



Ontology-based Modeling

Knut Hinkelmann

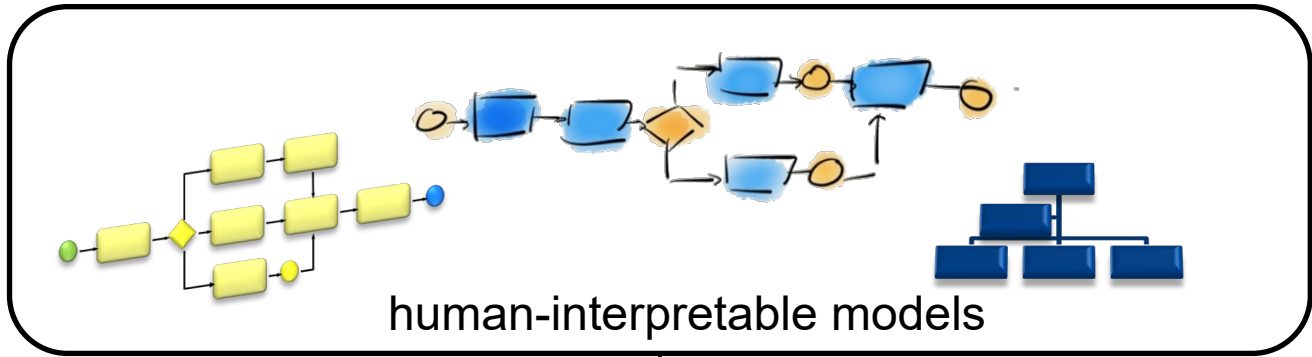


Human Problem Solving

*Communication/
Analysis/
Decision Making*



Models

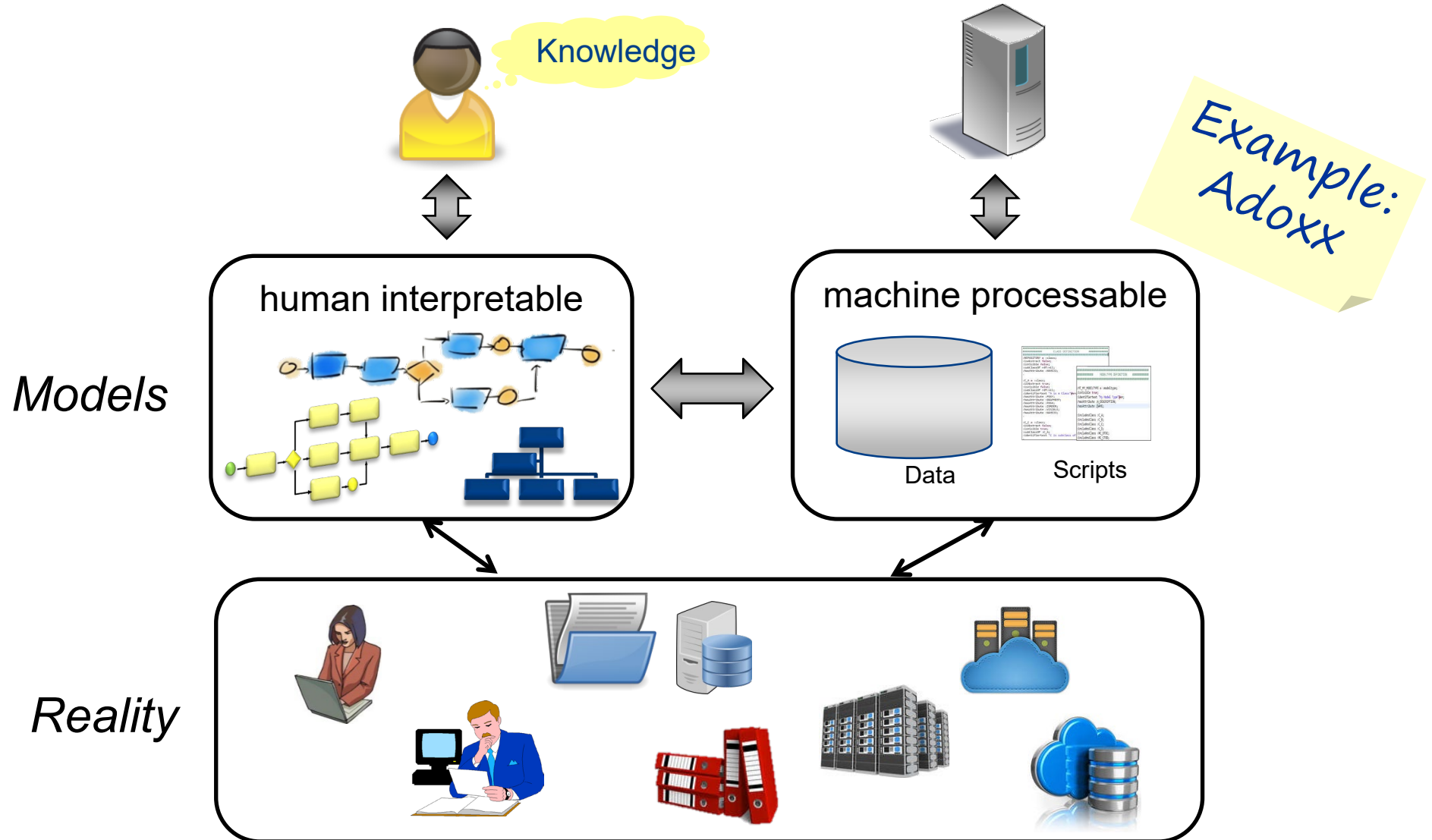


Reality

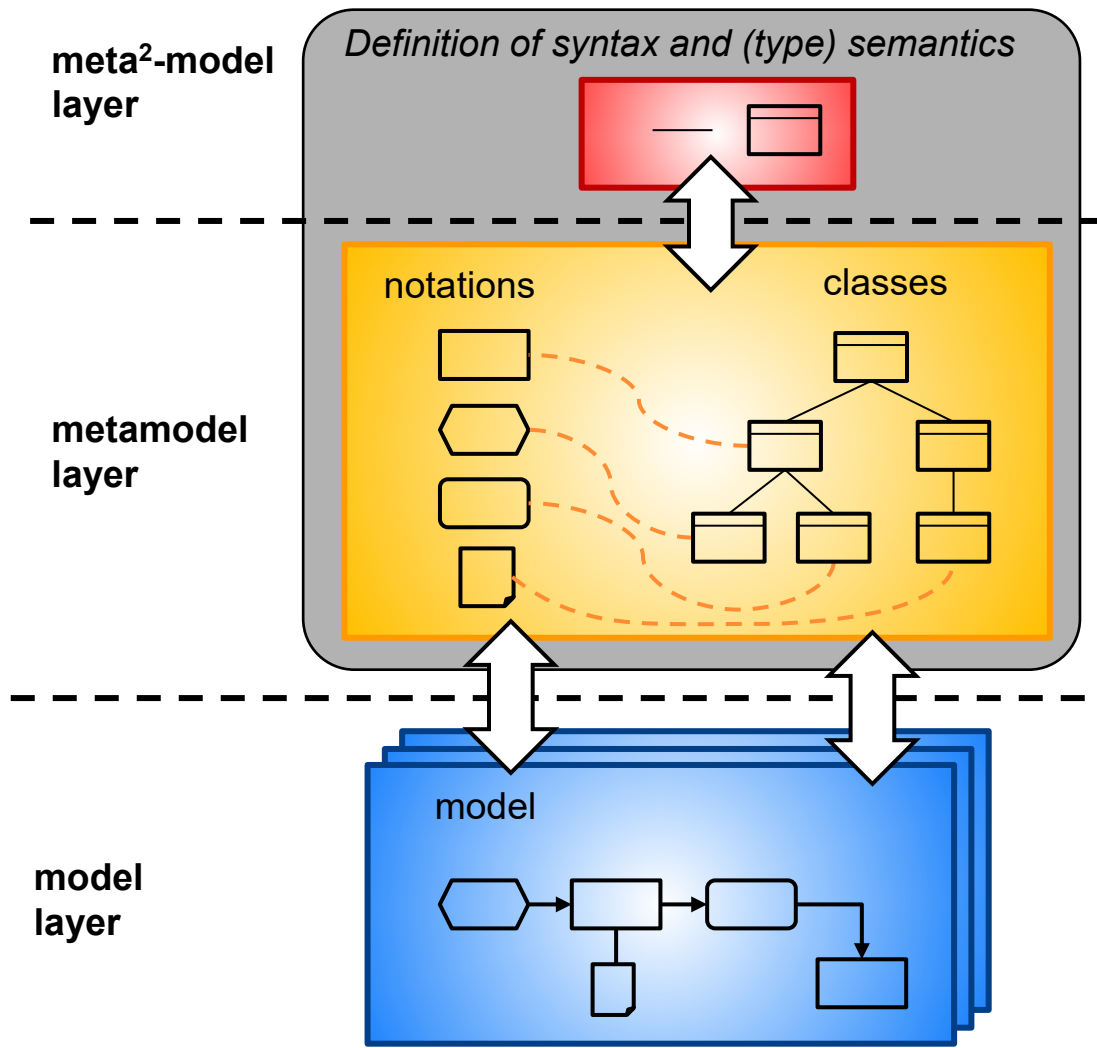


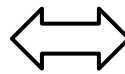
Models should allow automated analysis,
decision making and digitalization

Graphical Models represented in a Database



Modelling Environment




linguistic metamodeling: syntax, structure, type semantics

Can a machine understand what is in a model?

© 2000 Randy Glasbergen.
www.glasbergen.com



**"THE COMPUTER SAYS I NEED TO UPGRADE MY BRAIN
TO BE COMPATIBLE WITH ITS NEW SOFTWARE."**

Knowledge in Models

- The meaning of models is based on two kinds of knowledge:
 - ◆ Meta model: Concepts of the model language
 - ◆ Application Domain: Labels/names of the model elements

- Examples:



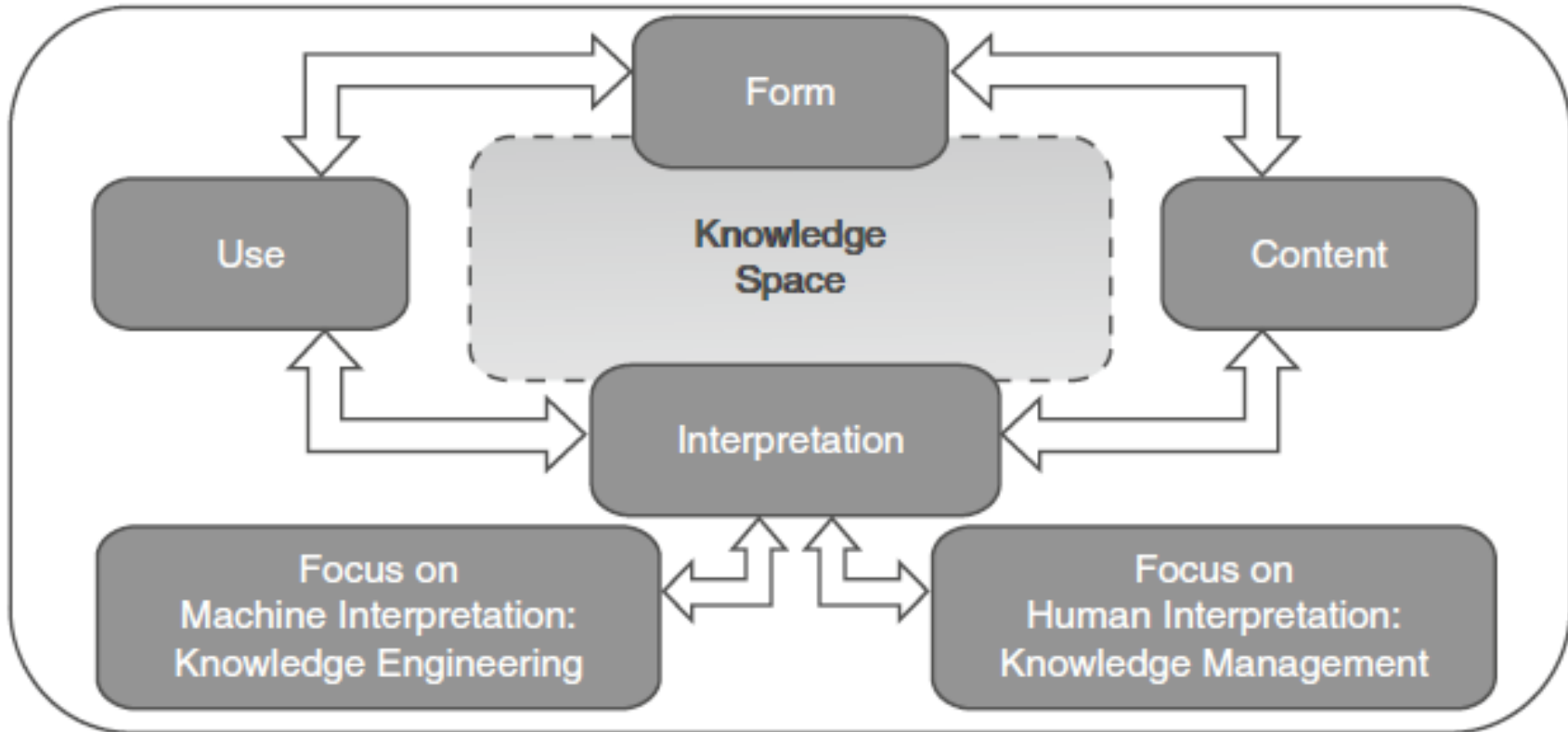
- ◆ Meta model: Application Component
- ◆ Application: «ERP System» is business software



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

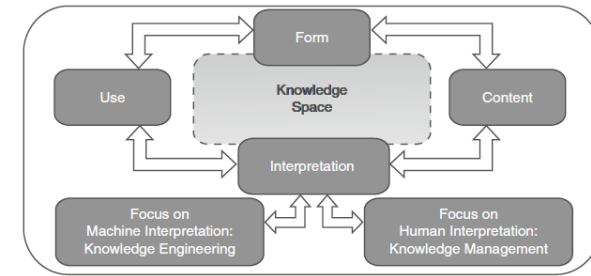
- The objective is to represent the knowledge so that it can be interpreted by a system for decision making and problem solving

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

- **Use:** Stakeholders and their concerns determine the relevant subset of the knowledge

Form: modeling language



- **Form:** Syntax and semantic of *meta model concepts*.

Content: Instantiation of 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

Knowledge Representation of Content: Linguistic View vs. Domain View

- **Linguistic View:** the specification that relates to a modelling language.

- **Domain View:** the specification that relates to a domain of discourse

ERP (Enterprise Resource Planning) is ...

... an Application
Component in ArchiMate



...a type of software system that organizations use to manage day-to-day business activities such as accounting, procurement, and supply chain operations. [...] Among the most widely used ERPs there are SAP, Oracle NetSuite, Microsoft Dynamics 365.

Content: Instantiation of Meta model + Application knowledge

- Humans «know» the meaning of the modeling objects.
 - ◆ Meta model: Concepts of the model language
 - ◆ Application domain: 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

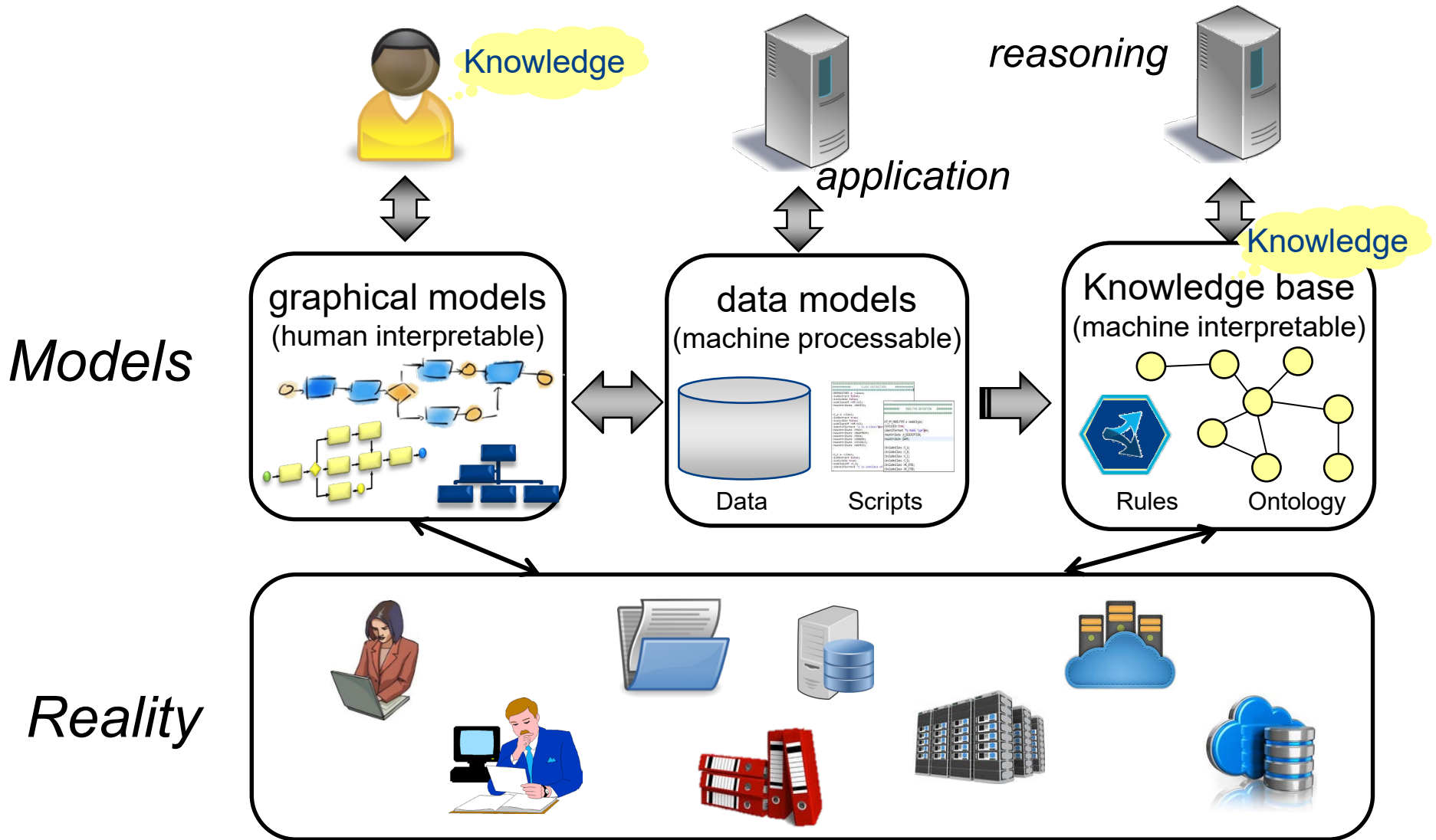


Semantic Lifting

Semantic Lifting – A Definition

- Semantic Lifting is a knowledge engineering technique that aims to annotate model constructs with ontology concepts or instances. Semantic lifting allows the formalization of the semantics of model constructs, thus enabling reasoning on and automation of knowledge contained in conceptual models

Semantic Lifting: Map Models into an Ontology



Semantic Lifting: Representing Content as Ontology

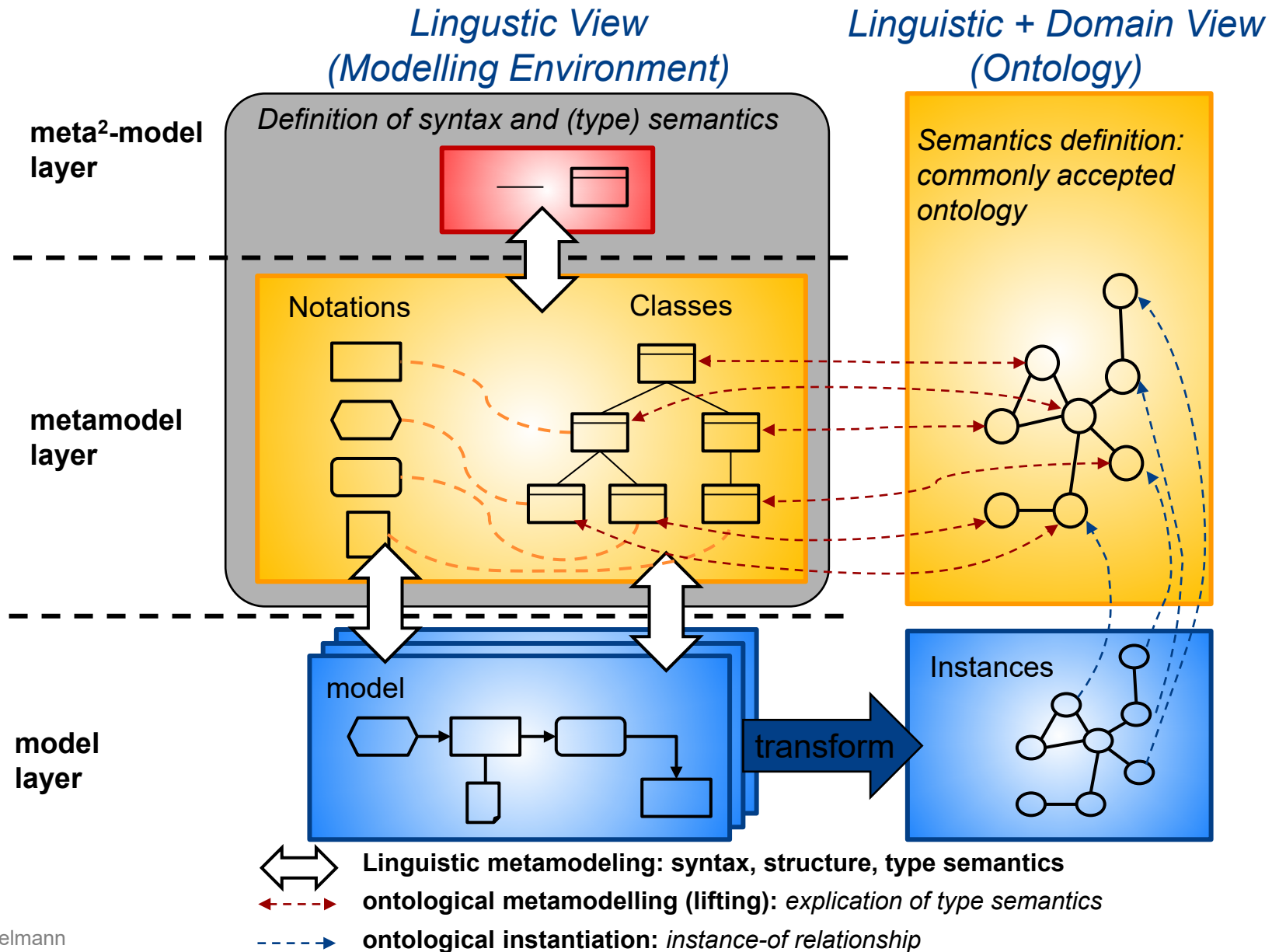
■ **Meta model Knowledge:**

- ◆ 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

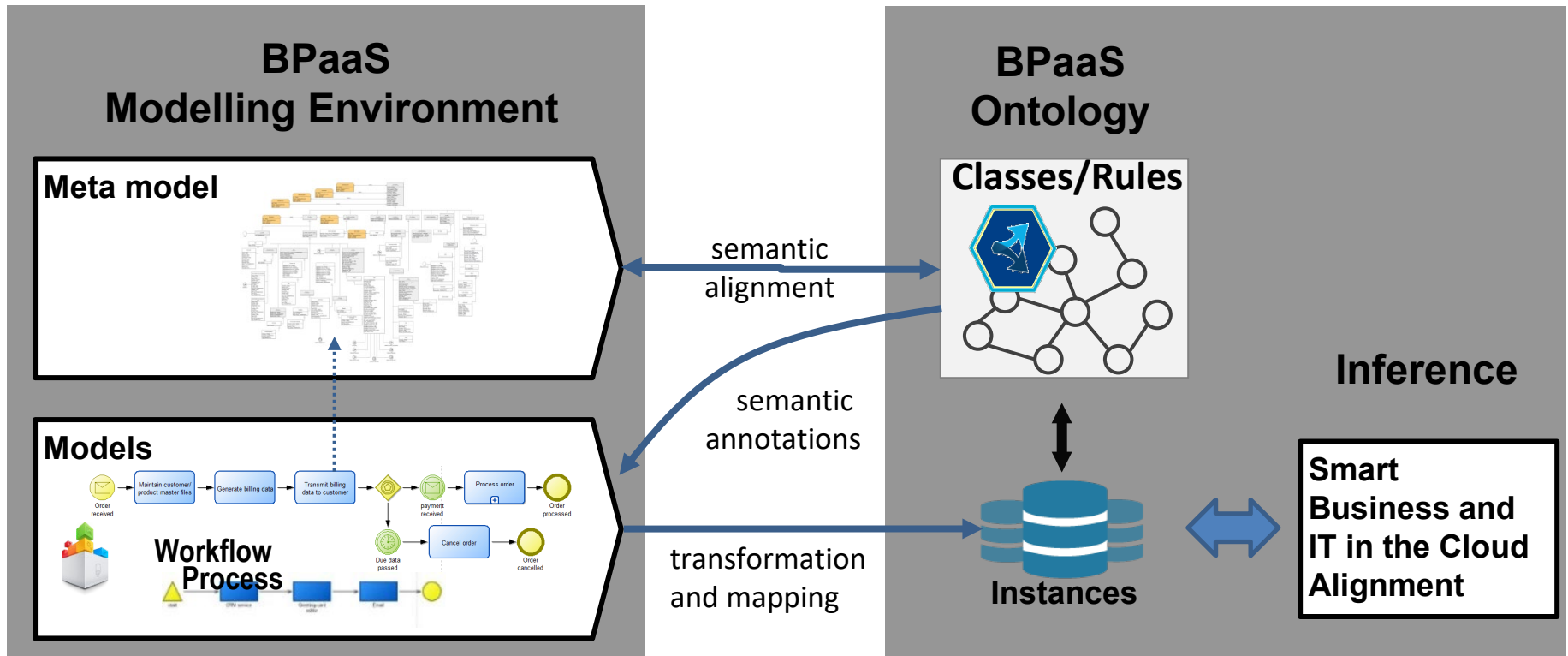
Semantic Lifting: Map Models into an Ontology



Example: Business Process as a Service

human interpretation
informal and semi-formal

machine interpretation
formal



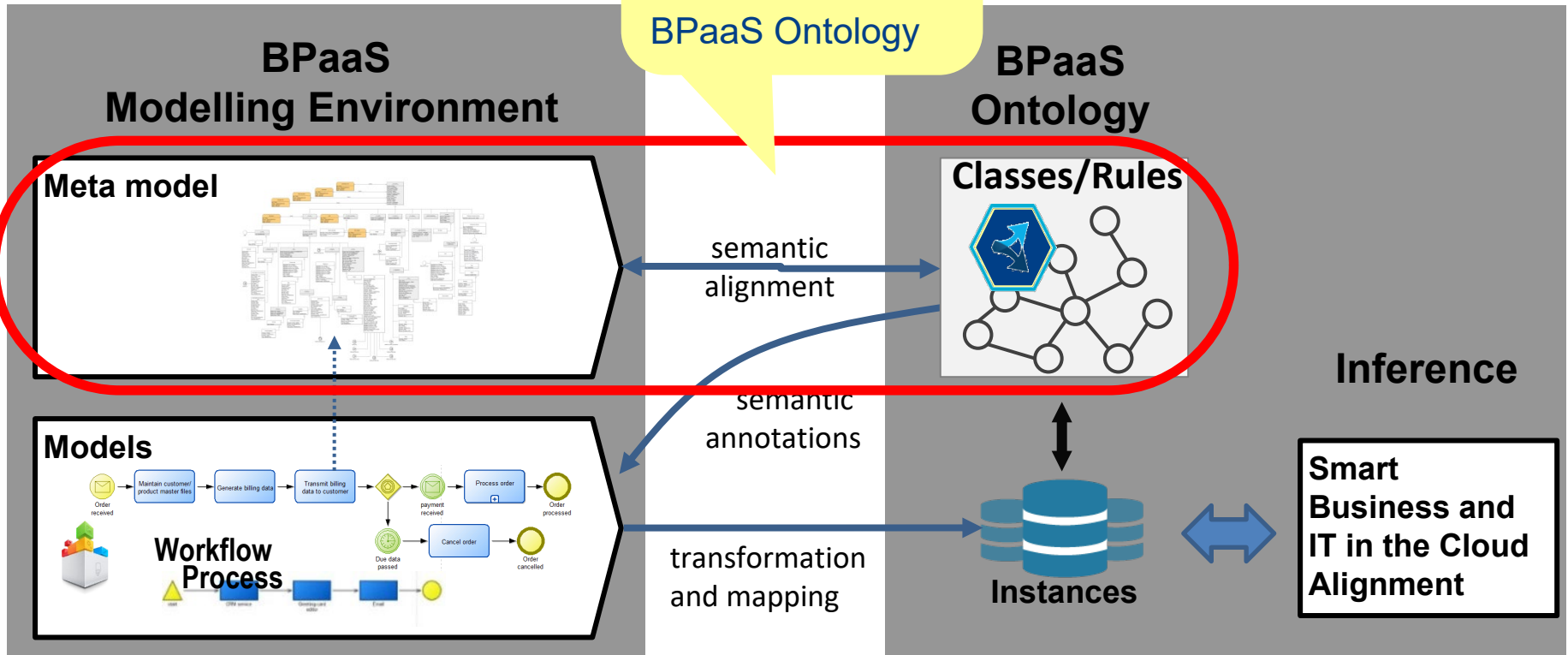
From: CoudSocket Project

Example: Business Process as a Service

human interpretation
informal and semi-formal

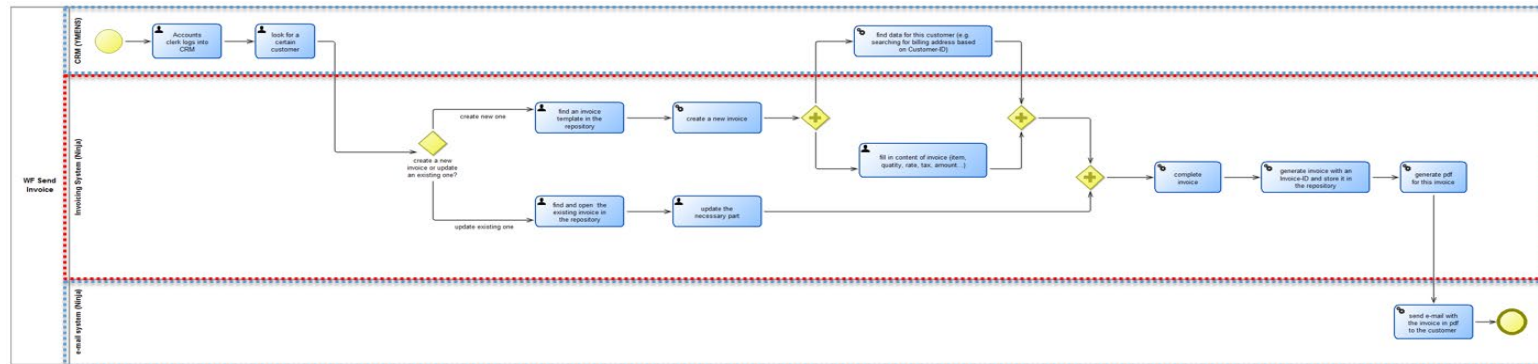
The semantics of the meta-model elements is defined in the BPaaS Ontology

machine interpretation
formal



From: CoudSocket Project

Example: Extend BPMN Element with Business Requirements



Functionality

<p>APQC</p> <p>APQC Annotation: apqc:9.2.2.3_Transmit_billing_data_to_customers_10796</p> <p>Set APQC</p> <p>Action</p> <p>Action Annotation: fbpdo:Send</p> <p>Set Action</p> <p>Object</p> <p>Object Annotation: fbpdo:Invoice</p> <p>Set Object</p>	<p>Description</p> <p>Functional</p> <p>Data Security Infrastructure</p> <p>Performance</p> <p>Support Service</p> <p>Payment</p>
--	---

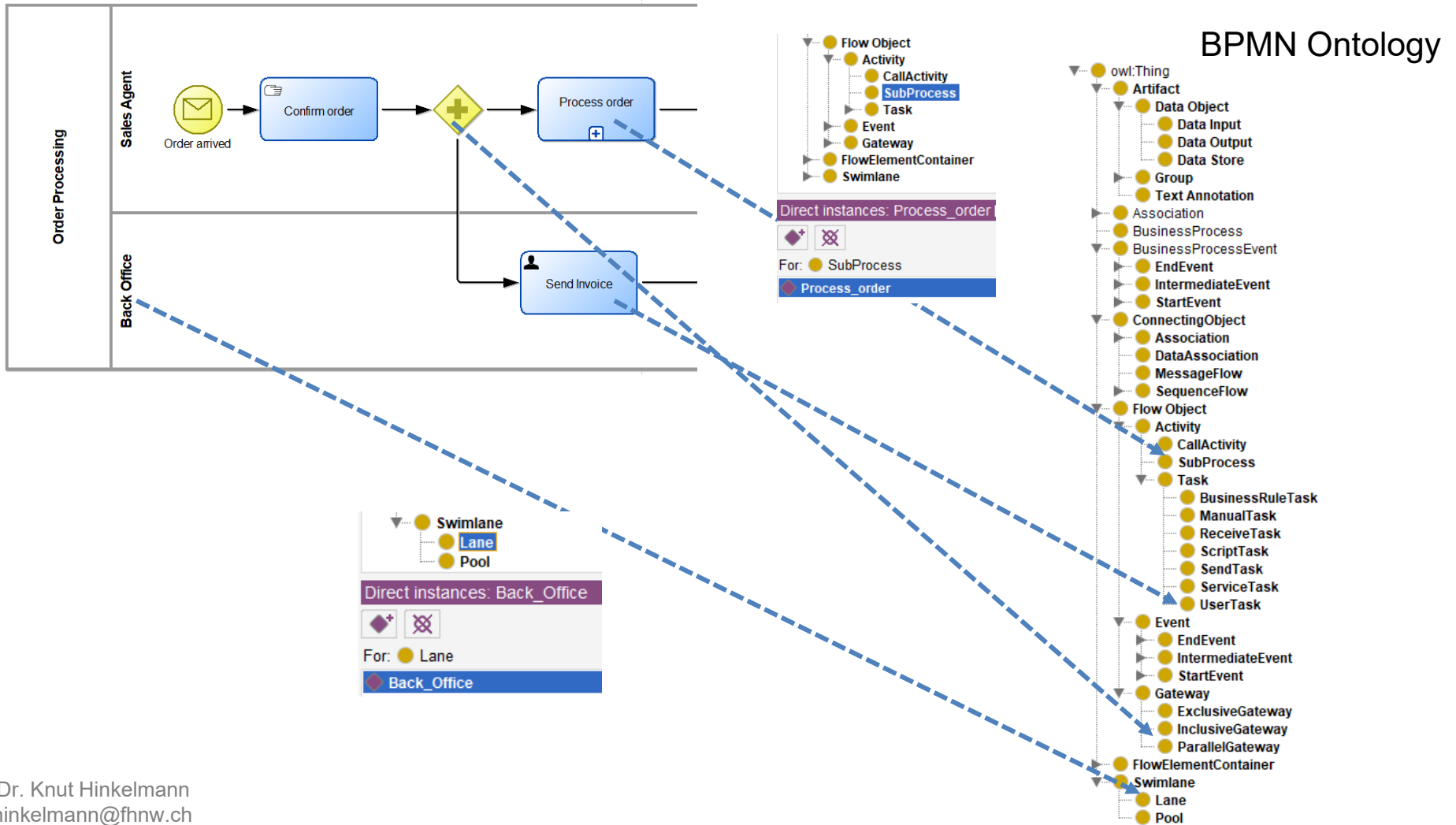
Non-functional requirements

<p>Availability</p> <p>Availability in %: 99.999</p> <p>Capacity</p> <p>Max Available Data Storage in GB per Month: 5.000000</p> <p>Maximum Simultaneous Connections: 500</p> <p>Maximum Simultaneous Service Users: 500</p> <p>Response Time</p> <p>Max Average Response Time: 00:00:00:00:01</p>	<p>Description</p> <p>Functional</p> <p>Data Security Infrastructure</p> <p>Performance</p> <p>Support Service</p> <p>Payment</p>
--	---

All Concepts are defined in the Ontology

Transformation and Mapping

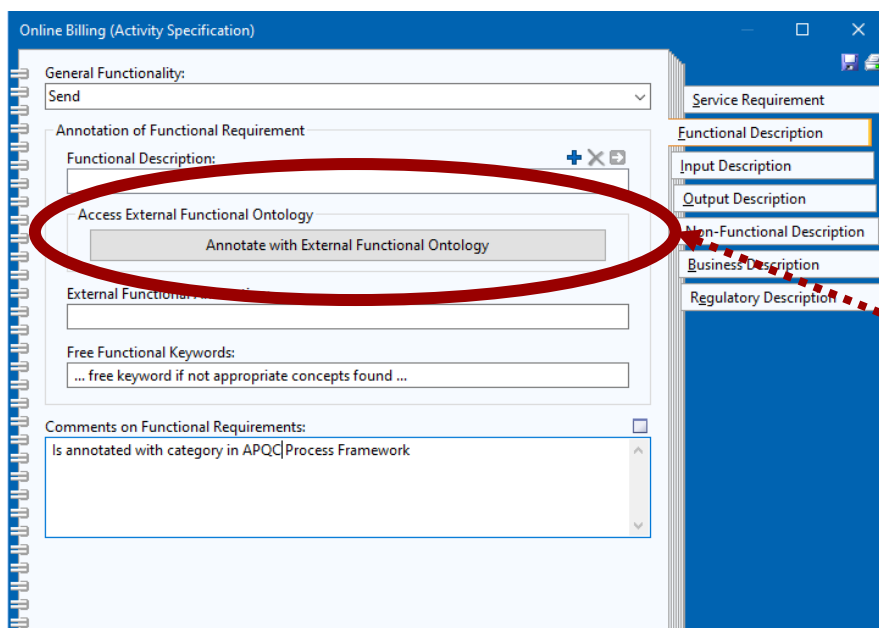
The model elements are exported as instances ontology classes



Semantic Annotation: Specifying Functionality

Annotate modeling elements with classes from the domain ontology

Example: Functionality of a Service



Domain Ontology:

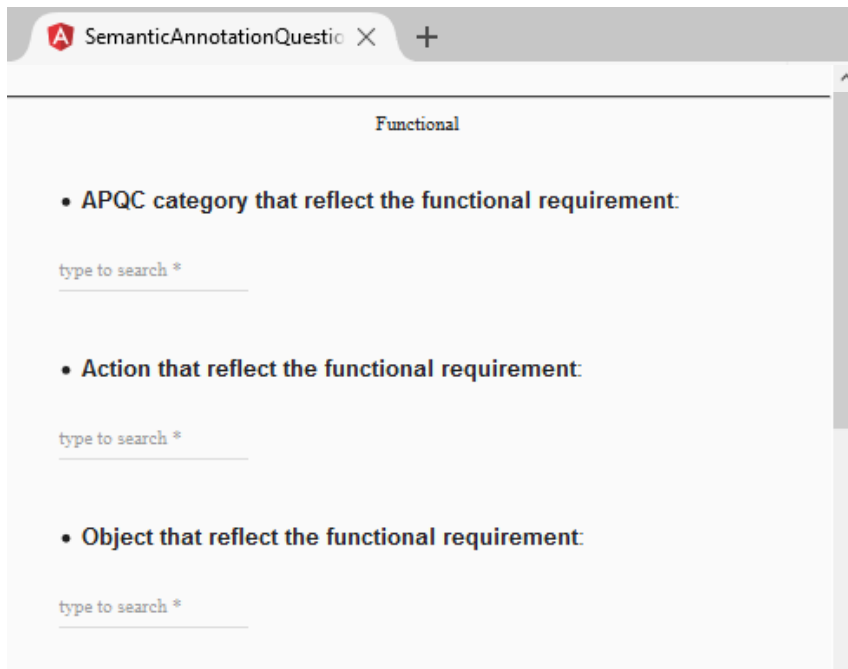
APQC Process Classification Framework

- American Productivity and Quality Center
 - Acquire, Construct, and Manage Assets
 - Deliver Physical Products
 - Deliver Services
 - Develop and Manage Business Capabilities
 - Develop and Manage Human Capital
 - Develop and Manage Products and Services
 - Develop Vision and Strategy
 - Manage Customer Service
 - Manage Enterprise Risk, Compliance, Remediation, and Resiliency
 - Manage External Relationships
 - Manage Financial Resources
 - Manage fixed-asset project accounting
 - Manage internal controls
 - Manage international funds/consolidation
 - Manage taxes
 - Manage treasury operations
 - Perform general accounting and reporting
 - Perform global trade services
 - Perform planning and management accounting
 - Perform revenue accounting
 - Invoice customer
 - **Generate customer billing data**
 - Maintain customer product master files
 - Post receivable entries
 - Resolve customer billing inquiries
 - Transmit billing data to customers
 - Manage and process adjustments/deductions
 - Manage and process collections
 - Process accounts receivable (AR)
 - Process customer credit
 - Process accounts payable and expense reimbursements
 - Process payroll
 - Manage Information Technology (IT)
 - Market and Sell Products and Services

Inferencing: Cloud Service Selection

Cloud Service Selection

Functionality

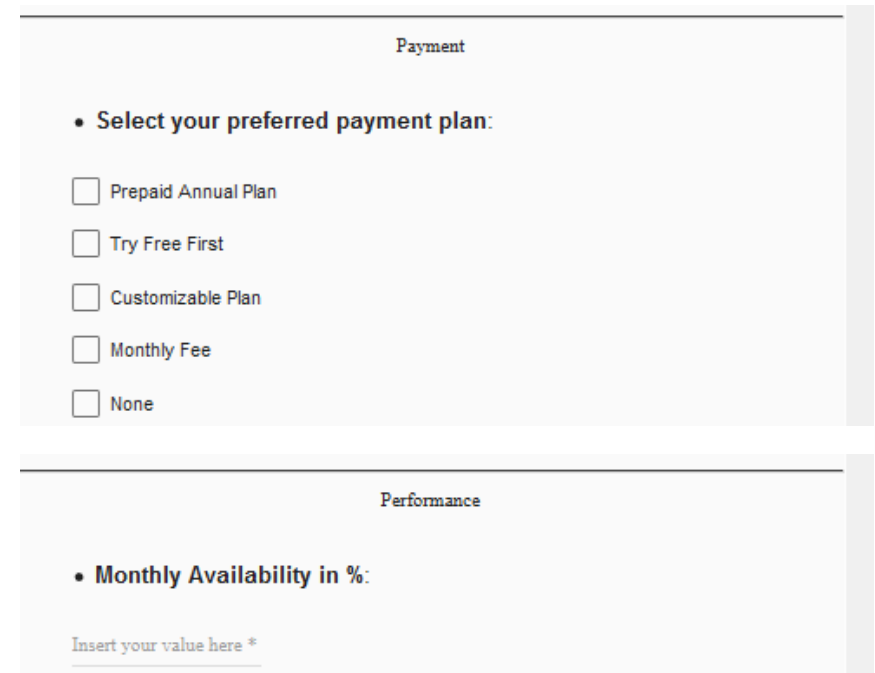


SemanticAnnotationQuestio X +

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 *

Discussion

- Drawbacks of Semantic Lifting...

Drawbacks of Semantic Lifting

- Separate Environments
 - ◆ Modelling and Metamodelling
 - ◆ Ontology
- Inconsistency
 - ◆ Metamodel and ontology must represent the same semantics but are maintained independently
 - ◆ 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

Can a machine understand what is in a model?

© 2000 Randy Glasbergen.
www.glasbergen.com



~~THE COMPUTER SAYS I NEED TO UPGRADE MY BRAIN
TO BE COMPATIBLE WITH ITS NEW SOFTWARE."~~

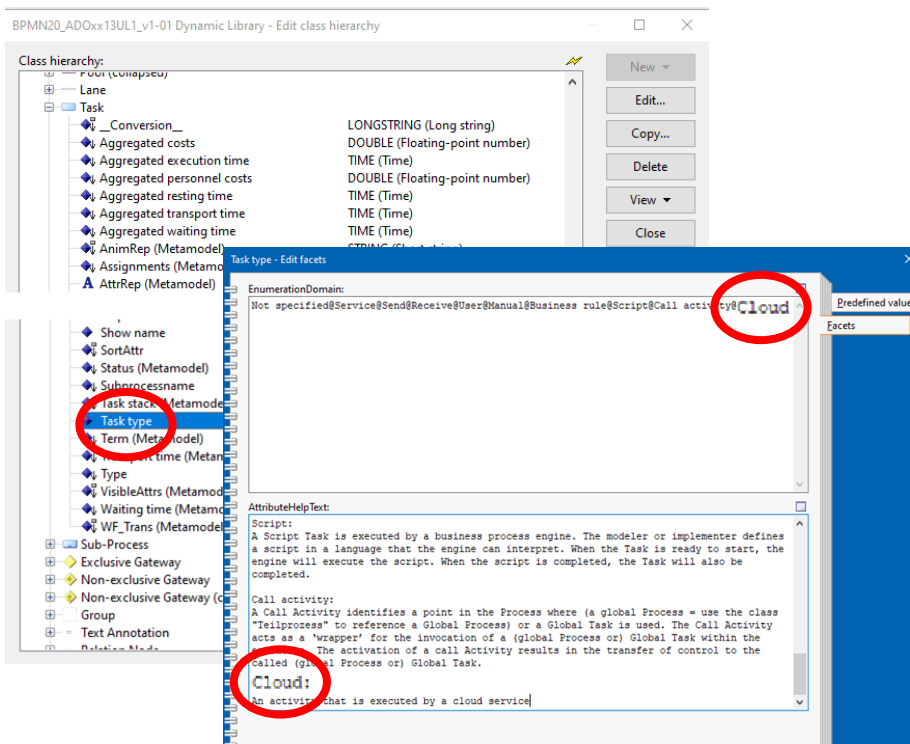
**THE COMPUTER NEEDS TO UPGRADE ITS ONTOLOGY
TO BE COMPATIBLE WITH THE NEW MODEL**

Example: New Model Element

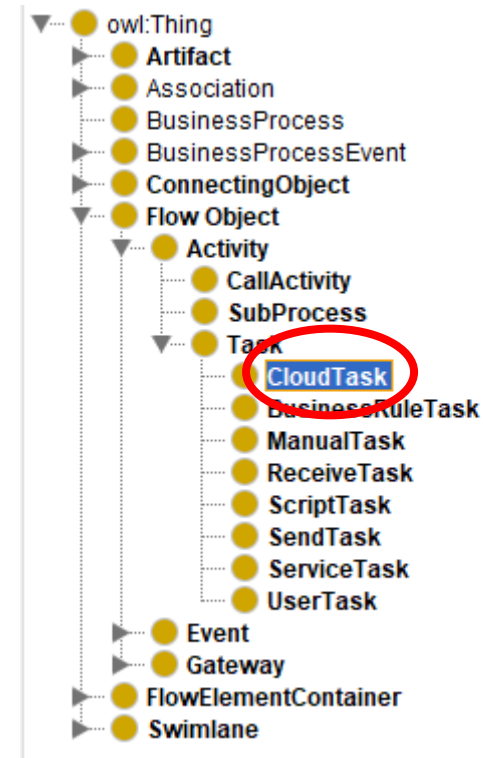
■ New task type: Cloud Task



Change in the meta model:

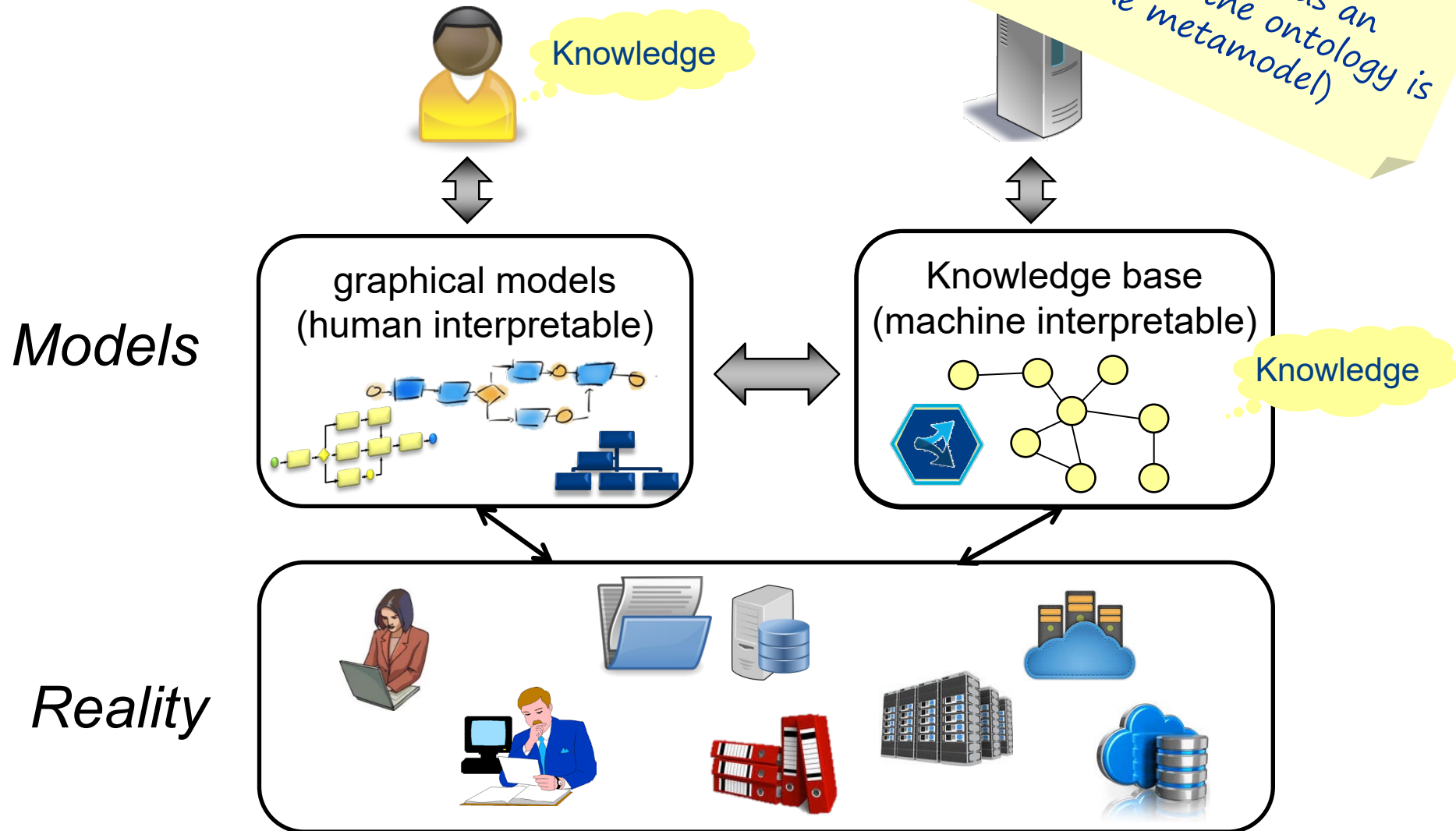


Change in the ontology:

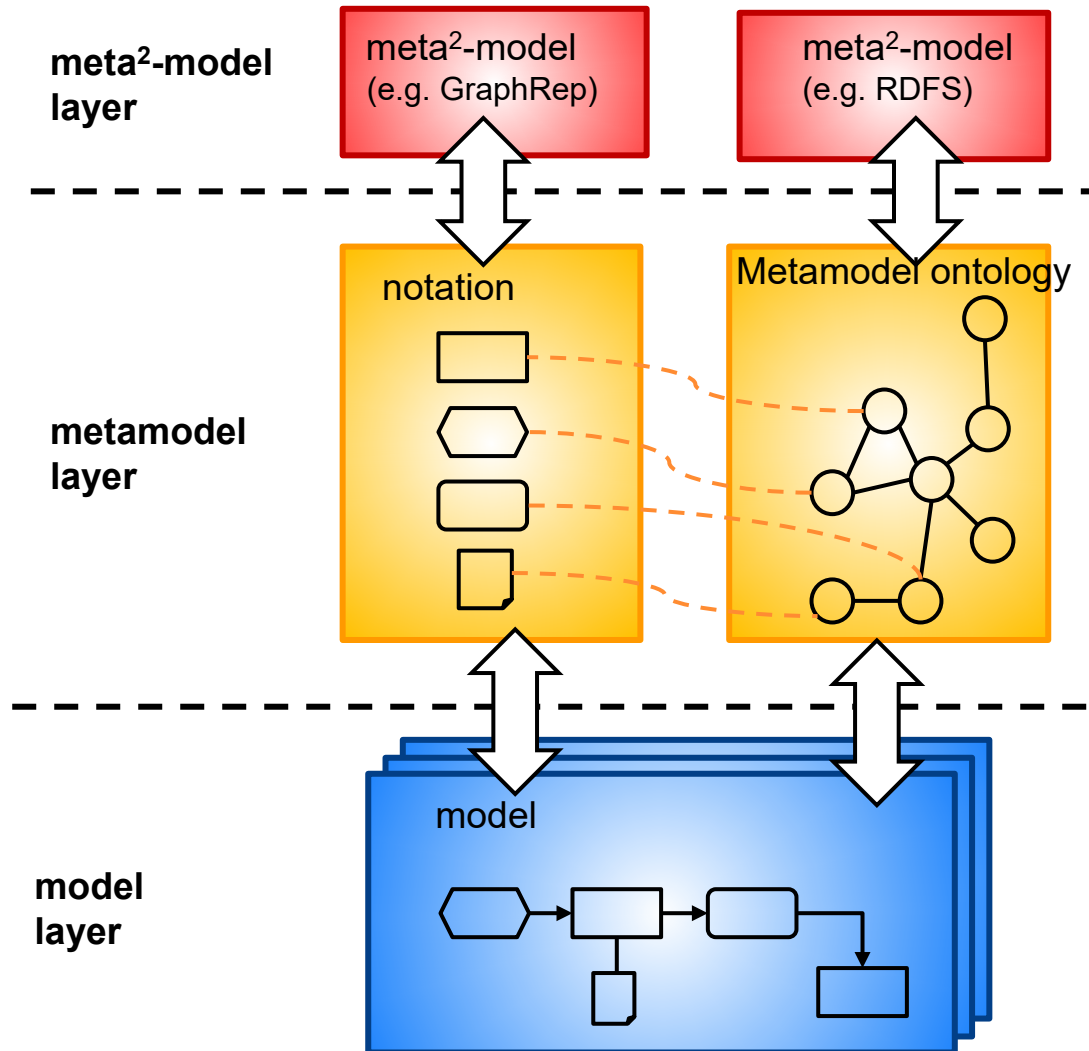


Ontology-based Metamodelling

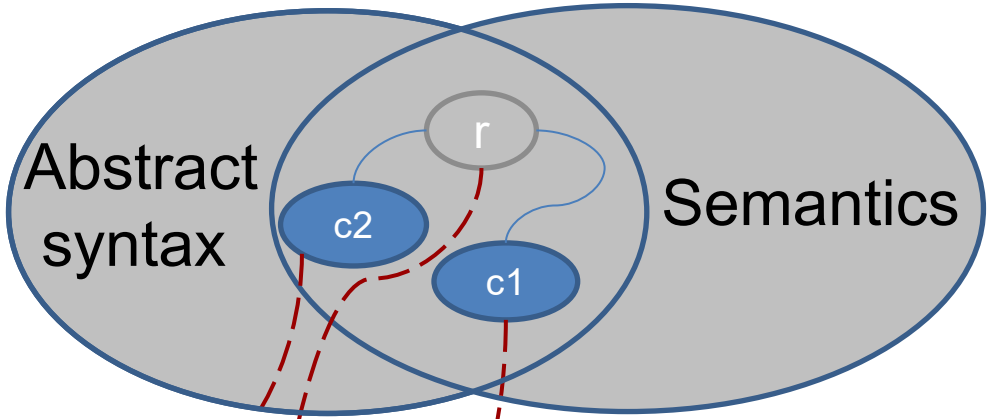
Ontology-based Metamodeling



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

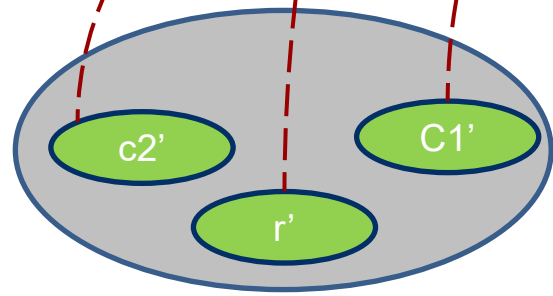


Ontology



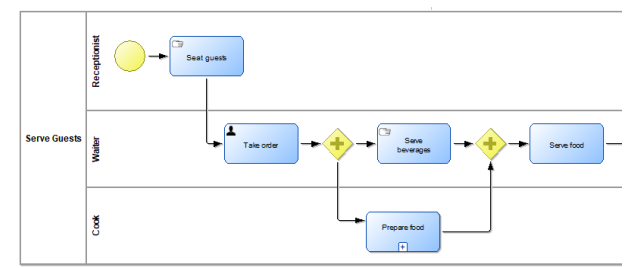
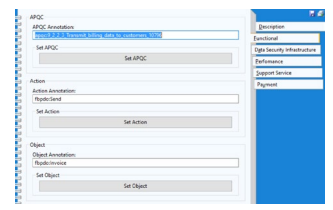
Metamodel layer

Model layer



Instances

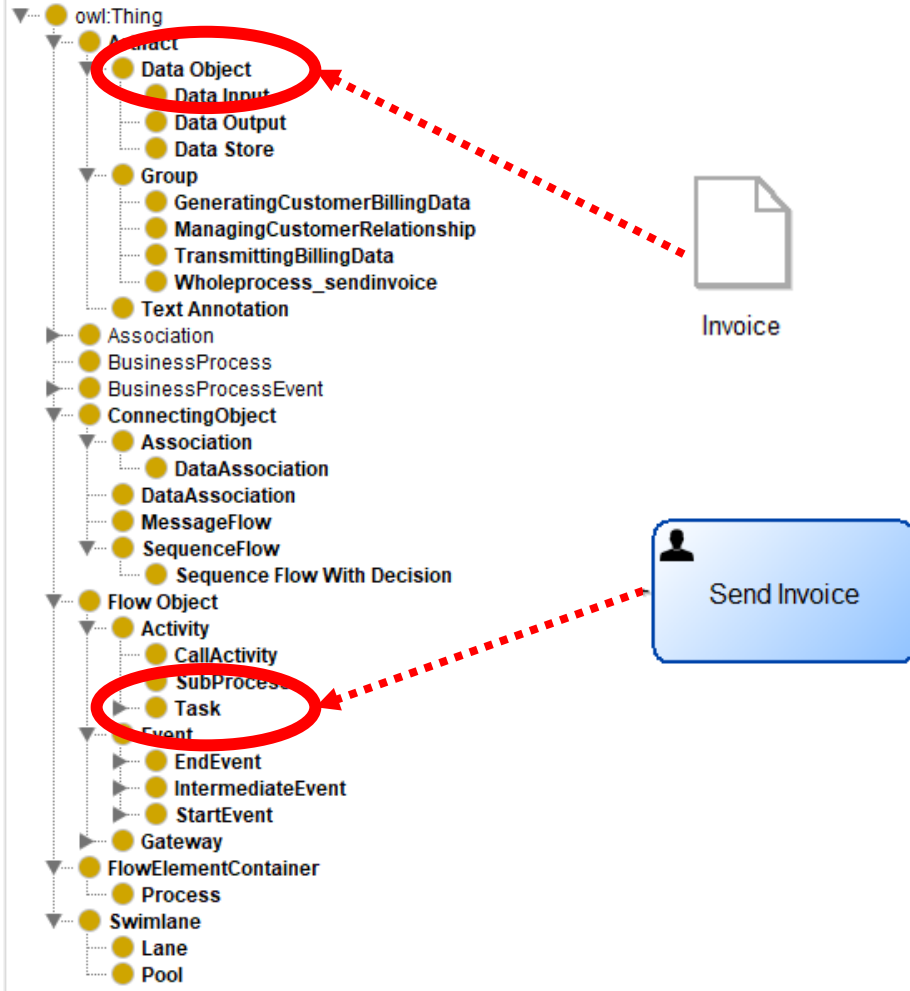
Model



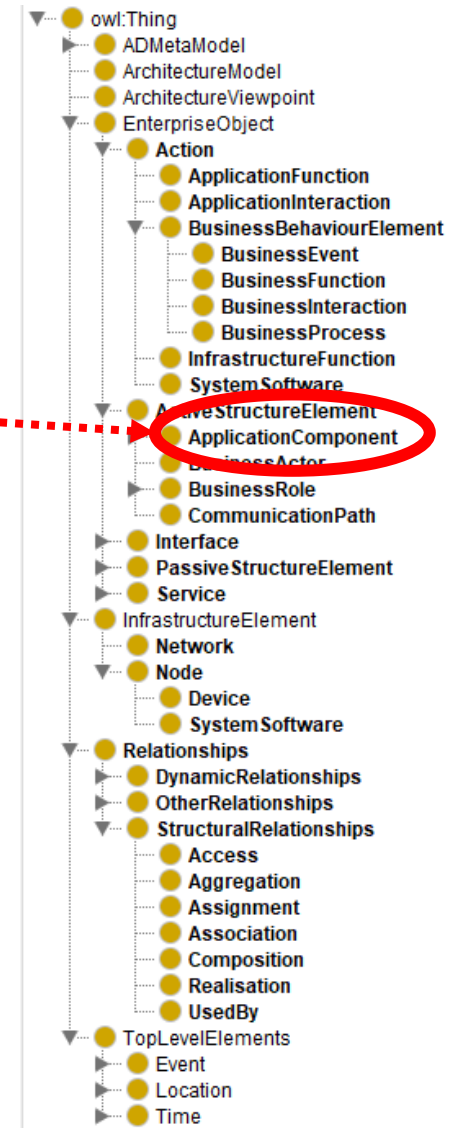
Thanks to Emanuele Laurenzi

Metamodel Ontologies

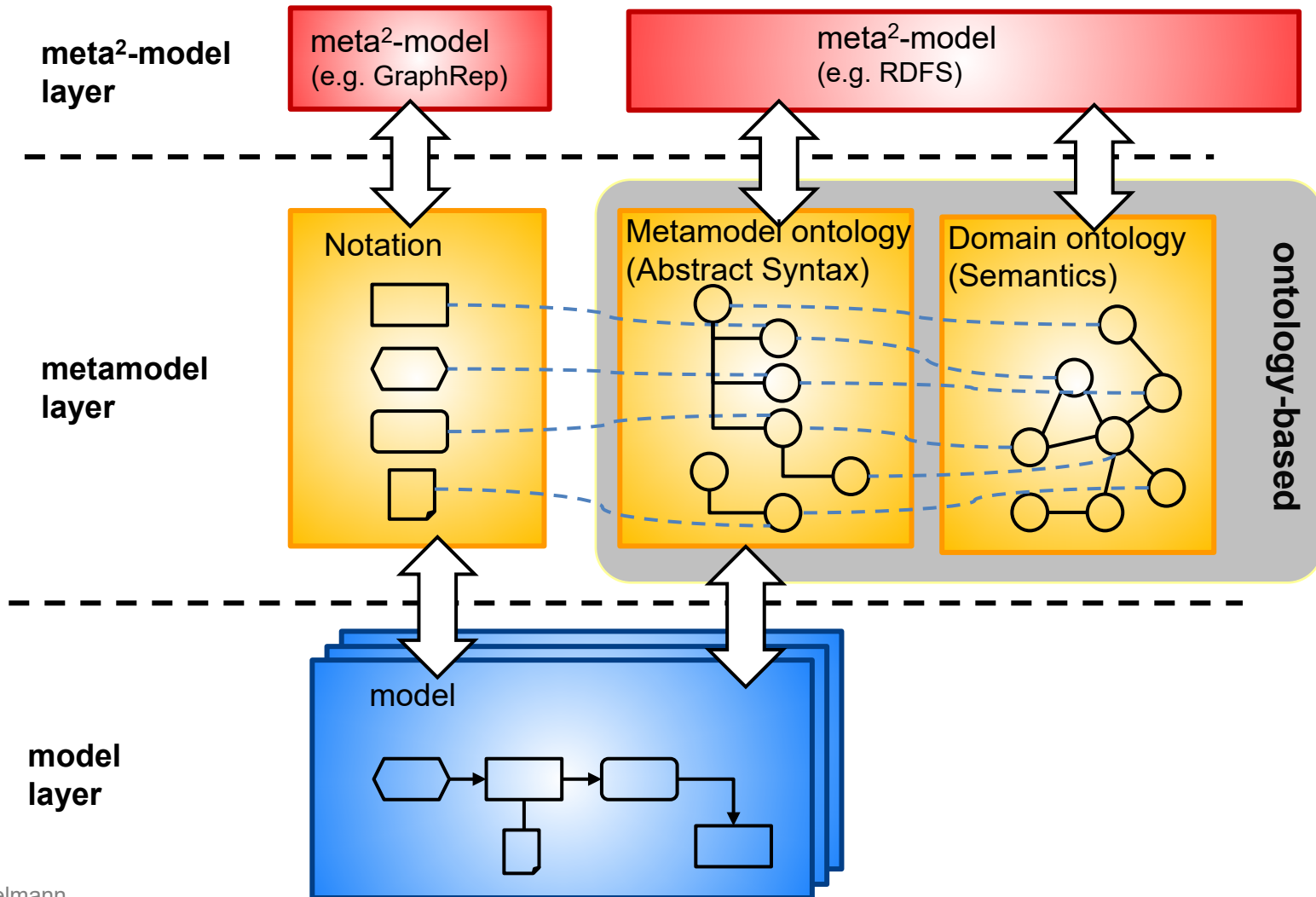
BPMN



Archimate



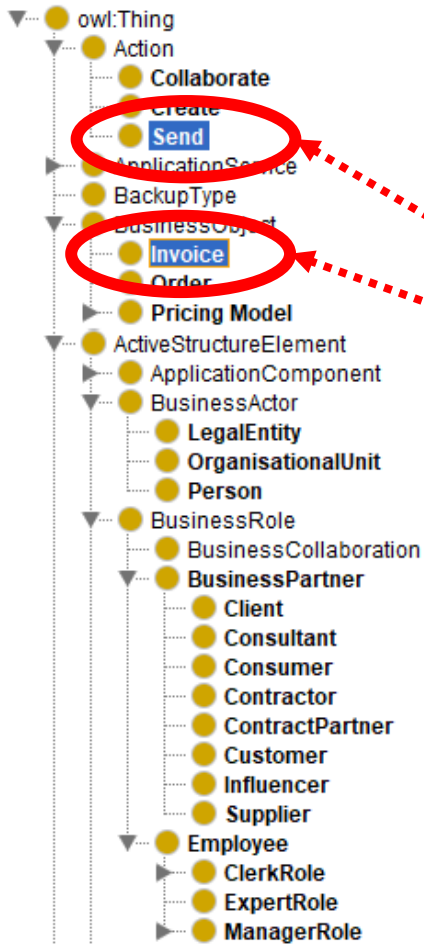
Ontology-based Metamodeling (2): Ontologies for Metamodel and Content



Domain Ontologies

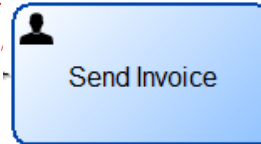
Domain Ontology: APQC Process Classification Framework

Enterprise Ontology (excerpt)

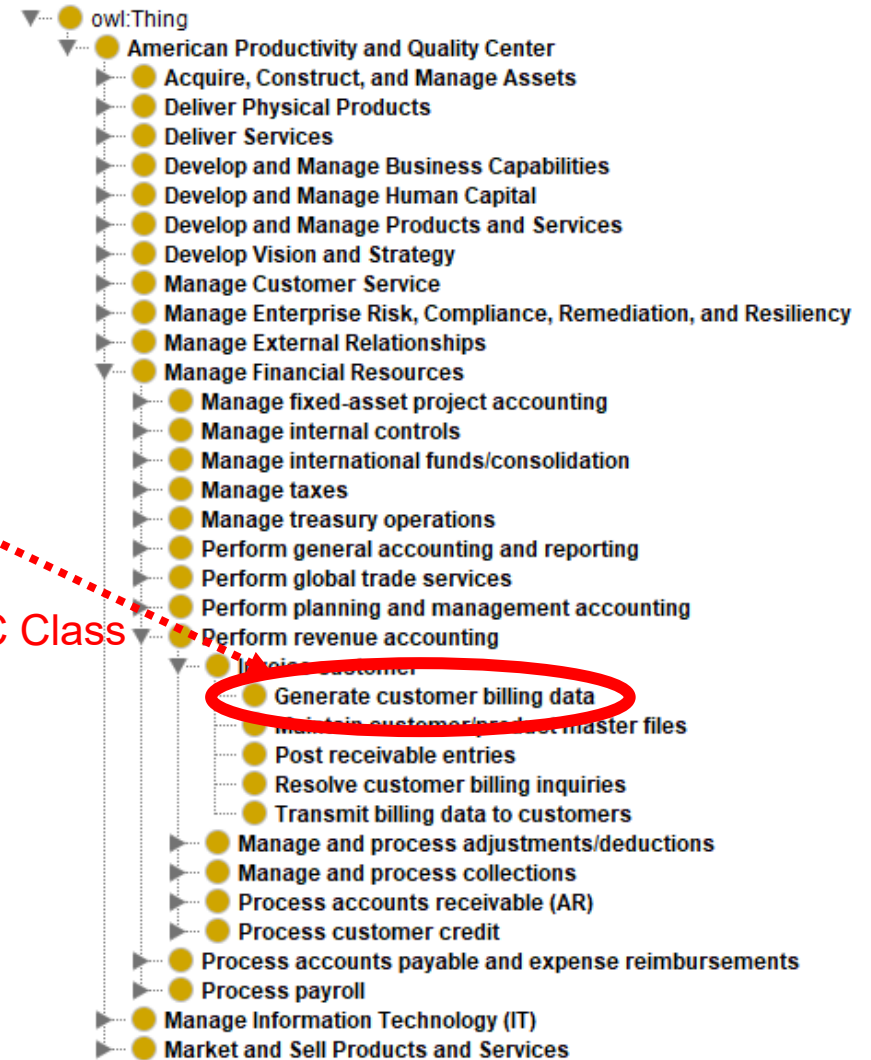


Action type

Object

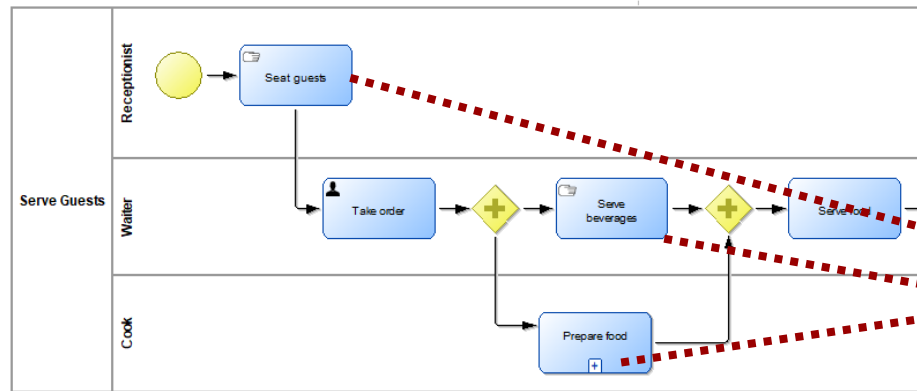


APQC Class



Ontology-Based Modeling

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



Class hierarchy: ManualTask ? | | | X

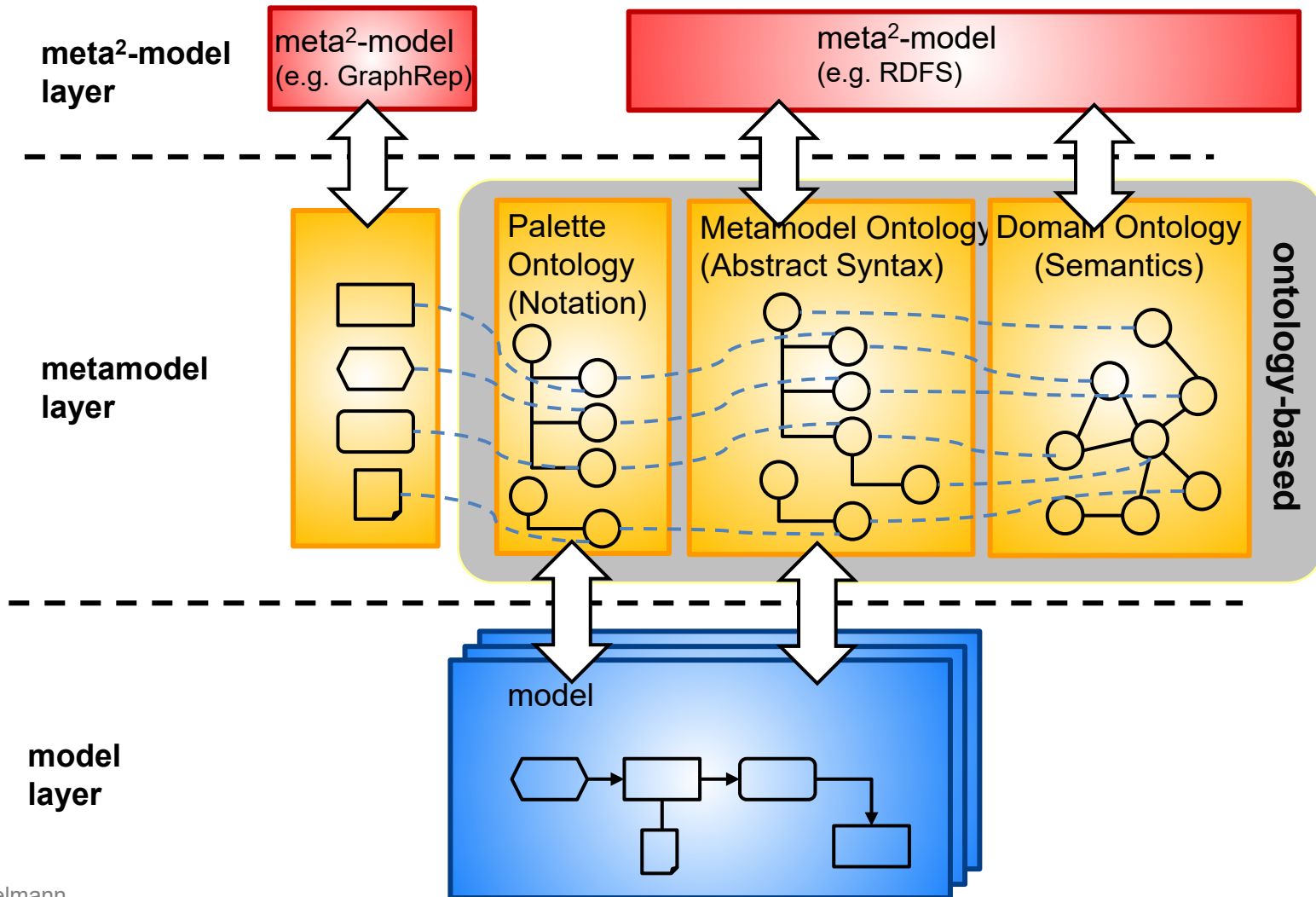
Asserted ▾

- 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 | | | X

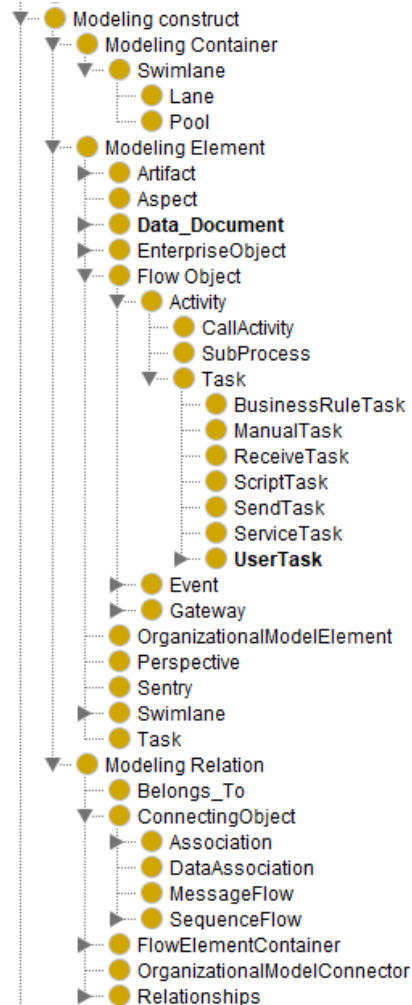
- ◆ Cook
- ◆ Prepare_Food
- ◆ Receptionist
- ◆ Seat_guests
- ◆ Serve_Beverages
- ◆ Serve_food
- ◆ Take_order
- ◆ Waiter

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



Palette Ontology

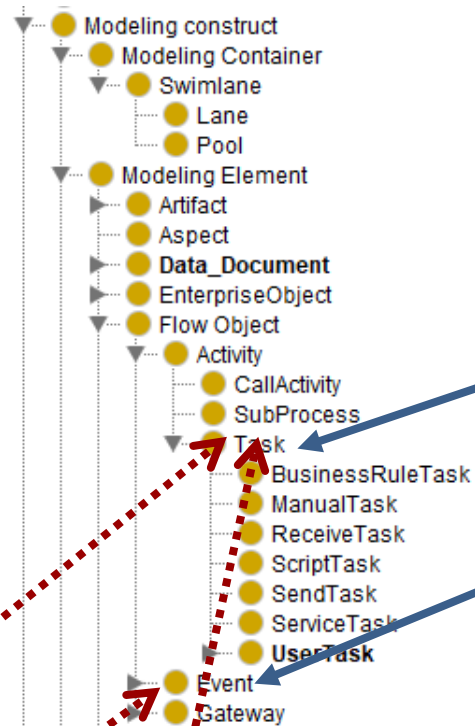
Palette Ontology (excerpt)



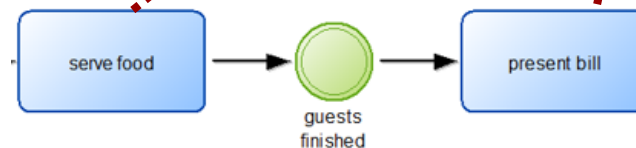
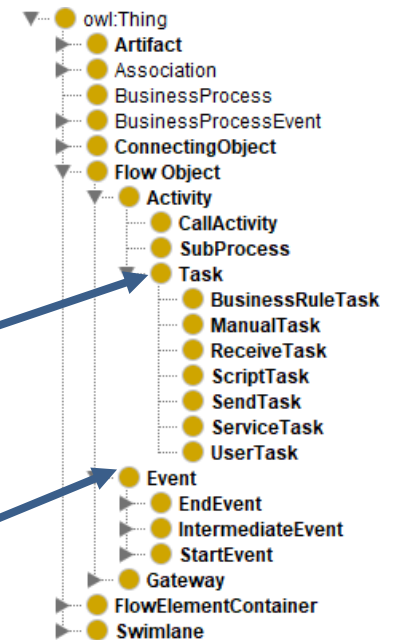
Representing Models in AOAME

- Models have several elements, named shape
- Each shape visualizes a modeling element
- Each modeling element is related to a meta model construct

Palette Ontology (excerpt)

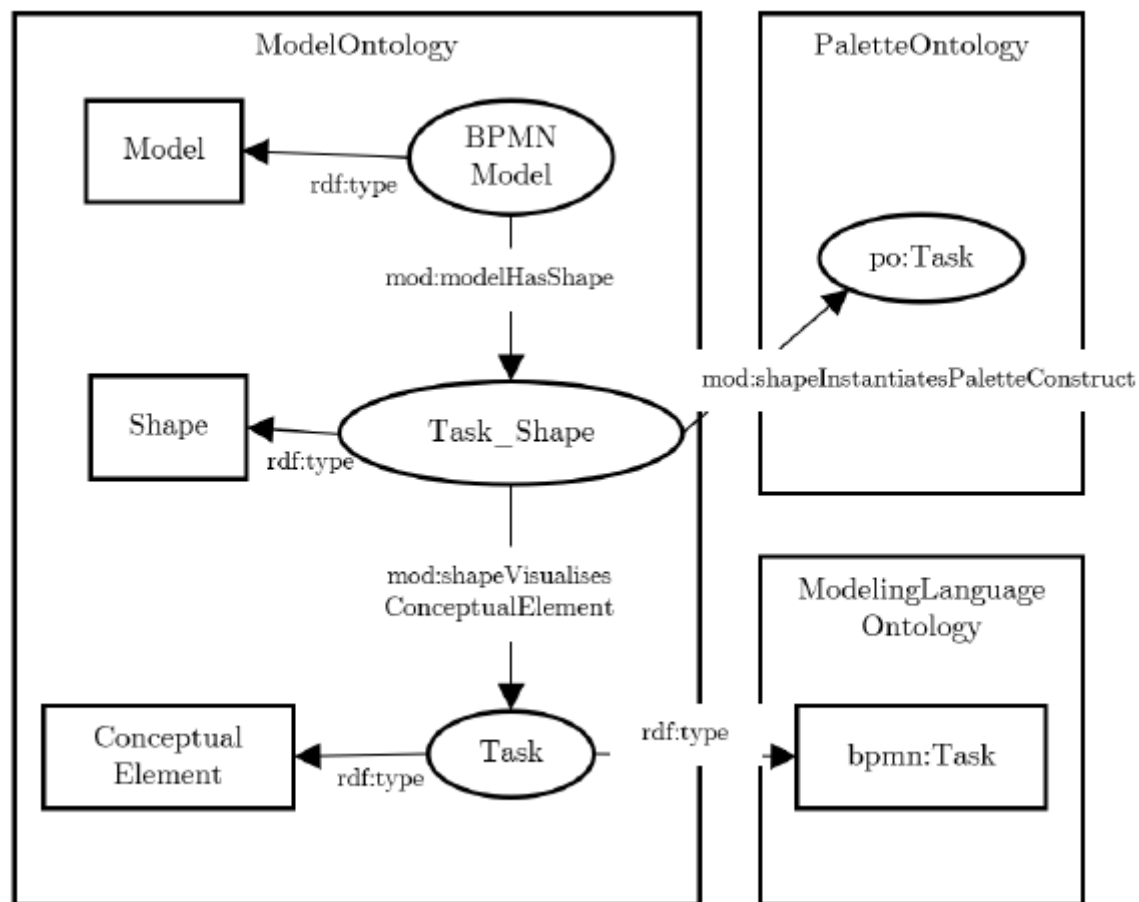


BPMN



Representing Models in AOAME

- Models have several elements, named shape
- Each shape visualizes a modeling element
- Each modeling element is related to a meta model construct
- Semantic alignment is built-in to the environment, because triples can be added for each conceptual element



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 ?task ?l
WHERE {
  ?model rdfs:label «Serve Guests».
  ?model mod:modelHasShape ?shape.
  ?shape mod:shapeVisualisesConceptualElement ?task.
  ?task rdf:type bpmn:Task .
  ?shape rdfs:label ?l.
}
```

Select the elements
(named shapes) in
the model

For the shapes find the
conceptual elements

Filter the elements for BPMN
Tasks and show the labels

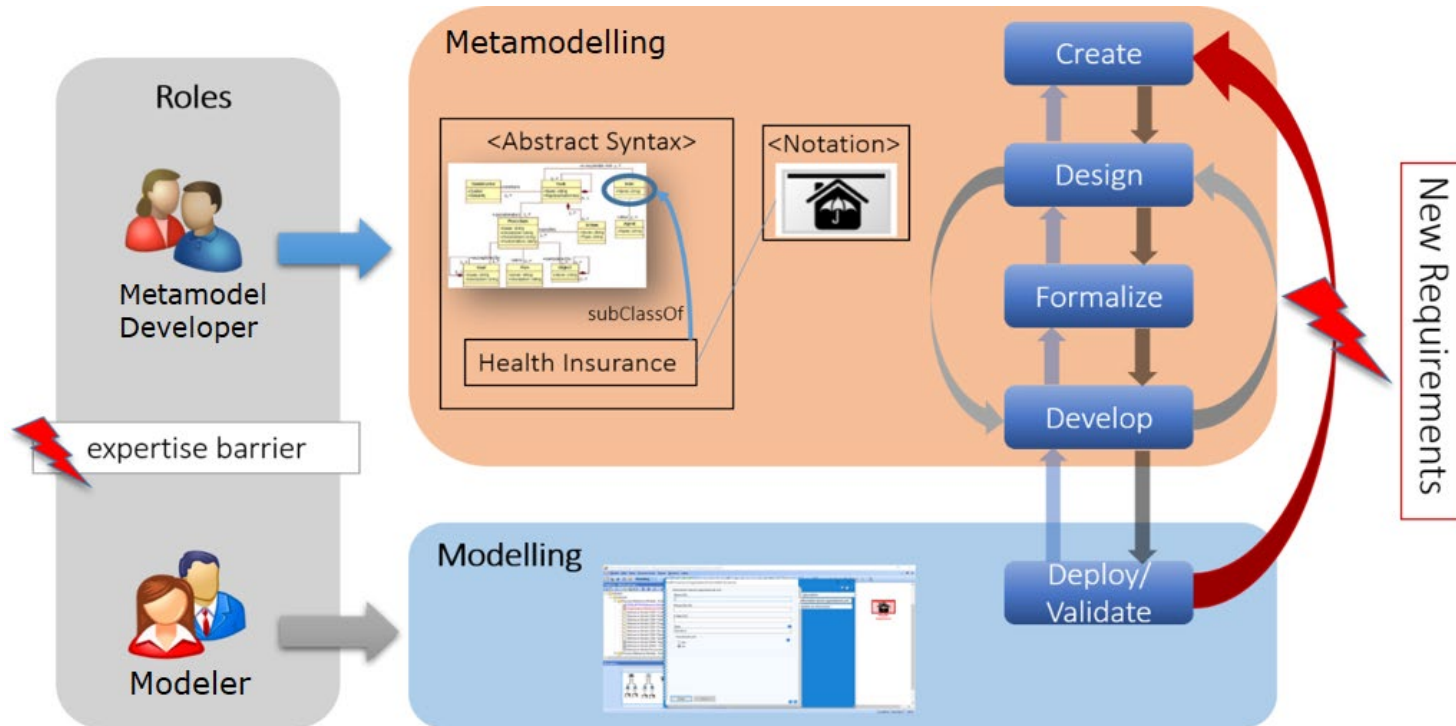


Agile Modelling

Objective

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

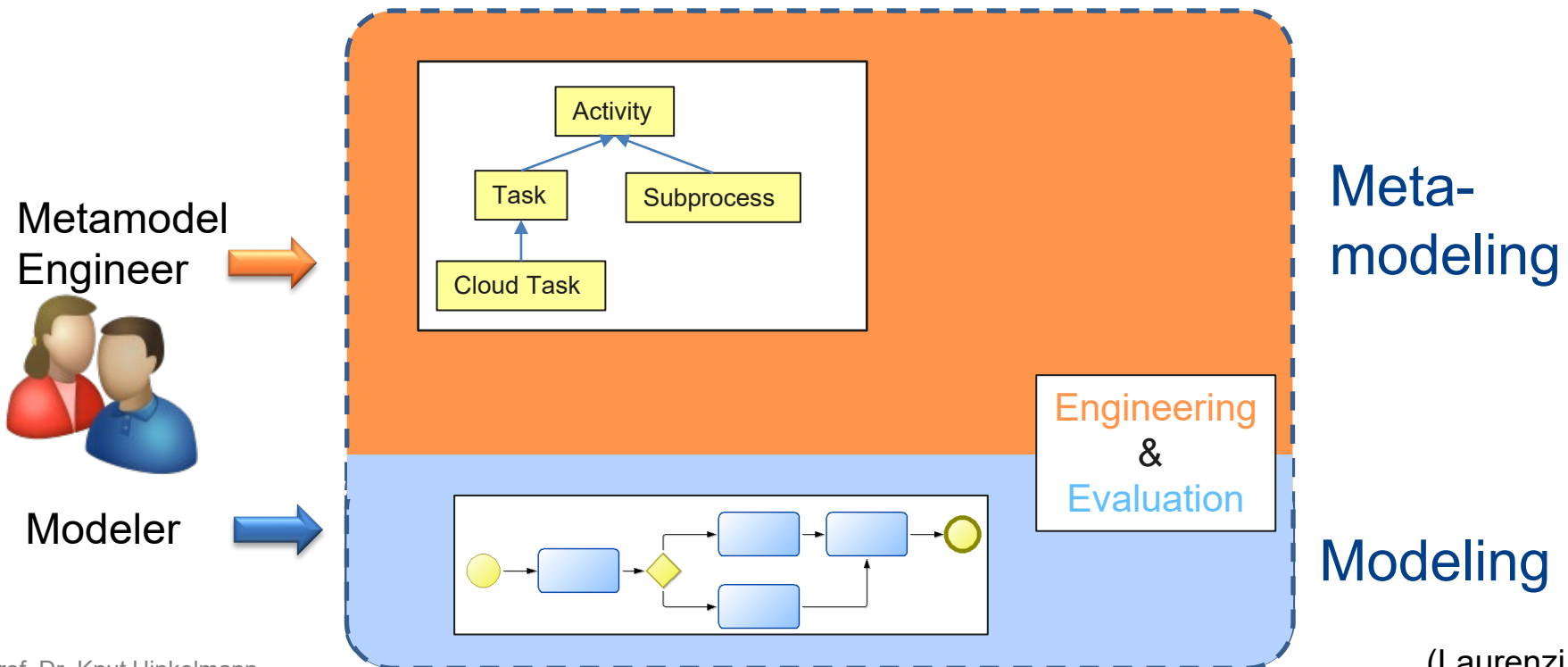
Challenge: Separation of metamodelling and modelling



- 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

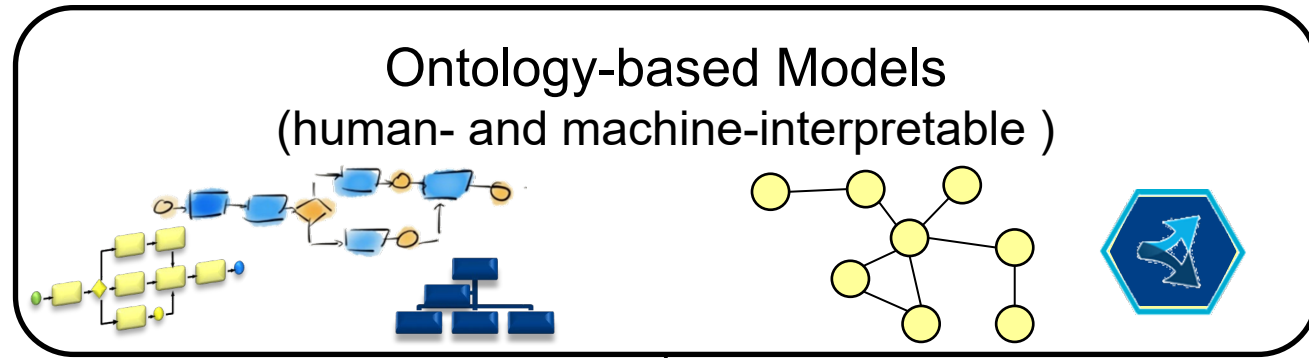
- Tight collaboration between metamodel developer and modeler
- Modeler can also take the role of metamodel developer



Agile and Ontology-Aided Modeling Environment (AOAME)



*Models +
Knowledge*



Reality



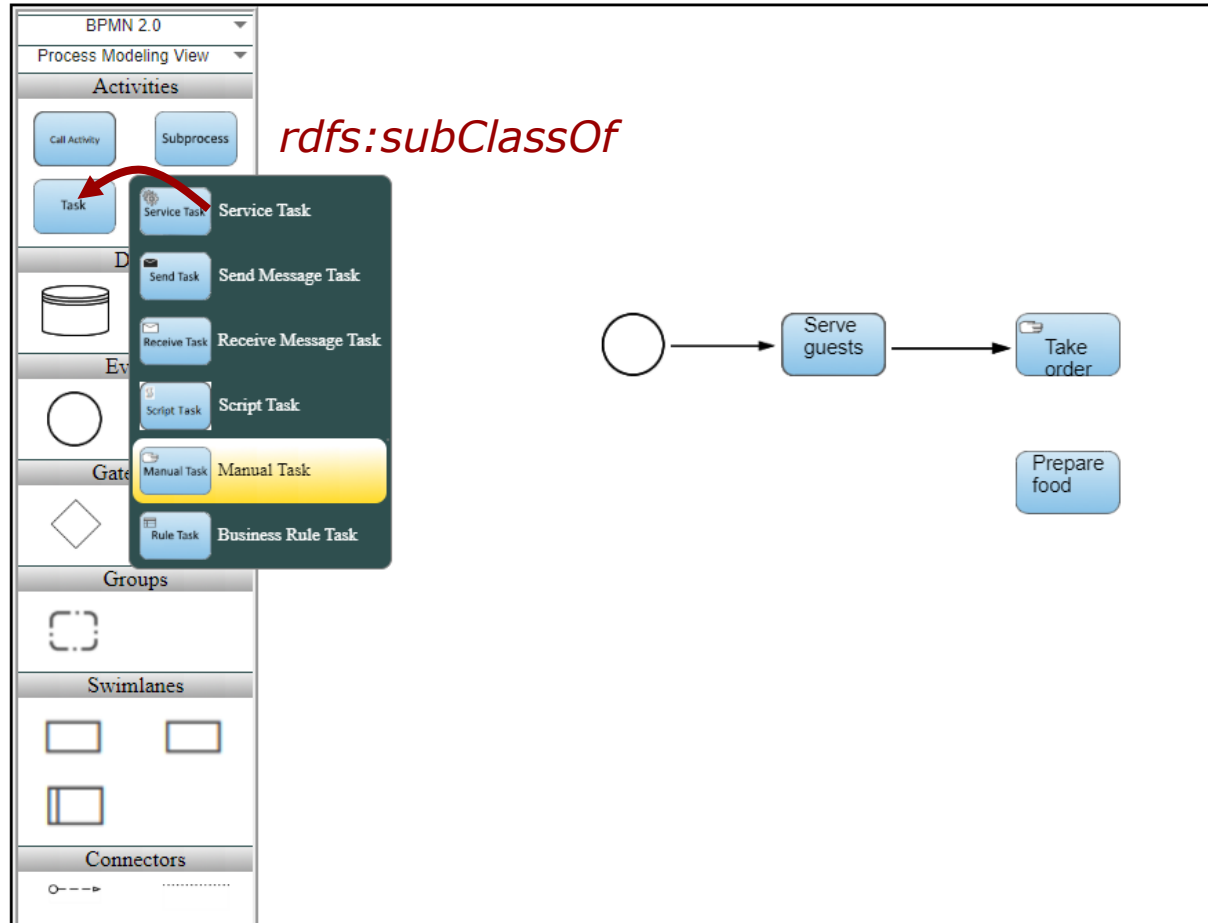
AOAME: *Agile and Ontology-Aided Modeling Environment*

- 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

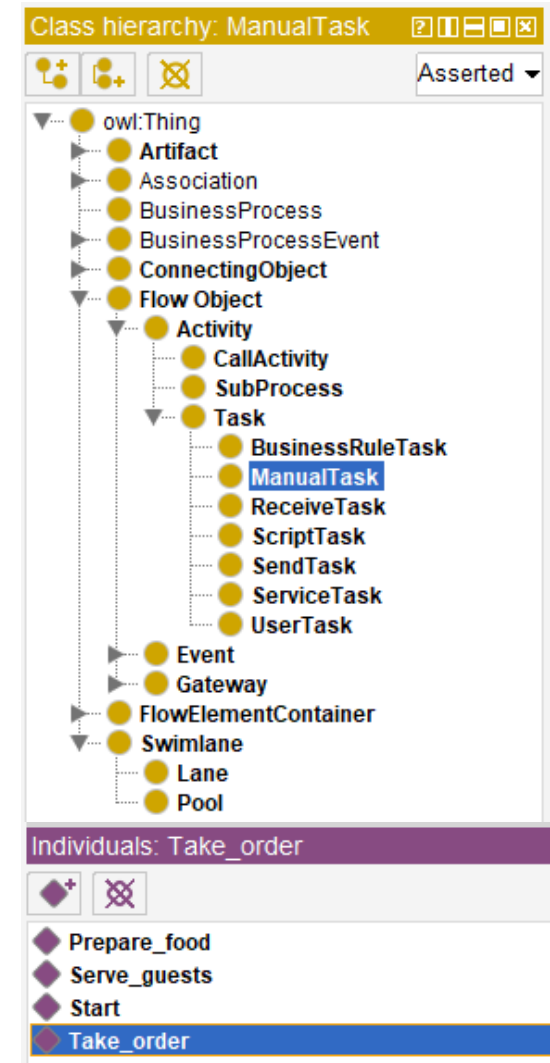
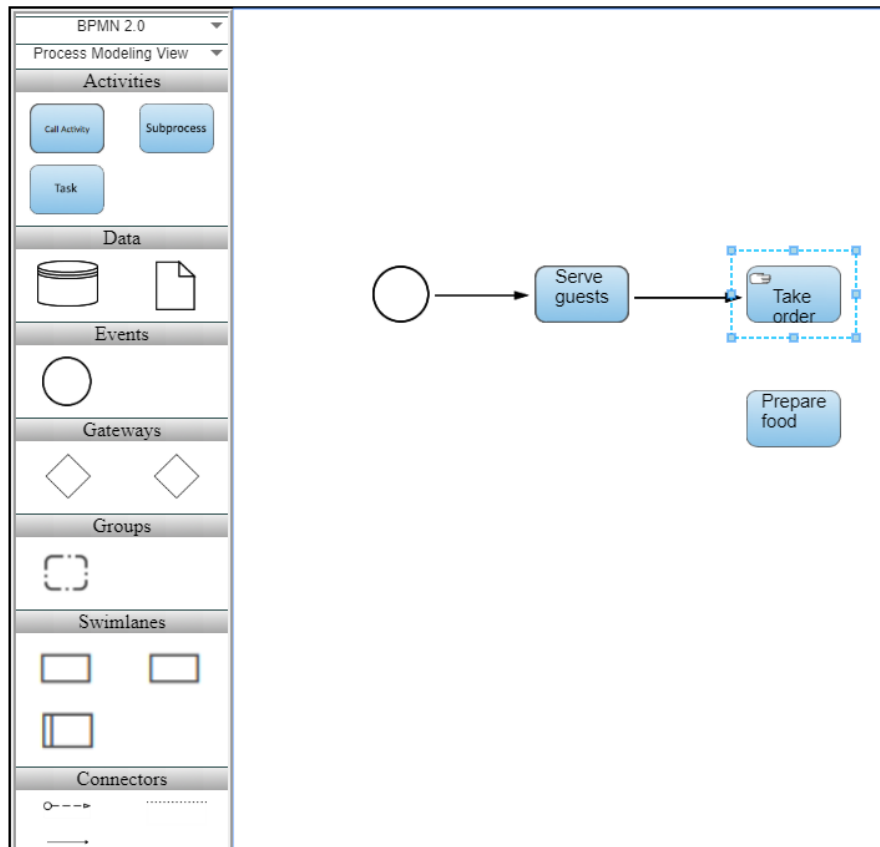
Palette

Model Editor

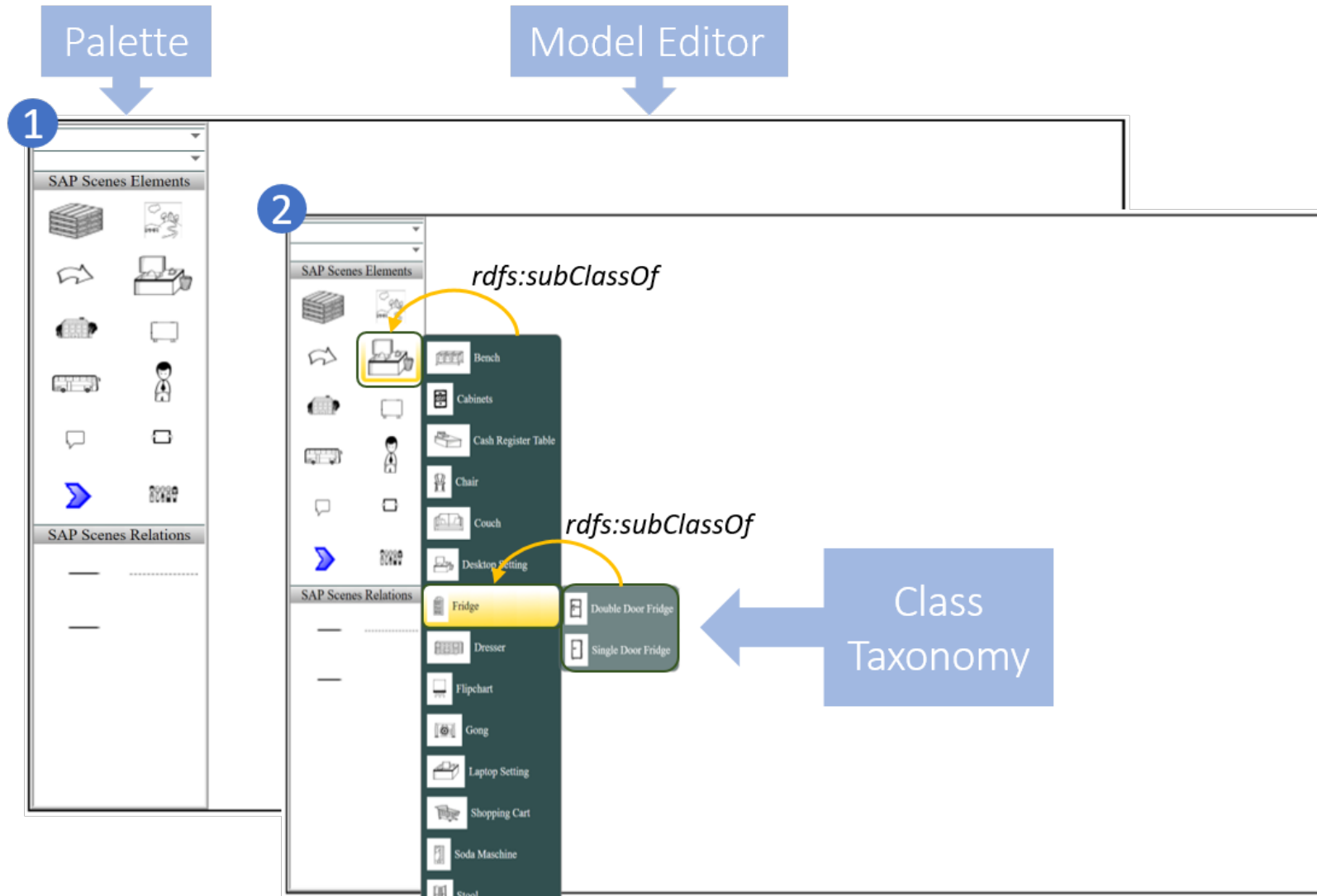


Ontology-Based Modelling

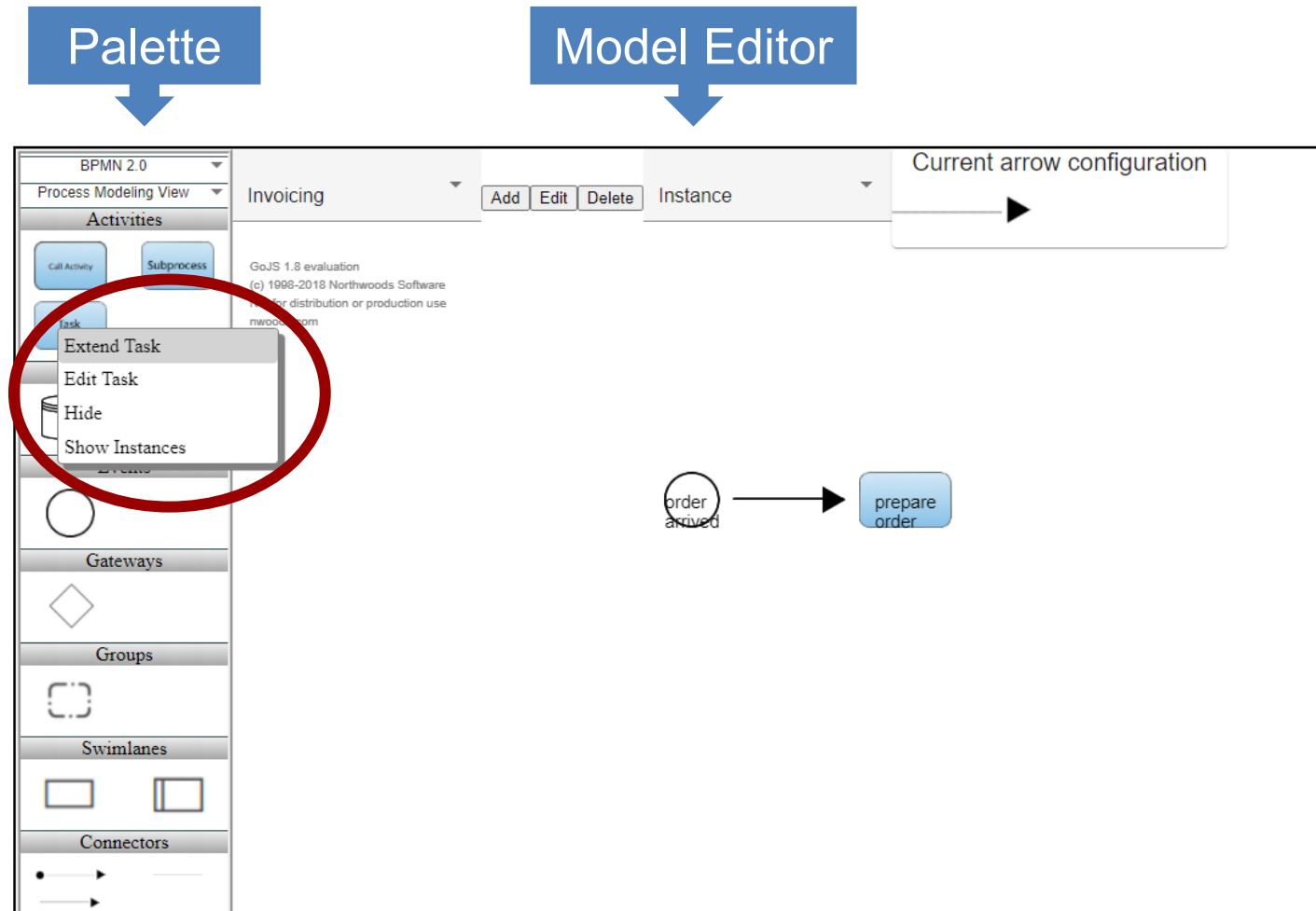
Model elements are directly created as instances in the ontology Modelling and ontology in a single environment



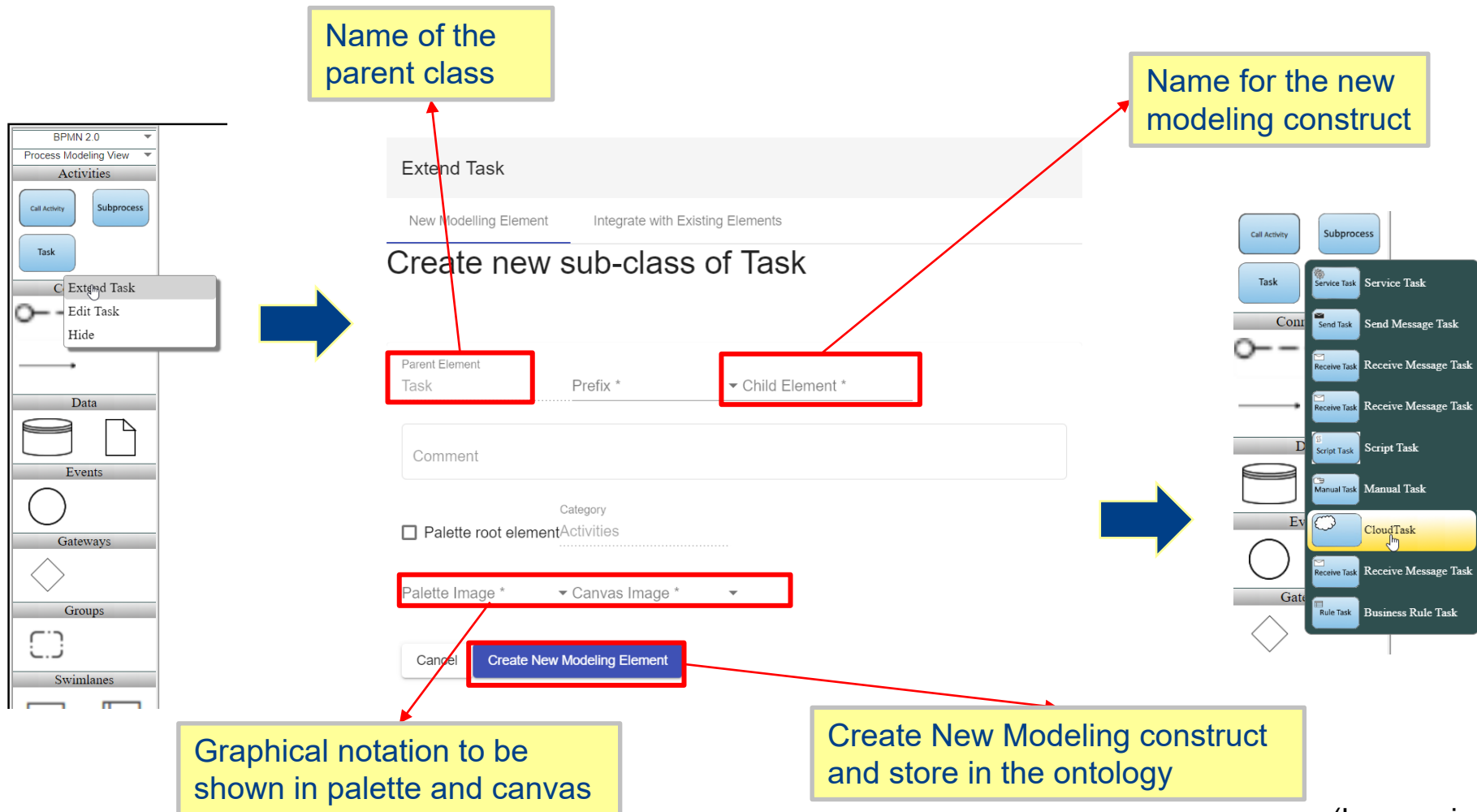
Modeling Elements are represented in a Class Hierarchy



Extending AOAME Modeling Languages – on the fly



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



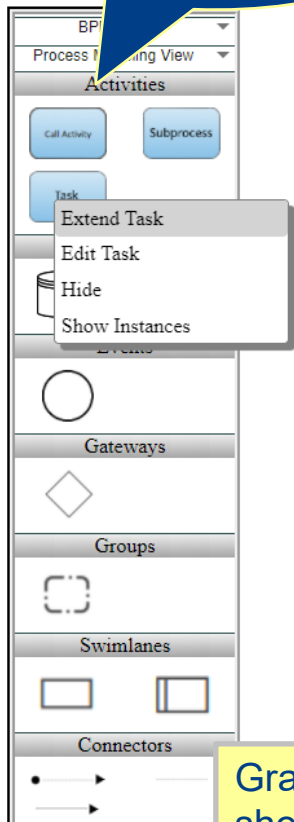
Integration of Meta-modeling and Modeling: On-the-fly Modeling Language Adaptation

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

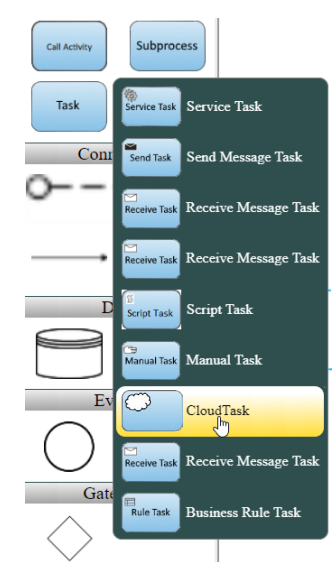
Parent Element: Task Prefix * Child Element *

Comment

Category: Palette root element Activities

Palette Image * Canvas Image *

Cancel **Create New Modeling Element**



Graphical notation to be shown in palette and canvas

Create New Modeling construct and store in the ontology

Semantic Alignment in AOAME

- With Semantic Mapping modeling elements can be connected to domain ontology

Edit

CloudTask Datatype Bridging Connector **Semantic Mapping**

Edit CloudTask

Prefix: bpmn New Label *: CloudTask

Comment

Palette Image (thumb...): Cloud Task Canvas Image *: From Arrow To Arro

Arrow Stroke

Cancel Save

Relations for CloudTask

Create New Relation

Create new ObjectProperty

Label *: paymentplan

bpaas:PaymentPlan

Create New Domain Element

Create Relation

Cancel Ok