UNIVERSITÄT BIELEFELD
Faculty of Technology

Insights from an OTTR-centric Ontology Engineering Methodology

Moritz Blum, Basil Ell, Philipp Cimiano
{mblum, bell, cimiano}@techfak.uni-bielefeld.de

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Grant No. 13XP5120B (DiProMag)

SIRIUS
Project No. 237898

14th Workshop on Ontology Design and Patterns (November 6, 2023)

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

Figure 1: Example Template Hierarchy

Reasonable Ontology Templates (OTTR)

a template library

a template definition

```
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 ] :: {
    ottr:Triple( ?sub, rdfs:subClassOf, ?super )
} .
```

a template definition

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


Moritz Blum, Basil Ell, Philipp Cimiano

Reasonable Ontology Templates (OTTR)

a template library

a template definition

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

a template definition

```
pz:Pizza[ ?name , ?label ] :: {
   ottr:Triple( ?name, rdf:type, owl:Class ),
   ax:SubClassOf( ?name, p:Pizza ),
   ottr: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:subClassOf p:Pizza .
p:Margherita rdfs:label "Margherita"@it .
```


Moritz Blum, Basil Ell, Philipp Cimiano

UNIVERSITÄT BIELEFELD
Faculty of Technology

Domain: Material Science 

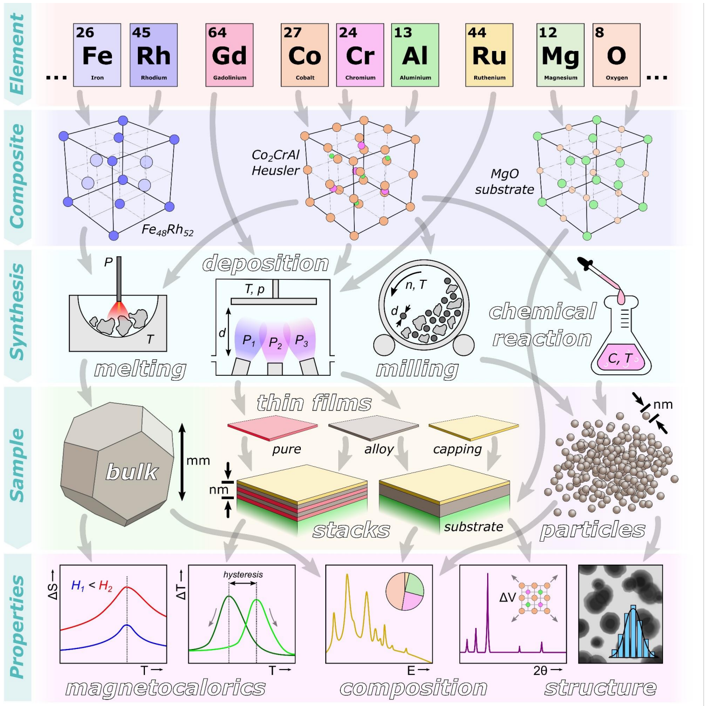


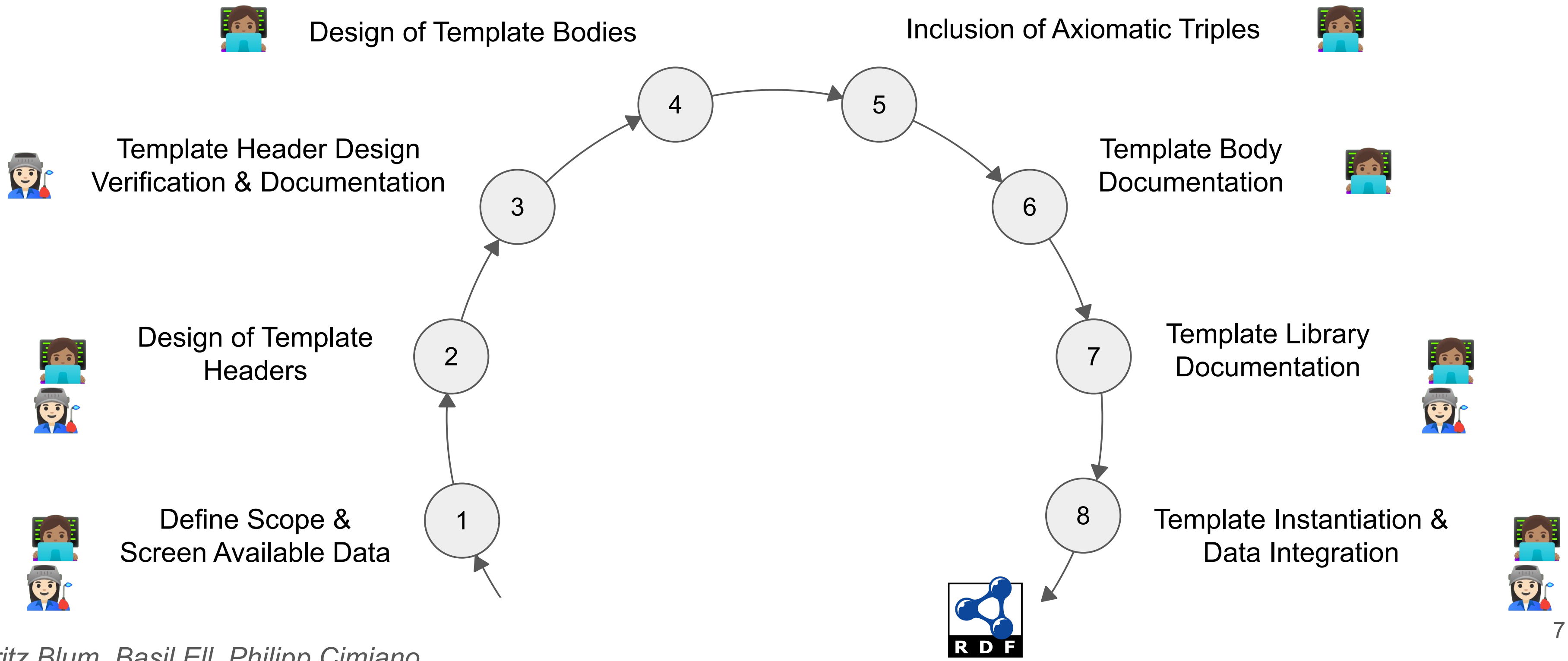
Figure 2: Material science workflow

- 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

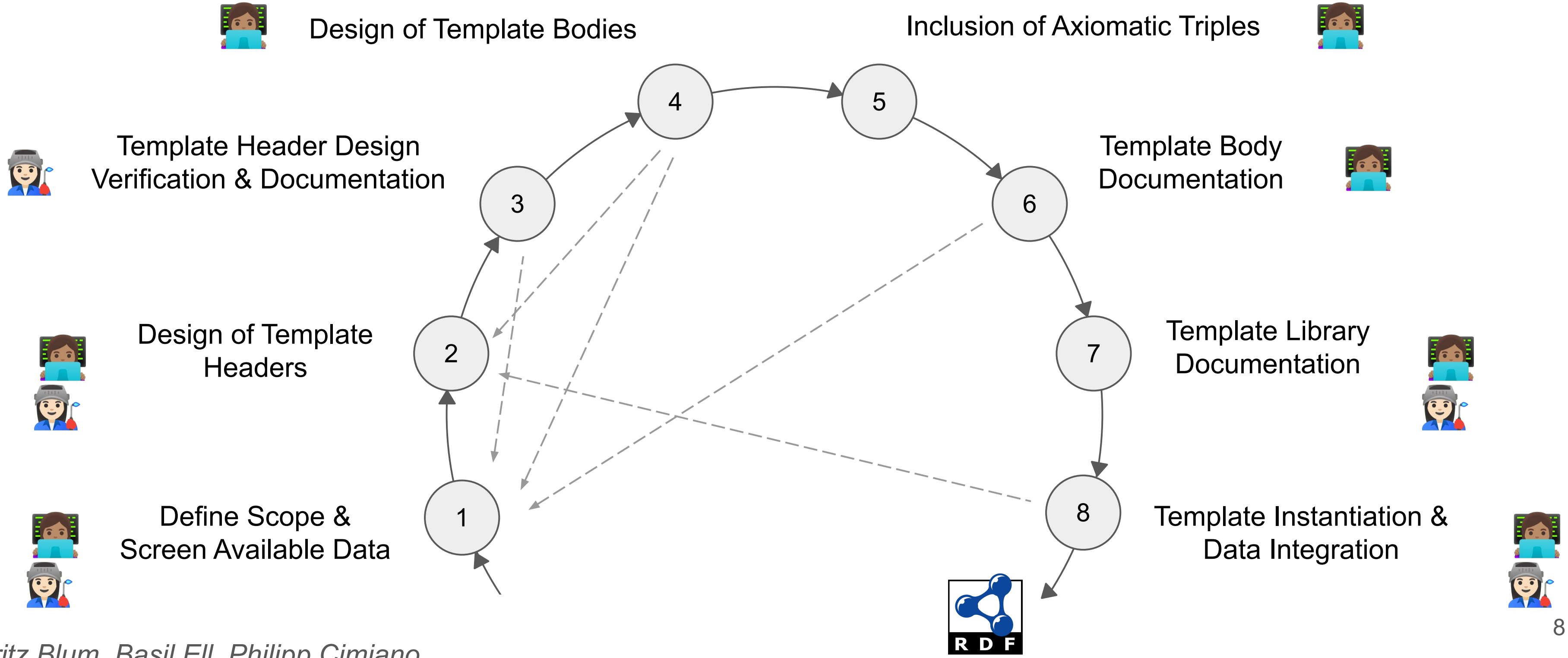
Project Website: https://www.dipromag.de/

Moritz Blum, Basil Ell, Philipp Cimiano

Methodology

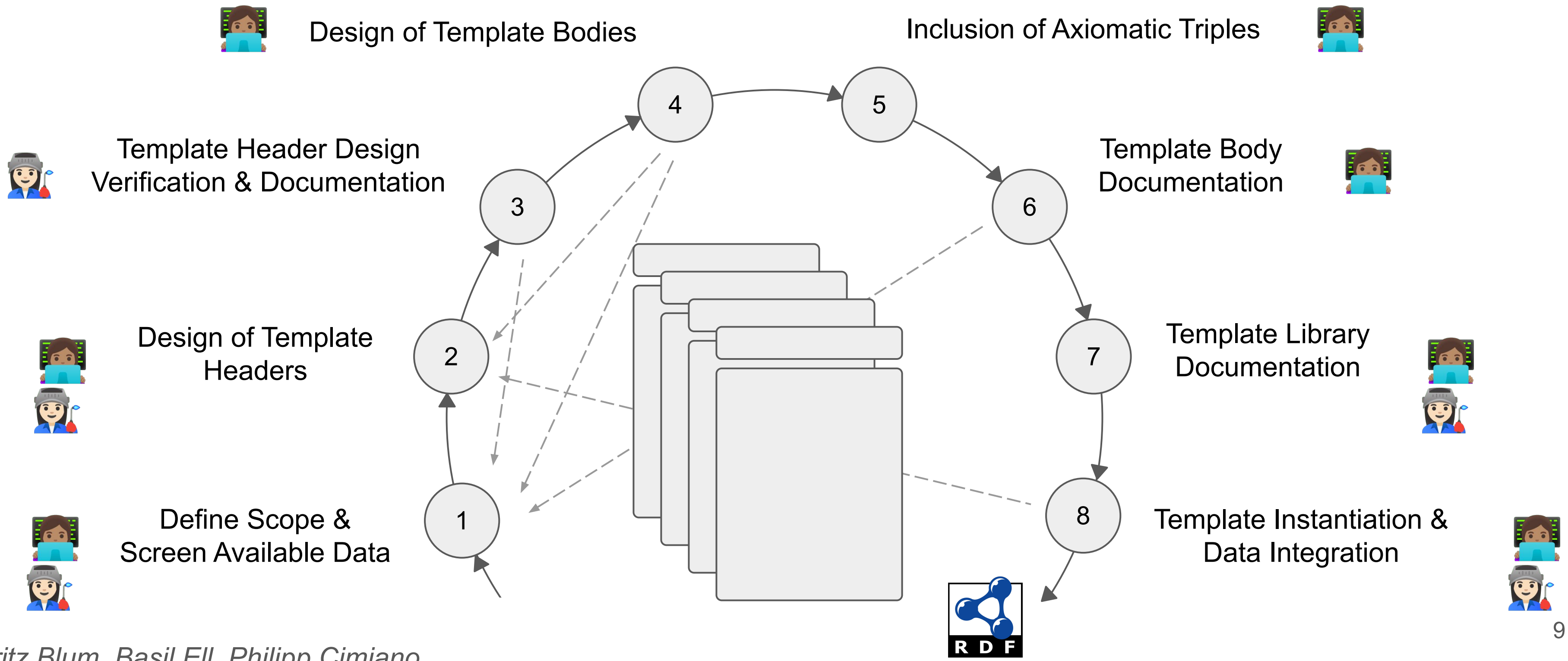


Methodology

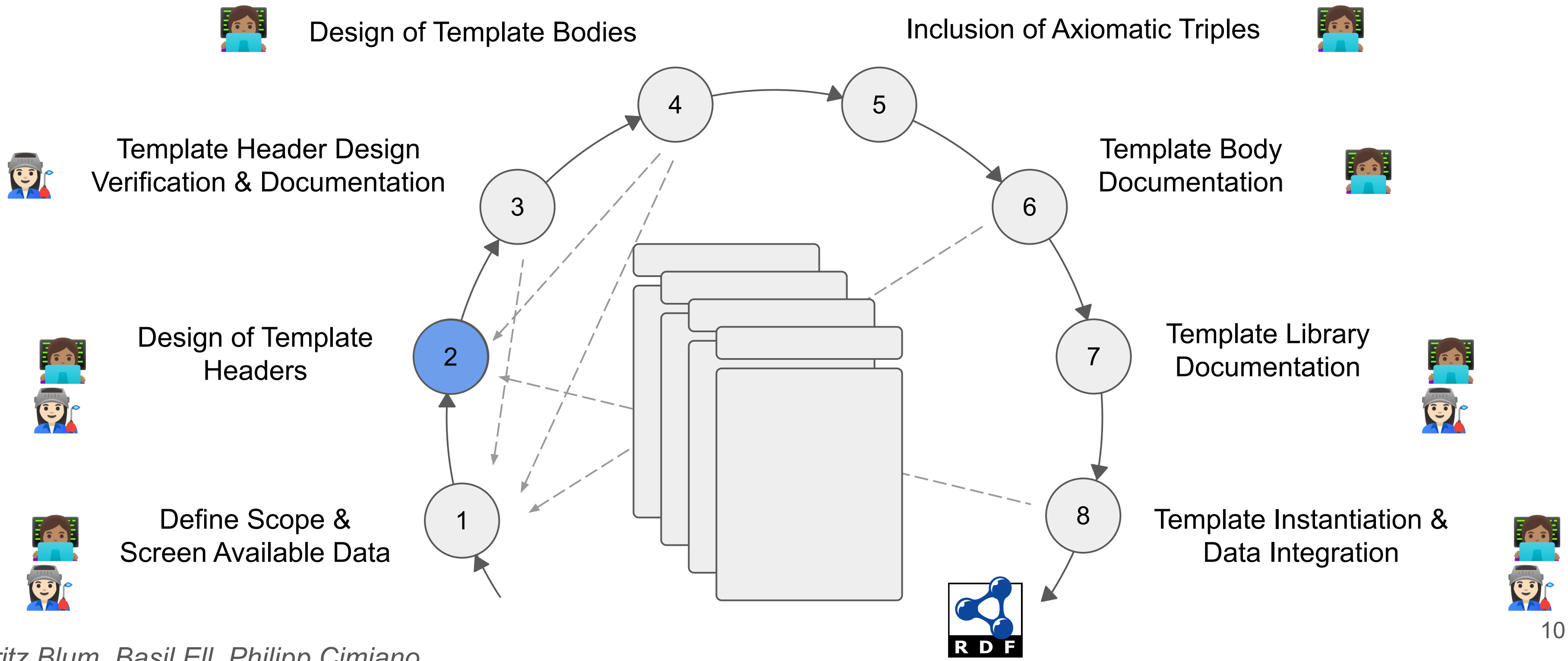


Moritz Blum, Basil Ell, Philipp Cimiano

Methodology



Methodology



Step 2 – Design of Template Headers

Template header: **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
    `ex:Pizza[ottr:IRI ?name, xsd:string ?label, ottr:IRI ?cheese, ottr:IRI ?cheese_age]`
    vs.
    `ex:Pizza[ottr:IRI ?name, xsd:string ?label, ottr:IRI ?cheese] &`
    `ex:Cheese[ottr:IRI ?cheese, ottr:IRI ?cheese_age]`

  - multiple similar but specific templates vs. few general templates
    `ex:PizzaLarge` & `ex:PizzaMedium` & `ex:PizzaSmall`
    vs.
    `ex:PizzaOfDiffSize`
- result = initial collection of template headers (improve iteratively)

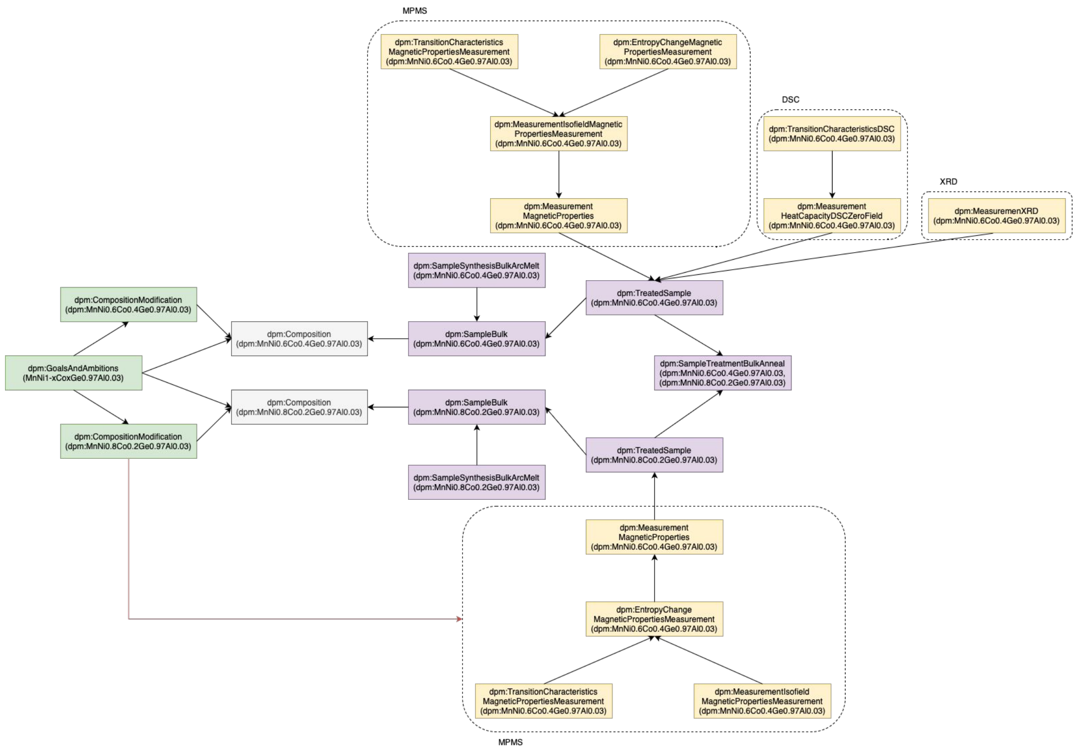
Moritz Blum, Basil Ell, Philipp Cimiano

UNIVERSITÄT BIELEFELD
Faculty of Technology

Step 2 – Design of Template Headers

Experiences:

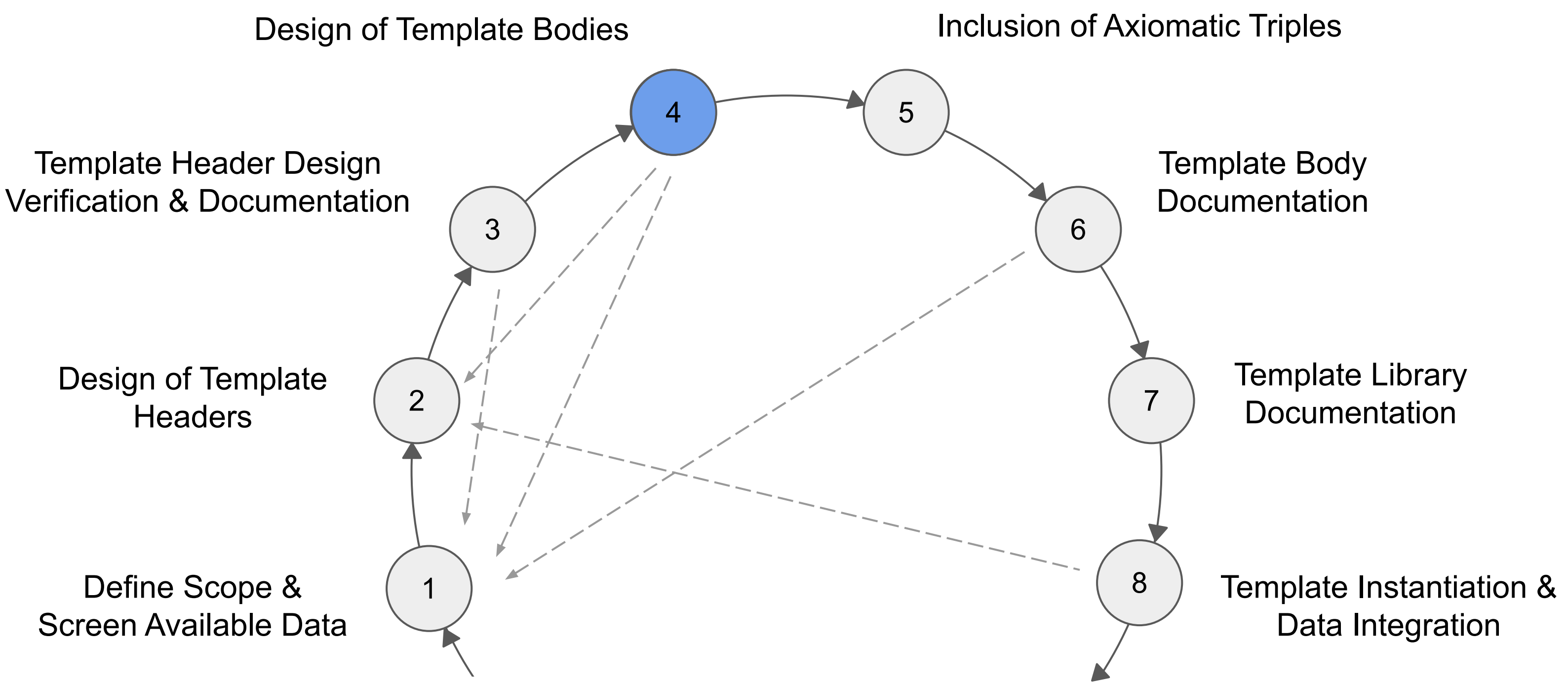
- derived template parameters directly from the available data
- one template per process step, i.e., one template per material type, one template per synthesis method, and one template per measurement method
- encountered trade-offs: splitting data across multiple templates concerns the correlations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



Moritz Blum, Basil Ell, Philipp Cimiano

Methodology

UNIVERSITÄT BIELEFELD
Faculty of Technology



Moritz Blum, Basil Ell, Philipp Cimiano

Step 4 – Design of Template Bodies

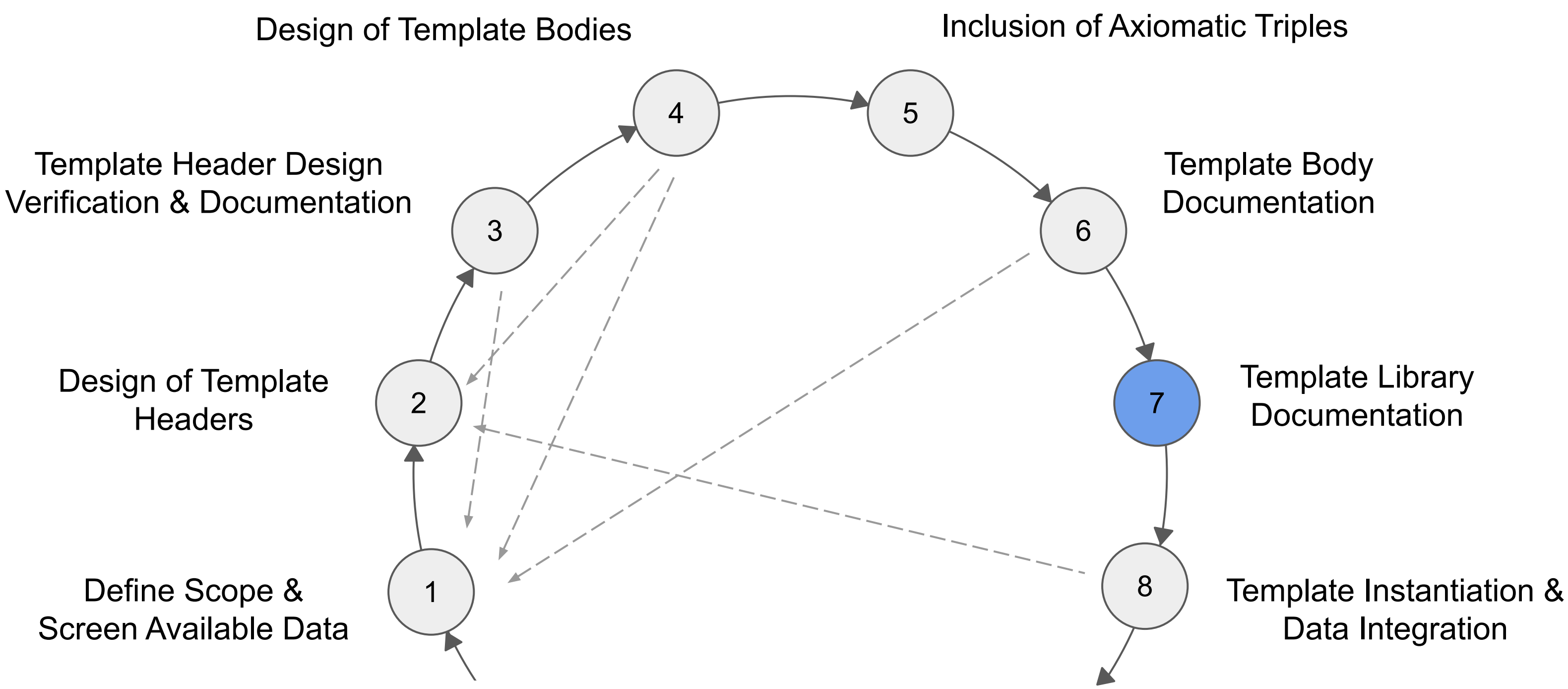
- straightforward, as a pre-structuring is already done
- use template header documentation
  - benefit from collected domain knowledge about, e.g., domain-specific terms
  - to identify sub-templates (🔄 iterative development – step back to template header design)
  - to identify and reuse existing ontologies
- 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

Moritz Blum, Basil Ell, Philipp Cimiano

Methodology



Moritz Blum, Basil Ell, Philipp Cimiano

Step 7 – Template Library Documentation

UNIVERSITÄT BIELEFELD
Faculty of Technology

Template: dpm:CreateComposition

URI: http://dipromag.de/ottr-templates/0.1/CreateComposition

Parameters

The parameters defined by the signature are listed here. Unfortunately parameter names are not informative; fixing it is a planned future feature. An optional parameter will accept the value ottr:none as an argument. A parameter which allows blanks will accept a blank node as argument value.

Index	Name	Type	Optional	Blanks allowed	Default value
1	material	ottr:IRI	no	yes	no
2	elements	NEList<ottr:IRI>	no	yes	no
3	stoichiometry_portion_unit_at_percent	NEList<xsd:float>	no	yes	no
4	user	ottr:IRI	no	yes	no
5	date	xsd:date	no	yes	no
6	comment	xsd:string	no	yes	""^^http://www.w3.org/2001/XMLSchema#string
7	verified	xsd:boolean	no	yes	"true"^^http://www.w3.org/2001/XMLSchema#boolean

Pattern

stOTTR

stOTTR serialisation of the template without annotation instances.

```
dpm:CreateComposition[
    ottr:IRI ?material,
    NEList<ottr:IRI> ?elements,
    NEList<xsd:float> ?stoichiometry_portion_unit_at_percent,
    ottr:IRI ?user,
    xsd:date ?date,
    xsd:string ?comment="",
    xsd:boolean ?verified="true"^^xsd:boolean
] :: {
    zipMin | dpm:CompositionElement(?material, ++?elements, ++?stoichiometry_portion_unit_at_percent),
    dpm:HasMetaData(?material, ?user, ?date, ?comment, ?verified),
    o-rdf:Type(?material, pmd:ChemicalComposition)
} .
```

The pattern of the template is illustrated by expanding a generated instance. Below the generated instance is shown in different serialisations, and its expansion is presented in different formats.

Generated instance

stOTTR

```
dpm:CreateComposition(x:argument1, (x:argument2-1, x:argument2-2), (_:argument3-1, _:argument3-2), x:argument4,
```

RDF/wOTTR

```
[ ottr:of      dpm:CreateComposition ;
  ottr:values  ( x:argument1 ( x:argument2-1 x:argument2-2 ) ( []
                                                              []
                                                            ) x:argument4
                 []
                 []
                 []
               )
] .
```

Visualisation of expanded RDF graph

Each resource node is linked to its IRI. Type relationships are not visualised, rather each node contains its type.

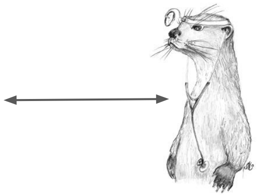
▼ Hierarchical horizontal layout (dot)

▶ Hierarchical vertical layout (dot)
▶ Spring model layout (neato)
▶ Spring model layout (fdp)
▶ Radial layout (twopi)
▶ Circular layout (circo)

Expanded RDF graph

```
x:argument1  rdf:type     pmd:ChemicalComposition ;
        dpm:comment       []  ;
        dpm:isVerified    []  ;
        dcterms:created   []  ;
        dcterms:creator   x:argument4 ;
        pmd:composedOf    [ rdf:type            dpm:Element , pmd:Object ;
                            dpm:hasStoichiometry  [ rdf:type   pmd:Measurement , pmd:Metadata ;
                                                    pmd:unit   dpm:at_percent ;
                                                    pmd:value  []
                                                  ] ;
                            pmd:relatesToElement  x:argument2-2
                          ] ;
```

Figure 3: OTTR documentation



- list of all templates
- template inclusion/call hierarchy
- individual template documentation

- instantiation order and naming guidelines, e.g., *“BulkSample”* + *\w{2} initials of the creator* + *YYMMDD* + *[0-9]{2} enumerator*

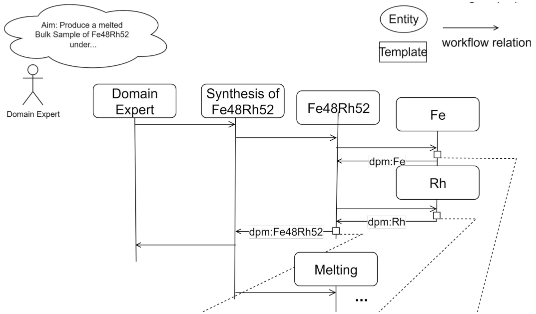
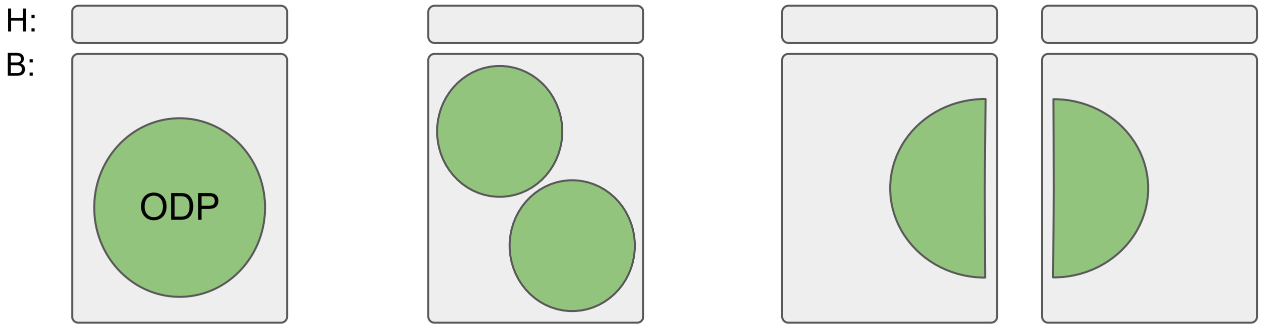


Figure 4: Template instantiation workflow

Moritz Blum, Basil Ell, Philipp Cimiano

ODPs are compatible with OTTR



a template can instantiate <u>one</u> ODP

a template can instantiate <u>multiple</u> ODP

multiple templates can instantiate one or more ODPs

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

UNIVERSITÄT BIELEFELD
Faculty of Technology

Conclusion

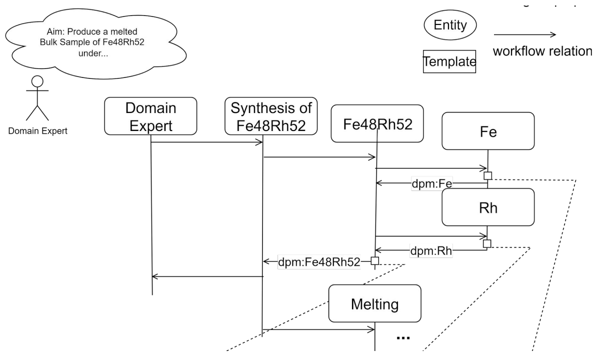
- ontology engineering methodology that relies on OTTR templates
  - key steps: defining the ontology's scope, …, instantiating templates and integrating data
  - our OTTR-based method should be combined with existing ontology engineering practices
  - compatible with ODPs
- experience from a project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

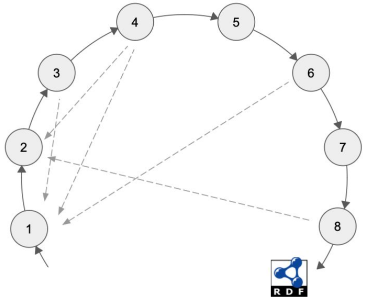
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Grant No. 13XP5120B (DiProMag)
SIRIUS
Project No. 237898

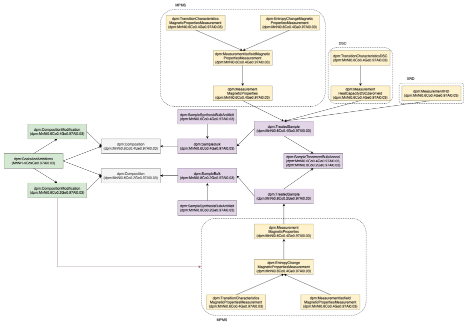
Moritz Blum, Basil Ell, Philipp Cimiano

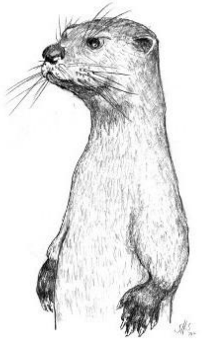
Thank you for your attention!

UNIVERSITÄT BIELEFELD
Faculty of Technology









GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Grant No. 13XP5120B (DiProMag)

SIRIUS
Project No. 237898

Moritz Blum, *Basil Ell, Philipp Cimiano*