Ontology-based Enterprise Modeling for Human and Machine Interpretation

Prof. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Image by [Pexels]() from [Pixabay]()

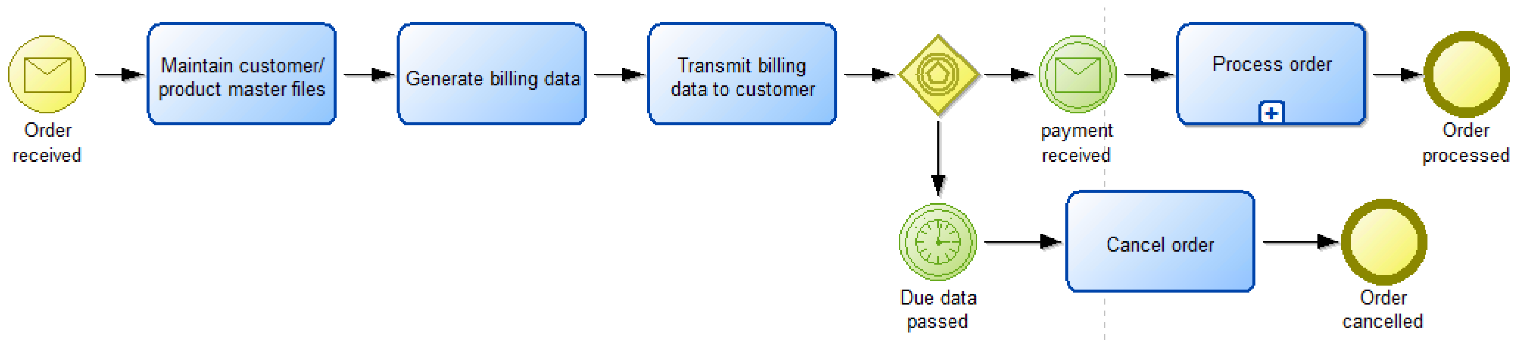
Why Modeling

- If the object you want to create or change is simple, then you can do it directly.
- For **complex** systems that are likely to **change** over time, you need a **model**.
- Without explicit modeling there is a *high risk that the implementation is not what is intended*

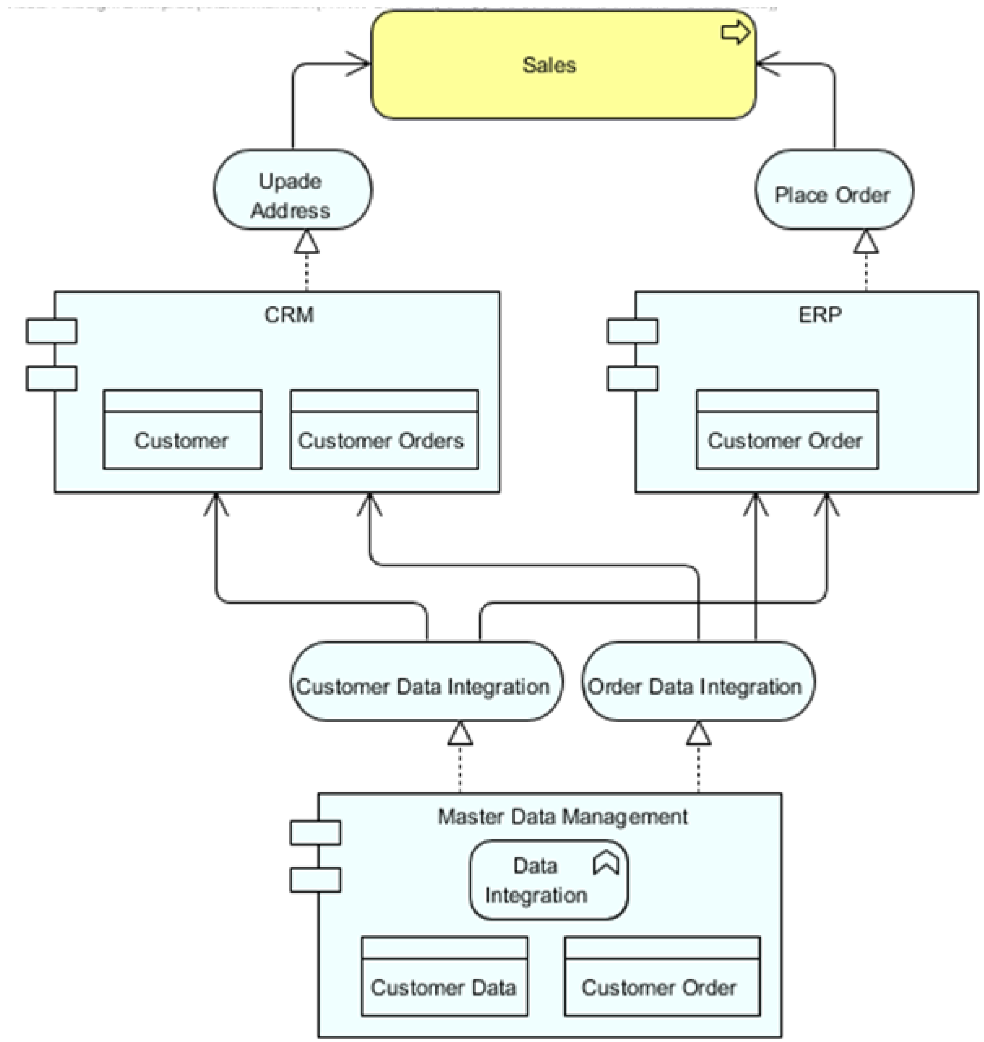
(John Zachmann, 2012)

Business Process Management

- Process Design
- Process Optimization
- Process Digitalization
- …



Enterprise Architecture

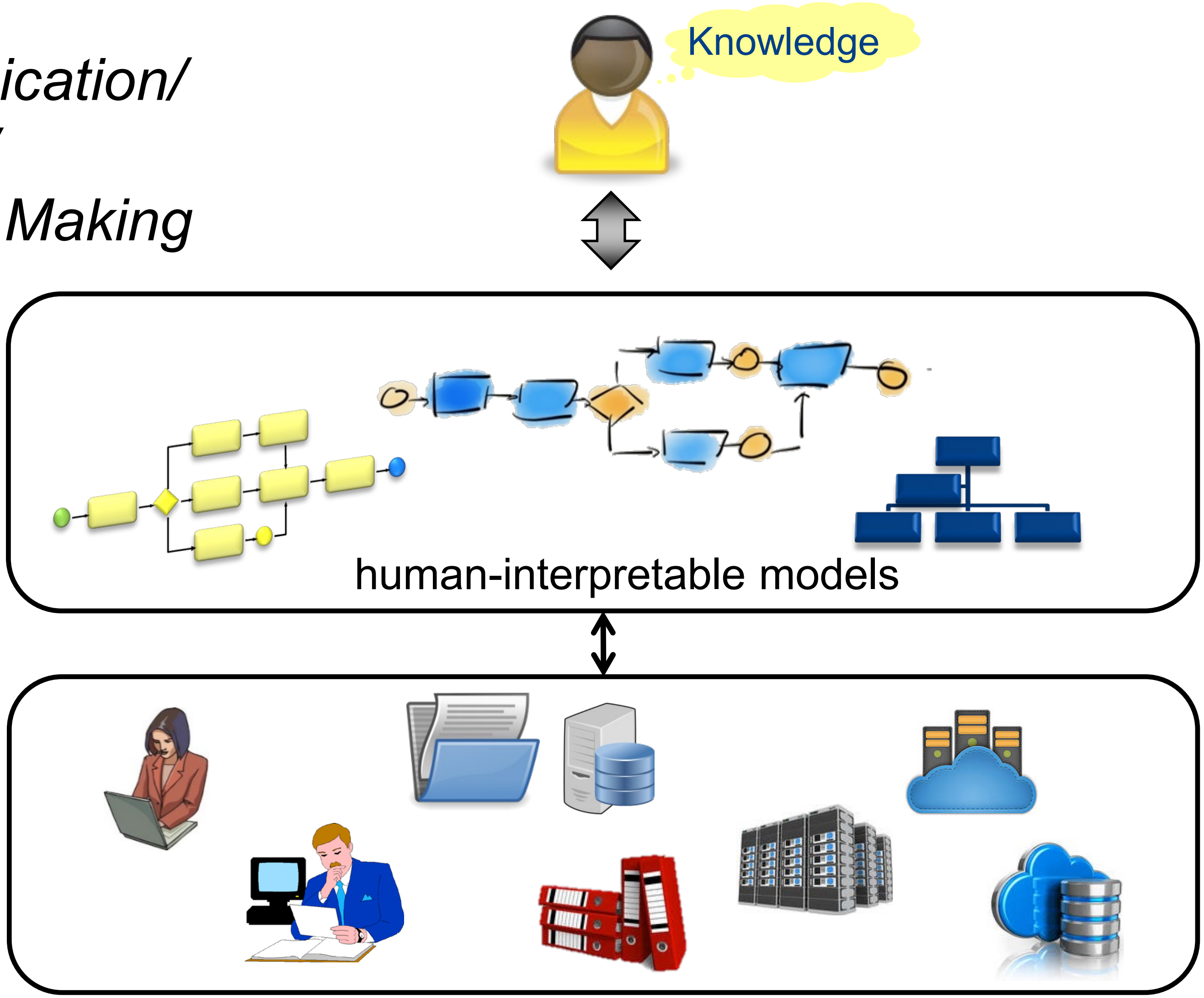


Graphical Models are appropriate for Humans

*Communication/
Analysis/
Decision Making*

Models

Reality



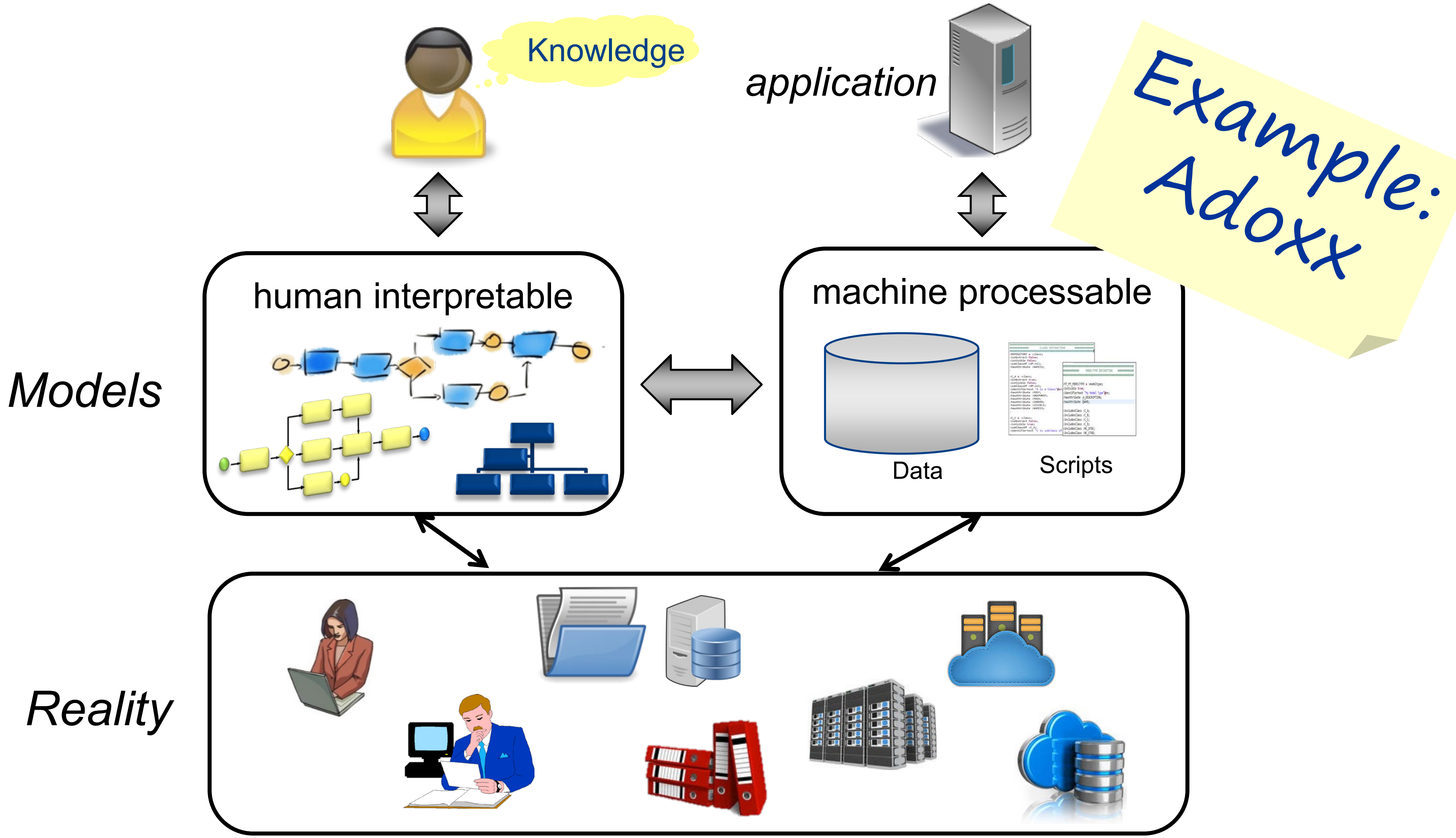
Models

- Models are not mere pictures; rather, they
  - provide a precise, meaningful description that can be visualized in different ways for different stakeholders;
  - can also be used to analyze the impact of changes, cost, risk, security, compliance and other relevant KPIs.

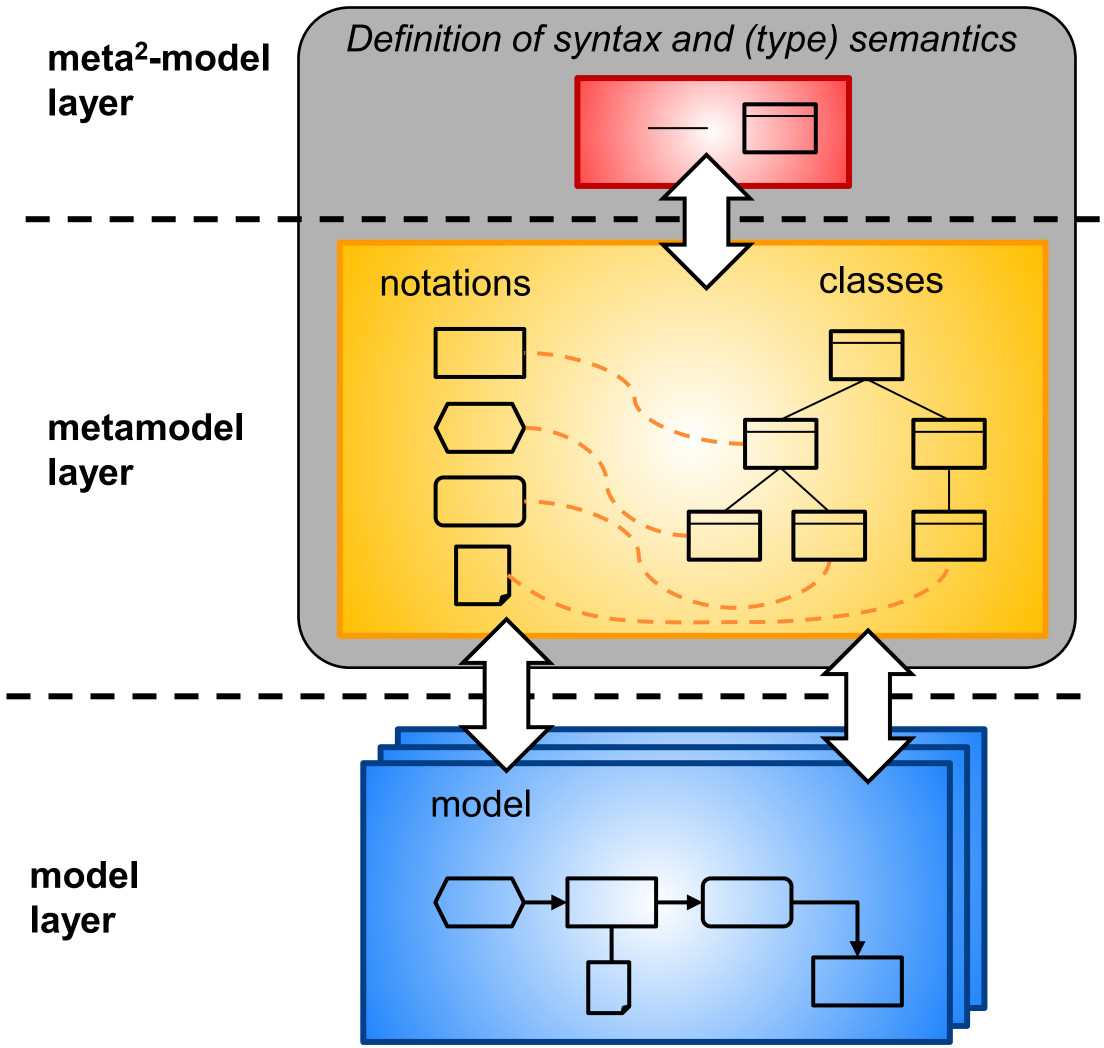
http://blog.bizzdesign.com/how-to-not-fail-when-implementing-strategy

Models should allow automated analysis, decision making and digitalization

Graphical Models are Represented in a Database

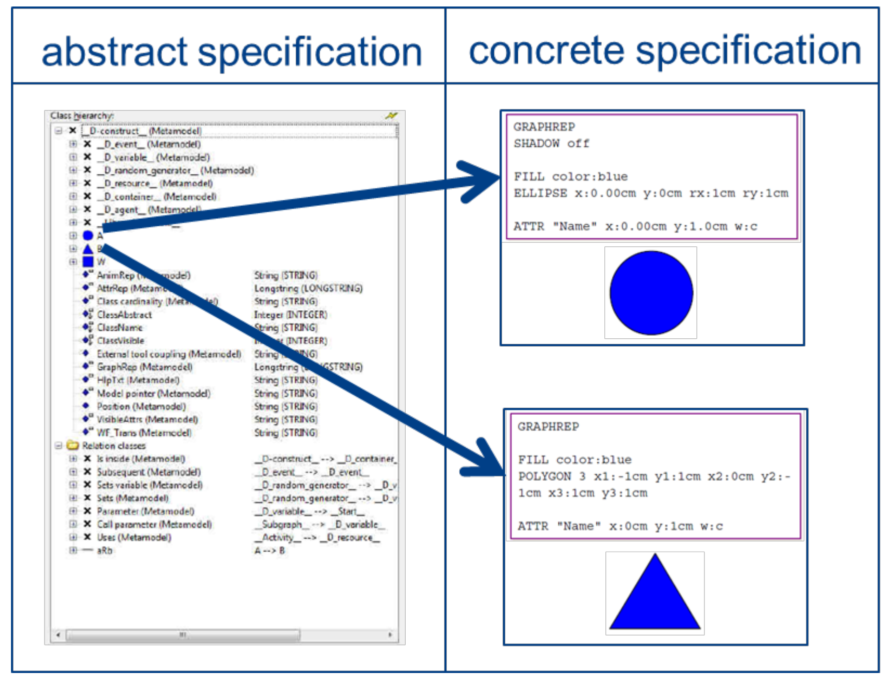


Modelling Environment



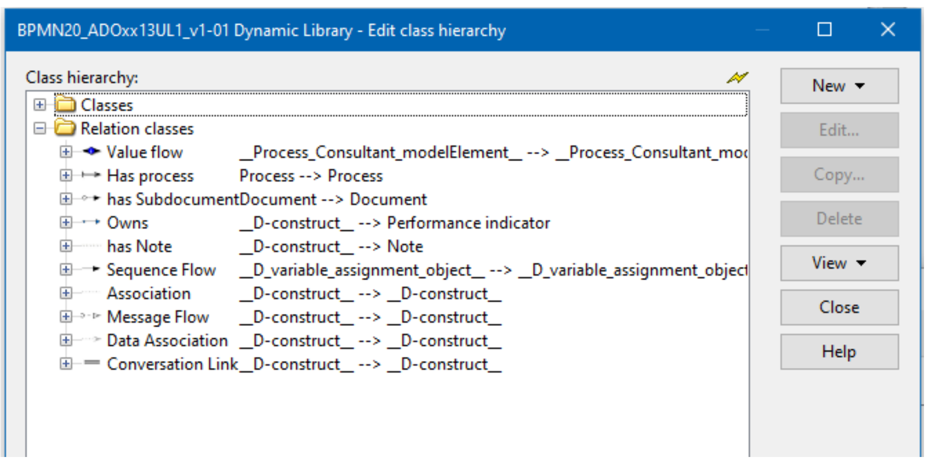
Metamodel: Semantics and Syntax of a Modeling Language

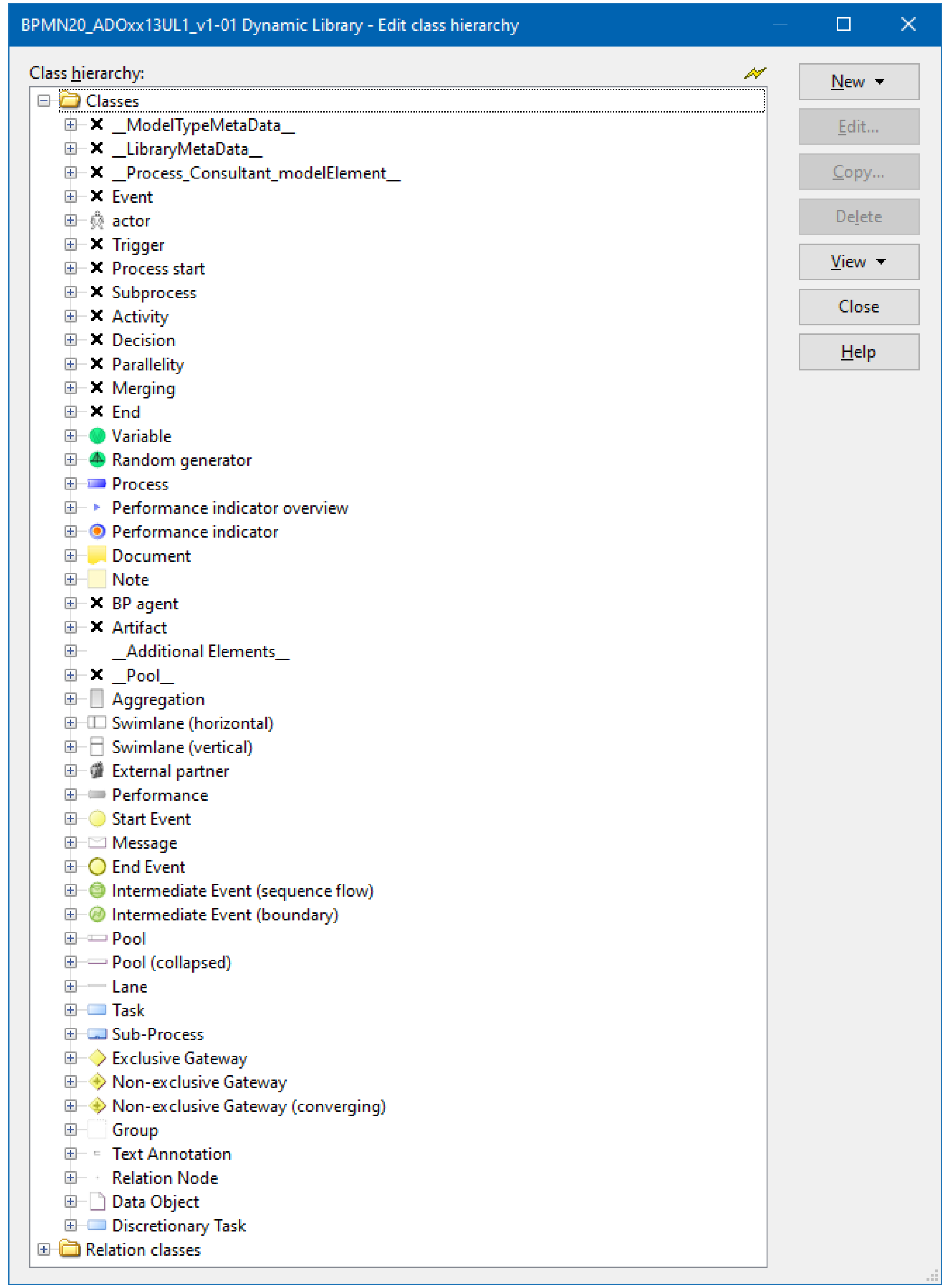
- The Semantics of a model language is defined by
  - Classes of elements and relations
  - Class hierarchy
  - Attributes of the elements
- The Syntax is defined by notation:
  - Adoxx: attribute GraphRep



Class Hierarchies

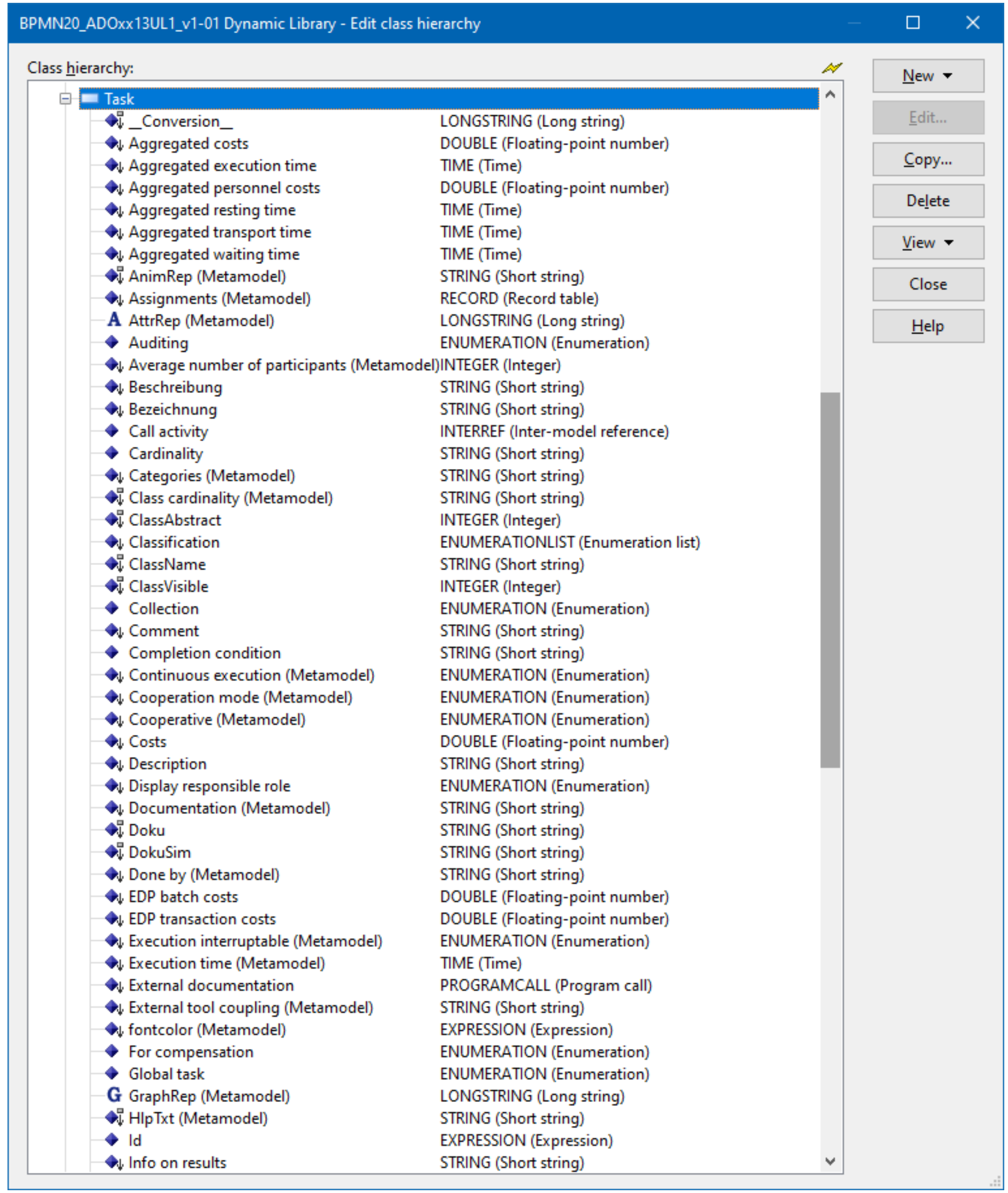
- ADOxx distinguishes
  - Classes
  - Relation classes





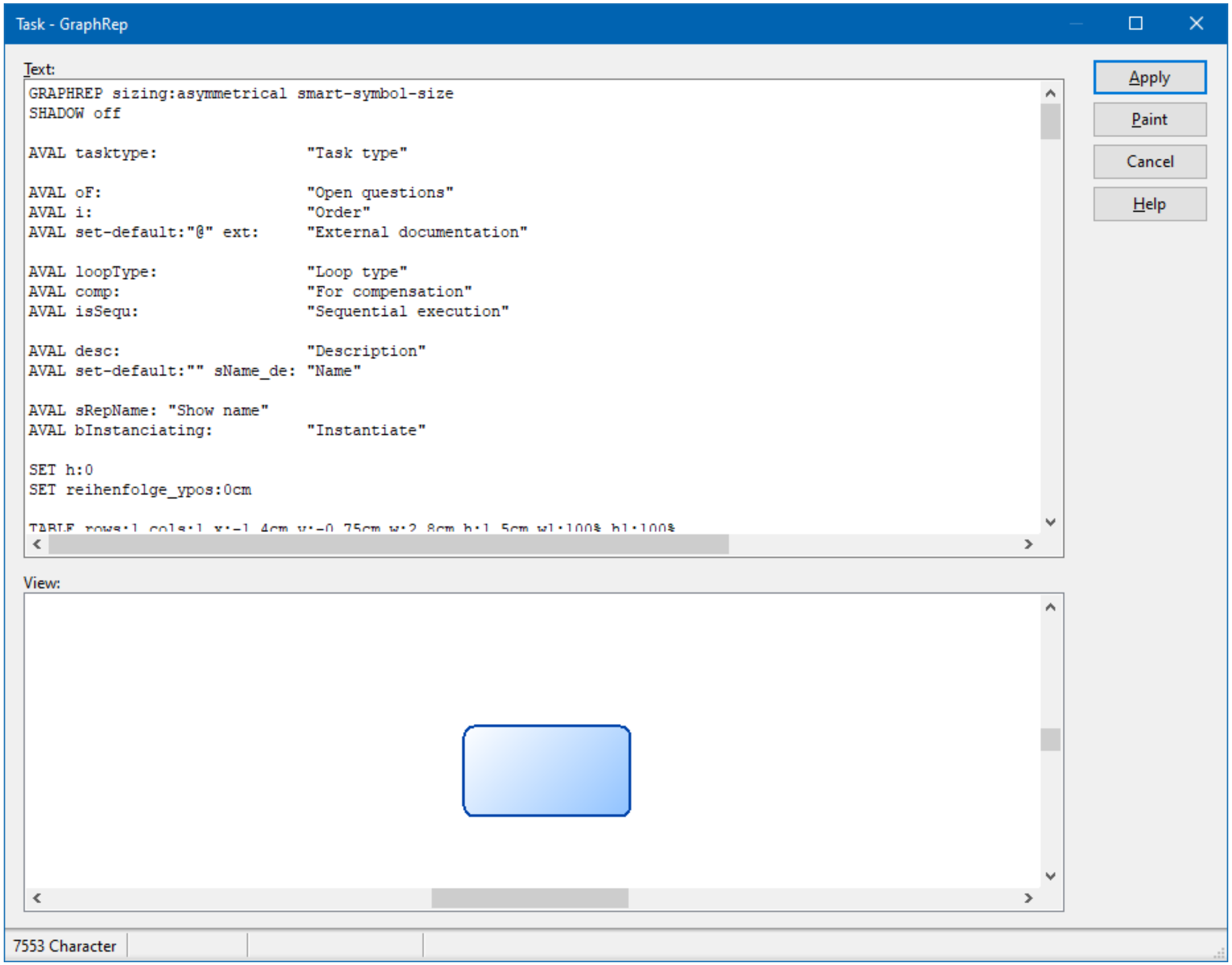
Attributes

- Kinds of Attributes
  - Properties of Models
  - Graphical Representation
  - References

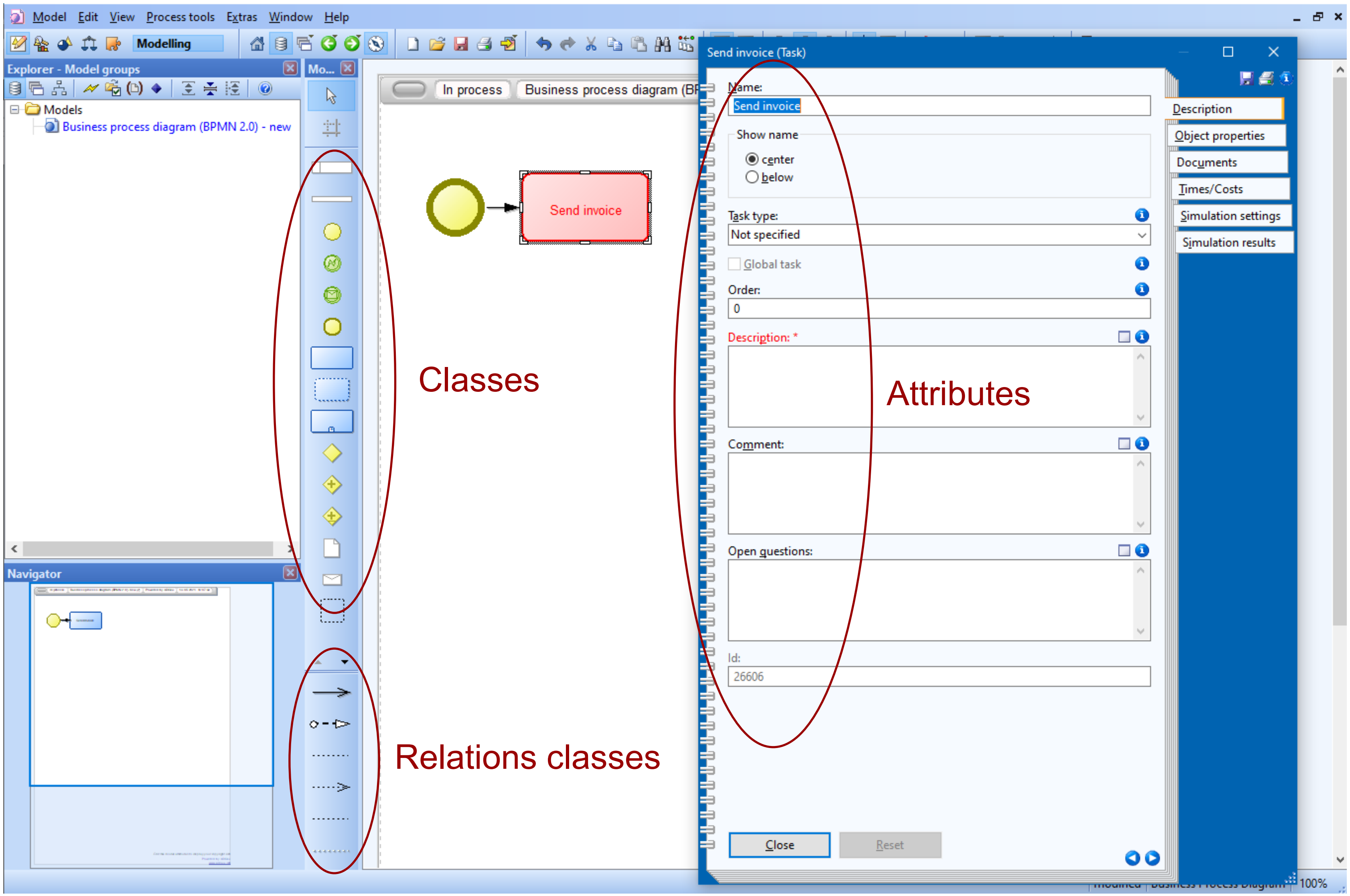


Notation

GraphRep: A script language for the graphical representation

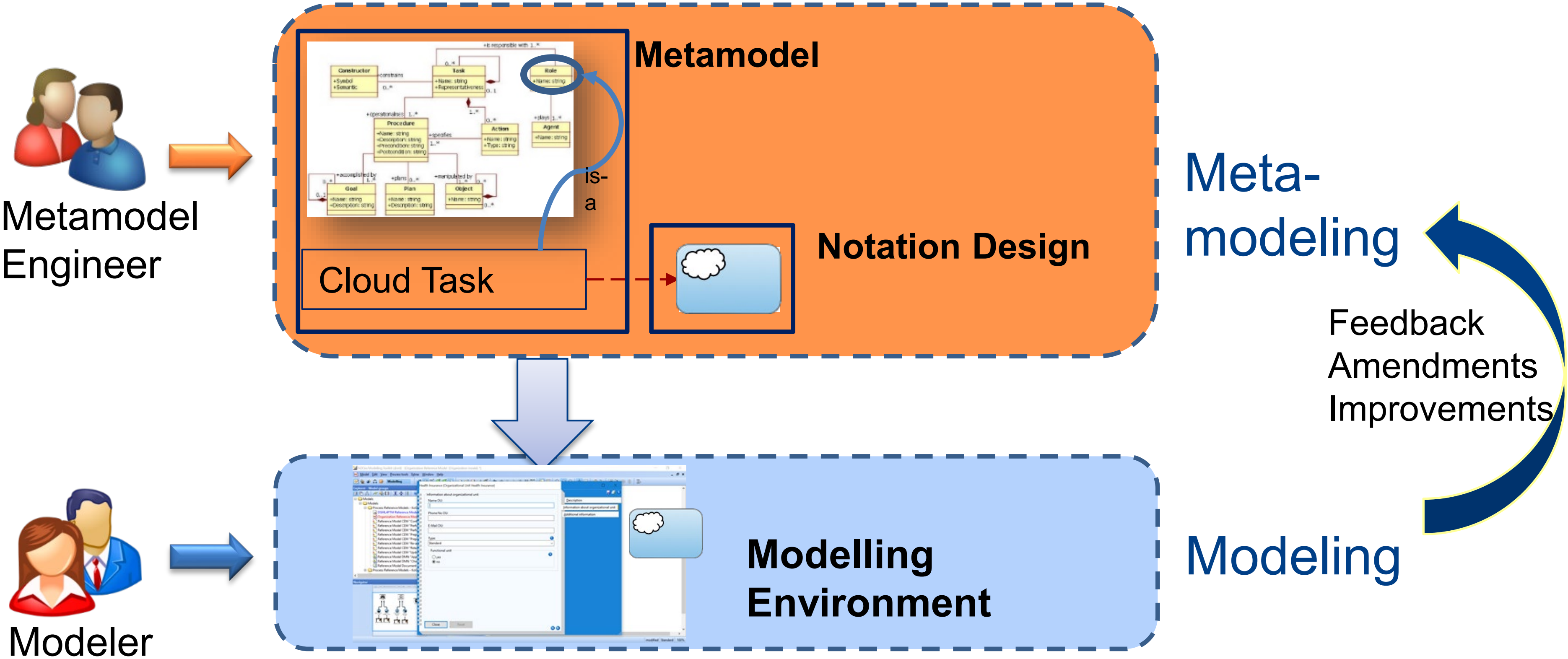


Appearance of Classes in the Modelling Toolkit

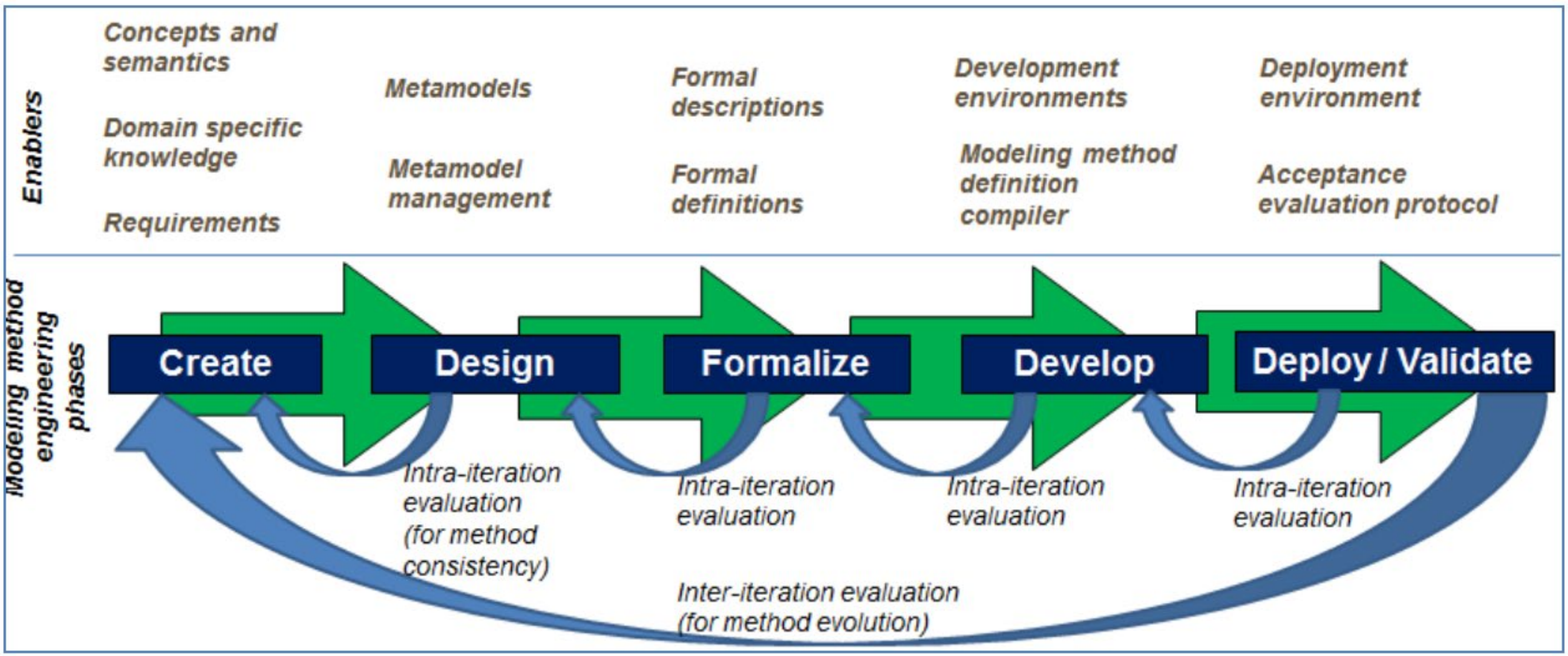


Change of Metamodel

- Example: new task type Cloud Task



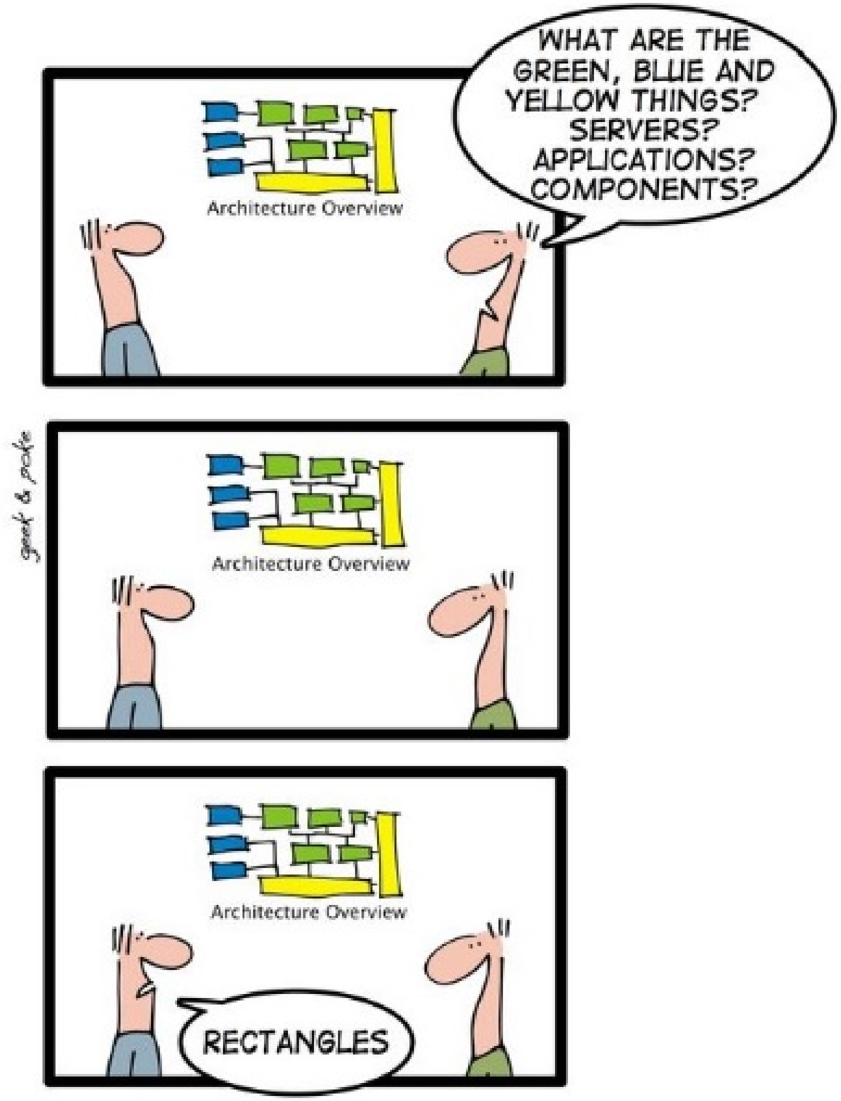
The AMME LifeCycle
Agile Modeling Method Engineering



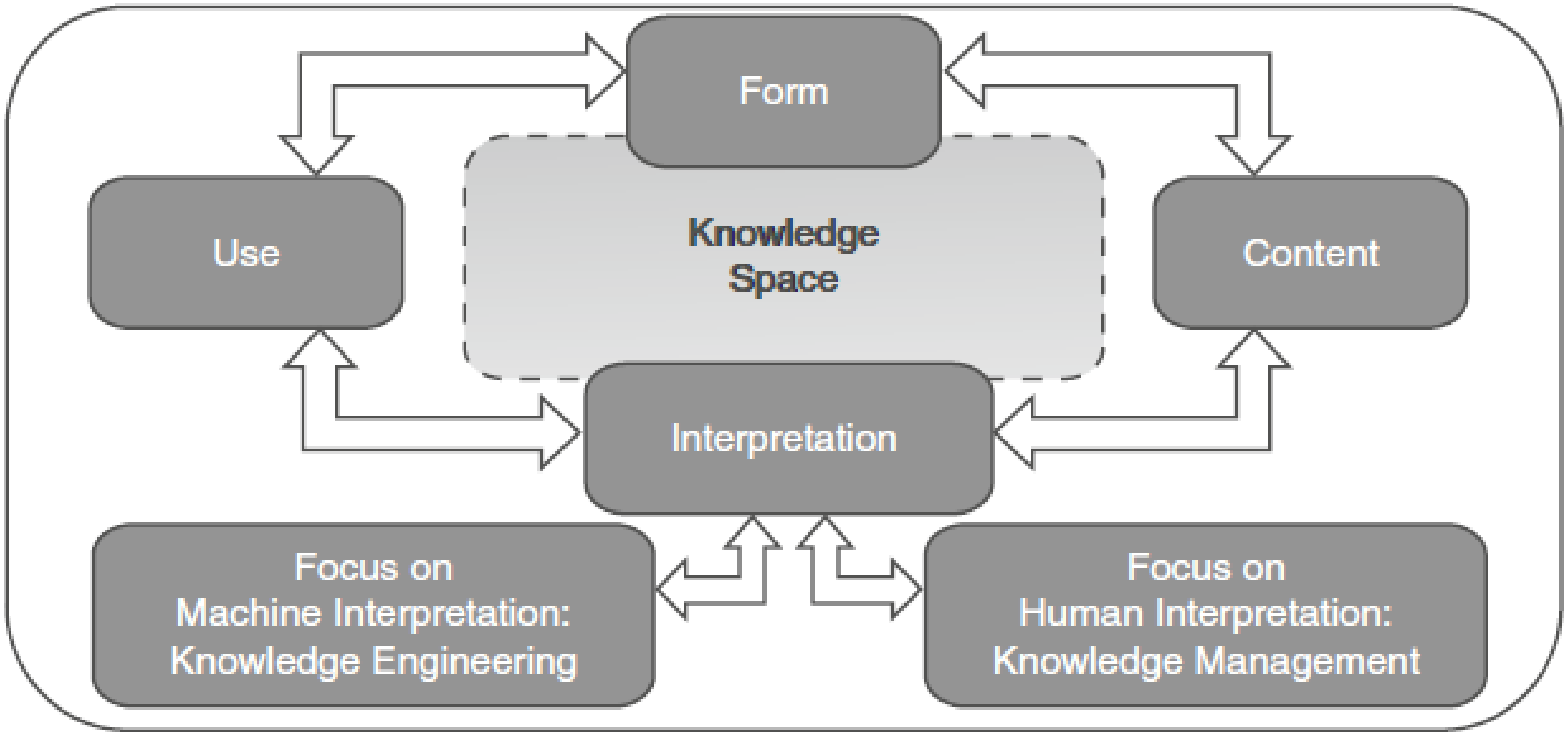
(Karagiannis 2015)

Knowledge in Models

Interpretation of Models

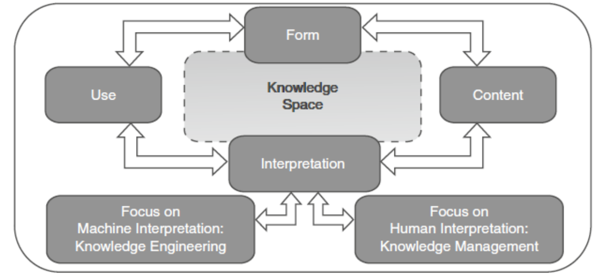


Dimensions of a Knowledge Space



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

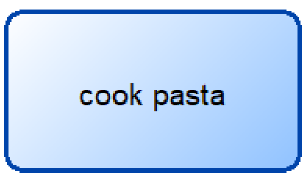
Content: Instantiation of the model elements

- ***Use***: Stakeholders and their concerns determine the relevant subset of the knowledge
- ***Form***: Syntax and semantic of **modeling language elements**.
- ***Content***: **Domain** in which knowledge engineering is applied, is represented in the labels
- ***Interpretation***: Giving meaning to a model:
  - Graphical models are cognitively adequate for human
  - Machines need more formal representation

Making the Knowledge in Models explicit

- Humans «know» the meaning of the modeling objects.
  - Elements of the model language
  - Labels represent domain knowledge
- Examples:

♦Model element: Application Compontent
♦Domain: «ERP System» is business software

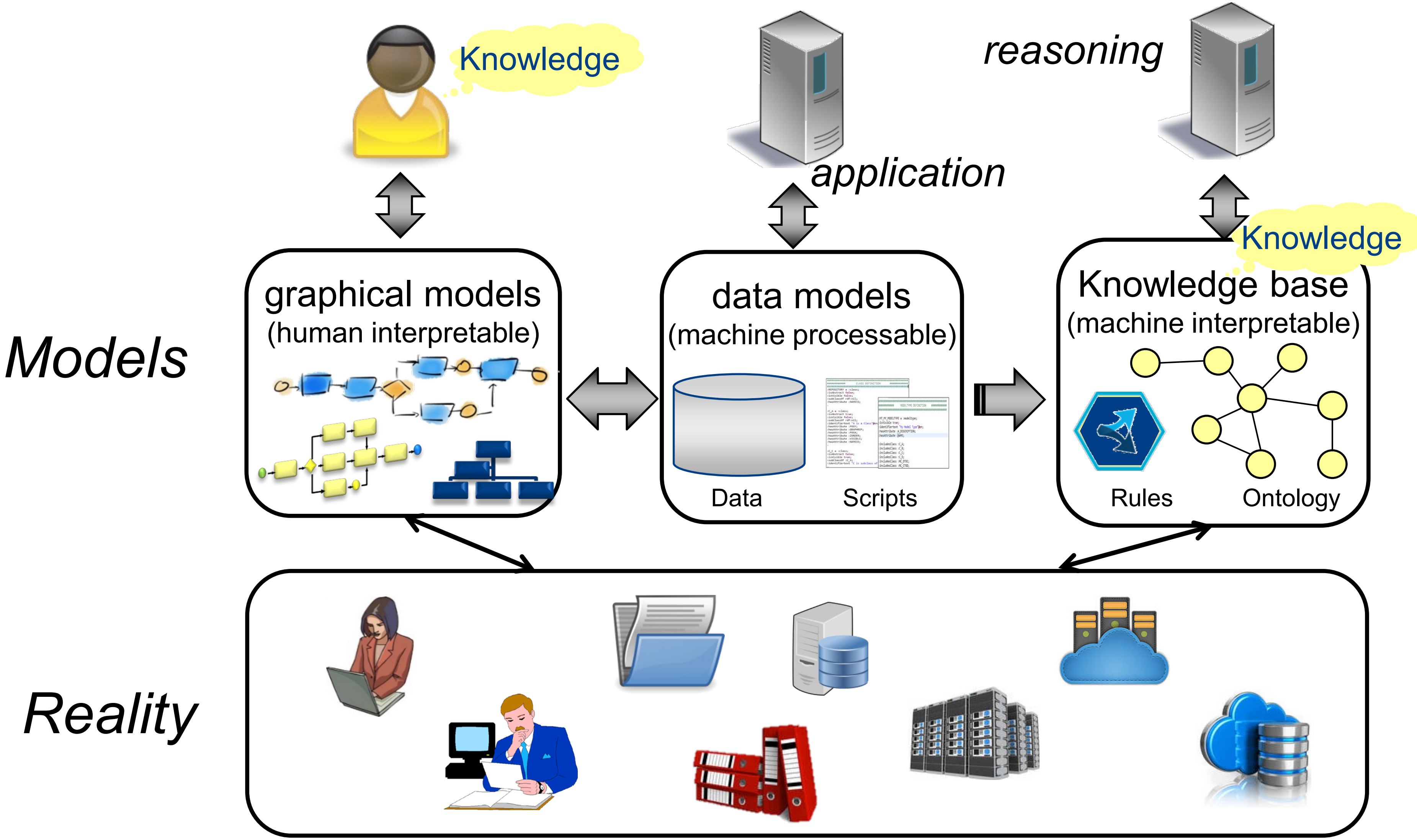


♦Model element: Task
♦Domain: «Cook pasta» is about preparing food

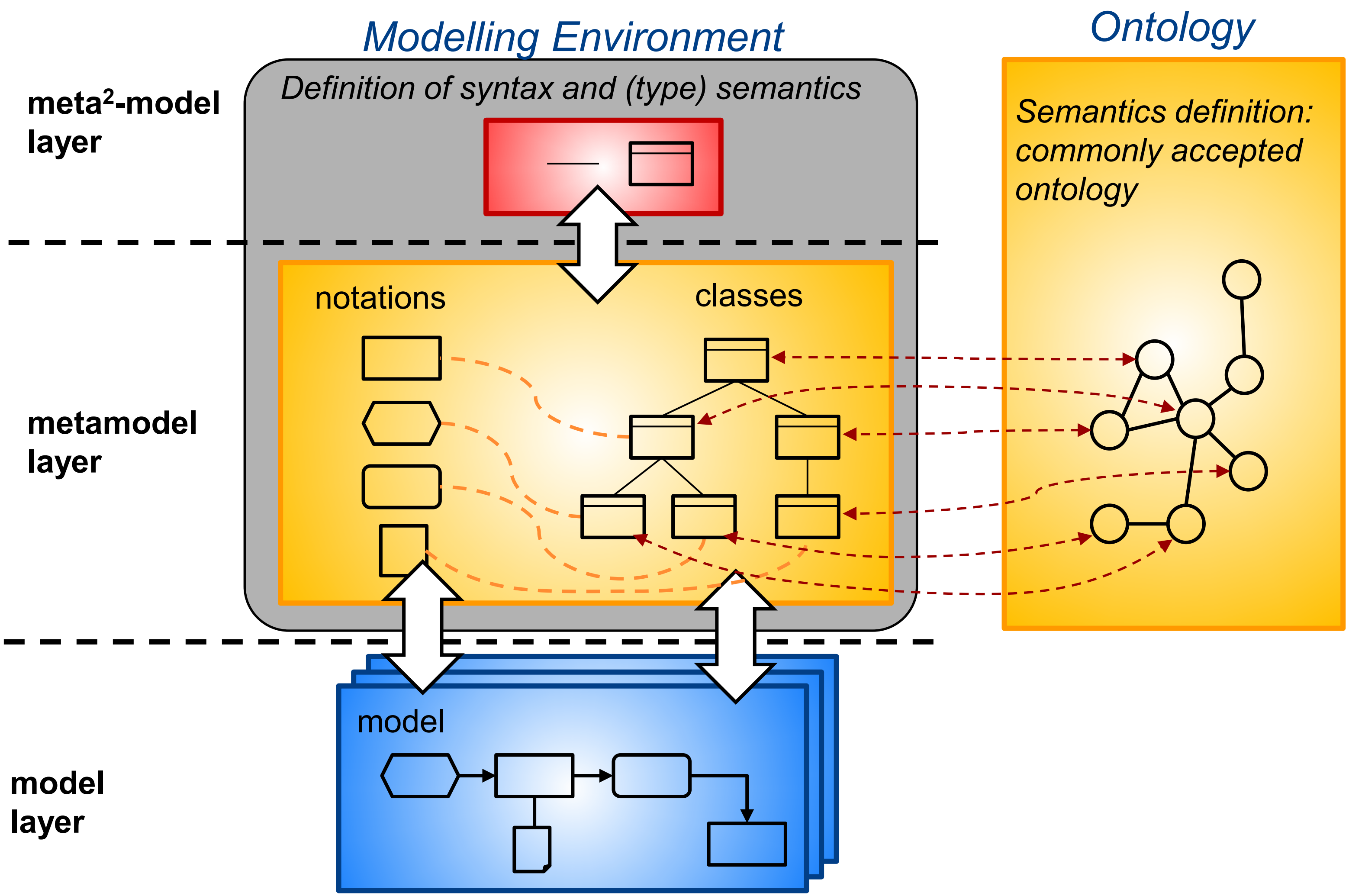
- The objective is to represent the knowledge so that it can be interpreted by a system for decision making and problem solving

Semantic Lifting

Semantic Lifting: Map Models into an Ontology



Semantic Lifting: Map Models into an Ontology

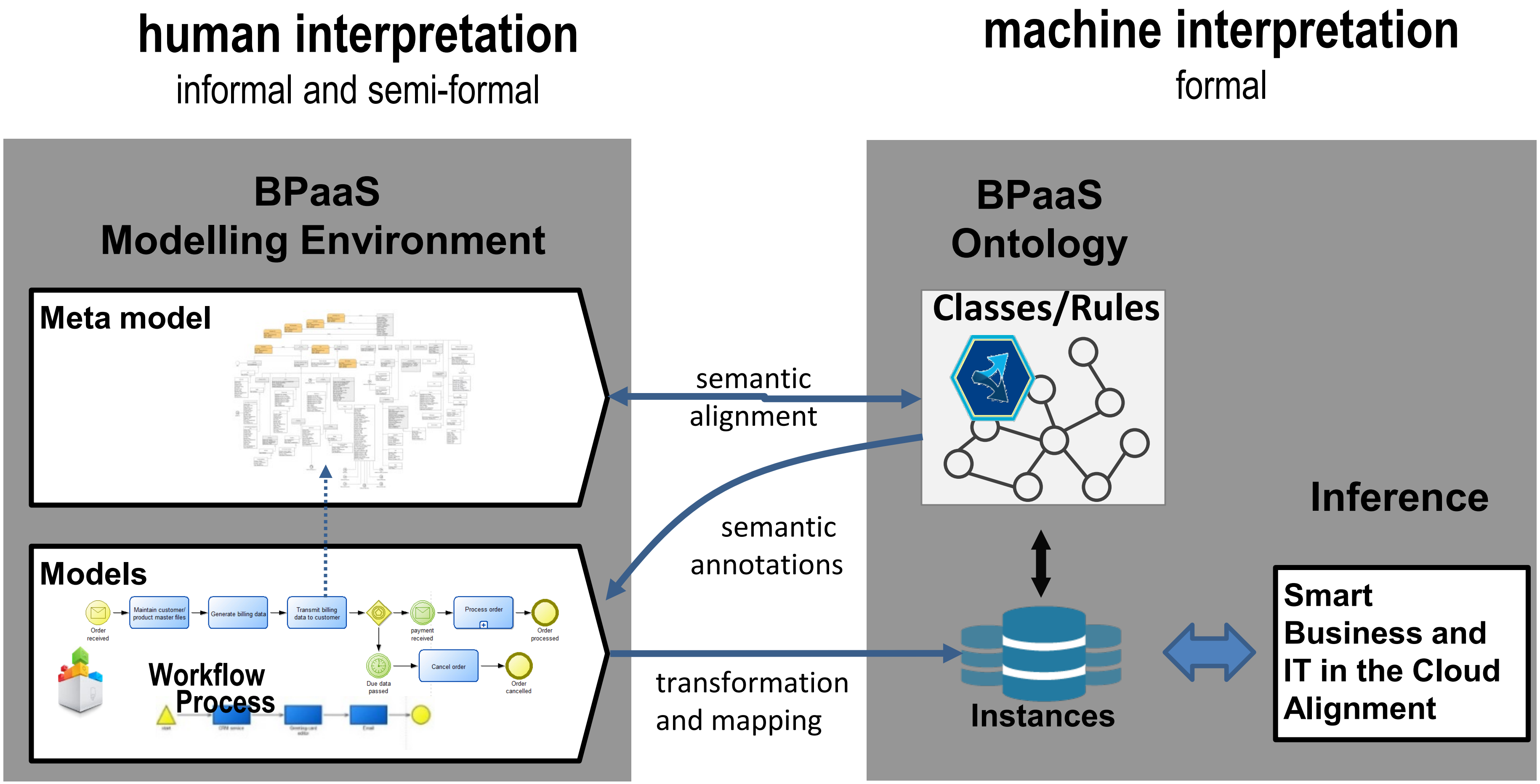


ontological metamodelling (lifting): *explication of type semantics*

Semantic Lifting

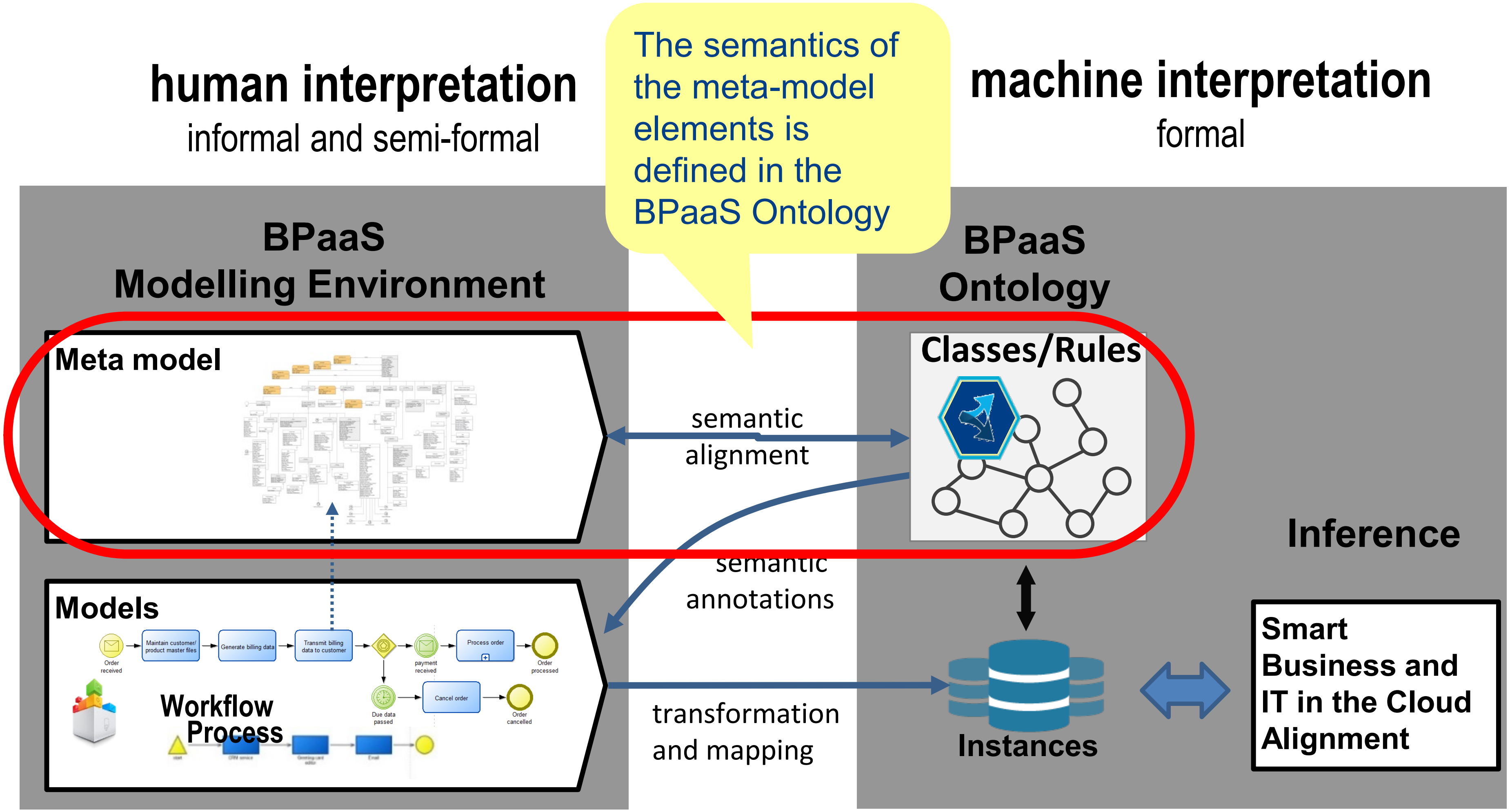
- Map models into an ontology
  - Semantics: Classes of the metamodel are aligned with classes in the ontology
  - Interpretation: For each element in a model an instance of the ontology is created
  - Content: Model elements are annotated with domain knowledge from ontology
  - Inference of the ontology can be applied to the knowledge base

Example: Business Process as a Service



From: CoudSocket Project

Example: Business Process as a Service



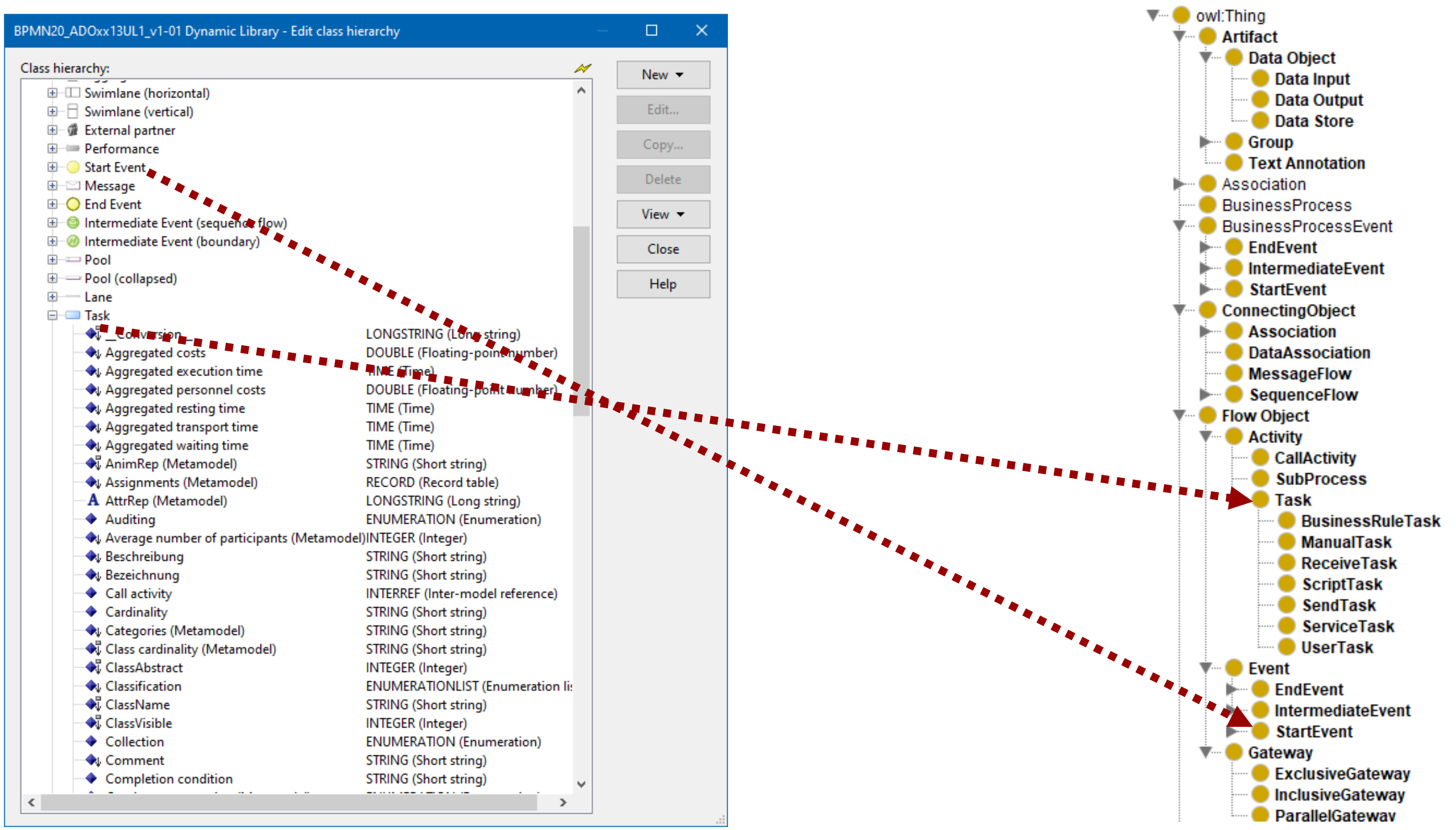
From: CoudSocket Project

Semantic Alignment

The ontology contains classes for all modelling elements

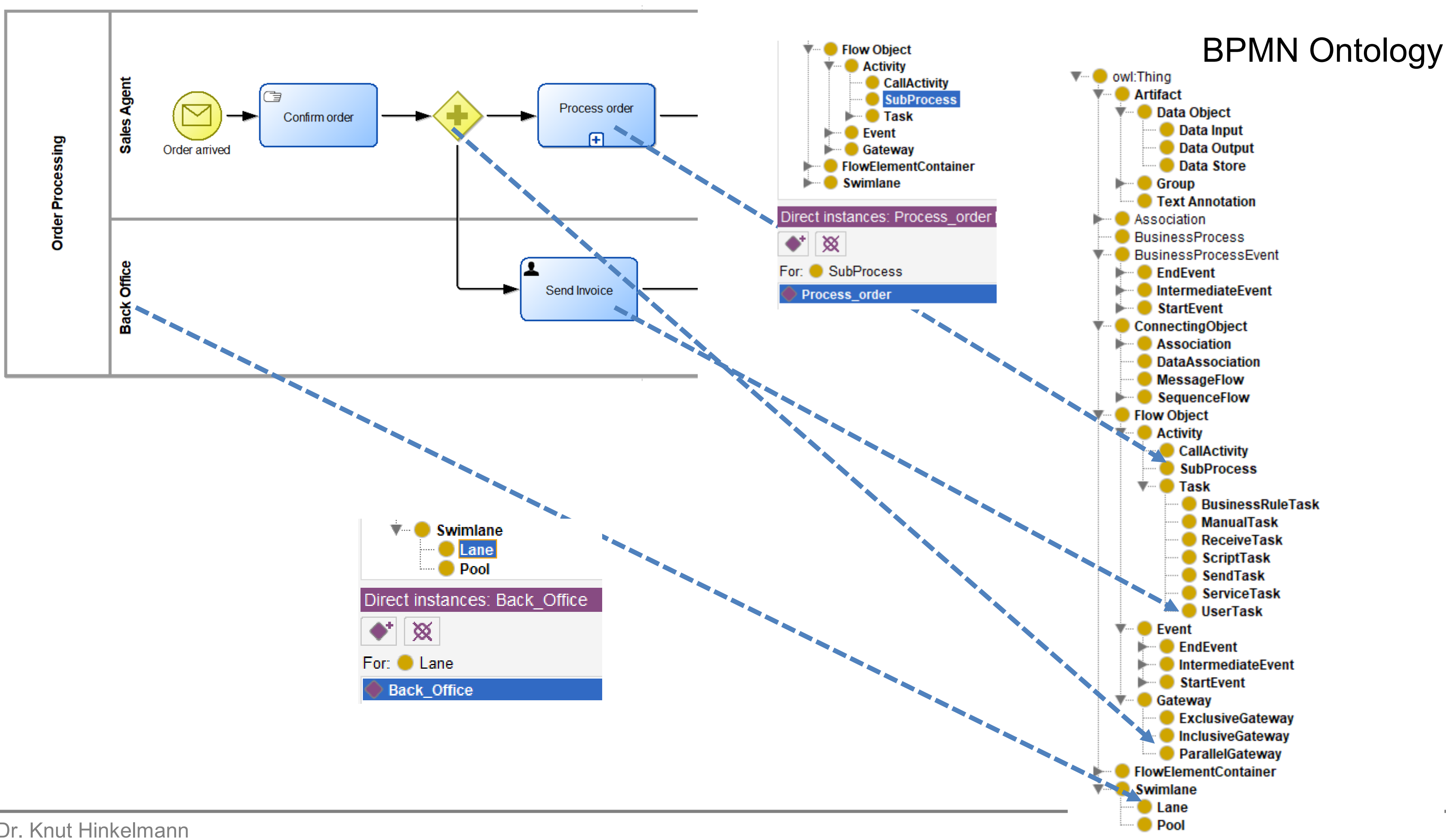
BPMN Modelling Language in ADOxx

BPMN Ontology



Transformation and Mapping

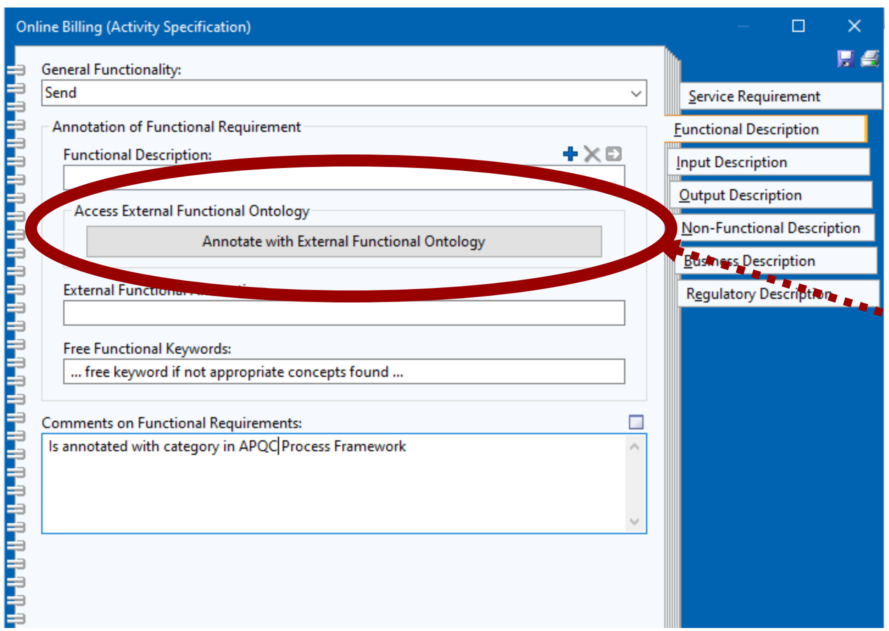
The model elements are exported as instances ontology classes



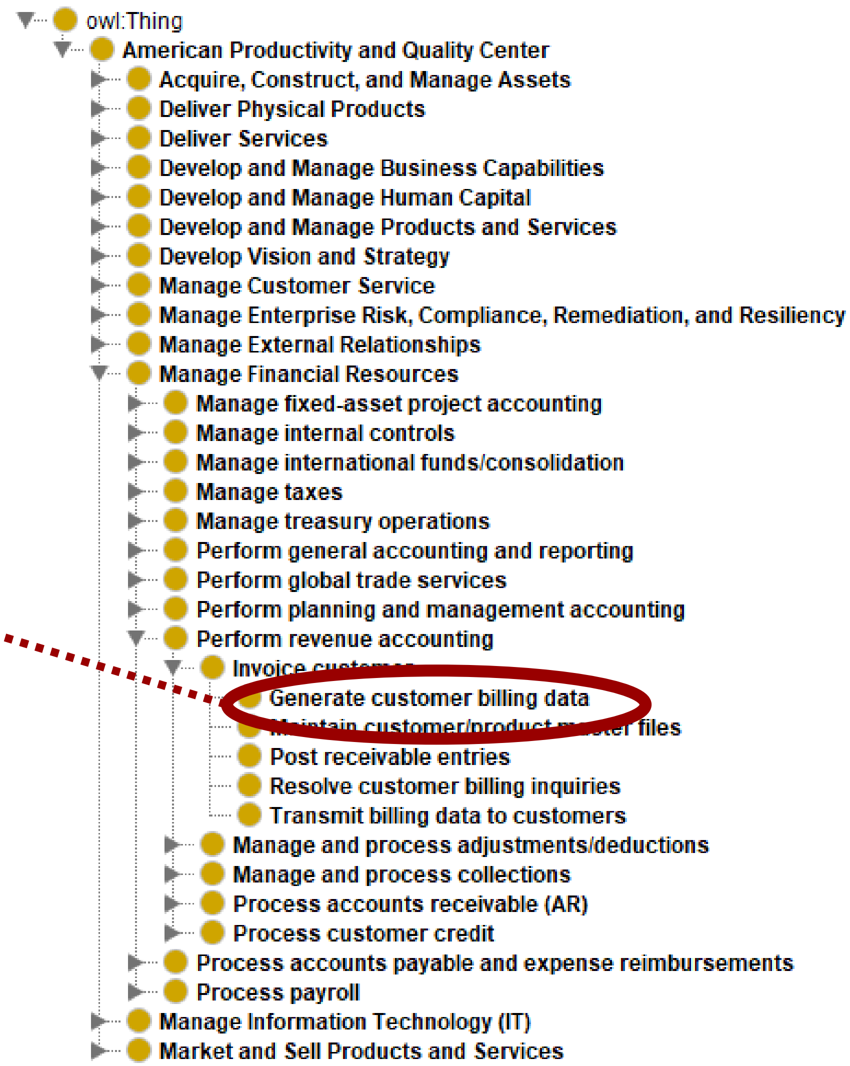
Semantic Annotations

Annotate modeling elements with classes from the domain ontology

Example: Functionality of a Service



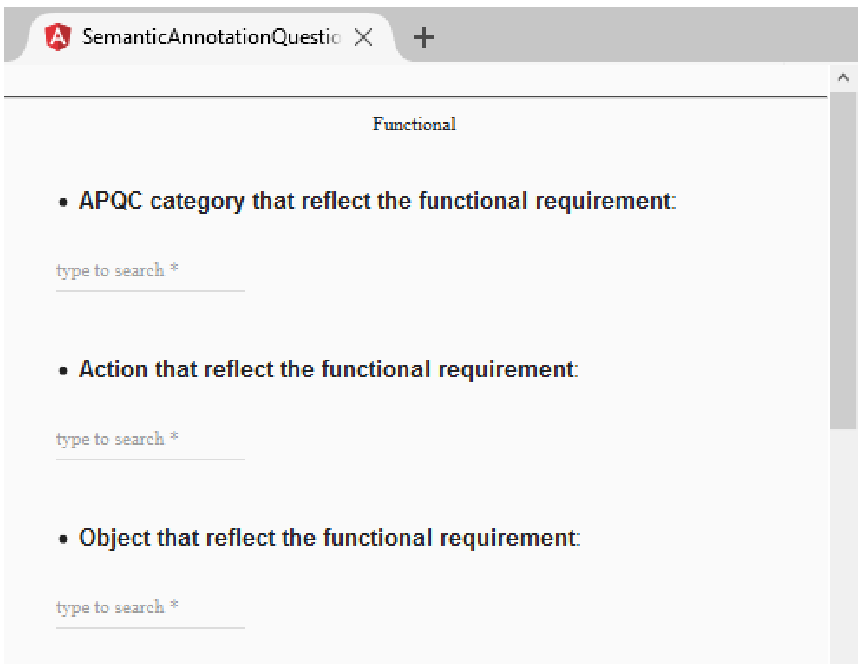
Domain Ontology:
APQC Process Classification Framework



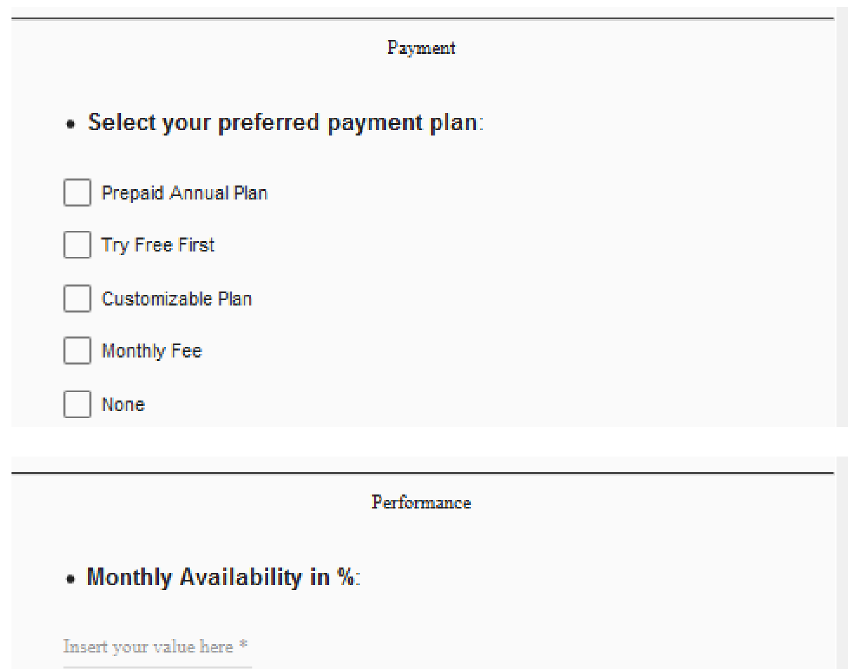
Application Example for Semantic Lifting

Cloud Service Selection

Functionality



Non-functional requirements



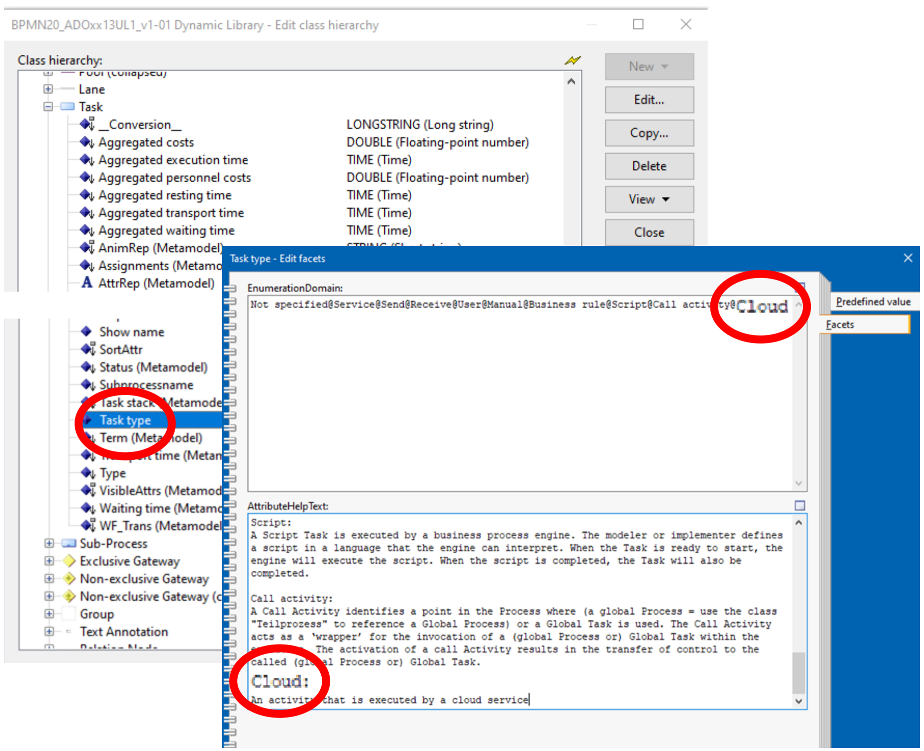
Drawbacks of Semantic Lifting

- 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

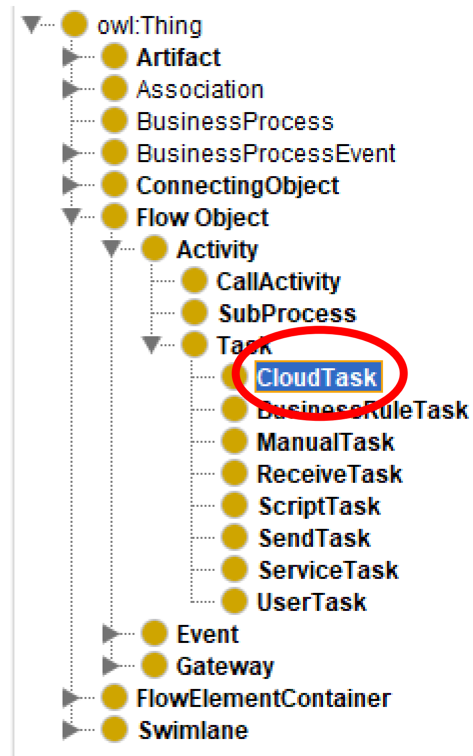
Example: New Model Element

- New task type: Cloud Task

Change in the meta model:

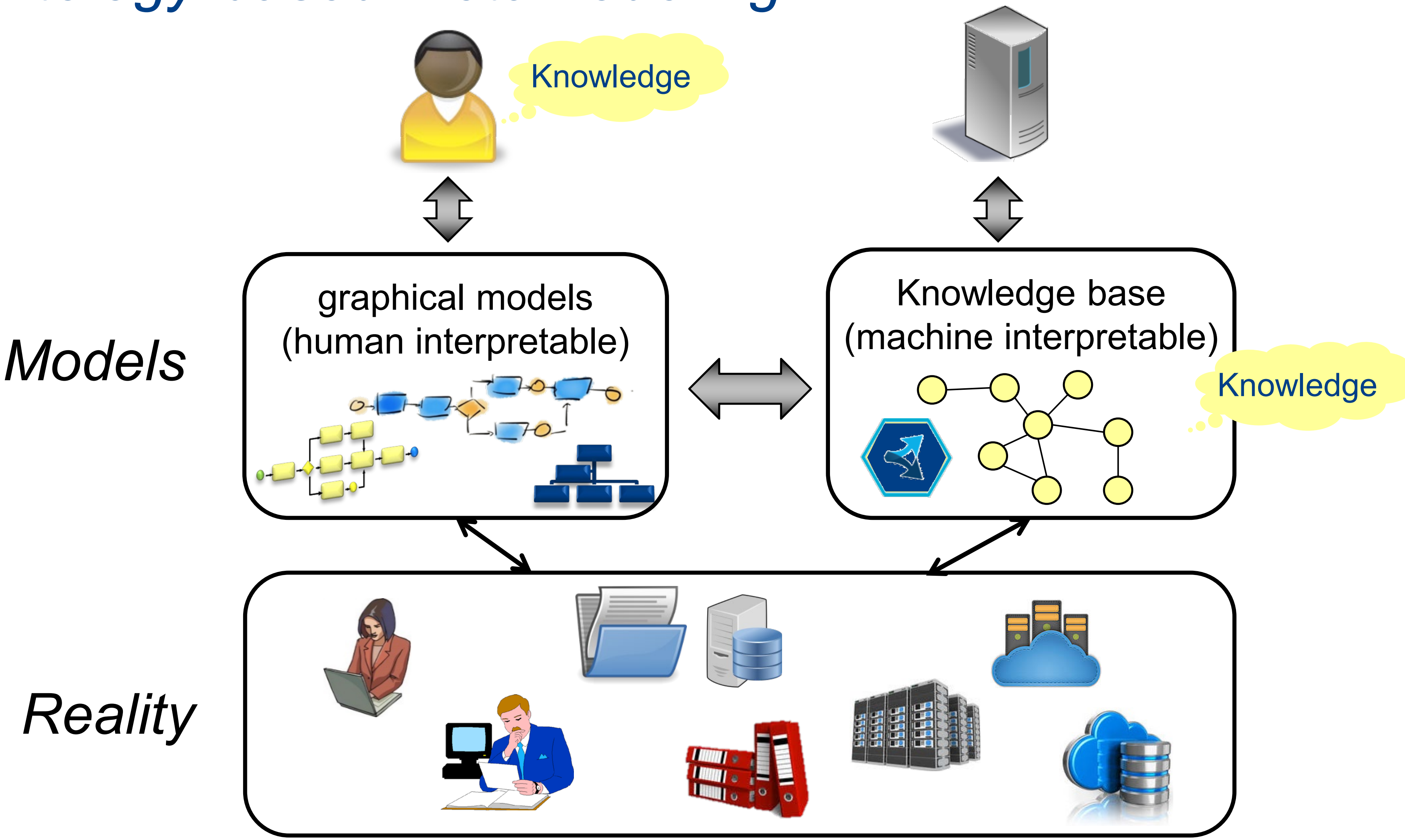


Change in the ontology:

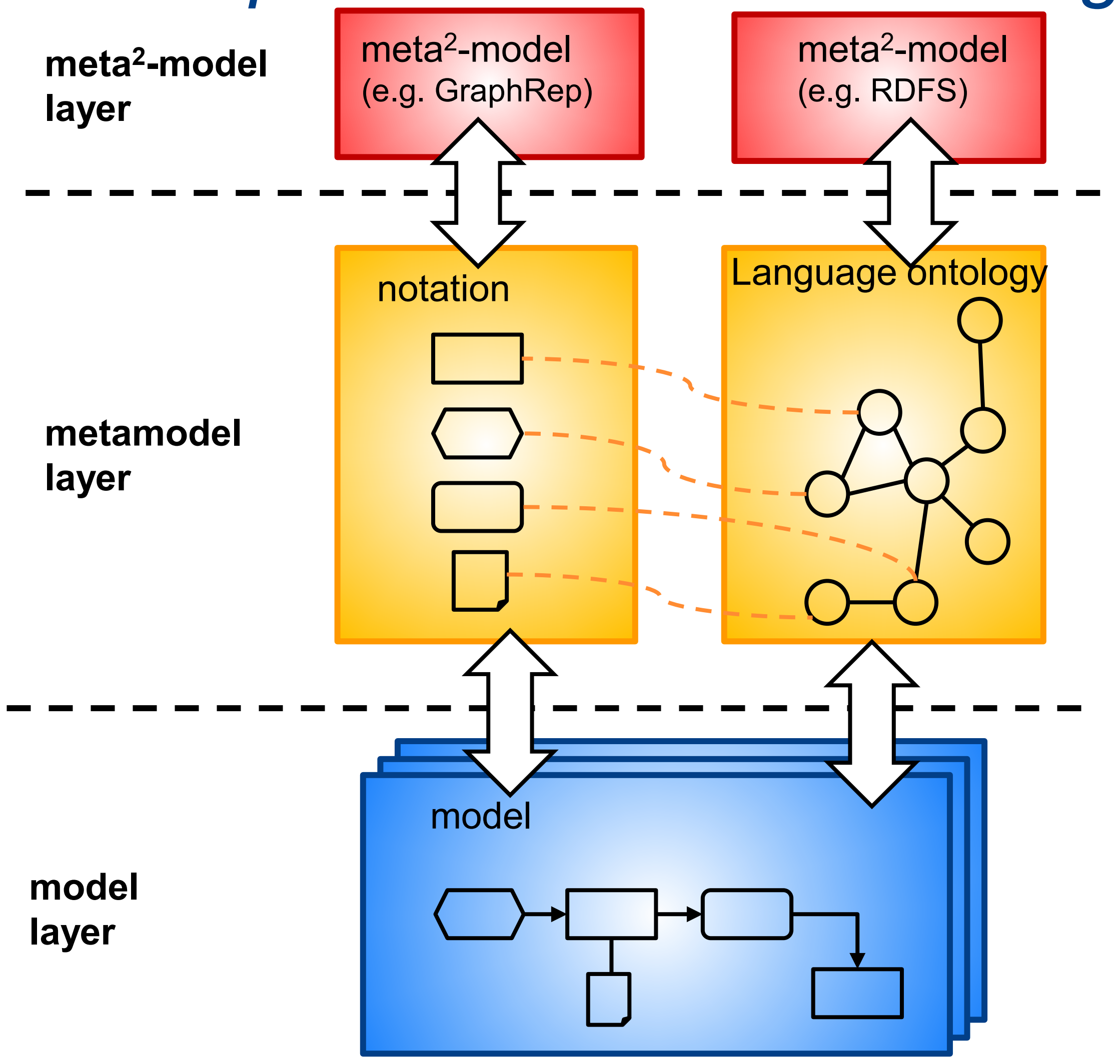


Ontology-based Metamodelling

Ontology-based Metamodeling



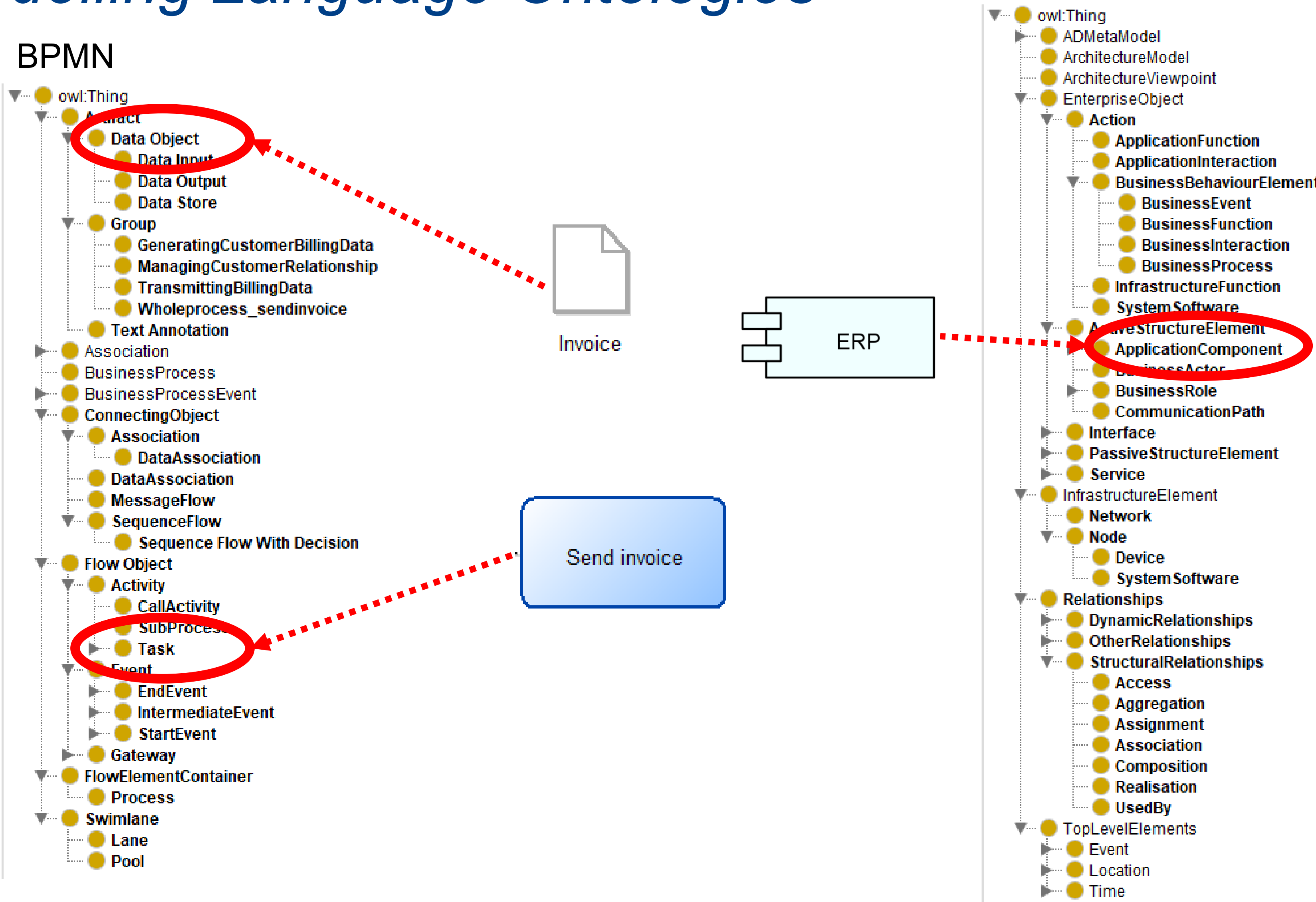
Ontology-based Metamodeling (1): Metamodel is represented as an Ontology



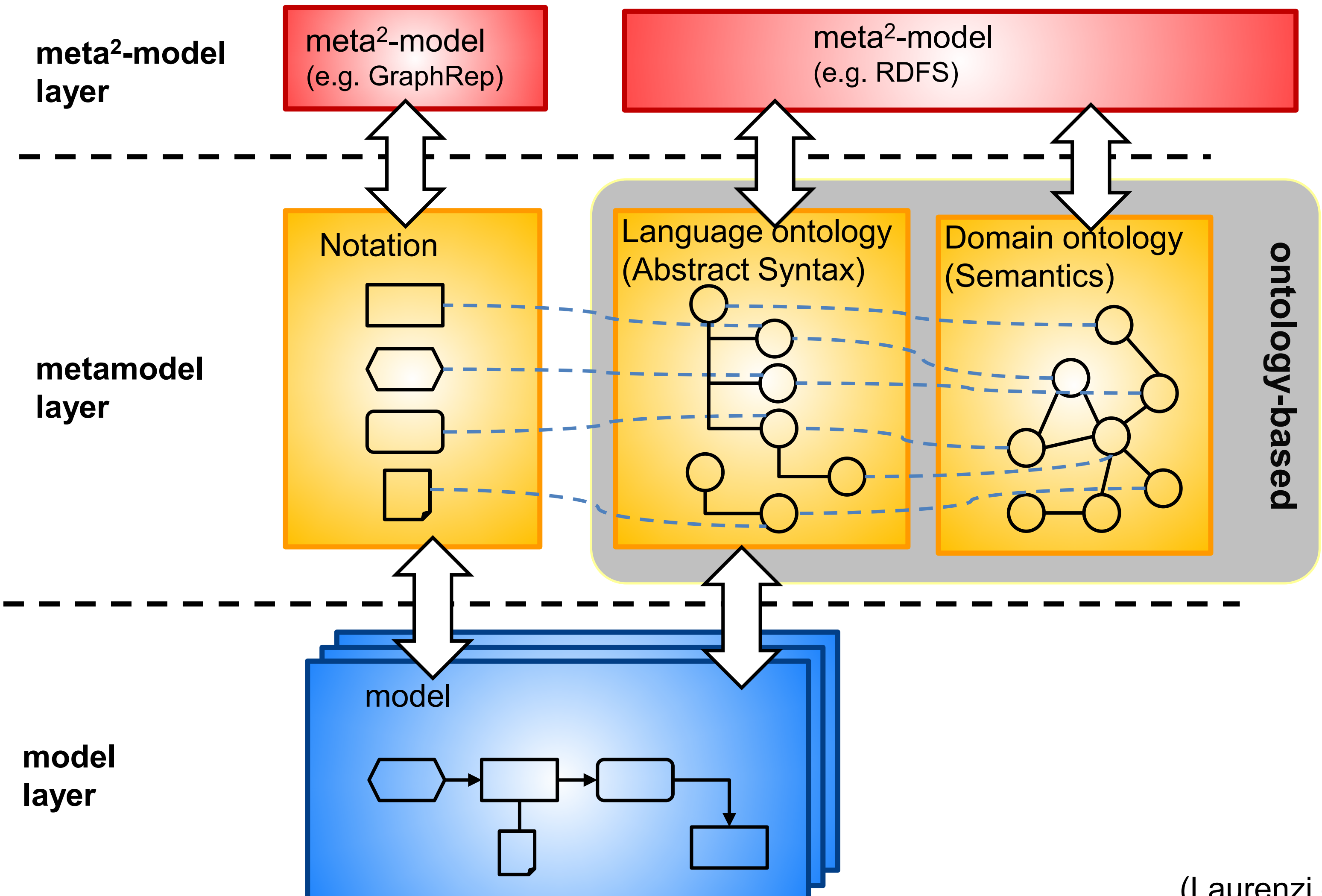
Modelling Language Ontologies

BPMN

Archimate



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

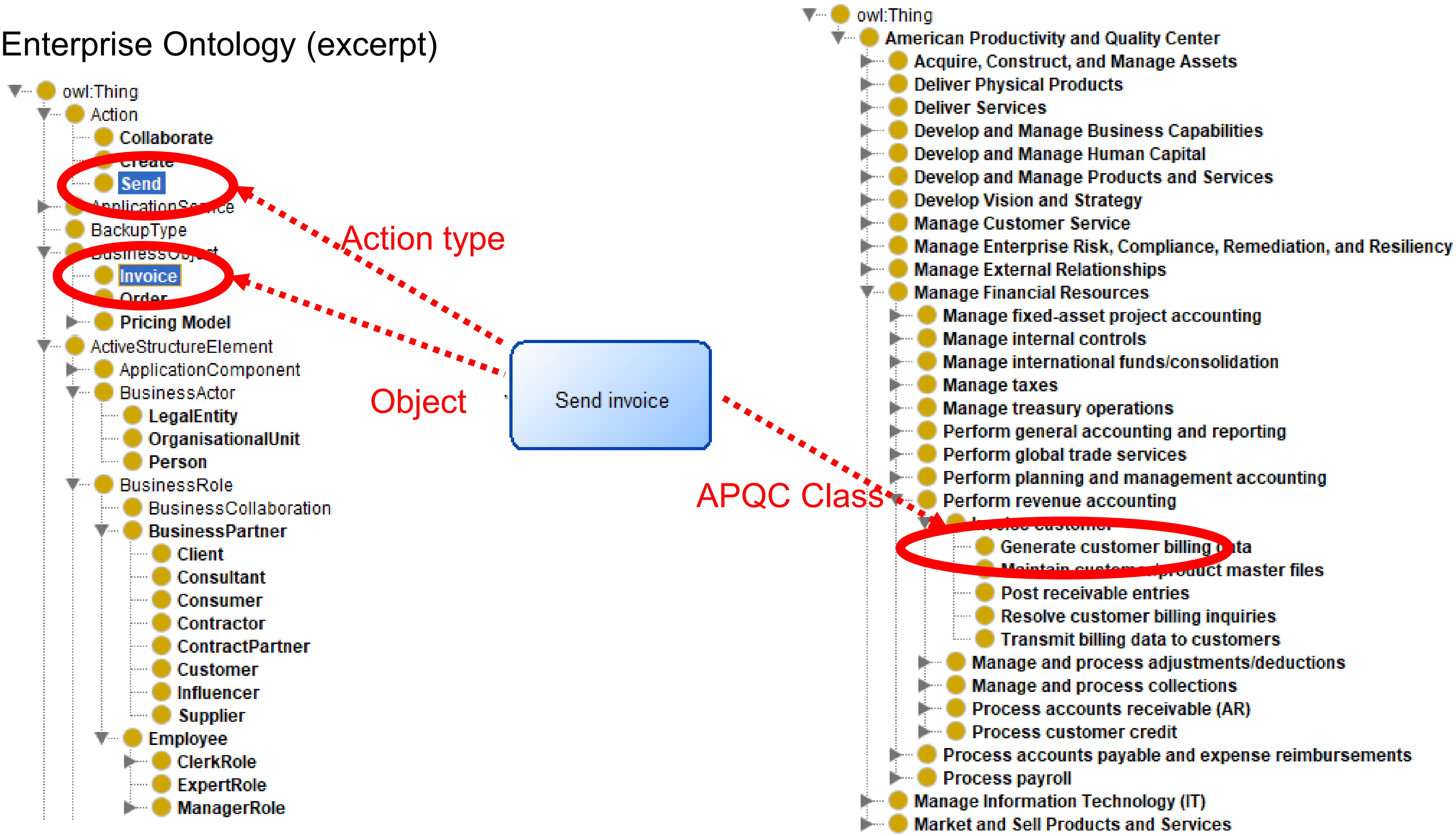


(Laurenzi et al. 2018)

Domain Ontologies

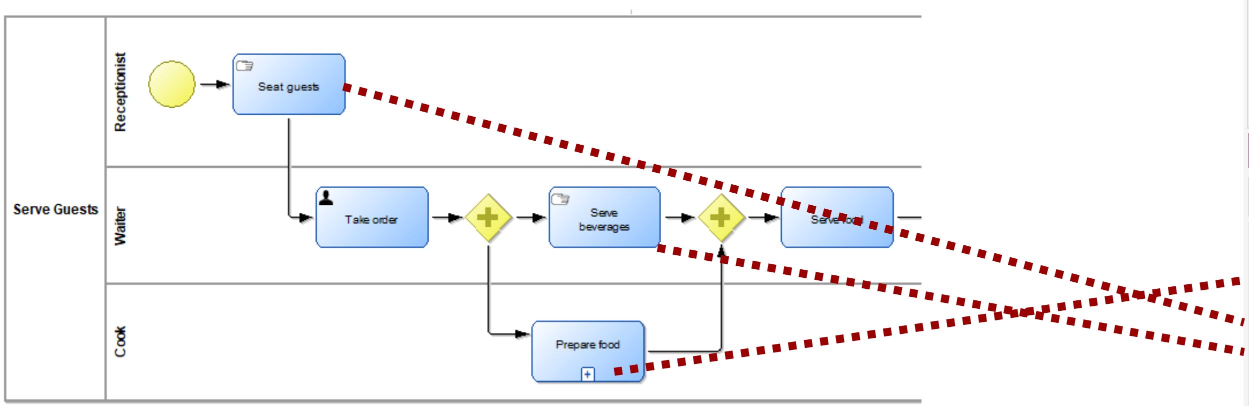
Enterprise Ontology (excerpt)

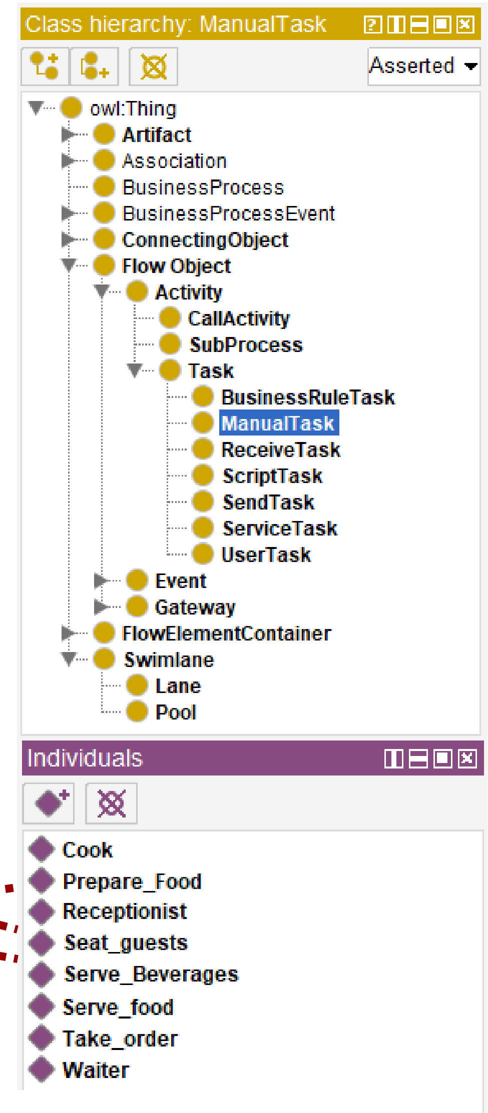
Domain Ontology: APQC Process Classification Framework



Ontology-Based Modeling

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





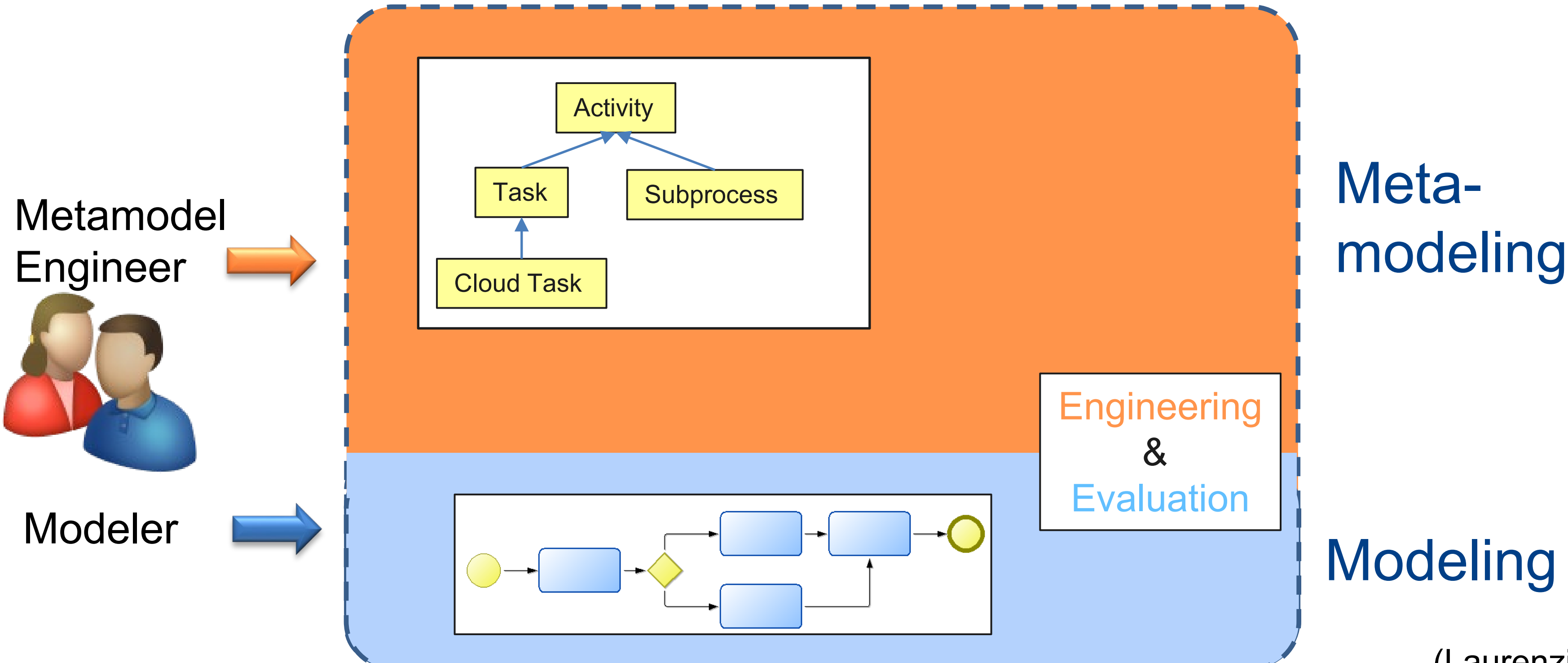
Agile Meta-Modeling

Objective

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

Integration Modeling and Metamodeling in a Single Environment

- Modeling and metamodeling in a single environment
- Tight collaboration between metamodel developer and modeler
- Modeler can also take the role of metamodel developer

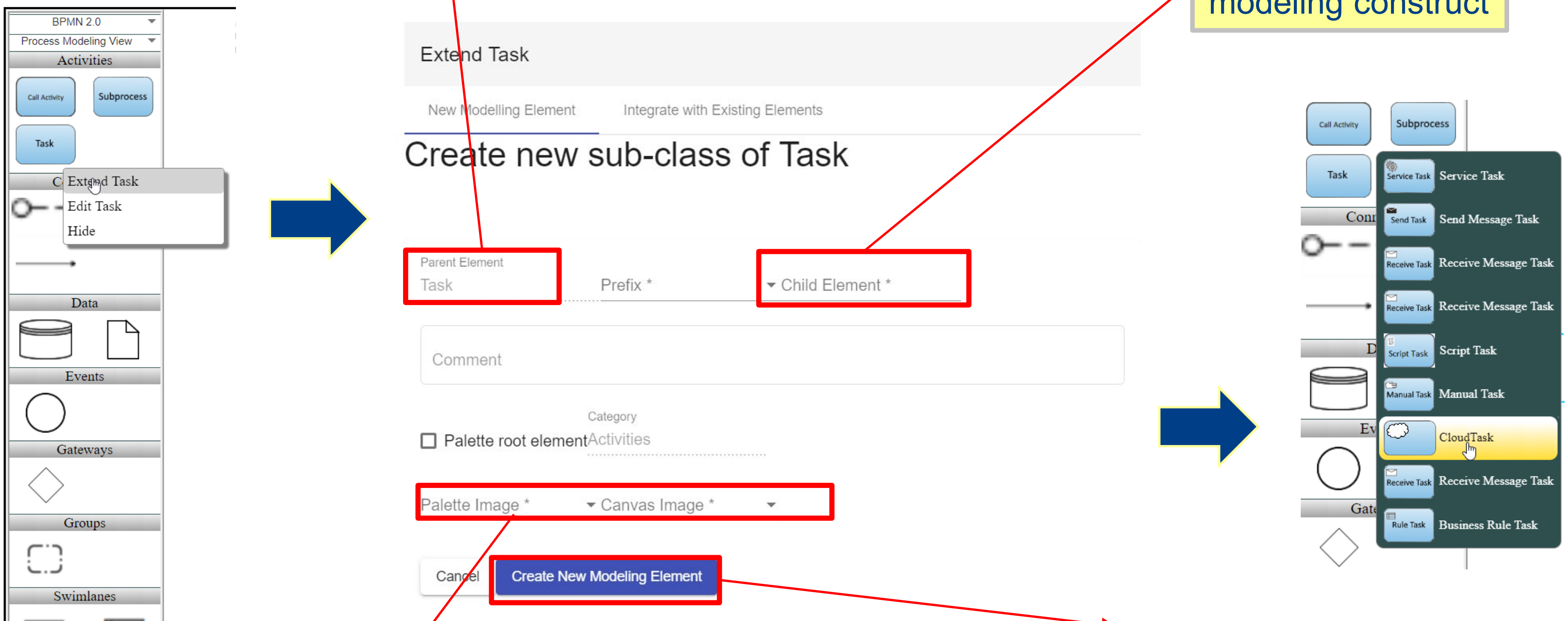


(Laurenzi et al. 2018)

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

Name of the parent class

Name for the new modeling construct



Graphical notation to be shown in palette and canvas

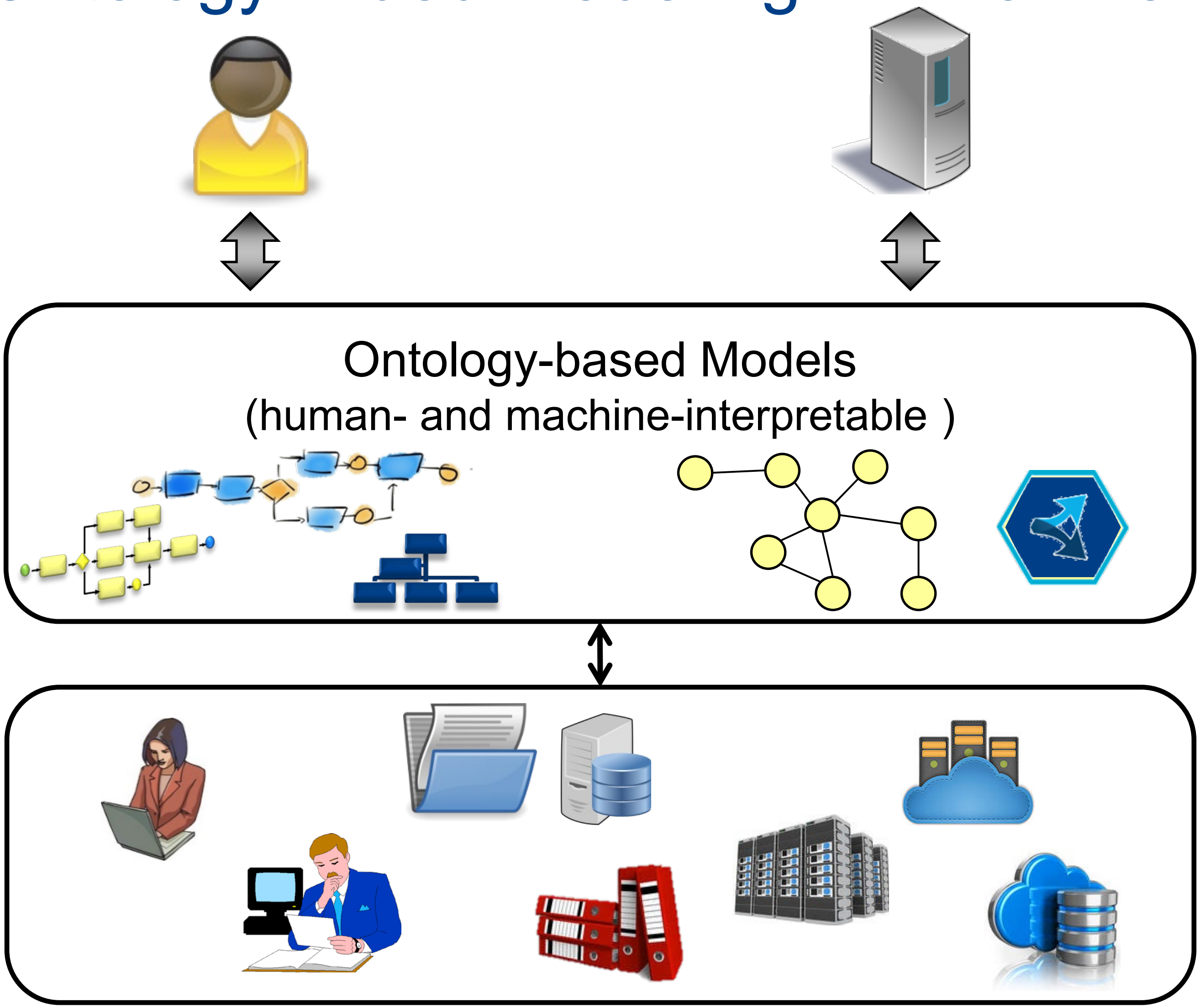
Create New Modeling construct and store in the ontology

(Laurenzi et al. 2018)

Agile and Ontology-Aided Modeling Environment (AOAME)

Models + Knowledge

Reality



(Laurenzi et al. 2018)