in OWL
  - abstractions, encapsulation, clear interface
  - nested
- template: $\langle \textit{parameters} \rangle \mapsto \mathcal{O}$
- instance: $\langle \textit{arguments} \rangle \mapsto \mathcal{O}_{\textit{arguments}}$
- OWL serialisation
  - OWL reasoning
  - Leverage W3C stack and tools
- Extensible framework for representing and transforming data
- Demo



OTTR

# Ontology templates

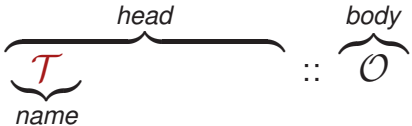
- Template:

# Ontology templates

- Template:

# Ontology templates

- Template:



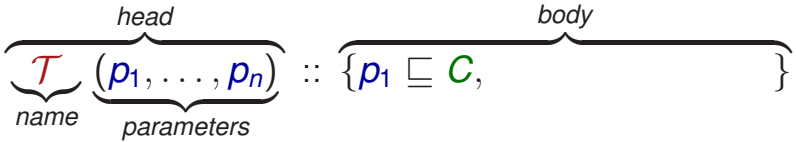
# Ontology templates

- Template:

$$\overbrace{\underbrace{\mathcal{T}}_{\textit{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\textit{parameters}}}^{\textit{head}} :: \overbrace{\mathcal{O}}^{\textit{body}}$$

# Ontology templates

- Template:




- Parameters can be any concept, role, or individual name or data value

# Ontology templates

- Template:

$$\overbrace{\underbrace{\mathcal{T}}_{\text{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\text{parameters}}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, \quad\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value

# Ontology templates

- Template:

$$\overbrace{\underbrace{\mathcal{T}}_{\text{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\text{parameters}}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value

# Ontology templates

- Template:

$$\overbrace{\underbrace{\mathcal{T}}_{\text{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\text{parameters}}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1\ \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}$$

# Ontology templates

- Template:

$$\overbrace{\underbrace{\mathcal{T}}_{\text{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\text{parameters}}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(p_4, \ldots)\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1\ \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3)\}$$

- Templates can be (non-cyclically) nested

# Ontology templates

- Template:

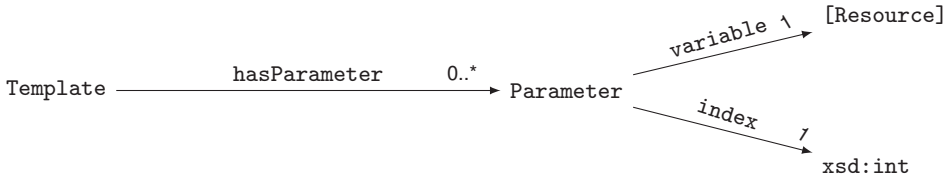
$$\overbrace{\underbrace{\mathcal{T}}_{\text{name}}\ \underbrace{(p_1, \ldots, p_n)}_{\text{parameters}}}^{\text{head}} :: \overbrace{\{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(p_4, \ldots)\}}^{\text{body}}$$

- Parameters can be any concept, role, or individual name or data value
- Template instance:

$$\mathcal{T}_1\ \underbrace{(p_1, p_2, p_3)}_{\text{arguments}} \Rightarrow \{p_1 \sqsubseteq C, D \sqsubseteq \exists p_2.E, F(p_3), \mathcal{T}_2(\ldots)\}$$

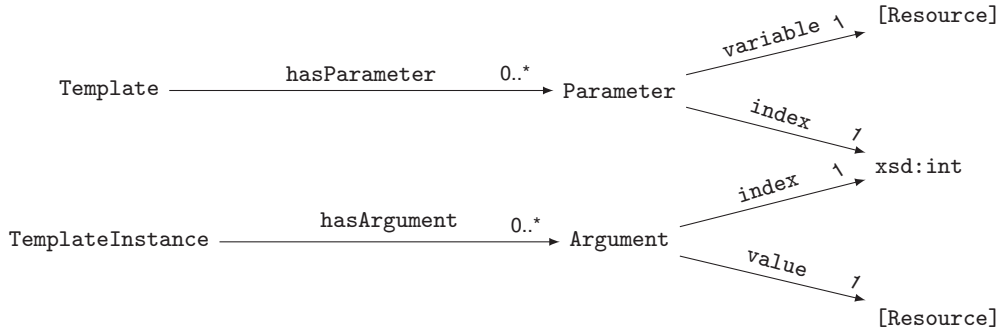
- Templates can be (non-cyclically) nested
- Instances are expanded recursively

# OTTR OWL vocabulary



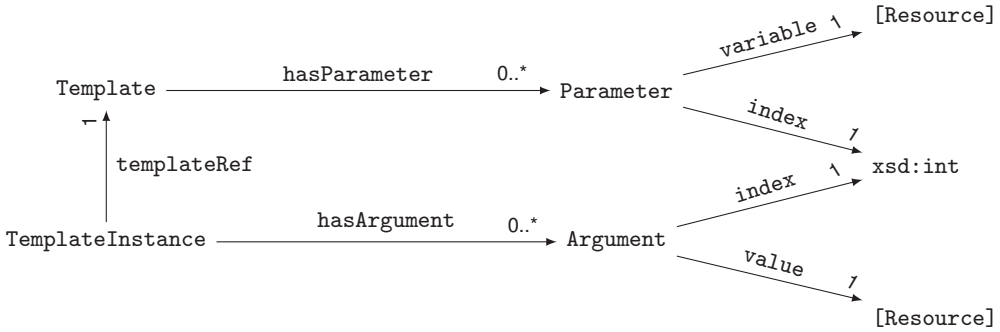

http://ns.ottr.xyz

# OTTR OWL vocabulary



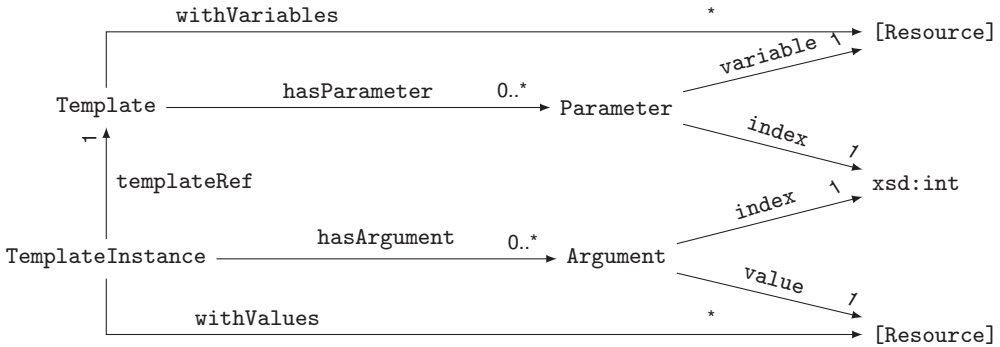

http://ns.ottr.xyz

# OTTR OWL vocabulary




http://ns.ottr.xyz

# OTTR OWL vocabulary




http://ns.ottr.xyz

$$\mathit{PartOf}(\mathit{Whole}, \mathit{Part}) :: \{\mathit{Whole} \sqsubseteq \exists\, \mathit{hasPart}.\mathit{Part}\}$$

$$\mathit{PartOf}(\mathit{Whole}, \mathit{Part}) :: \{\mathit{Whole} \sqsubseteq \exists\, \mathit{hasPart}.\mathit{Part}\}$$

$$\mathit{PartOf}(\mathit{Hammer}, \mathit{Handle})$$

$$\mathit{PartOf}(\mathit{Whole}, \mathit{Part}) :: \{\mathit{Whole} \sqsubseteq \exists\, \mathit{hasPart}.\mathit{Part}\}$$

$$\mathit{PartOf}(\mathit{Hammer}, \mathit{Handle}) \Rightarrow$$
$$\{\mathit{Hammer} \sqsubseteq \exists\, \mathit{hasPart}.\mathit{Handle}\}$$

$PartOf(Whole, Part) :: \{Whole \sqsubseteq \exists\, hasPart.Part\}$

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .


  ### body:
:Part  a owl:Class .
:Whole a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

$PartOf(Whole, Part) :: \{Whole \sqsubseteq \exists\, hasPart.Part\}$

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .


  ### body:
:Part  a owl:Class .
:Whole a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

$PartOf(Hammer, Handle)$

*PartOf*(*Whole*, *Part*) :: {*Whole* $\sqsubseteq \exists$ *hasPart*.*Part*}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .


  ### body:
:Part  a owl:Class .
:Whole a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
```

*PartOf*(*Whole*, *Part*) :: {*Whole* $\sqsubseteq \exists$ *hasPart*.*Part*}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
:Part  a owl:Class .
:Whole a owl:Class ;
    rdfs:subClassOf [ a owl:Restriction ;
        owl:onProperty ex:hasPart ; owl:someValuesFrom :Part ] .
```

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

*PartOf*(*Whole*, *Part*) :: {*SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*)}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
   ottr:withValues ( :Whole ex:hasPart :Part ) .
```

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

*PartOf*(*Whole*, *Part*) :: {*SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*)}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
   ottr:withValues ( :Whole ex:hasPart :Part ) .
```

- *RDF* macros
- not only OWL
- use for LOD

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

*PartOf*(*Whole*, *Part*) :: {*SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*)}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
   ottr:withValues ( :Whole ex:hasPart :Part ) .
```

- *RDF* macros
  - not only OWL
  - use for LOD
- Expansion:
  - copy template graph
  - paragraph $\mapsto$ argument
  - recurse

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

*PartOf*(*Whole*, *Part*) :: {*SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*)}

```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
   ottr:withValues ( :Whole ex:hasPart :Part ) .
```

*PartOf*(*Hammer*, *Handle*)

```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

- *RDF* macros
  - not only OWL
  - use for LOD
- Expansion:
  - copy template graph
  - paragraph $\mapsto$ argument
  - recurse
- Arguments
  - any RDF resource
  - incl. e.g., RDF(S)/OWL voc.
  - template IRIs
  - lists
    - variable-length args
    - multiple instances

*PartOf*(*Whole*, *Part*) :: {*SubSomeValuesFrom*(*Whole*, *hasPart*, *Part*)}

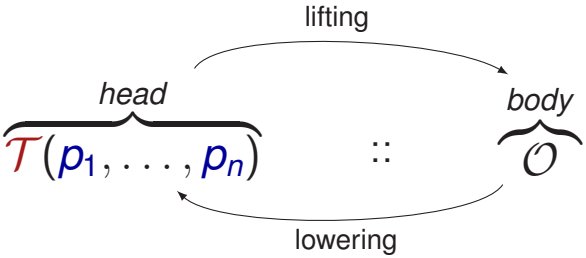
```
  ### head:
<http://draft.ottr.xyz/i17/partof> a ottr:Template ;
    ottr:hasParameter [ ottr:index 1; ottr:variable :Whole ] ,
                      [ ottr:index 2; ottr:variable :Part  ] .
    ## ottr:withVariables ( :Whole :Part ) .

  ### body:
[] ottr:templateRef ottr-owl:SubSomeValuesFrom ;
   ottr:withValues ( :Whole ex:hasPart :Part ) .
```

*PartOf*(*Hammer*, *Handle*)

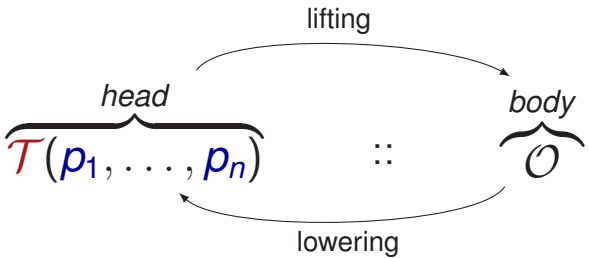
```
[] ottr:templateRef <http://draft.ottr.xyz/i17/partof> ;
   ottr:hasArgument [ ottr:index 1; ottr:value ex:Hammer ] ,
                    [ ottr:index 2; ottr:value ex:Handle ] .
   ## ottr:withValues ( ex:Hammer ex:Handle ) .
```

- *RDF* macros
  - not only OWL
  - use for LOD
- Expansion:
  - copy template graph
  - paragraph $\mapsto$ argument
  - recurse
- Arguments
  - any RDF resource
  - incl. e.g., RDF(S)/OWL voc.
  - template IRIs
  - lists
    - variable-length args
    - multiple instances
- Reasoning
  - body
  - head
  - expanded
  - instantiated

# Extensible framework



# Extensible framework

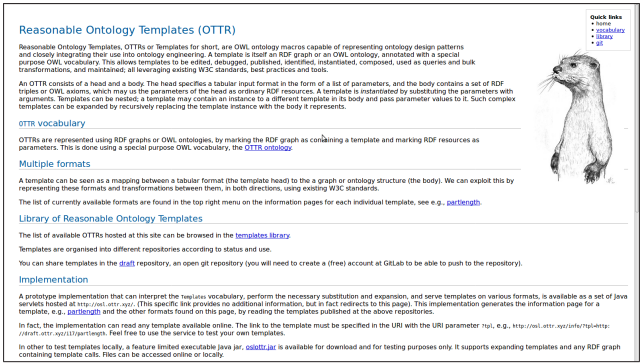



| head instance | head format | lifting & lowering | body format |
|---|---|---|---|
| XML, XSLx | XSD+SAWSDL | XSLT | OWL |
| OTTRs (RDF) | (SHACL, ShEx) | SPARQL | OWL |

*All formats generated from templates!*

# Resources

- http://www.ottr.xyz



Reasonable Ontology Templates (OTTR)

Quick links
- home
- vocabulary
- library
- git

Reasonable Ontology Templates, OTTRs or Templates for short, are OWL ontology macros capable of representing ontology design patterns and closely integrating their use into ontology engineering. A template is itself an RDF graph or an OWL ontology, annotated with a special purpose OWL vocabulary. This allows templates to be edited, debugged, published, identified, instantiated, composed, used as queries and bulk transformations, and maintained; all leveraging existing W3C standards, best practices and tools.

An OTTR consists of a head and a body. The head specifies a tabular input format in the form of a list of parameters, and the body contains a set of RDF triples or OWL axioms, which may us the parameters of the head as ordinary RDF resources. A template is *instantiated* by substituting the parameters with arguments. Templates can be nested; a template may contain an instance to a different template in its body and pass parameter values to it. Such complex templates can be expanded by recursively replacing the template instance with the body it represents.

OTTR vocabulary

OTTRs are represented using RDF graphs or OWL ontologies, by marking the RDF graph as containing a template and marking RDF resources as parameters. This is done using a special purpose OWL vocabulary, the OTTR ontology.

Multiple formats

A template can be seen as a mapping between a tabular format (the template head) to the a graph or ontology structure (the body). We can exploit this by representing these formats and transformations between them, in both directions, using existing W3C standards.

The list of currently available formats are found in the top right menu on the information pages for each individual template, see e.g., partlength.

Library of Reasonable Ontology Templates

The list of available OTTRs hosted at this site can be browsed in the templates library.

Templates are organised into different repositories according to status and use.

You can share templates in the draft repository, an open git repository (you will need to create a (free) account at GitLab to be able to push to the repository).

Implementation

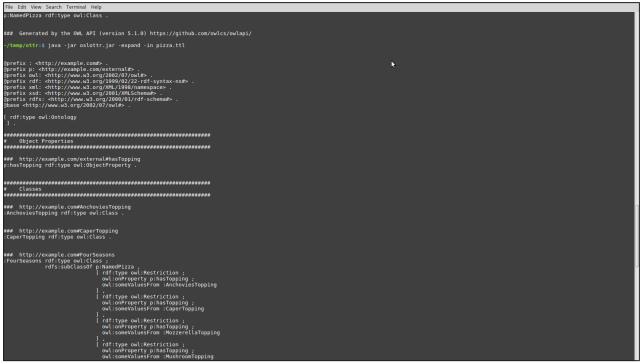
A prototype implementation that can interpret the `Templates` vocabulary, perform the necessary substitution and expansion, and serve templates on various formats, is available as a set of Java servlets hosted at `http://osl.ottr.xyz/`. (This specific link provides no additional information, but in fact redirects to this page). This implementation generates the information page for a template, e.g., partlength and the other formats found on this page, by reading the templates published at the above repositories.

In fact, the implementation can read any template available online. The link to the template must be specified in the URI with the URI parameter `tpl`, e.g., `http://osl.ottr.xyz/info?tpl=http://draft.ottr.xyz/137/partlength`. Feel free to use the service to test your own templates.

In other to test templates locally, a feature limited executable java jar, oslottr.jar is available for download and for testing purposes only. It supports expanding templates and any RDF graph containing template calls. Files can be accessed online or locally.

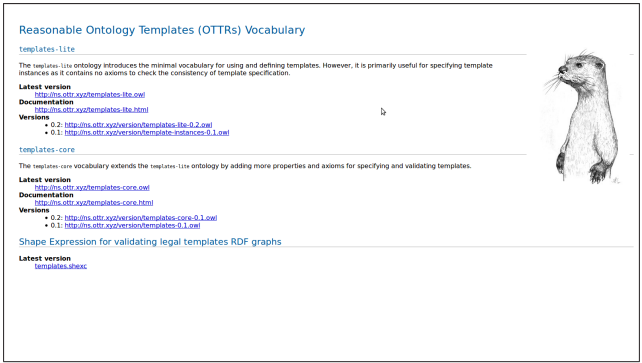
# Resources

- `http://www.ottr.xyz`
  - CLI java tool



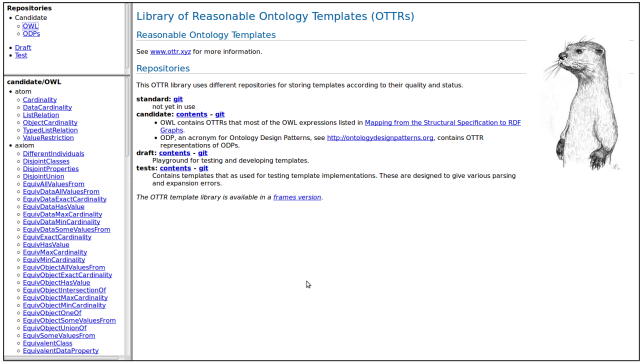
# Resources

- `http://www.ottr.xyz`
  - CLI java tool
- OTTR vocabulary:
  `http://ns.ottr.xyz`



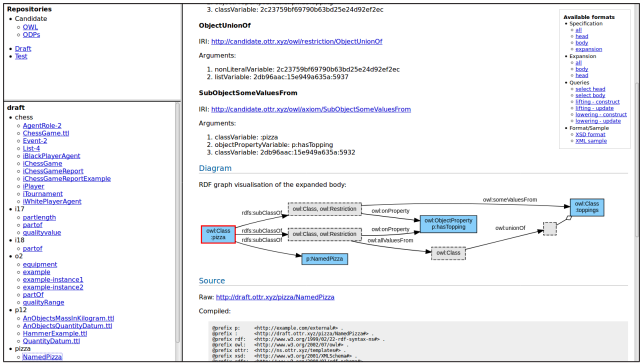
# Resources

- `http://www.ottr.xyz`
  - CLI java tool
- OTTR vocabulary:
  `http://ns.ottr.xyz`
- Library of OTTRs:
  `http://library.ottr.xyz`



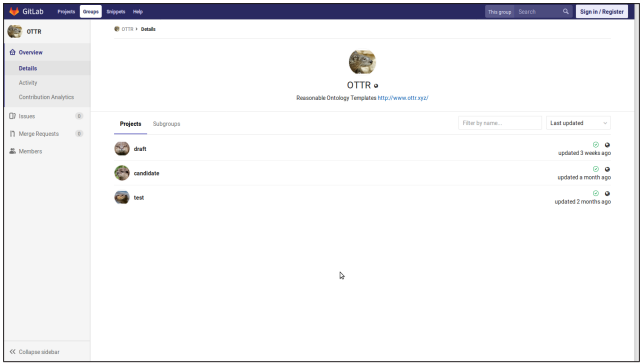
# Resources

- `http://www.ottr.xyz`
  - CLI java tool
- OTTR vocabulary:
  `http://ns.ottr.xyz`
- Library of OTTRs:
  `http://library.ottr.xyz`
- Online web application:
  `http://osl.ottr.xyz`



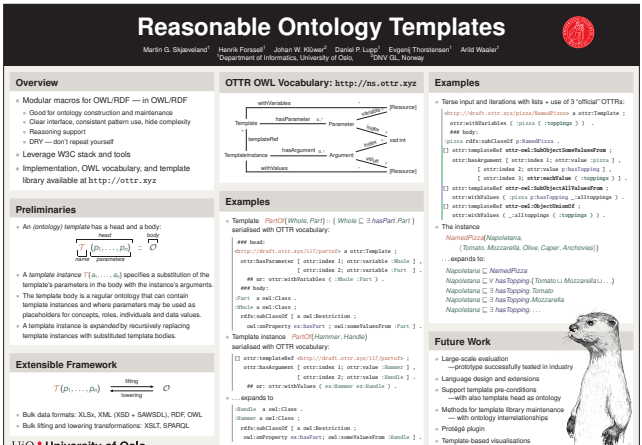
# Resources

- `http://www.ottr.xyz`
  - CLI java tool
- OTTR vocabulary:
  `http://ns.ottr.xyz`
- Library of OTTRs:
  `http://library.ottr.xyz`
- Online web application:
  `http://osl.ottr.xyz`
- Git:
  `http://gitlab.com/ottr/`



# Resources

- `http://www.ottr.xyz`
  - CLI java tool
- OTTR vocabulary:
  `http://ns.ottr.xyz`
- Library of OTTRs:
  `http://library.ottr.xyz`
- Online web application:
  `http://osl.ottr.xyz`
- Git:
  `http://gitlab.com/ottr/`
- ISWC Poster



# The Future and Future work

- Engineering ontologies using only OTTR templates
- API for OWL (— or AI for OWL)

- Ontology experts construct OTTRs
- “Knowledge engineers” combine OTTRs to user-facing templates
- Domain experts use tabular format

# The Future and Future work

- Engineering ontologies using only OTTR templates
- API for OWL (— or AI for OWL)

- Ontology experts construct OTTRs
- “Knowledge engineers” combine OTTRs to user-facing templates
- Domain experts use tabular format

- Large-scale evaluation
- Further development of templates and tools

- Theory for structuring a library templates, e.g., what is a subtemplate?
- Template library maintenance

- Ontology as pre-condition: template heads as ontologies too

- Protege plugin
- Excel plugin