

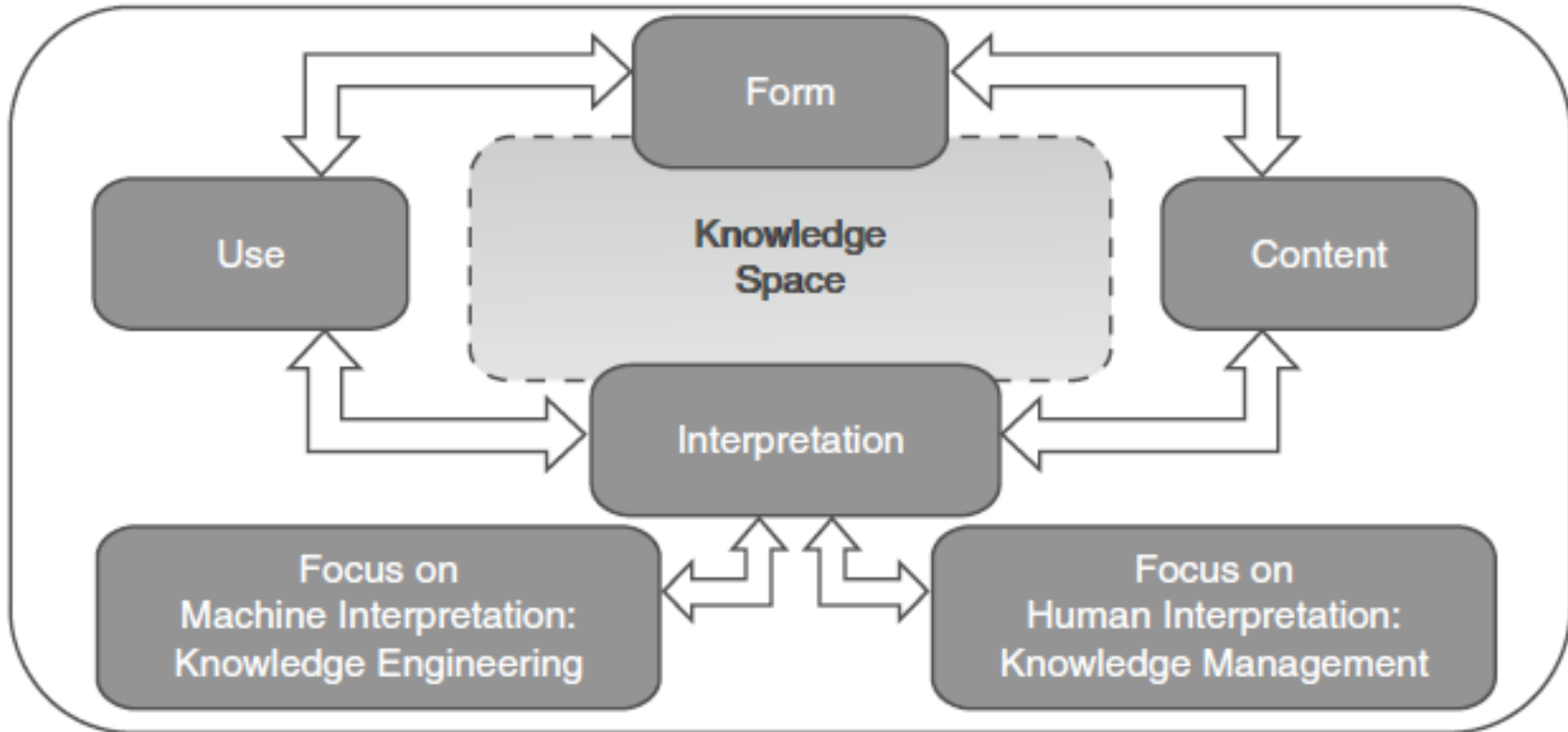


# *Ontology-based Metamodeling*

*Knut Hinkelmann*

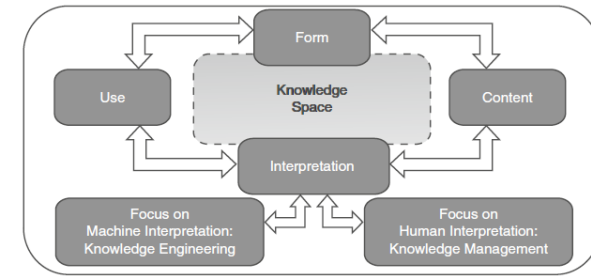


# Dimensions of a Knowledge Space



Karagiannis, D., & Woitsch, R. (2010). Knowledge Engineering in Business Process Management. In *Handbook on Business Process Management 2* (pp. 463–485). Springer.

# Dimensions of the Knowledge Space



## Use:

- process optimization requires knowledge about time and costs
- selection of a cloud service require knowledge about data and functionality

## Form: modeling language



## Content: Instantiation of concepts



- **Use:** Stakeholders and their concerns determine the relevant subset of the knowledge
- **Form:** Syntax and semantic of *meta model concepts*.
- **Content:** *Instantiation* of meta model concepts for a specific *application* (represented in the labels)
- **Interpretation:** Giving meaning to a model:
  - ◆ Graphical models are cognitively adequate for human
  - ◆ Machines need more formal representation

## *Content: Instantiation of Meta model + Application knowledge*

- Humans «know» the meaning of the modeling objects.
  - ◆ Meta model: Concepts of the model language
  - ◆ Application: Labels/names of the model elements

### ■ Examples:



- ◆ Meta model: Application Component
- ◆ Application: «ERP System» is business software



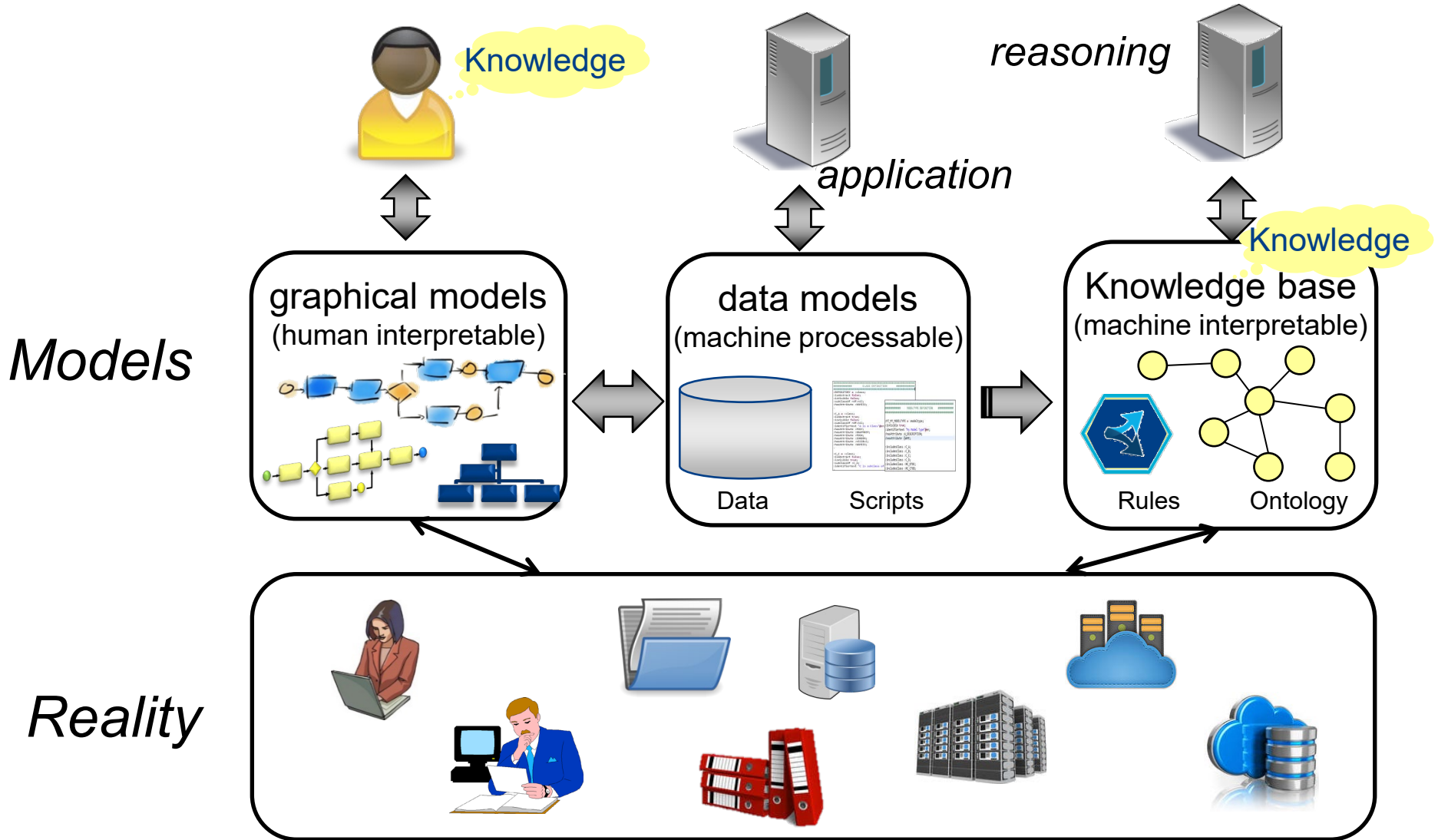
- ◆ Meta model: Task
- ◆ Application: «Cook pasta» is about preparing food

- The objective is to represent the knowledge so that it can be interpreted by a system for decision making and problem solving



# *Semantic Lifting*

# Semantic Lifting: Map Models into an Ontology



# *Semantic Lifting: Representing Content as Ontology*

## ■ **Meta model Knowledge:**

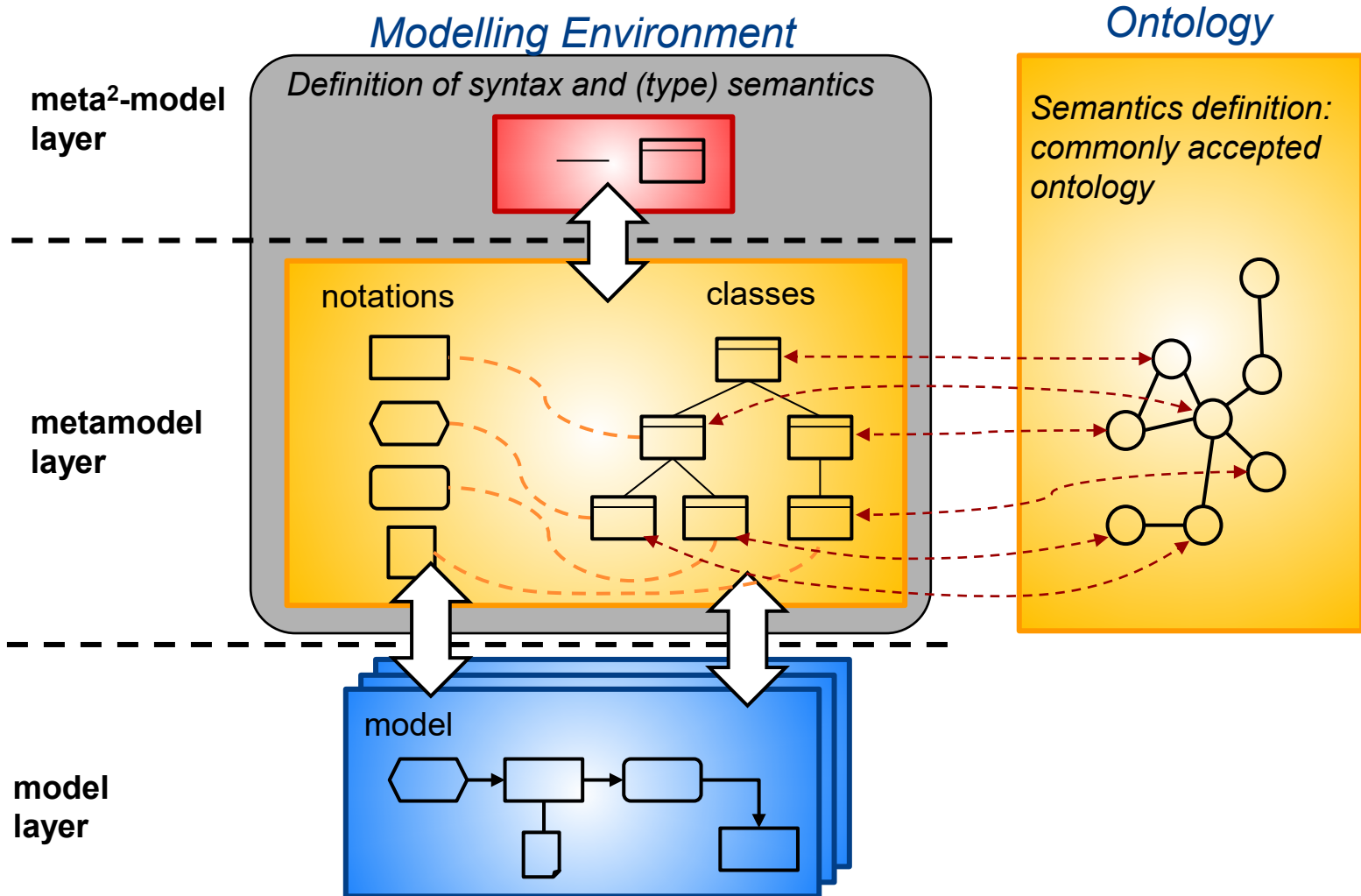
- ◆ Concepts of the meta model have corresponding class in an ontology
- ◆ For each element in a model an instance of the corresponding ontology class is created

## ■ **Knowledge about application domain:**

- ◆ Model elements are annotated with domain knowledge from application domain ontology

- **Ontology reasoning can be applied to the content knowledge in the models**

# Semantic Lifting: Map Models into an Ontology



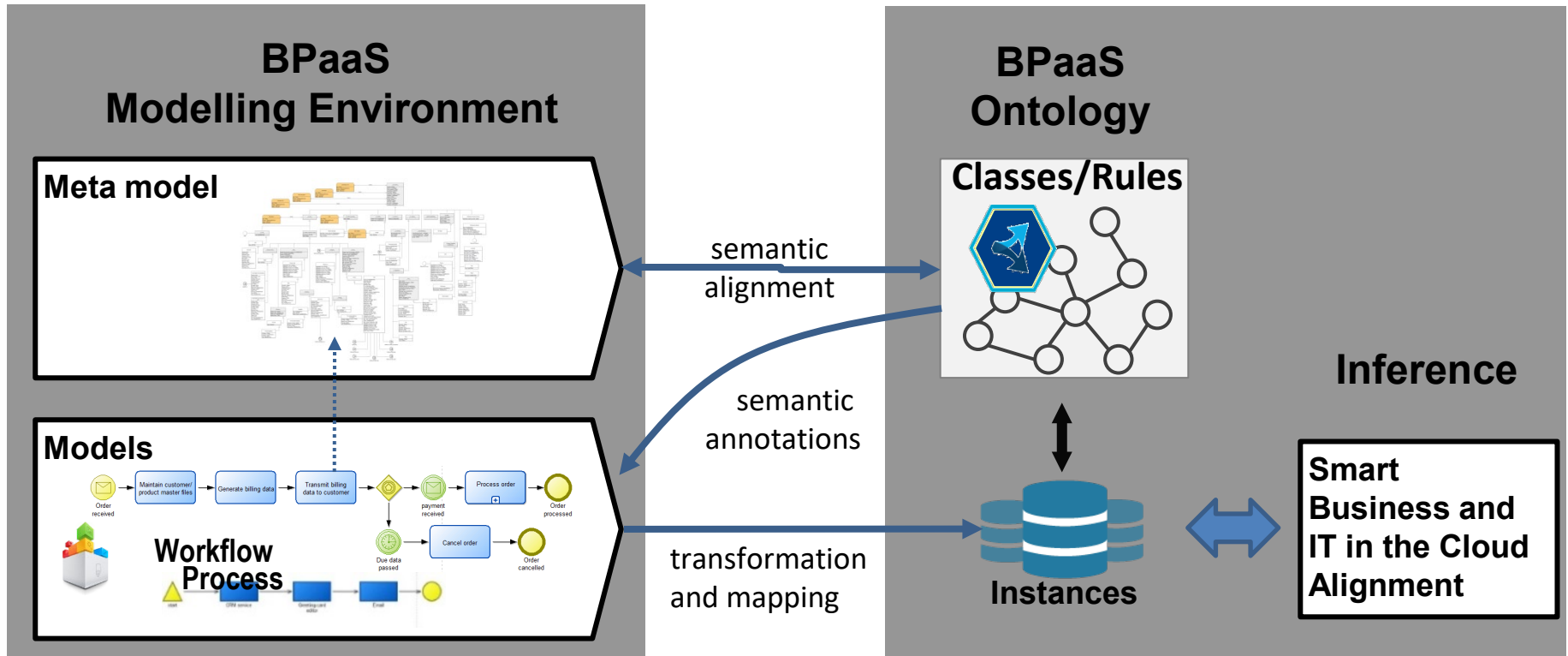
← - - - → **ontological metamodeling (lifting): explication of type semantics**



# Example: Business Process as a Service

**human interpretation**  
informal and semi-formal

**machine interpretation**  
formal



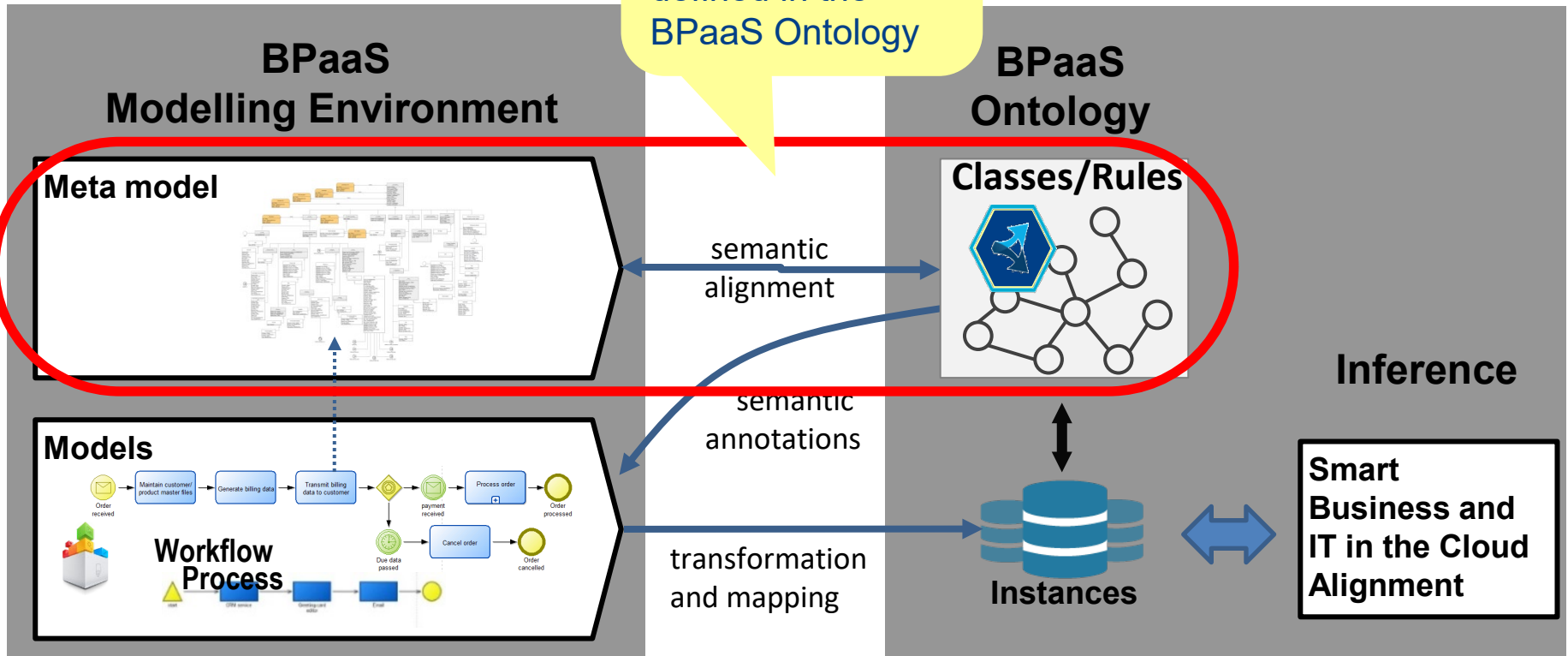
From: CoudSocket Project

# Example: Business Process as a Service

**human interpretation**  
informal and semi-formal

The semantics of the meta-model elements is defined in the BPaaS Ontology

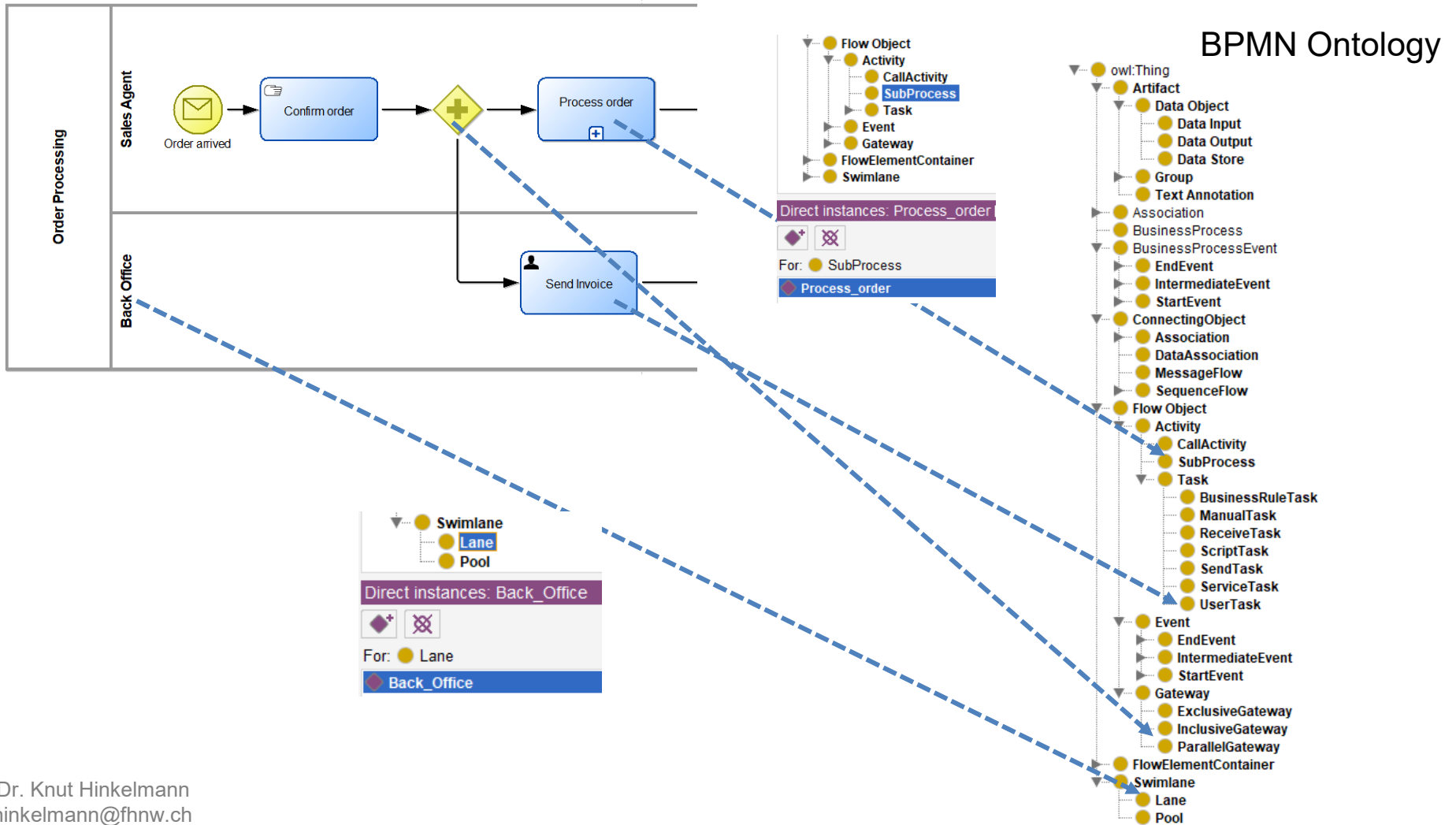
**machine interpretation**  
formal



From: CoudSocket Project

# Transformation and Mapping

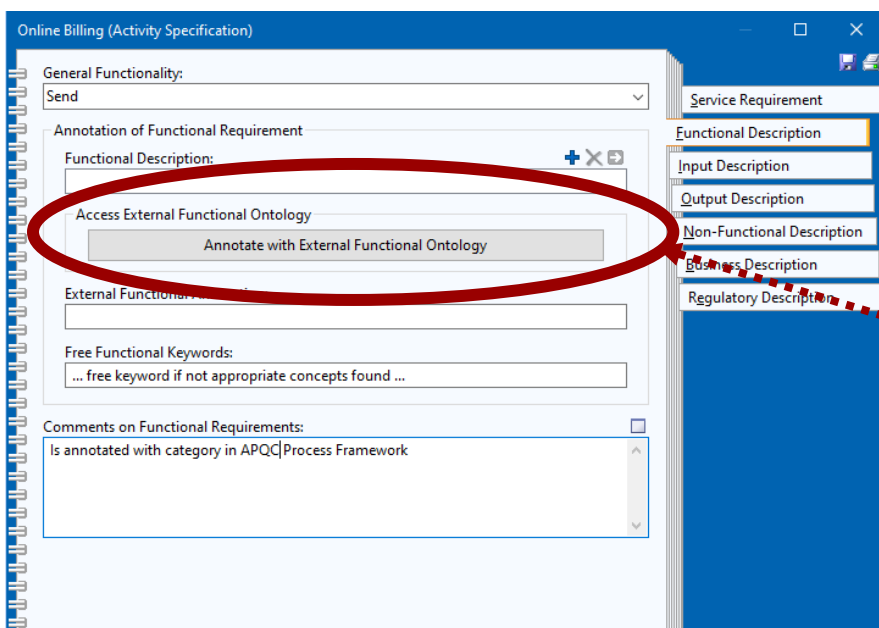
The model elements are exported as instances ontology classes



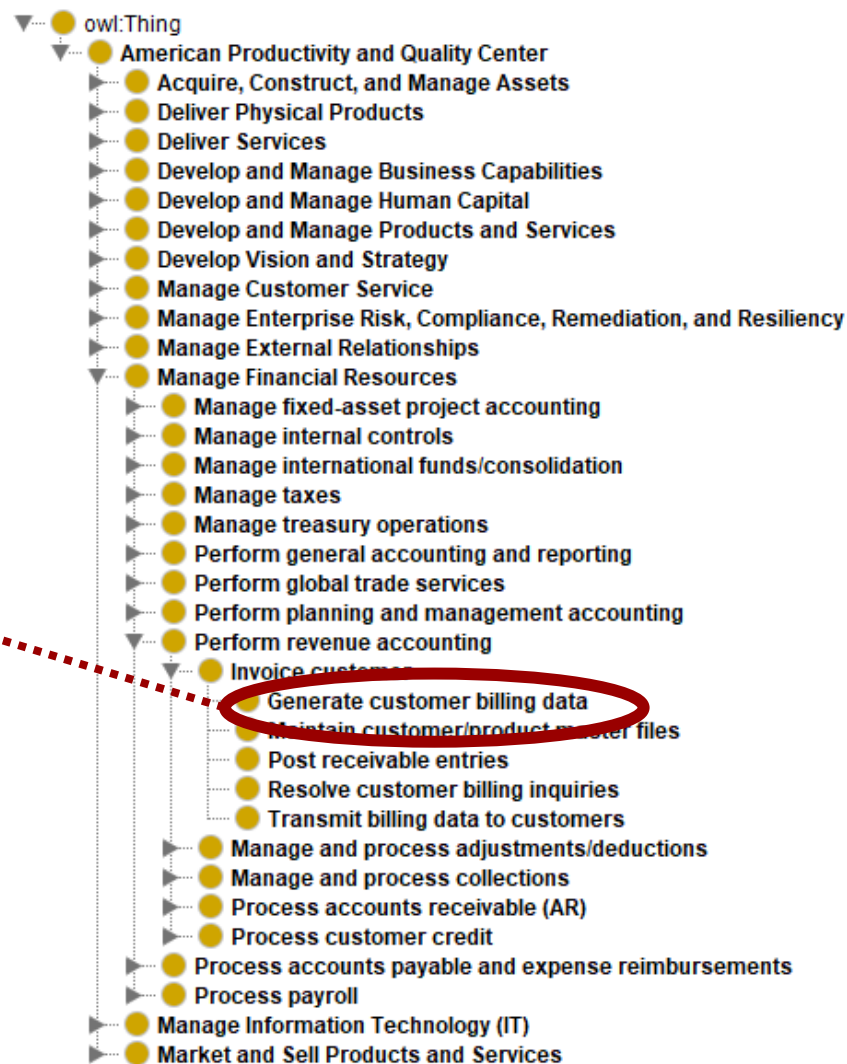
# Semantic Annotations

Annotate modeling elements with classes from the domain ontology

## Example: Functionality of a Service



Domain Ontology:  
APQC Process Classification Framework



# Inferencing: Cloud Service Selection

## Cloud Service Selection

### Functionality

SemanticAnnotationQuestio X +

Functional

- APQC category that reflect the functional requirement:  
type to search \*
- Action that reflect the functional requirement:  
type to search \*
- Object that reflect the functional requirement:  
type to search \*

### Non-functional requirements

Payment

- Select your preferred payment plan:
  - Prepaid Annual Plan
  - Try Free First
  - Customizable Plan
  - Monthly Fee
  - None

Performance

- Monthly Availability in %:  
Insert your value here \*

## *Drawbacks of Semantic Lifting*

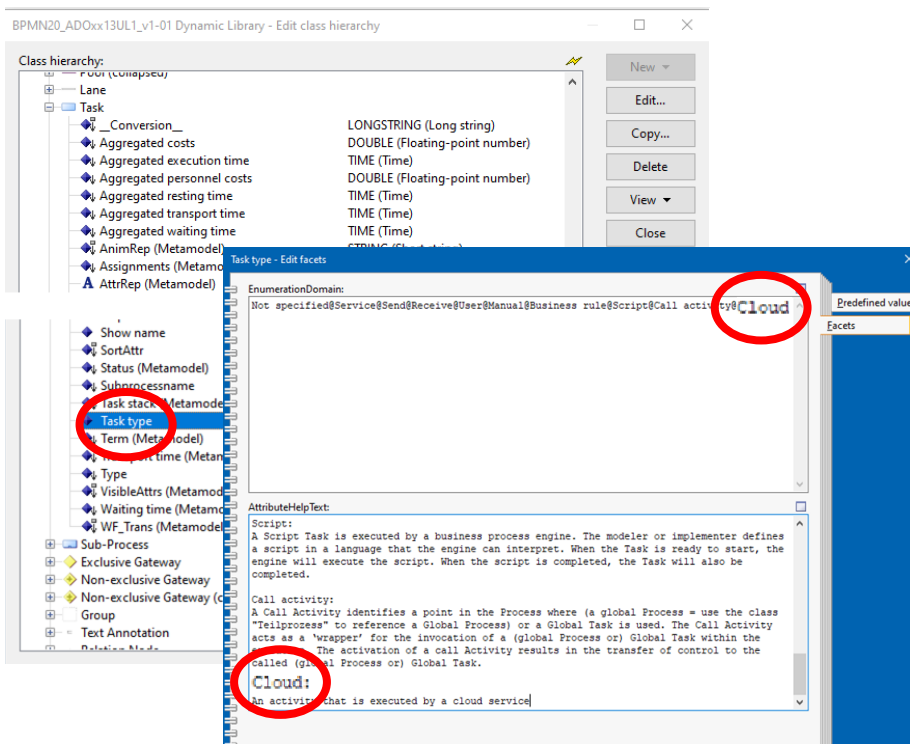
- Separate Environments for
  - ◆ Modelling
  - ◆ Knowledge Base (Inferencing)
- Inconsistency: Both metamodel and ontology must be aligned but are maintained independently:
  - ◆ Metamodel and ontology must represent the same semantics
  - ◆ Each change in metamodel must be reproduced in the ontology and vice versa
- Effort: After each change the models must be translated again into the ontology instances

# Example: New Model Element

## ■ New task type: Cloud Task



Change in the meta model:



Class hierarchy: ... Task type (Metamodel) ...

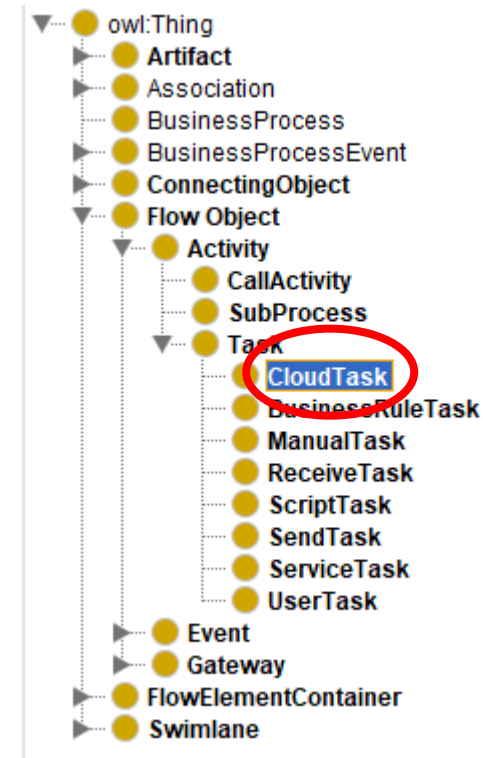
Task type - Edit facets

EnumerationDomain: ... Cloud

Predefined value: Cloud

AttributeHelpText: ... Cloud: An activity that is executed by a cloud service

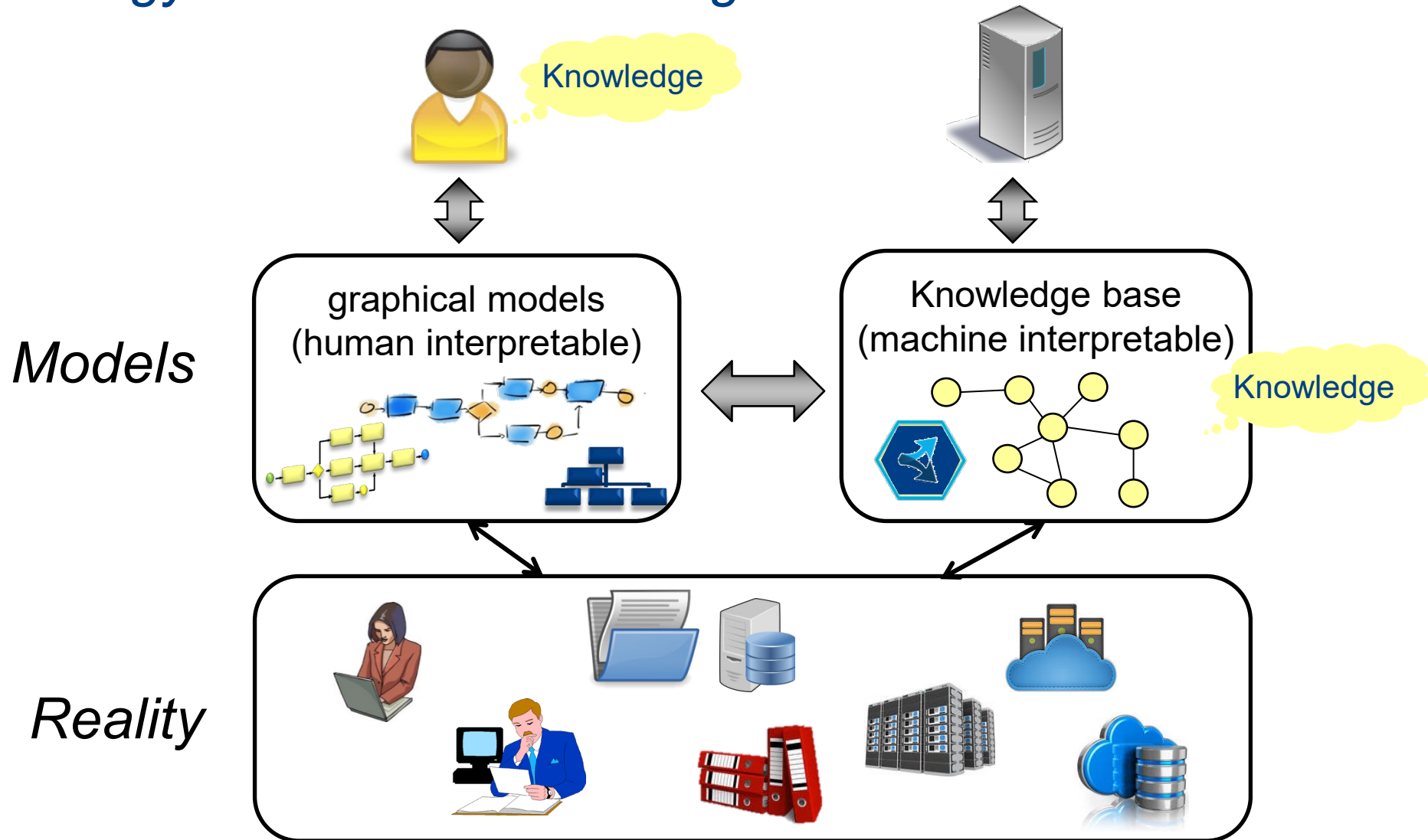
Change in the ontology:



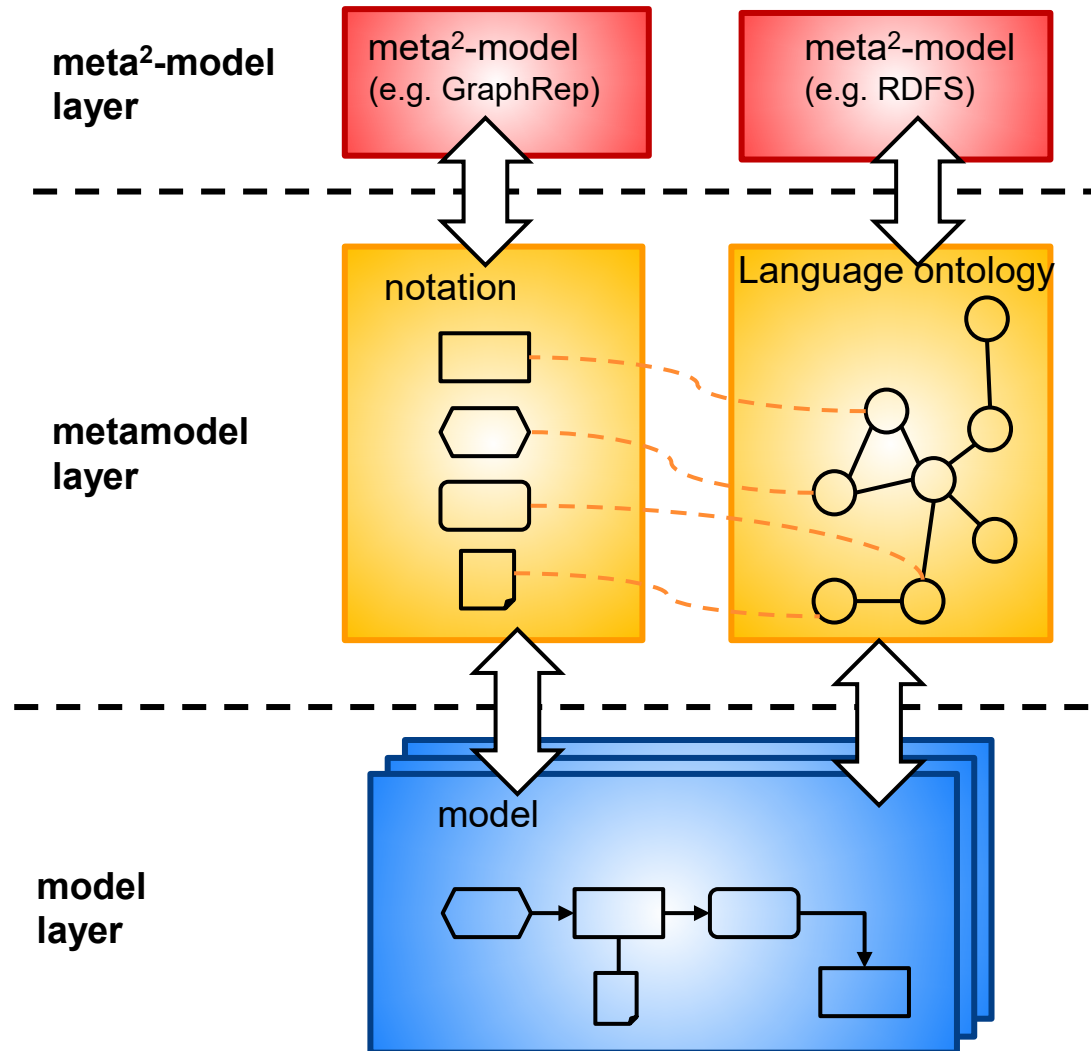
# ***Ontology-based Metamodelling***



# Ontology-based Metamodeling

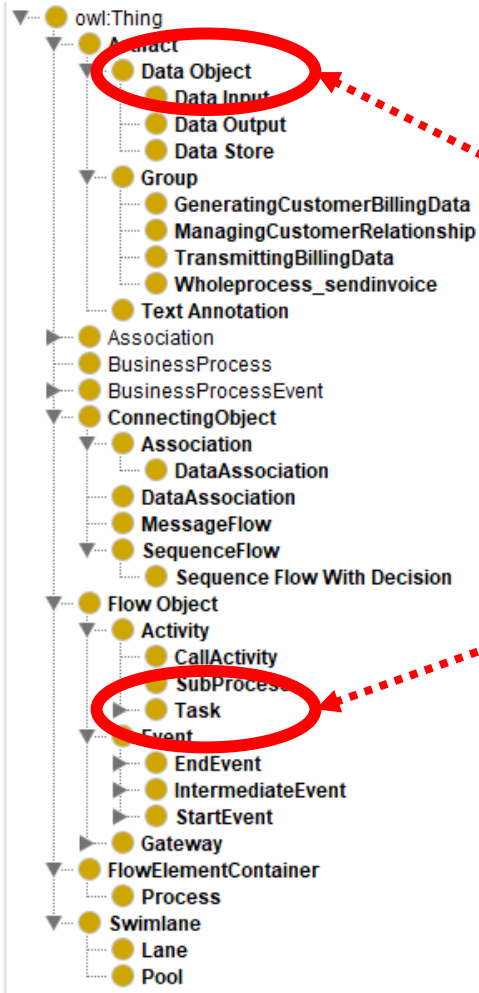


# Ontology-based Metamodeling (1): Metamodel is represented as an Ontology

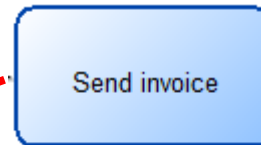


# Modelling Language Ontologies

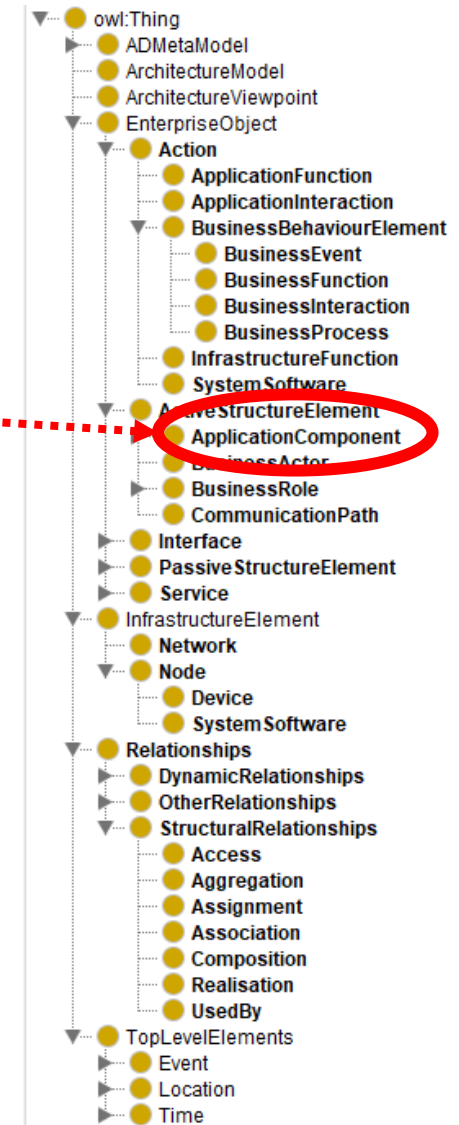
## BPMN



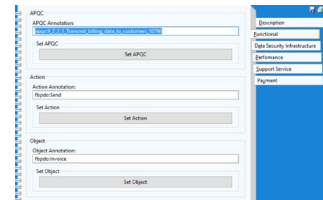
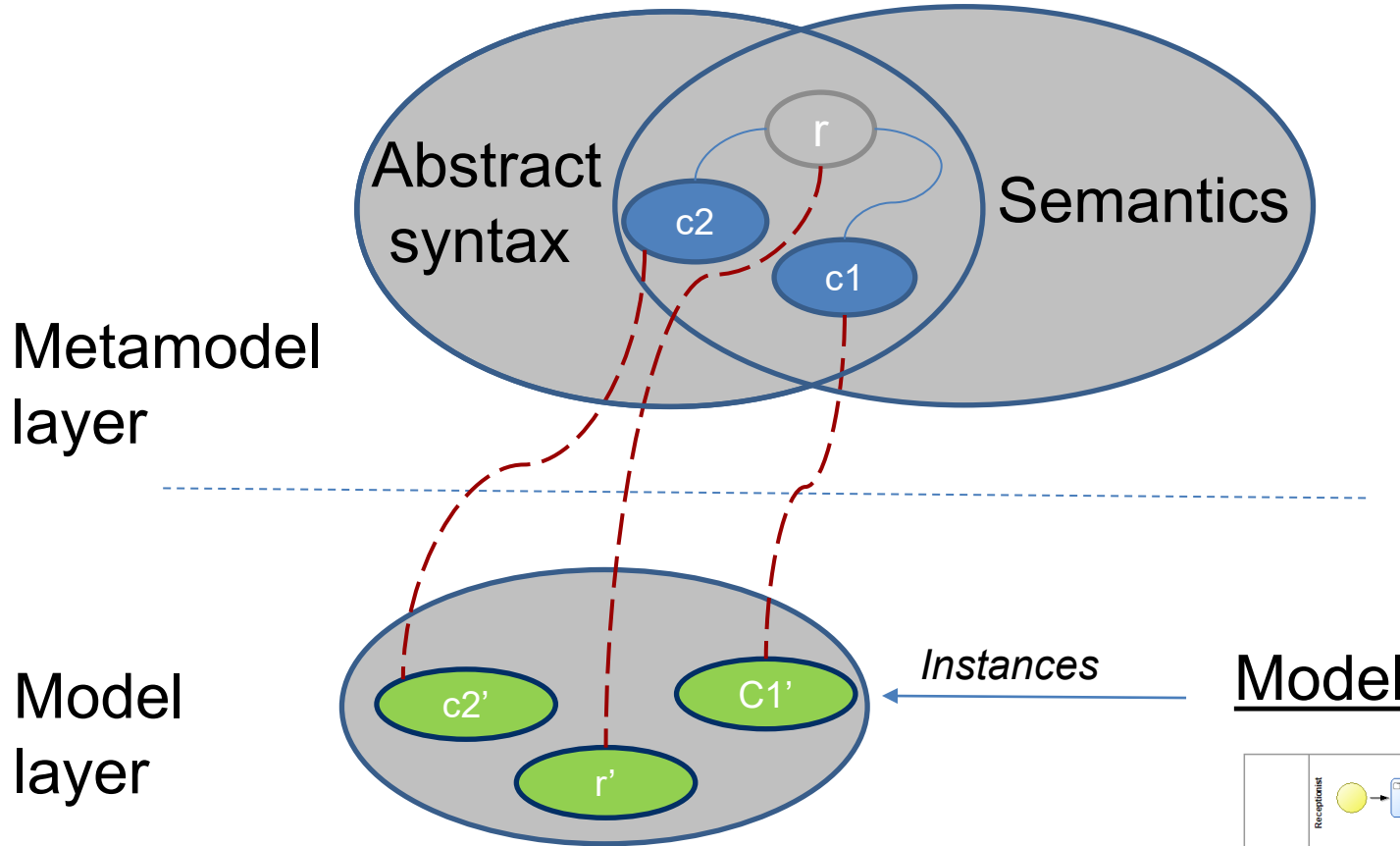
Invoice



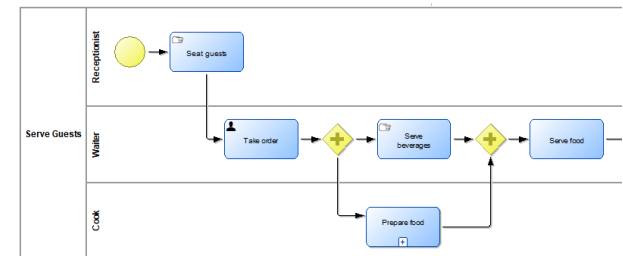
## Archimate



# Ontology

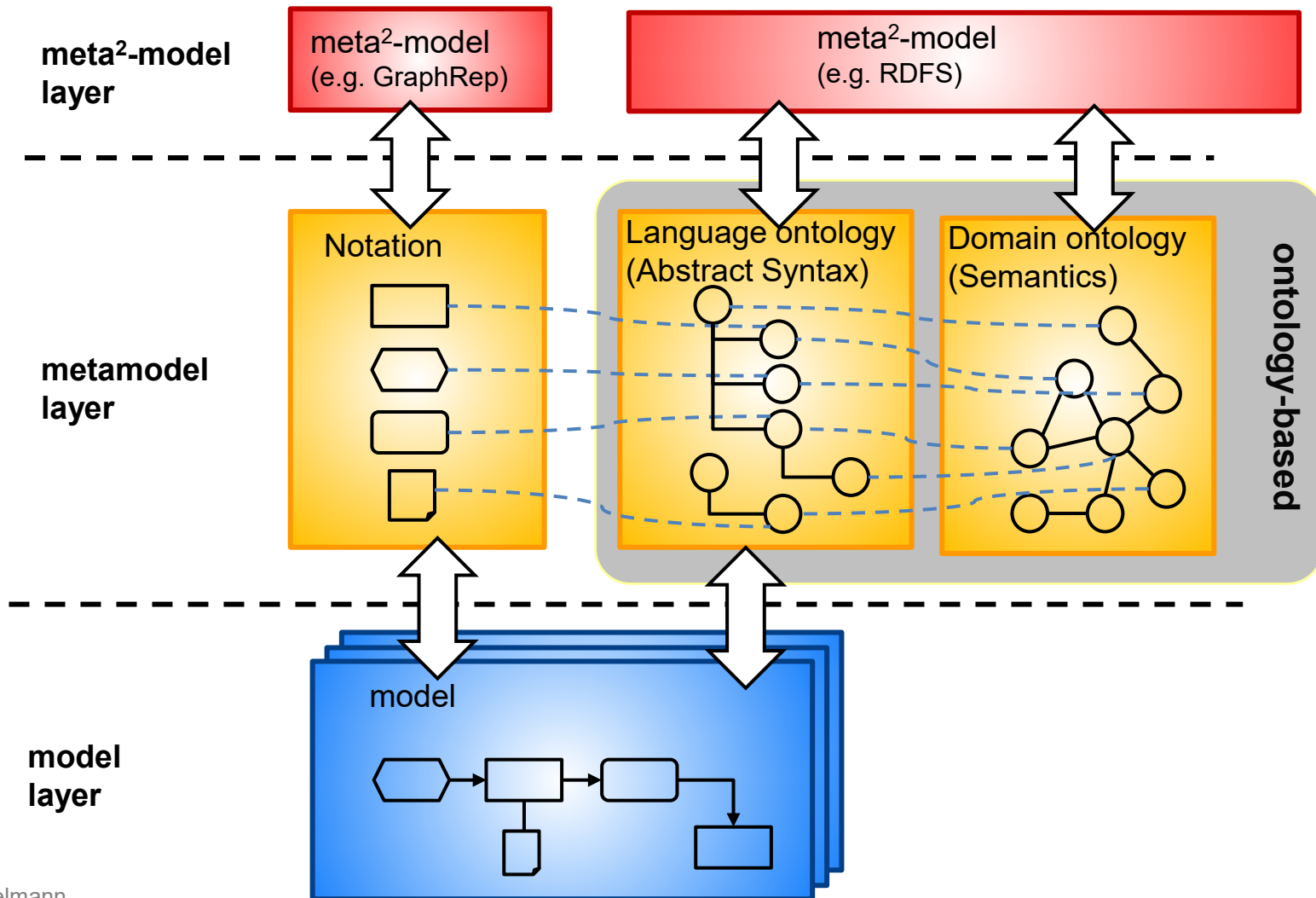


# Model



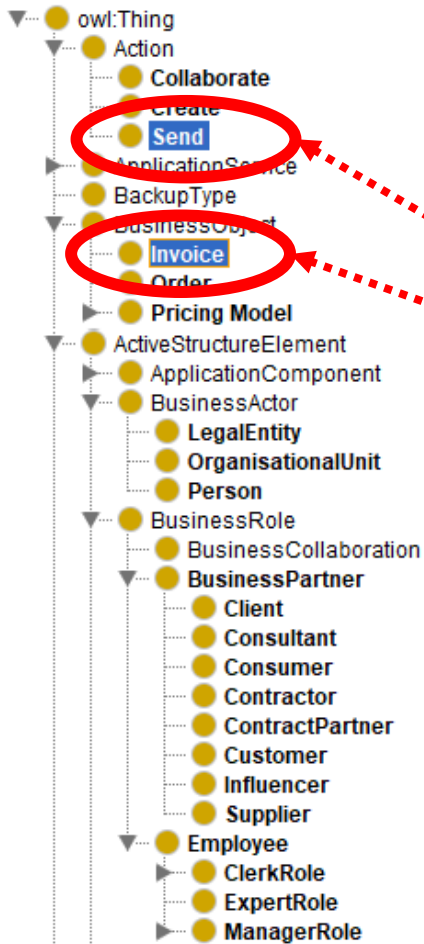
Thanks to Emanuele Laurenzi

# Ontology-based Metamodeling (2): Ontologies for Metamodel and Content



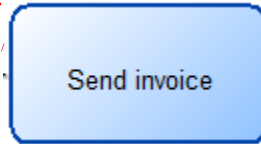
# Domain Ontologies

## Enterprise Ontology (excerpt)



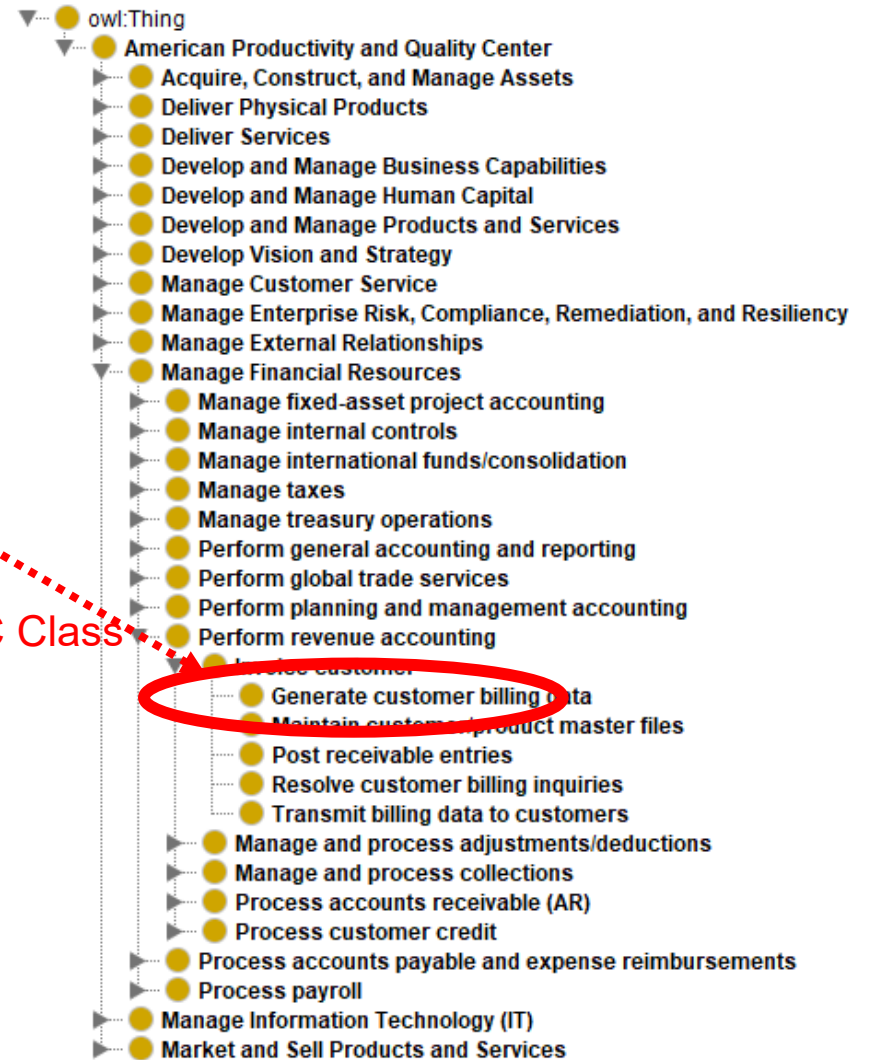
Action type

Object



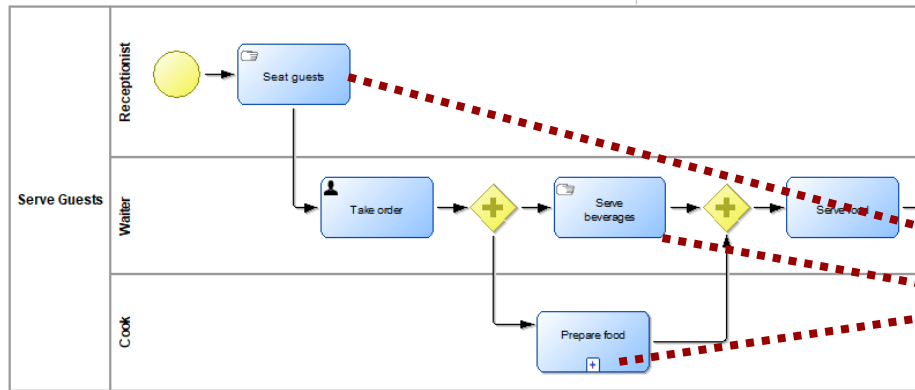
APQC Class

## Domain Ontology: APQC Process Classification Framework



# Ontology-Based Modeling

- Single environment for modelling and ontology
- Model elements are directly created as instances in the ontology



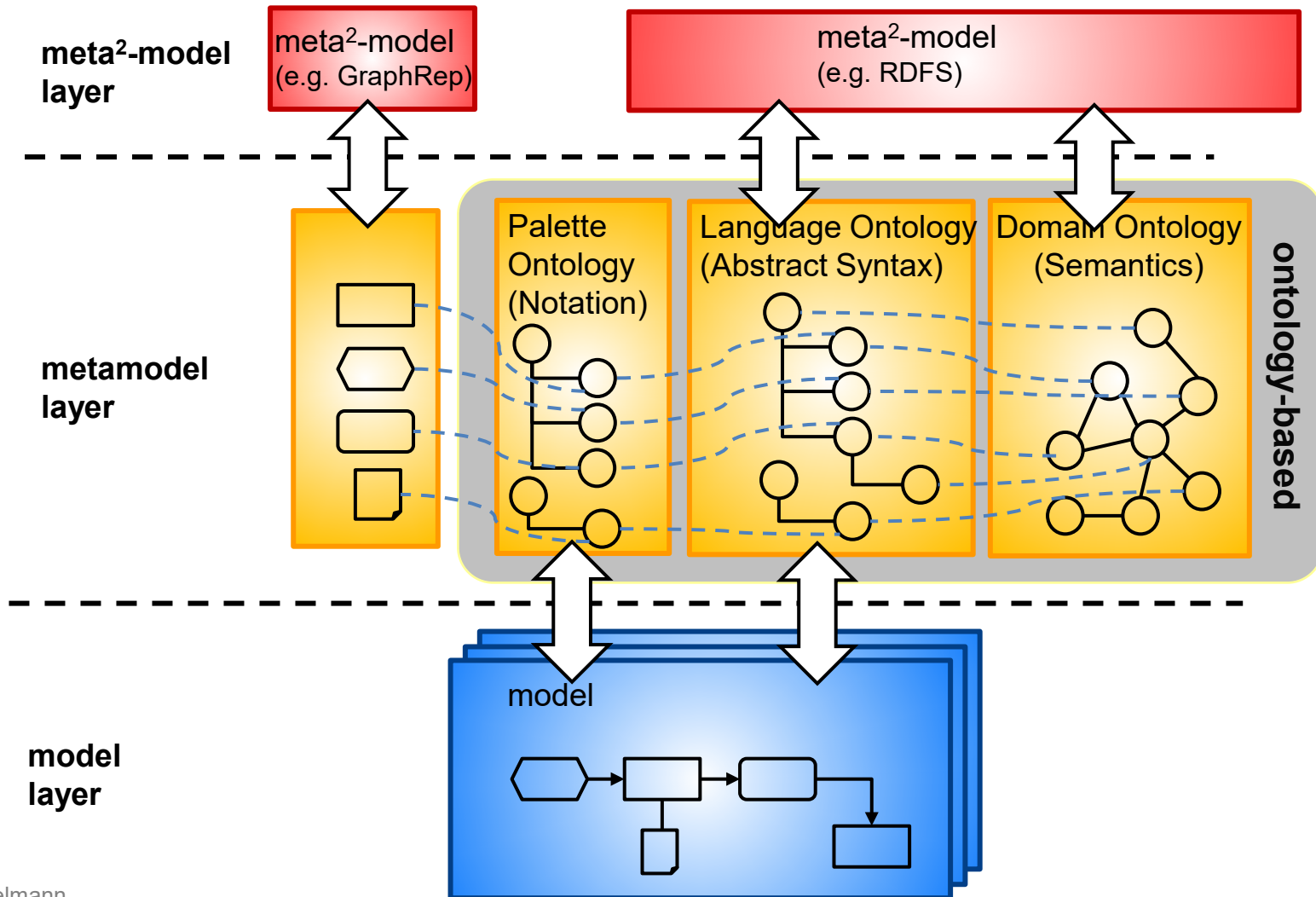
Class hierarchy: ManualTask Asserted

- owl:Thing
  - Artifact
    - Association
    - BusinessProcess
      - BusinessProcessEvent
      - ConnectingObject
      - Flow Object
        - Activity
          - CallActivity
          - SubProcess
            - Task
              - BusinessRuleTask
              - ManualTask**
              - ReceiveTask
              - ScriptTask
              - SendTask
              - ServiceTask
              - UserTask
  - Event
  - Gateway
  - FlowElementContainer
  - Swimlane
    - Lane
    - Pool

Individuals

- ◆ Cook
- ◆ Prepare\_Food
- ◆ Receptionist
- ◆ Seat\_guests
- ◆ Serve\_Beverages
- ◆ Serve\_food
- ◆ Take\_order
- ◆ Waiter

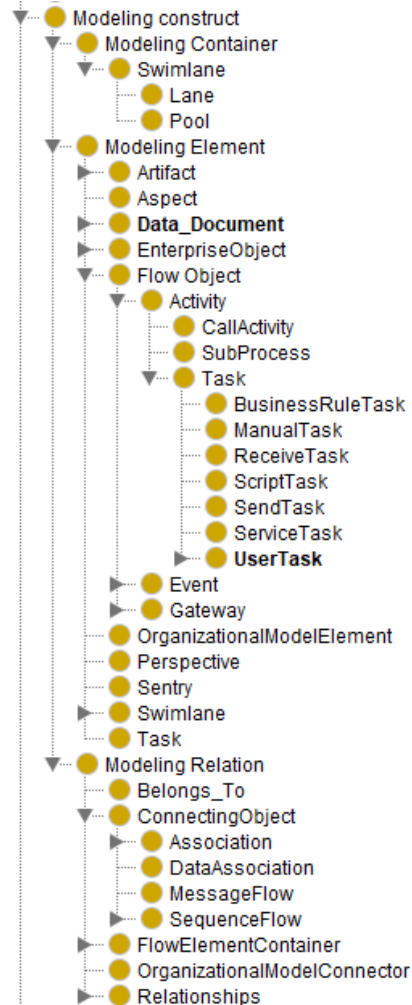
# Ontology-based Metamodeling (3): Ontologies for Language, Metamodel and Content





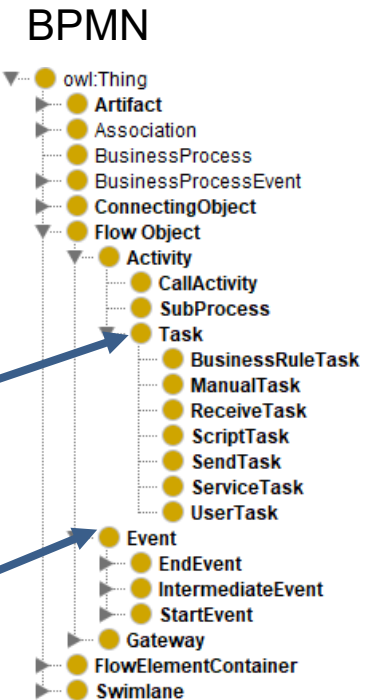
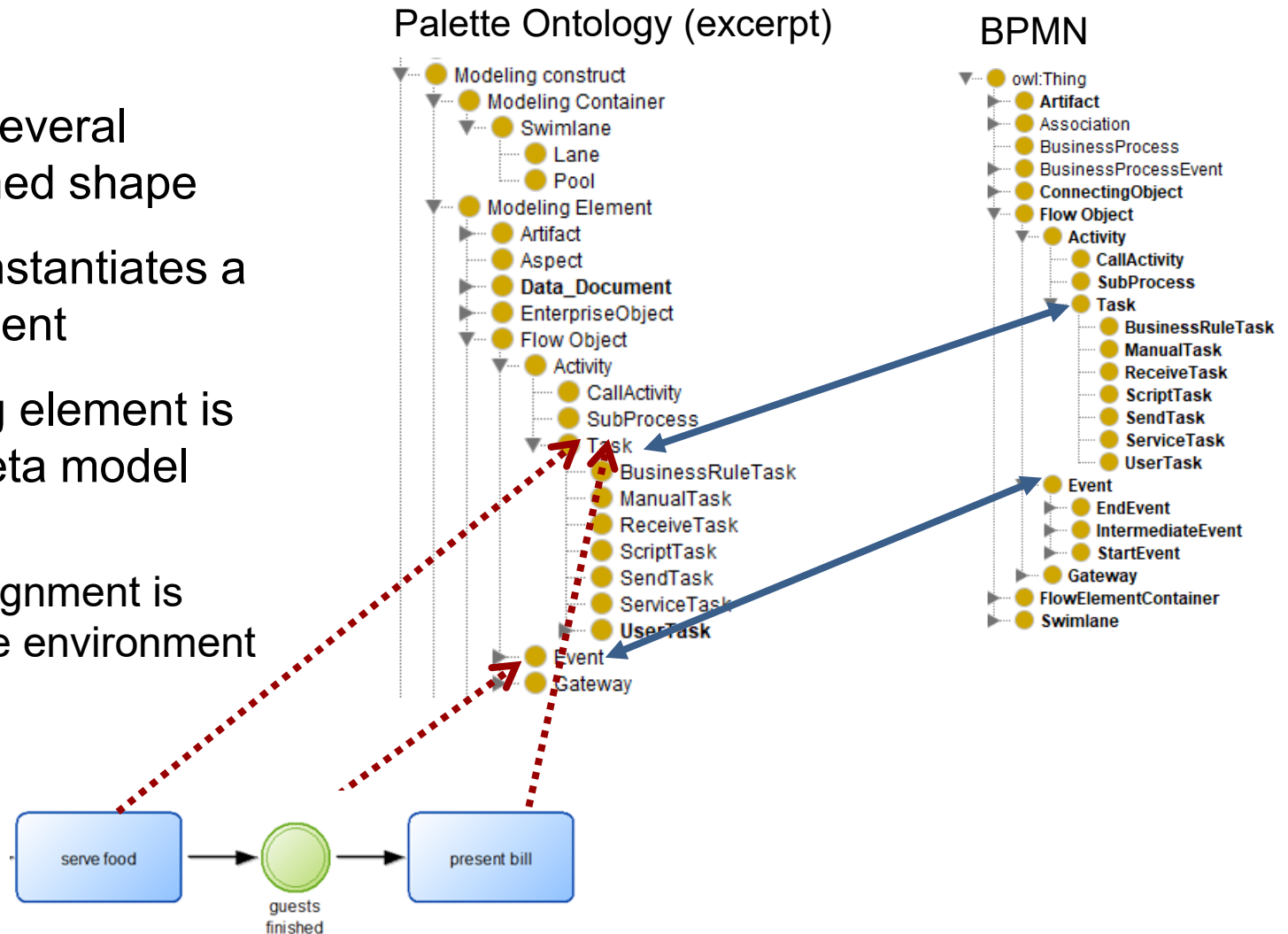
# Palette Ontology

## Palette Ontology (excerpt)



# Representing Models in AOAME

- Models have several elements, named shape
- Each shape instantiates a modeling element
- Each modeling element is related to a meta model construct
  - ◆ Semantic alignment is built-in to the environment



## Example Query

«Which task elements are in the model Serve Guests»?

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mod: <http://fhnw.ch/modelingEnvironment/ModelOntology#>
PREFIX lo: <http://fhnw.ch/modelingEnvironment/LanguageOntology#>
PREFIX po: <http://fhnw.ch/modelingEnvironment/PaletteOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bpmn: <http://ikm-group.ch/archiMEO/BPMN#>
```

```
SELECT ?model ?shape ?l
WHERE {
?model rdfs:label "Serve Guests".
?model mod:modelHasShape ?shape.
?shape mod:shapeInstantiatesPaletteConstruct ?x.
?x po:paletteConstructIsRelatedToModelingLanguageConstruct bpmn:Task.
?shape rdfs:label ?l
}
```

Select the elements  
(named shapes) in  
the model

For the shapes find the  
model elements  
(= Palette Constructs)

Filter the palette constructs for  
BPMN Tasks and show the labels

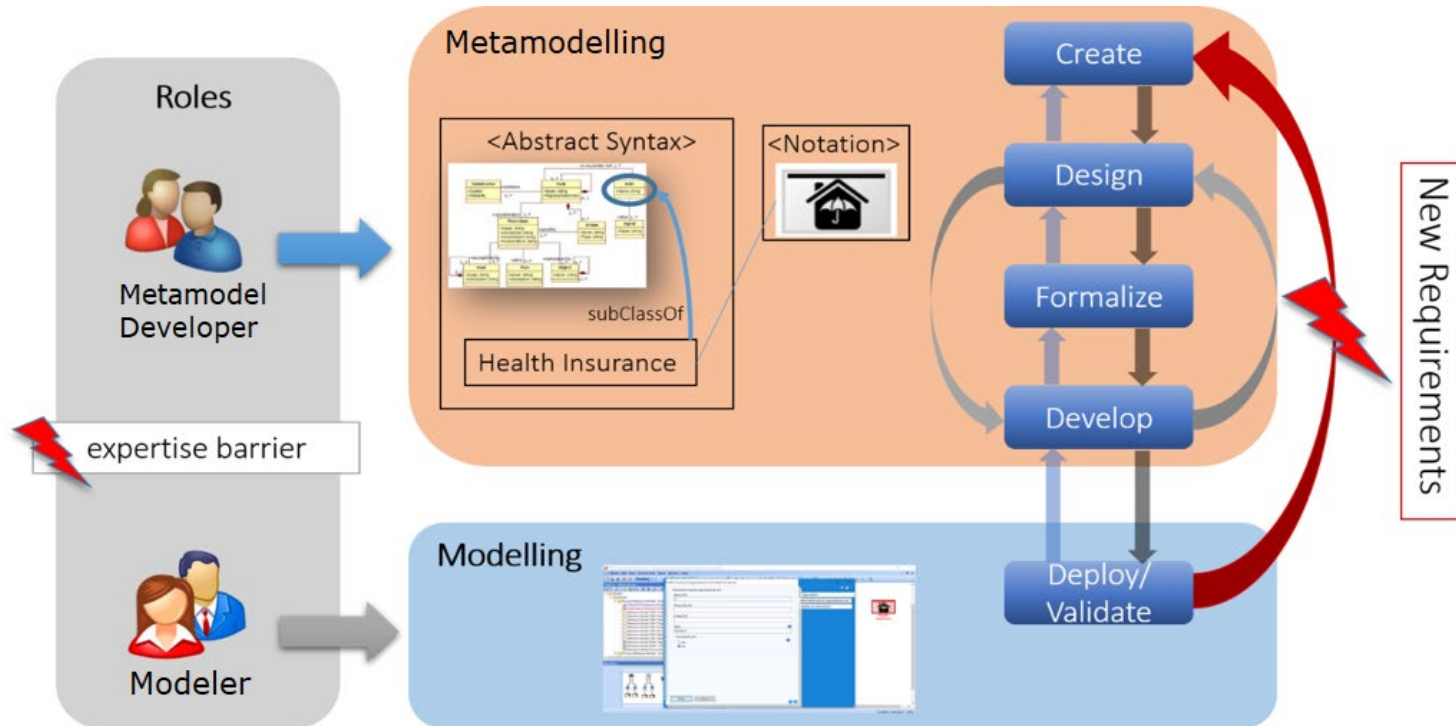


# *Agile Modelling*

# *Objective*

Adapt modeling languages and ensure a precise shared interpretation of new modeling constructs to both **humans and machines**

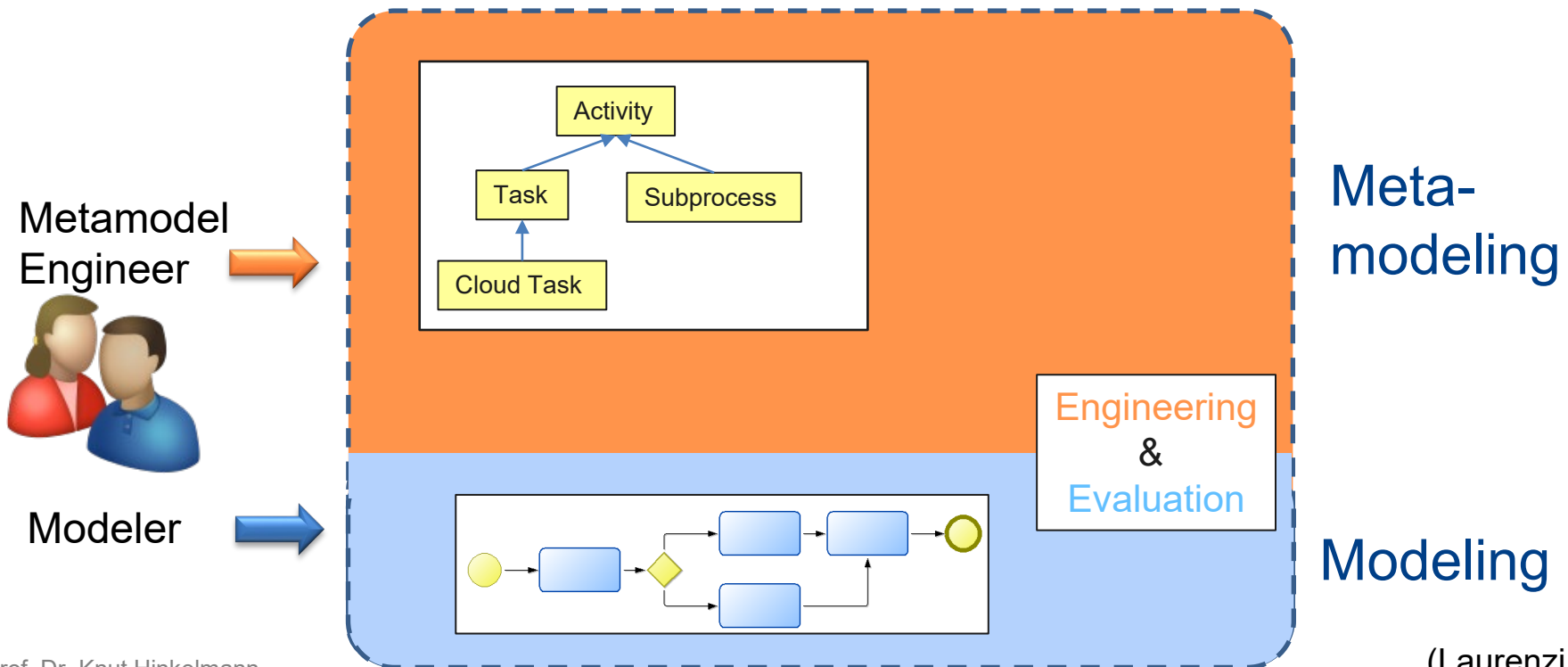
# Challenge: Separation of metamodelling and modelling



- Challenge 1: Metamodeling is a joint effort between metamodel experts and domain experts
- Challenge 2: Sequentialization of metamodeling and modeling is time consuming

# Integration Modeling and Metamodeling in a Single Environment

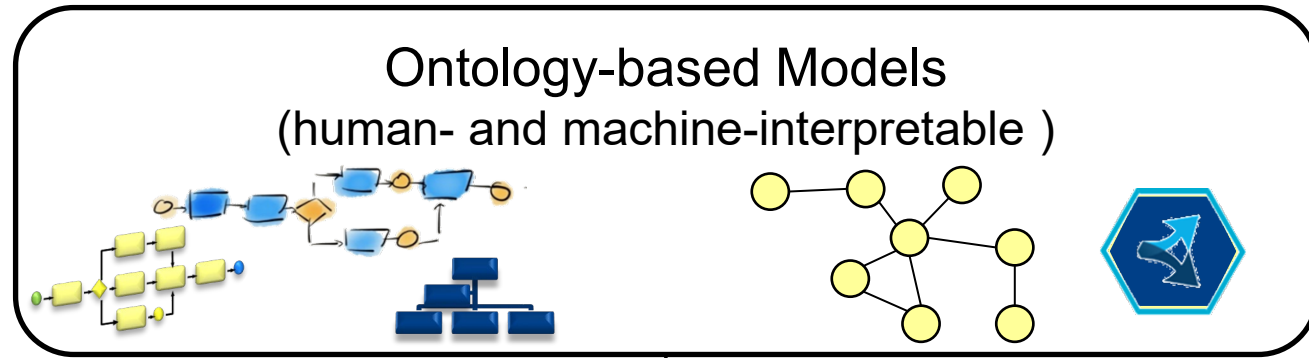
- Tight collaboration between metamodel developer and modeler
- Modeler can also take the role of metamodel developer



# Agile and Ontology-Aided Modeling Environment (AOAME)



*Models + Knowledge*



*Reality*





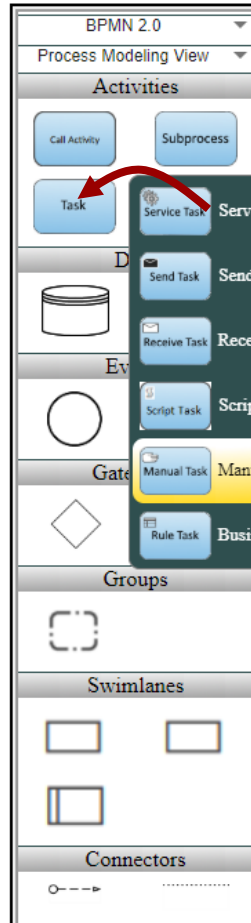
# AOAME: *Agile and Ontology-Aided Modeling Environment*

- AOAME is a a prototypical implementation for Agile and Ontology-Aided Modeling
- It is based on the PhD Thesis of Emanuele Laurenzi
- Implementation of the current version by
  - ◆ Emanuele Laurenzi
  - ◆ Charuta Pande
  - ◆ Devid Montecchiari
  - ◆ Egemen Kaba

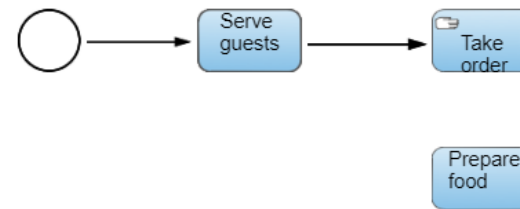
# Ontology-Based Modeling in AOAME

Palette

Model Editor

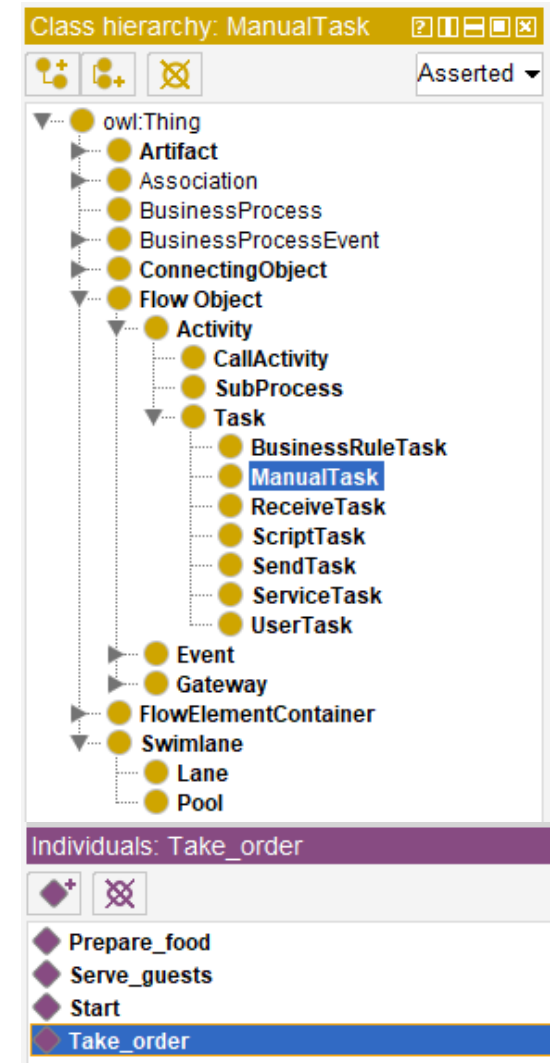
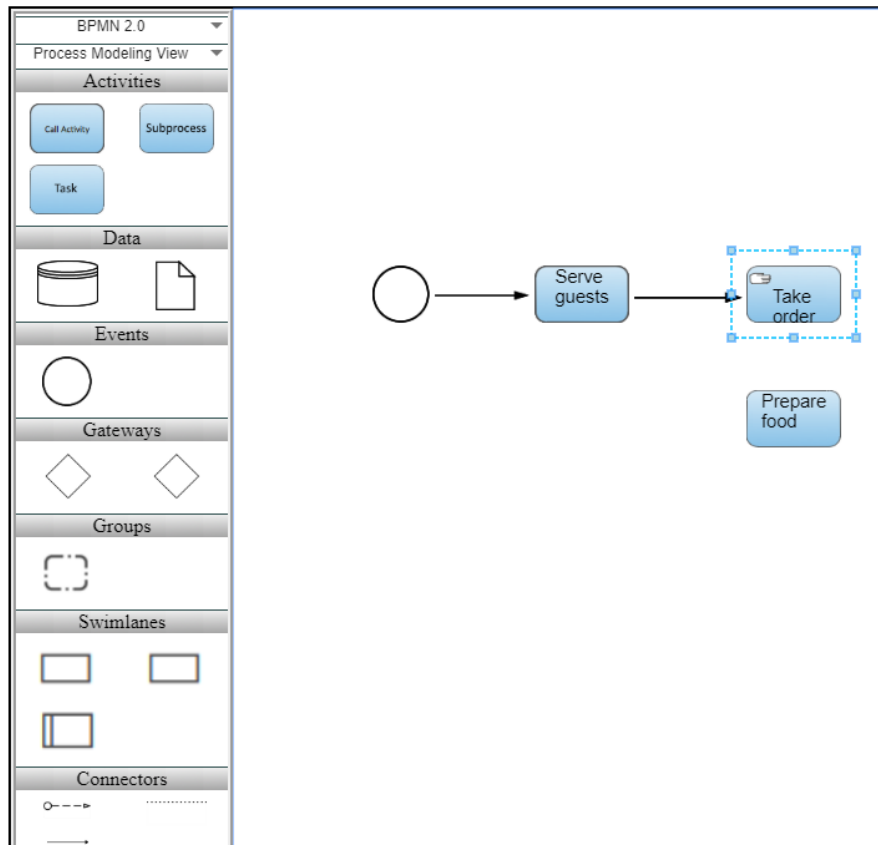


*rdfs:subClassOf*

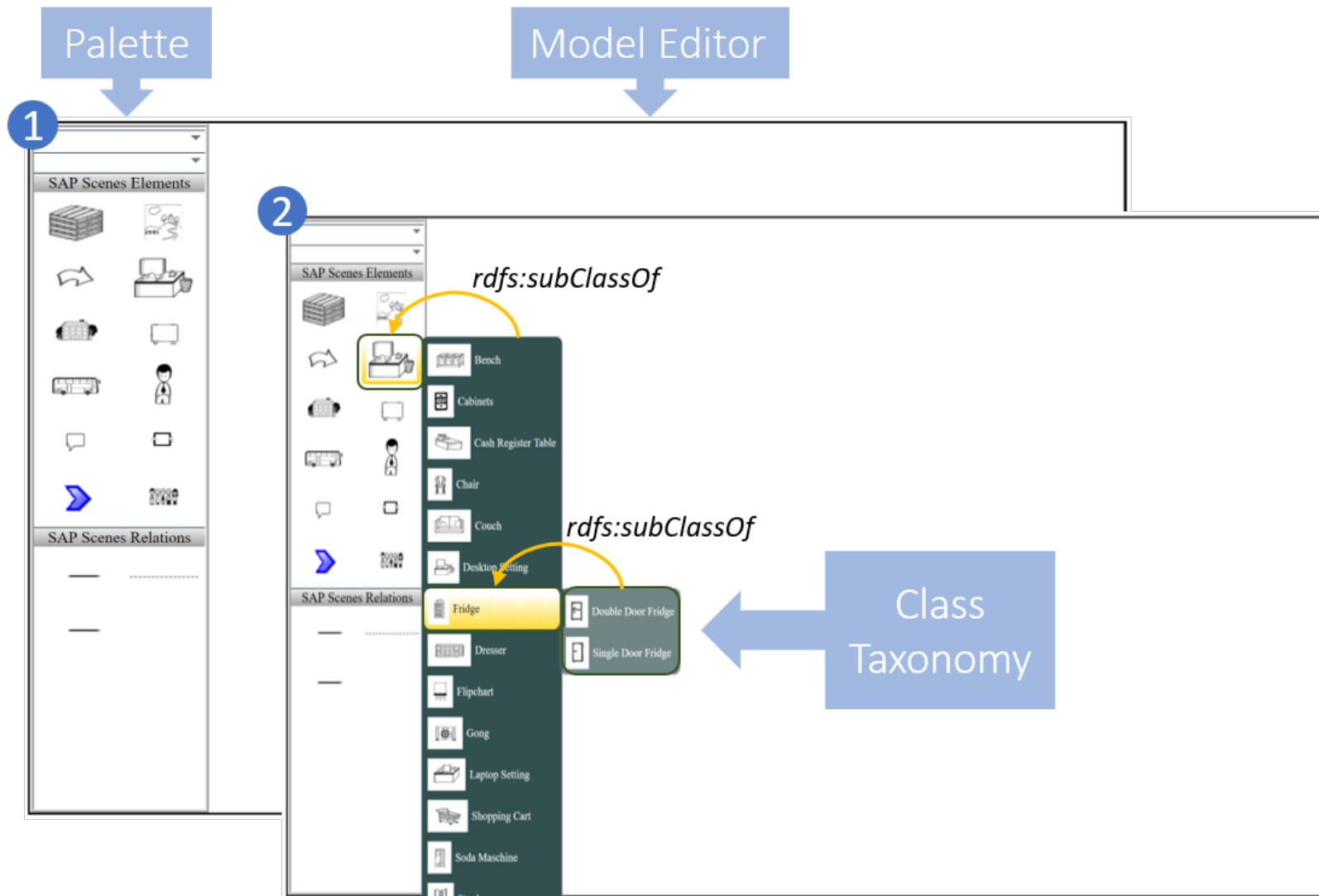


# Ontology-Based Modelling

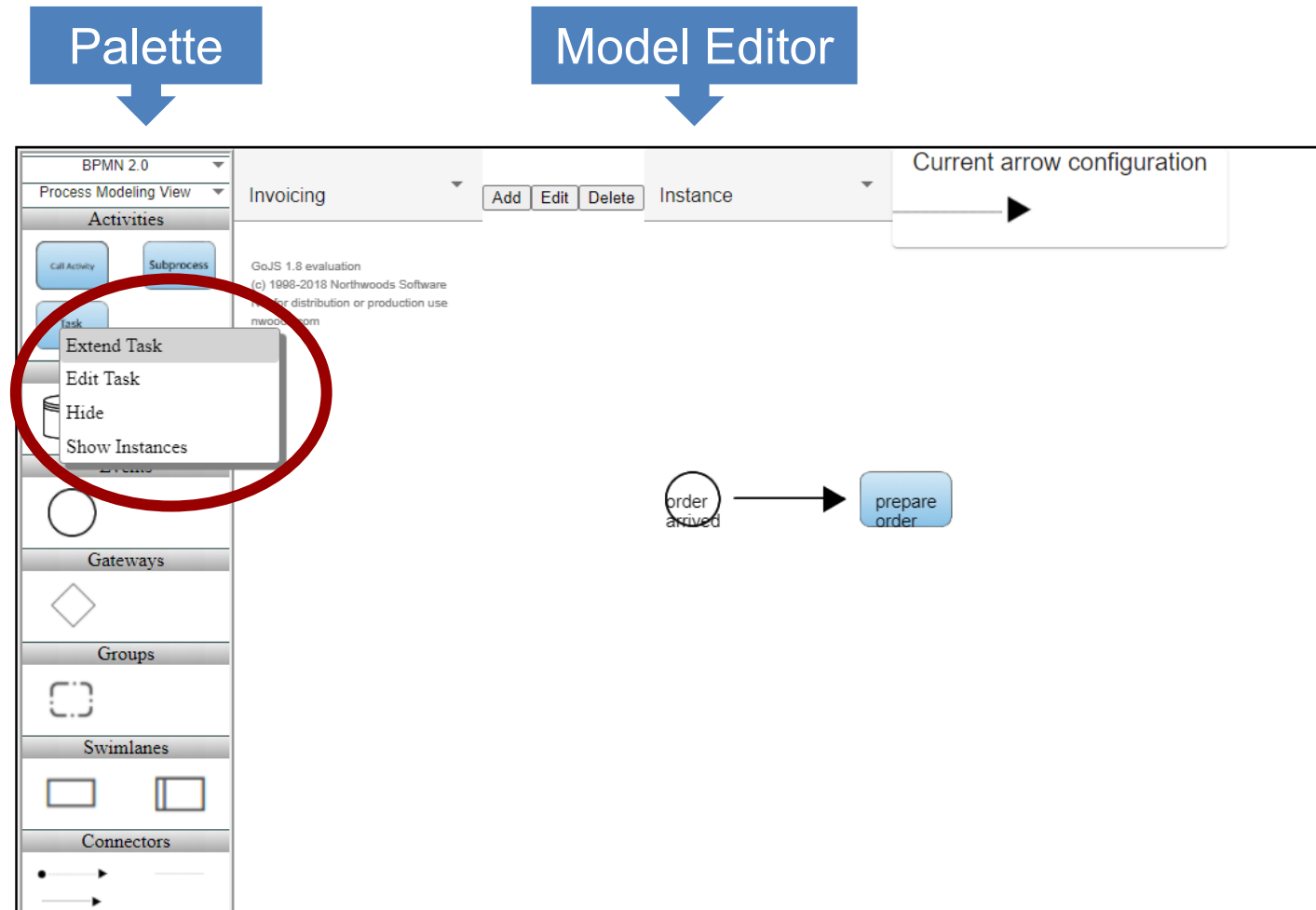
Model elements are directly created as instances in the ontology  
Modelling and ontology in a single environment



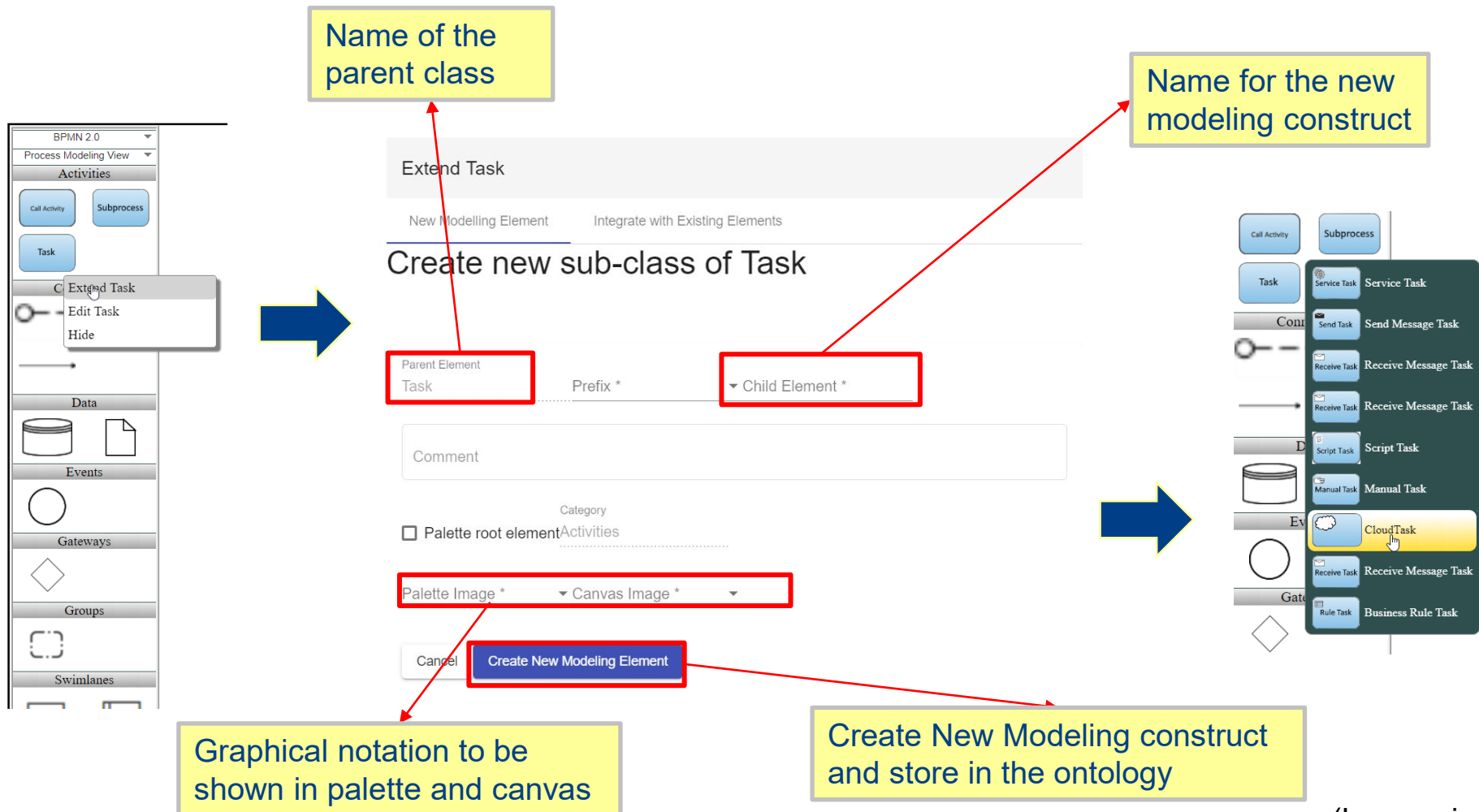
# Modeling Elements are represented in a Class Hierarchy



# Extending AOAME Modeling Languages – on the fly



# Integration of Meta-modeling and Modeling: On-the-fly Modeling Language Adaptation



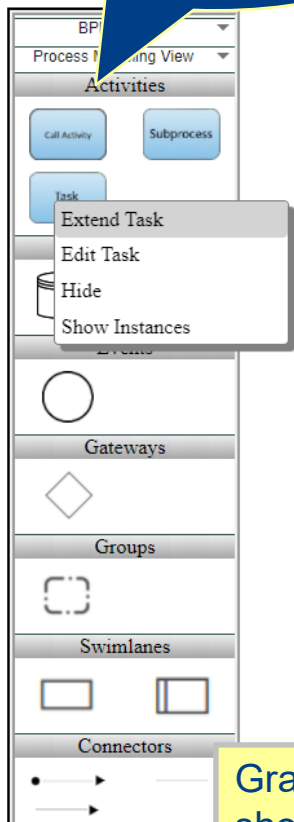
# Integration of Meta-modeling and Modeling: On-the-fly Modeling Language Adaptation

Ontology-based palette

Name of the parent class

Name for the new modeling construct

Ontology-based metamodel



Extend Task

New Modelling Element Integrate with Existing Elements

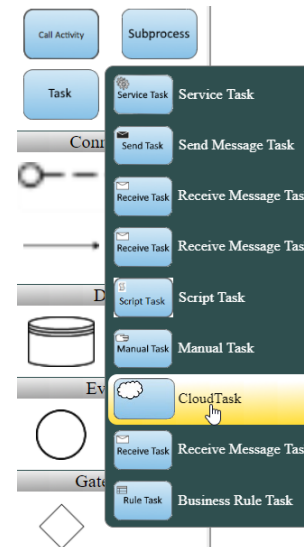
### Create new sub-class of Task

Parent Element  Prefix \*  Child Element \*

Comment

Category  Palette root element

Palette Image \*  Canvas Image \*



Graphical notation to be shown in palette and canvas

Create New Modeling construct and store in the ontology

# ***Hands-on Agile and Ontology-Aided Modeling in AOAME***