

Conceptual Modeling

Knut Hinkelmann



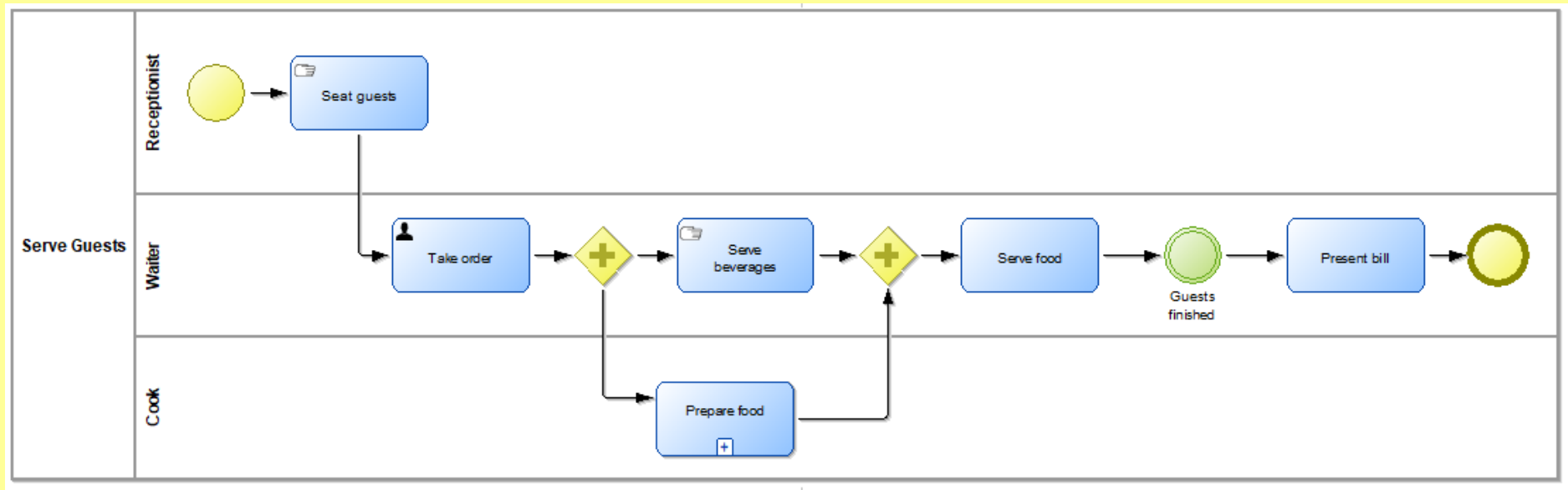
Image by [Bessi](#) from [Pixabay](#)

Visual Communication

- A picture is worth a thousand words
- Graphical Models are easier to understand than text

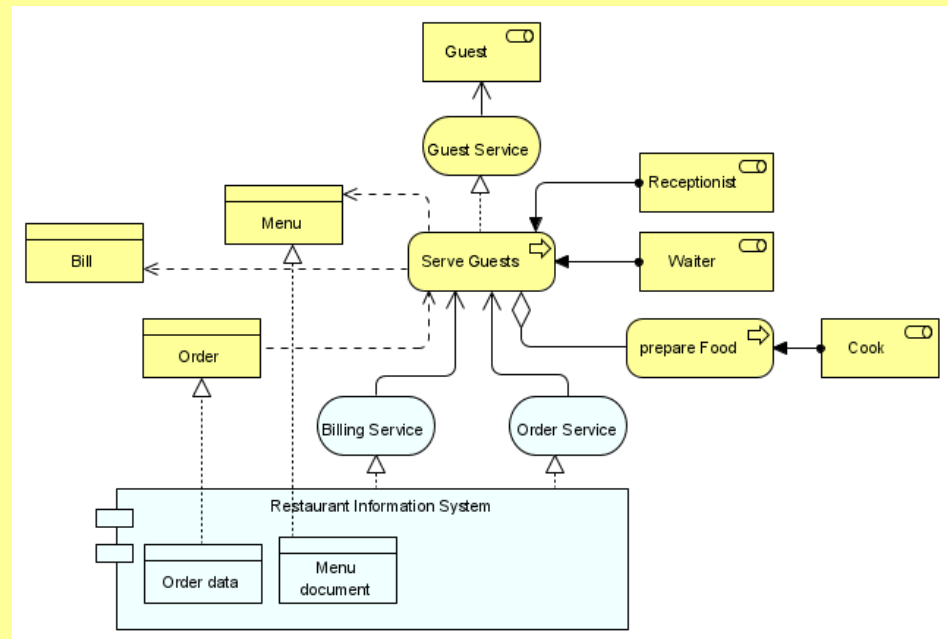


Example: Interpretation of a Process Model



- Who seats the guests?
- Which task is executed after «Serve food»?
- Which activities are executed in parallel?

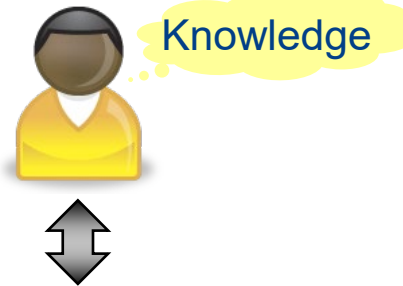
Example: Interpretation of an Architecture Model



- Who is involved in the process «Serve Guests»?
- Which Business Processes are served by the Restaurant Information System?
- In which application is the order data stored?

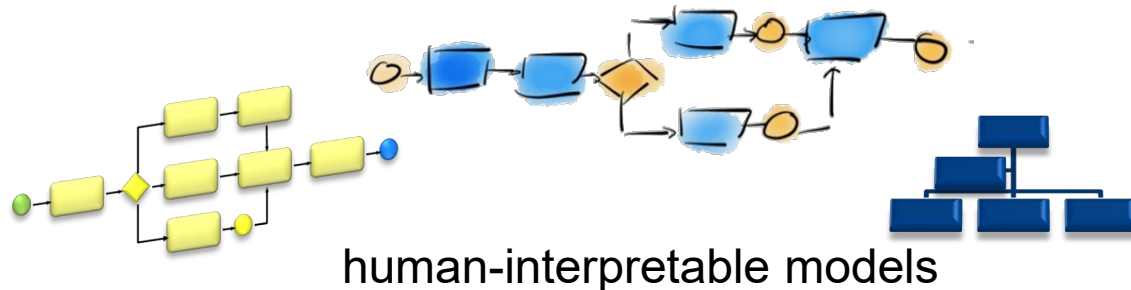
Graphical Models are appropriate for Humans

*Communication/
Analysis/
Decision Making*



Examples:
Visio
Powerpoint

Models

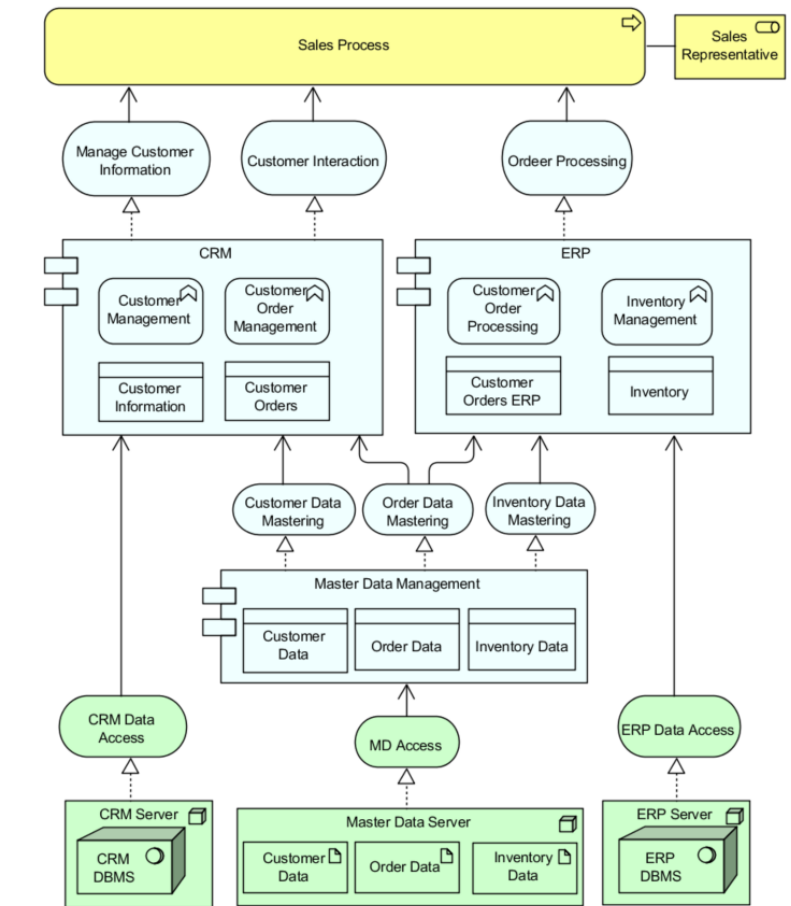
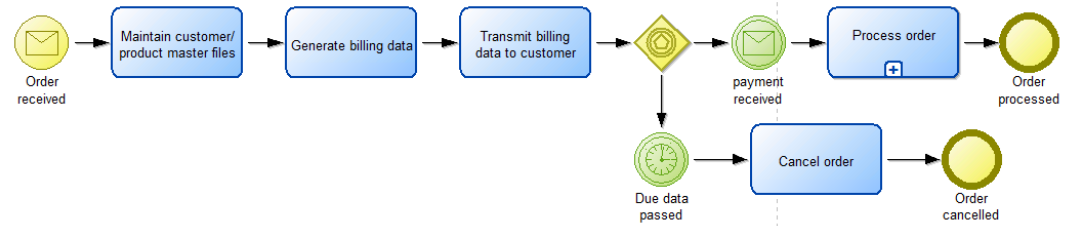
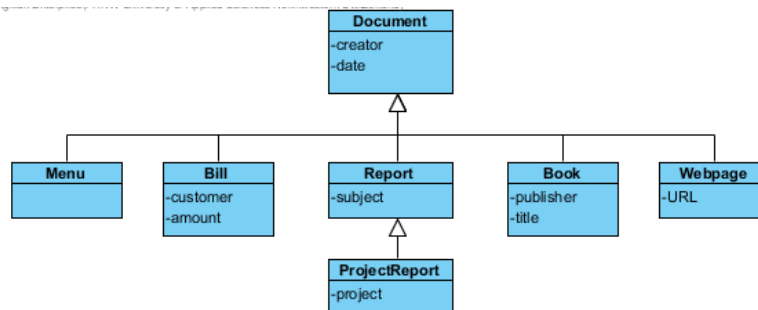
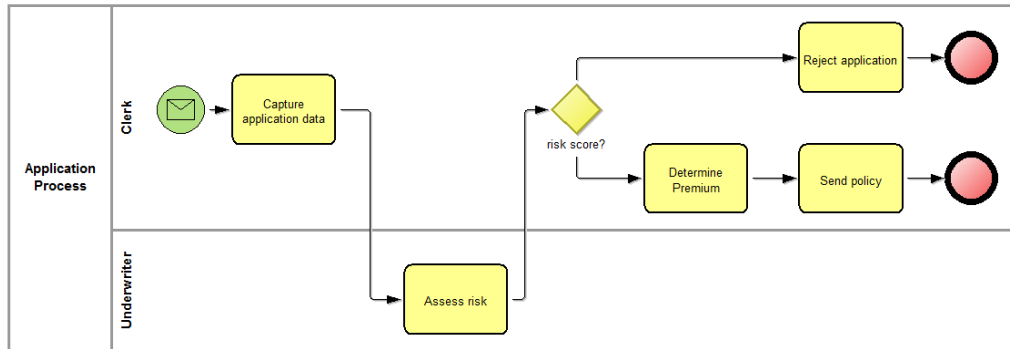
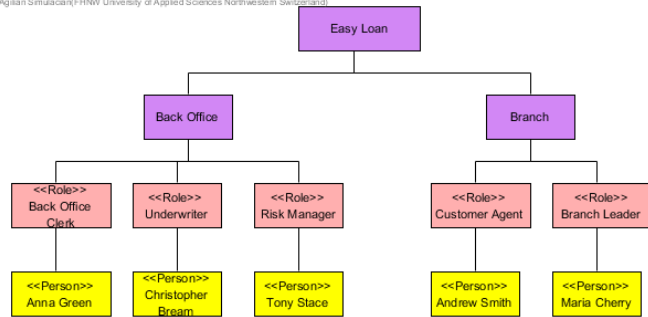


Reality

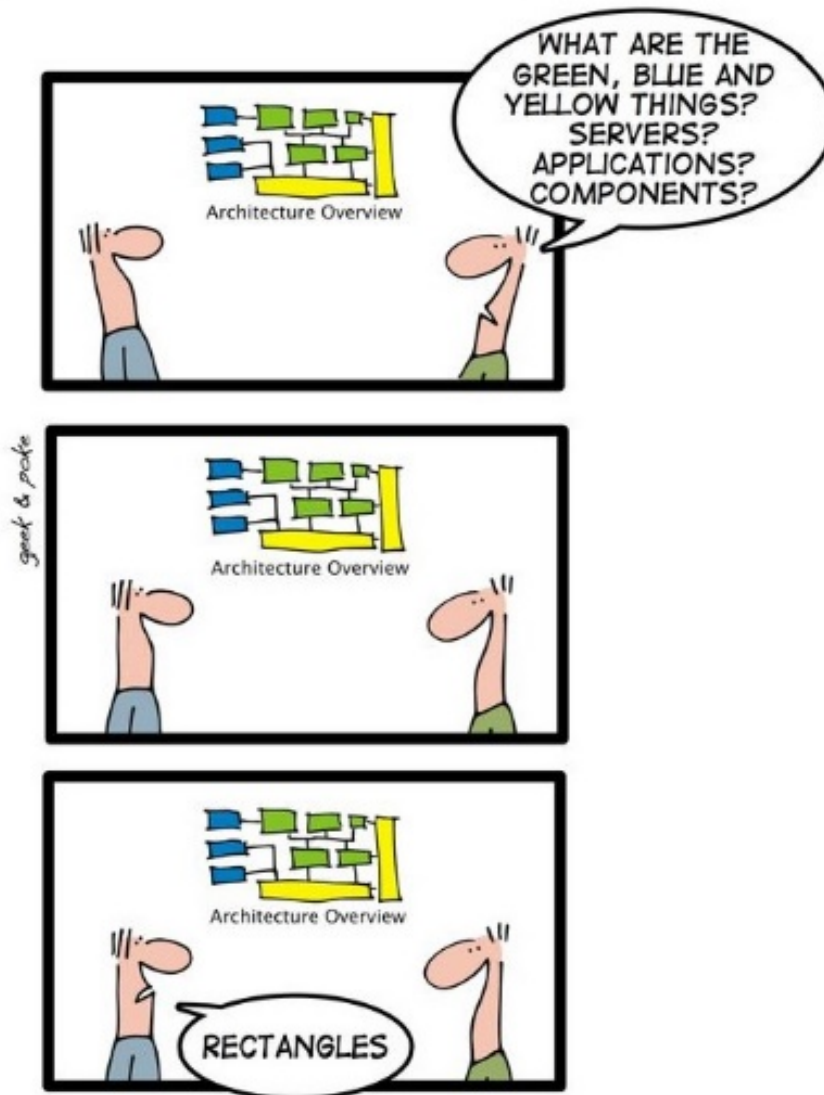


Enterprise Models

Agilan Simulator (FH NW University of Applied Sciences Northwestern Switzerland)



Interpretation of Models

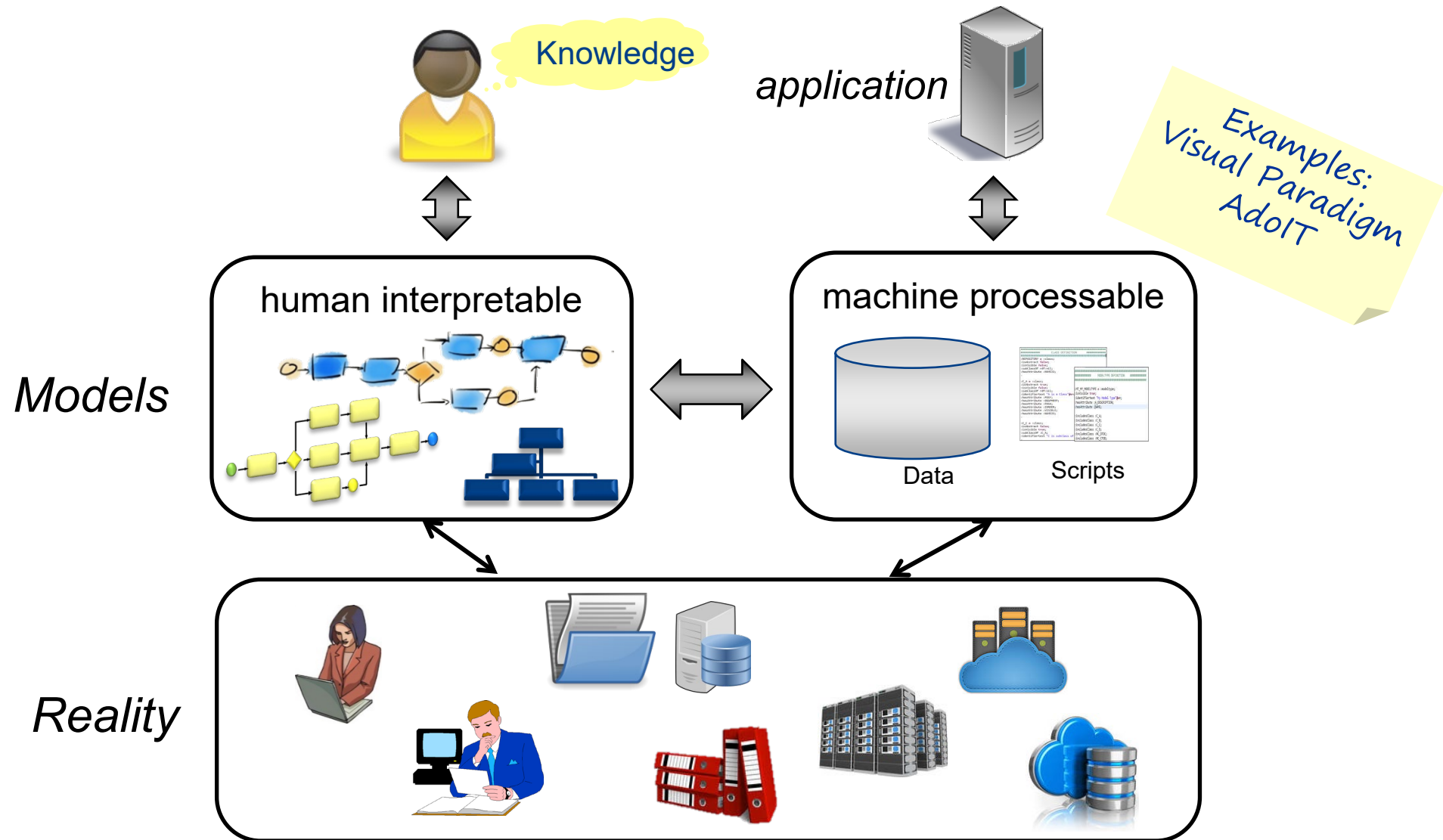


Models

- 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.

Models should allow automated analysis,
decision making and digitalization

Graphical Models are Represented in a Database



Conceptual Modeling and Metamodelling

Types of Modelling Languages

- **General-purpose** modelling languages can be used to represent any kind of knowledge
 - ◆ Examples:
 - Class diagrams (e.g. from UML)
 - Knowledge graphs (RDFS)
- **Domain-specific** languages have *predefined* concepts that are specific for a domain
 - ◆ Examples of domain-specific modelling languages:
 - **BPMN:**
 - Elements: task, event, gateway,
 - Relationships: sequence flow, message flow, association, ...
 - **ArchiMate:**
 - Elements: process, actor, role, business object, ...
 - Relationships: uses, realizes, ...
- **Conceptual modeling** = using a domain-specific modeling language



Strengths and Weaknesses of Domain-specific Modelling Languages

■ Strengths

- ◆ Comprehensibility of models
 - Concepts are adequate for stakeholders
- ◆ Guidance for modelers
 - Predefined concepts determine what is relevant for a model
 - Modeling language determines correct usage of elements
- ◆ Standardisation: Reuse of models
 - Common concepts for a domain (e.g. BPMN, ArchiMate)

■ Weaknesses

- ◆ Restricted to a specific domain
 - Only what can be expressed with the modelling elements can be modeled

Models, Modelling, Modeling Language

Model

A reproduction of the part of reality which contains the essential aspects to be investigated.

Conceptual Modelling

Creating models using predefined concepts.

Meta Model

The concepts of the modeling language are predefined in a so-called meta-model

Modelling Language

Notation/Visualization of the concepts that can be used for modeling

Modelling Language

- A **modelling language** specifies the notation for the concepts, from which a model can be made.
- There are different kinds of notations
 - ◆ For graphical models the notation consists of *visualization* of the concepts
 - ◆ Textual models consist of words
 - ◆ Mathematical models use symbols
 - ◆ physical model are composed of physical elements

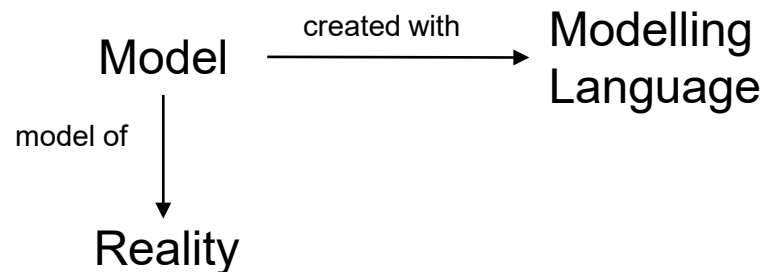
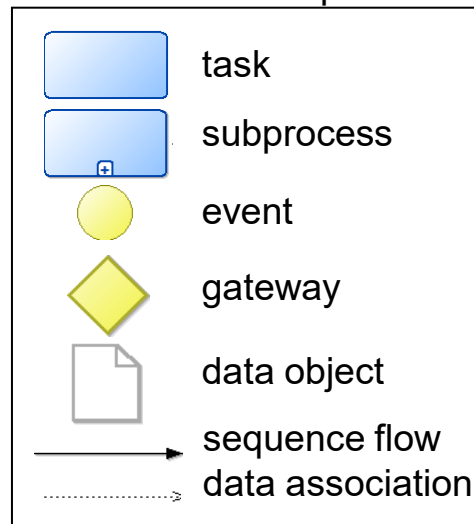


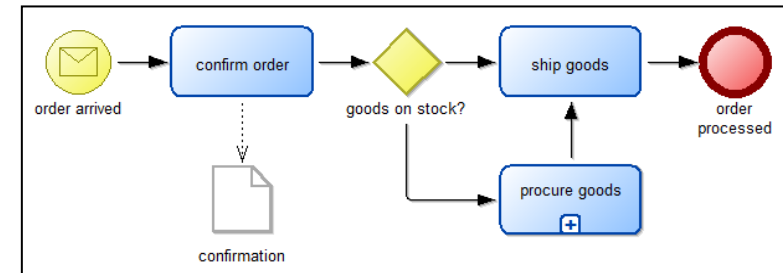
Illustration: Modeling Language for Business Processes

Modelling Language:

Notation/appearance of meta-model concept



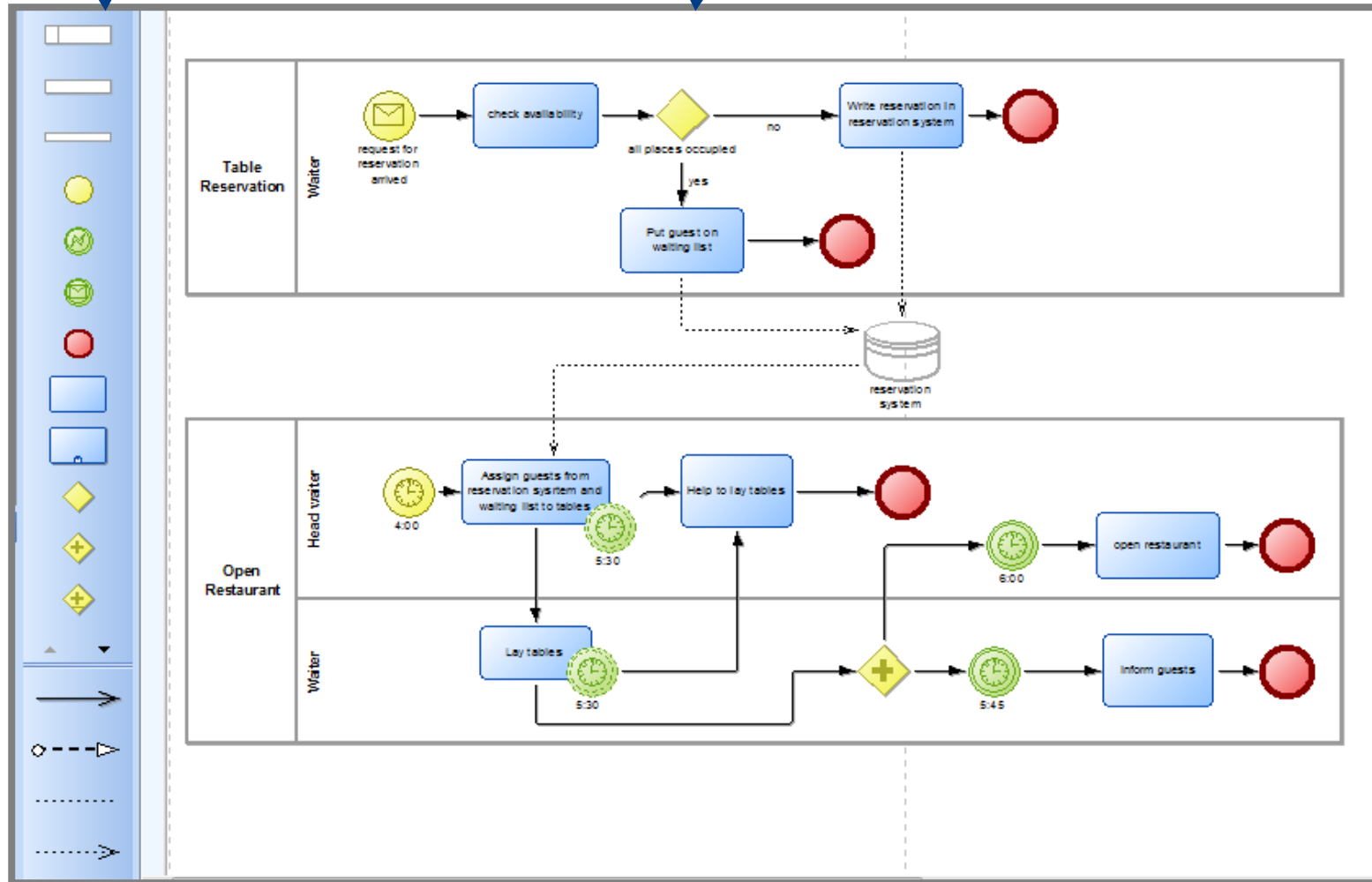
Model:



A model contains instances of the concepts defined in the meta-model. The object „confirm order“ represents a real entity; it is an instance of the concept «task»

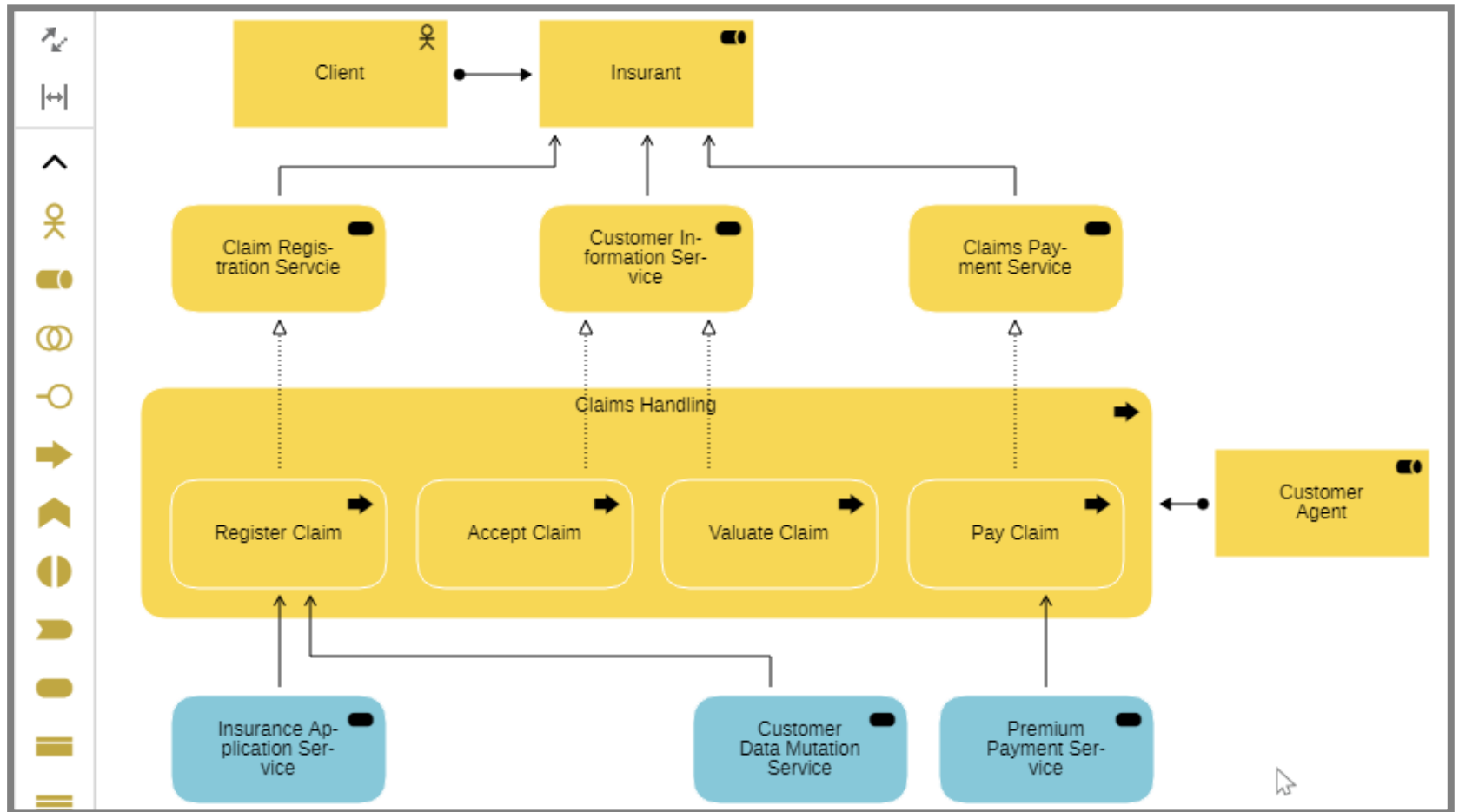
Modelling
Language

Model

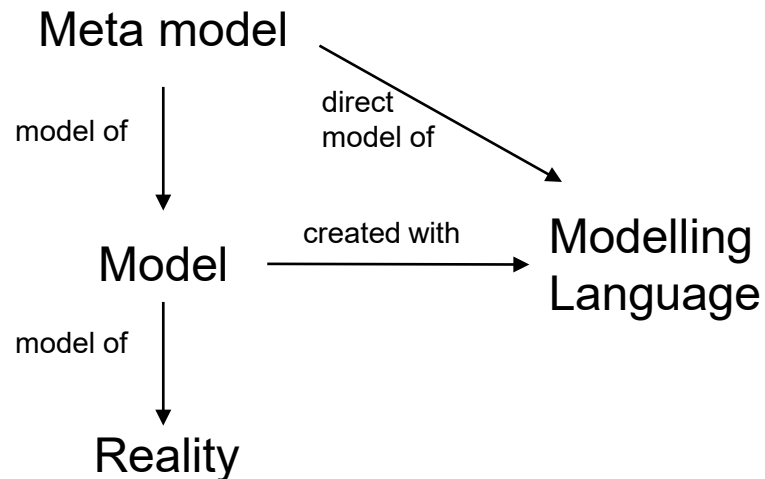


Modelling
Language

Model



Meta-model



- A meta-model defines ...
 - ... Concepts that can be used to create a model
 - ... Attributes of concepts
 - ... Rules to combine concepts
- The meta-model represents the general knowledge about the domain

Illustration: Modeling Language for Business

Processes Metamodel:

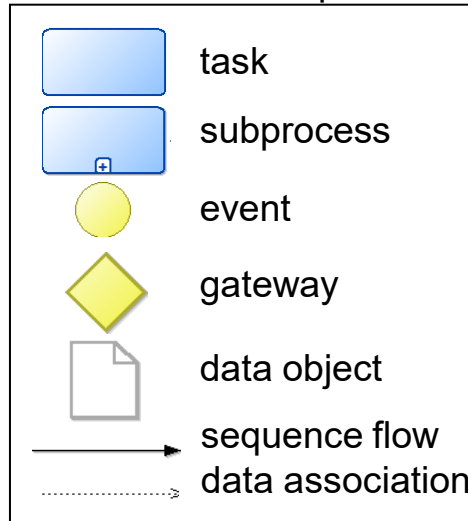
Concepts which can be used to create models.

Example: A process model consists of concepts for

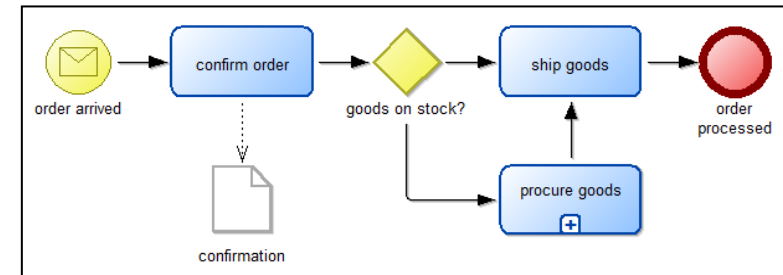
- Model elements:
event, task, subprocess, gateway, data object
- Relationships:
sequence flow, data association.

Modelling Language:

Notation/appearance of meta-model concept



Model:

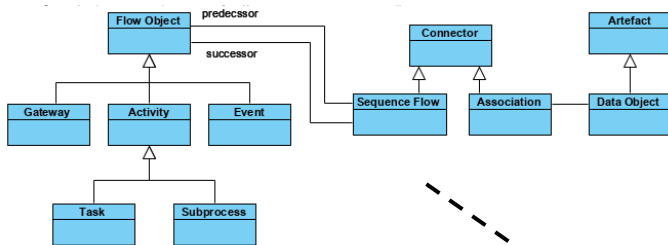


A model contains instances of the concepts defined in the meta-model. The object „confirm order“ represents a real entity; it is an instance of the concept «task»

Metamodels can be defined as Class Diagrams

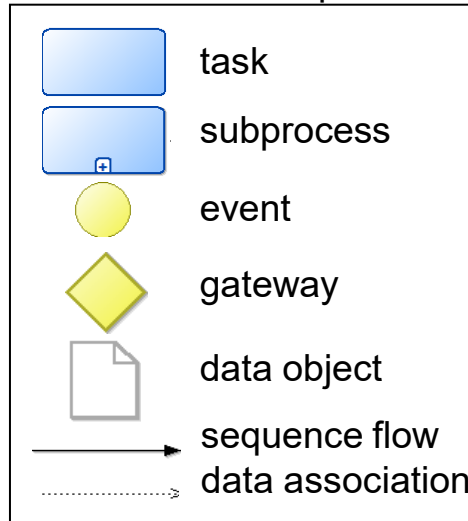
Metamodel:

Concepts which can be used to create models.

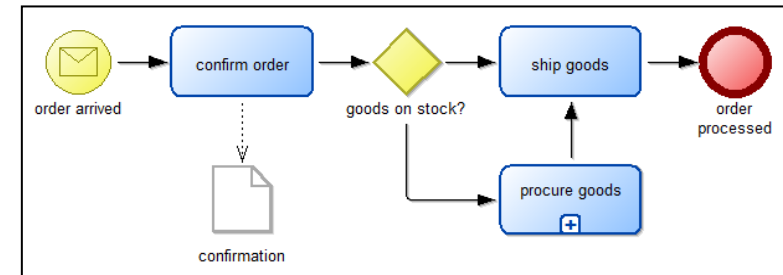


Modelling Language:

Notation/appearance of meta-model concept



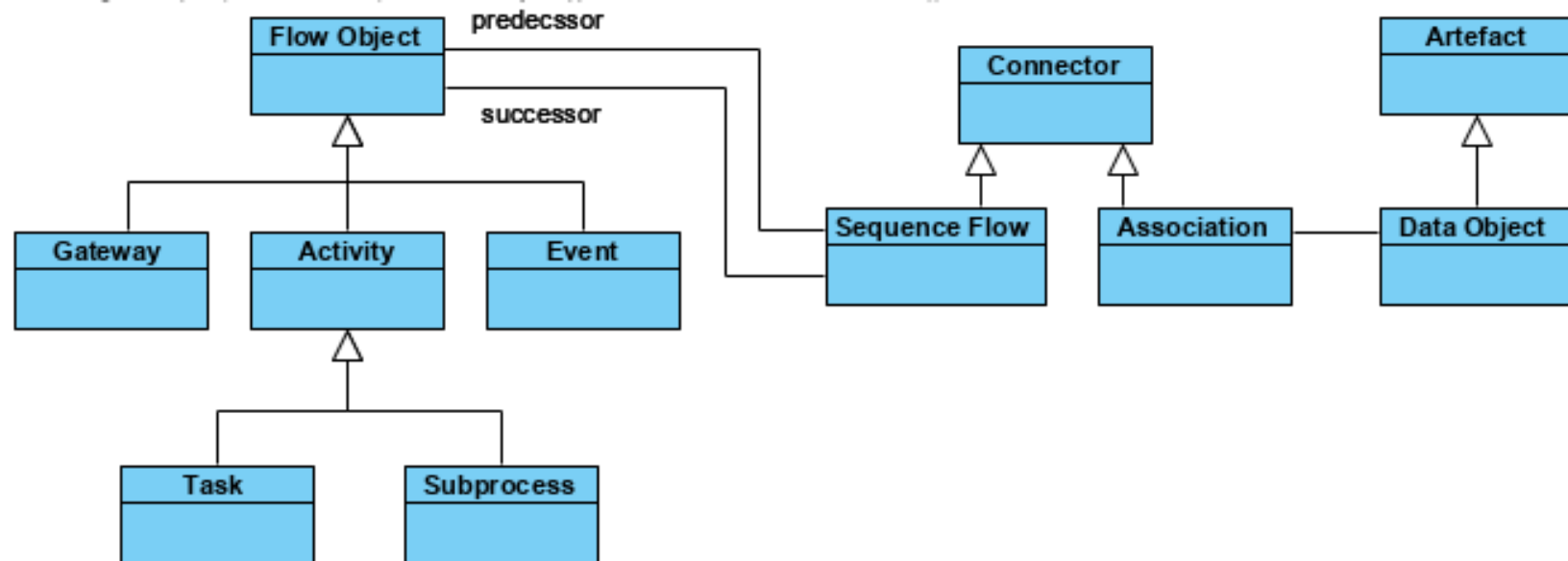
Model:



A model contains instances of the concepts defined in the meta-model. The object „confirm order“ represents a real entity; it is an instance of the concept «task»

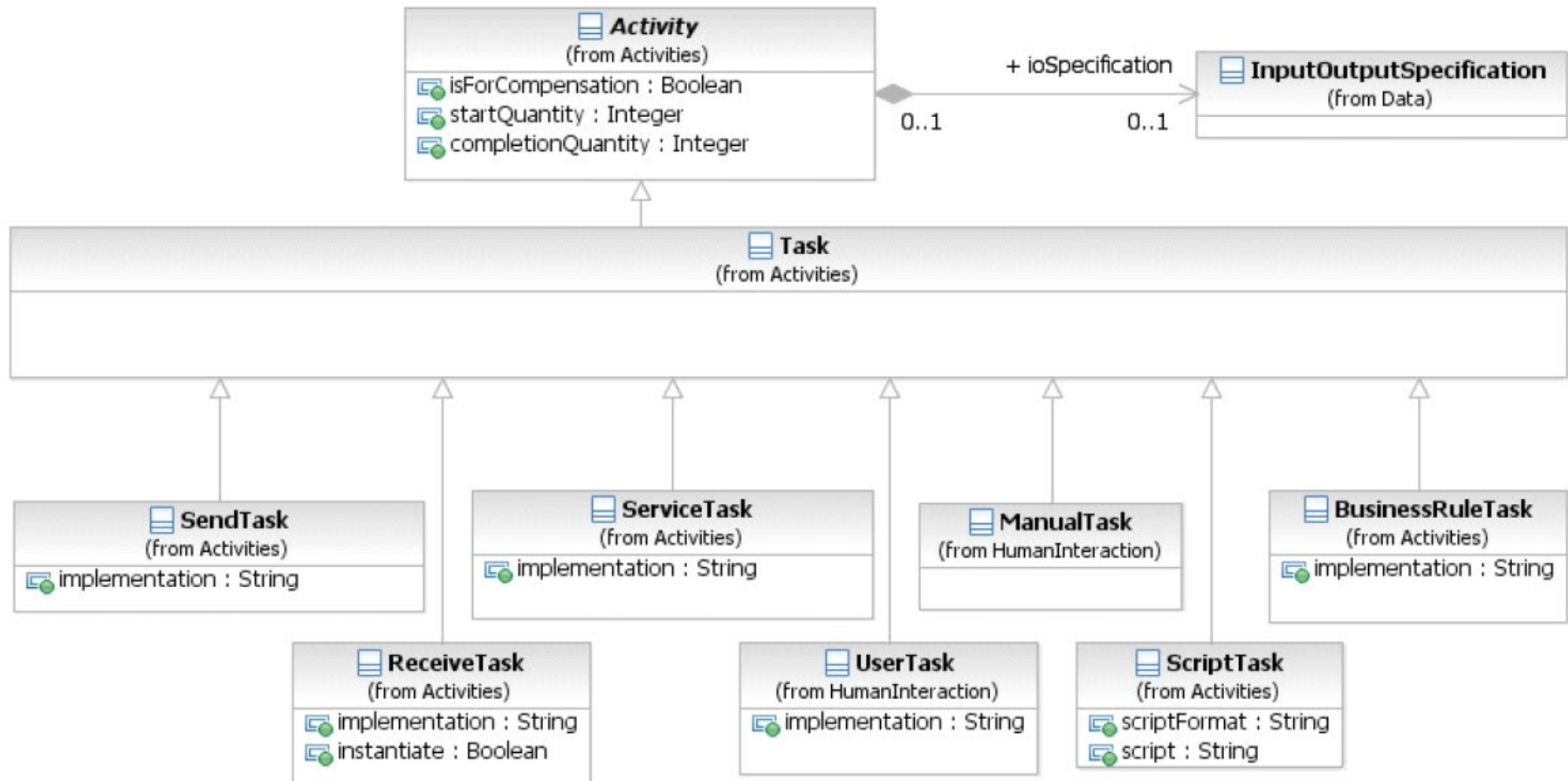
Metamodels can be defined as Class Diagrams

A Metamodeling language one can described meta models
Metamodels can be represented as class diagrams

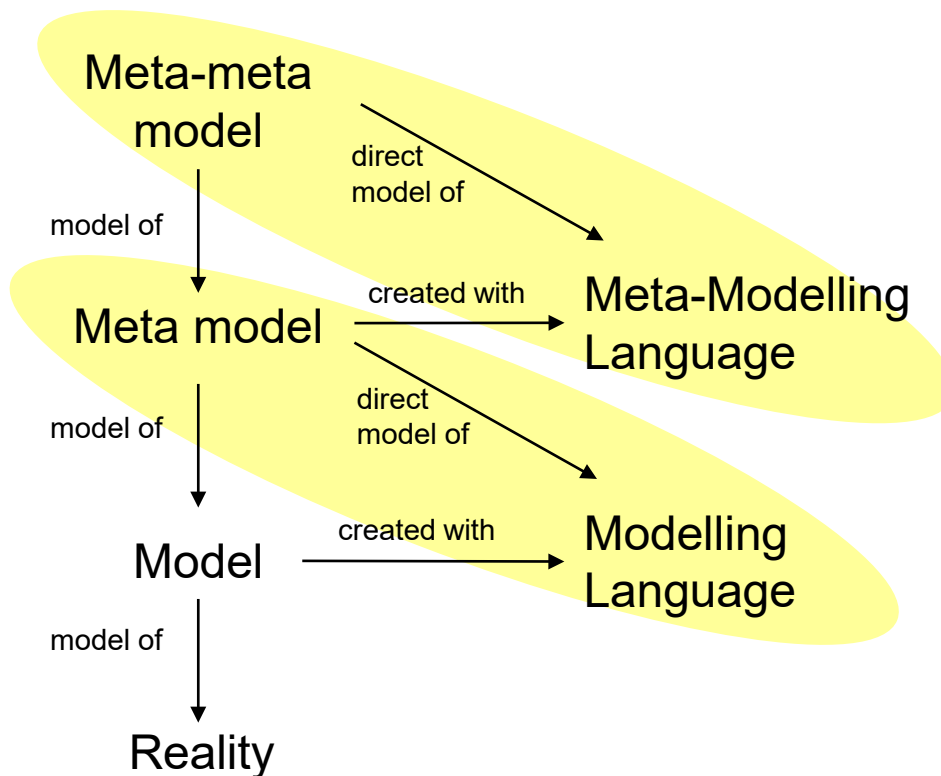


(UML Class diagrams were originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

Subset of the BPMN Metamodel as UML Class Diagram

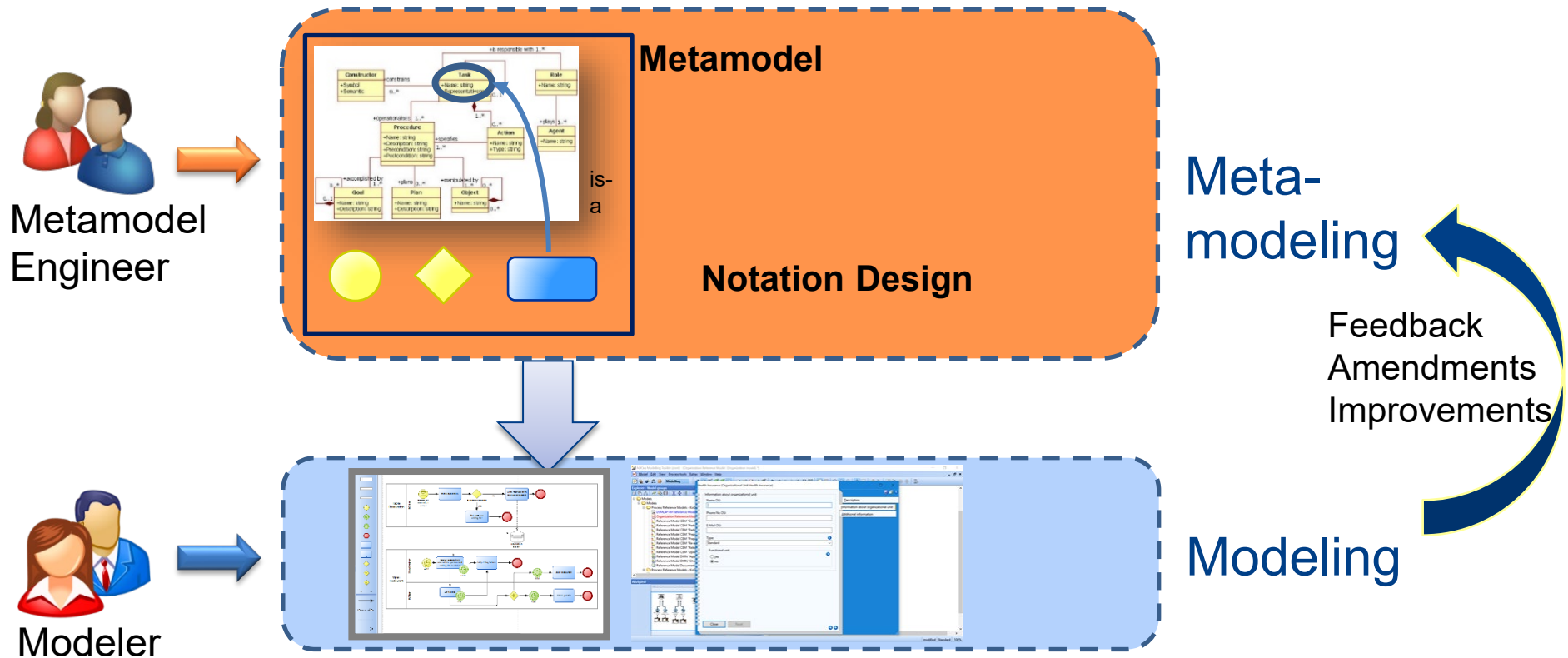


Meta-meta model



- The meta model must again be described in some language, which is specified in a meta-meta model
- A **meta-meta model** defines the concepts for describing a meta model
- Graphical models usually have to kinds of concepts
 - ◆ Modeling elements
 - ◆ Relationships
- Examples for meta-modeling languages are
 - ◆ class diagrams.
 - ◆ Knowledge graphs
- Note: Meta-modeling languages are general-purpose modeling languages

Modeling and Metamodeling



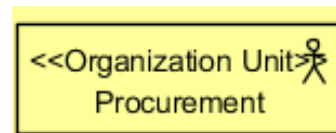
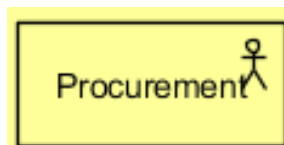
What do we do if there is no Domain-specific Modelling Language

- If there is no appropriate domain-specific modelling language for a domain of interest, we can
 - ◆ Metamodelling
 - Customization of an existing modeling language
 - Define a new domain-specific modelling language
 - ◆ Use a general-purpose modeling language

Customization of a Modeling Language

Customization in Archimate

- ArchiMate can be customized for specific usage by
 - ◆ Specialization of elements and relationships
 - ◆ Adding specific attributes, e.g. for cost calculations
- In Visual Paradigm the specialization can be done with stereotypes
 - ◆ Example: Specialization of Actor



Examples of Specializations

Specialisation can be made for elements and relations on all layers

- A **Business Actor** could be
 - ◆ Individual
 - ◆ Organization Unit
 - ◆ Company
- **Product** could be
 - ◆ Physical Product
 - ◆ Digital Product
- A **Data Object** could mean
 - ◆ Document
 - ◆ Structured Data
- **Application Interface** could be
 - ◆ Application-to-Application Interface
 - ◆ User Interface
- **Network** could be
 - ◆ WiFi Network
 - ◆ Wide Area Network
- **Equipment** could be
 - ◆ Machine
 - ◆ Vehicle

Using a General-Purpose Modeling Language

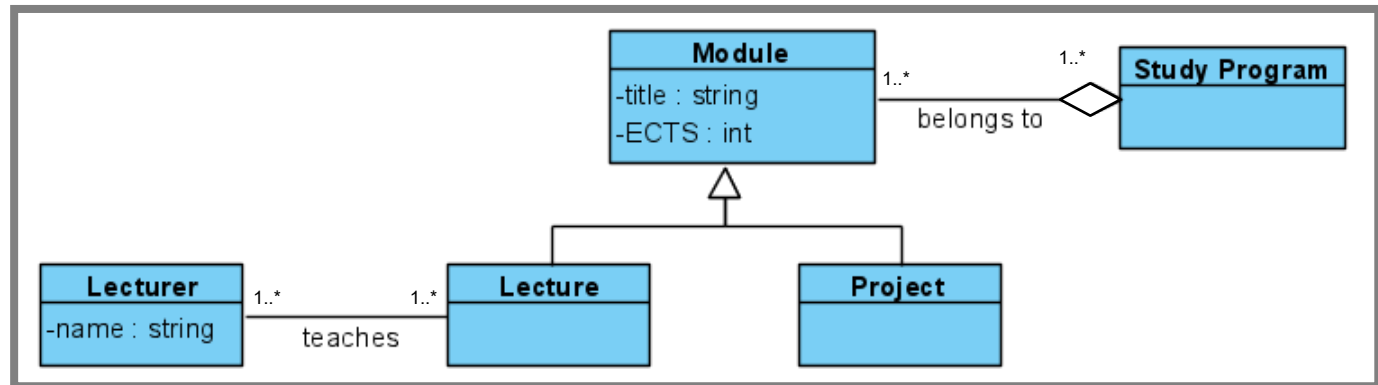
General-Purpose Modeling Languages

- General-purpose modeling languages can be used to represent any kind of knowledge
- There are a wide range of general-purpose modeling languages
 - ◆ Natural language allows to express any knowledge
 - ◆ Formal languages: Typically a subset of Logic
 - ◆ Class Diagrams
 - Artificial Intelligence: Ontologies, Knowledge Graphs
 - Data Modeling: Entity Relationship Diagrams
 - Object-Oriented Programming: UML Class Diagrams

Modeling with a General-purpose Modeling Language

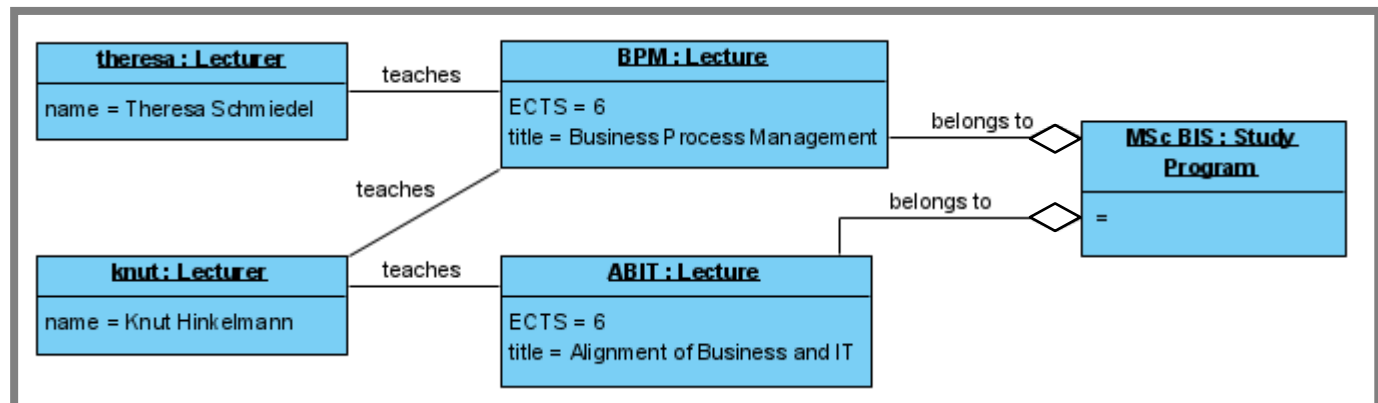
- UML Class Diagrams are general-purpose modeling languages
- A class diagram contains **Classes** with **Attributes** and **Associations**

Classes
(=metamodel)



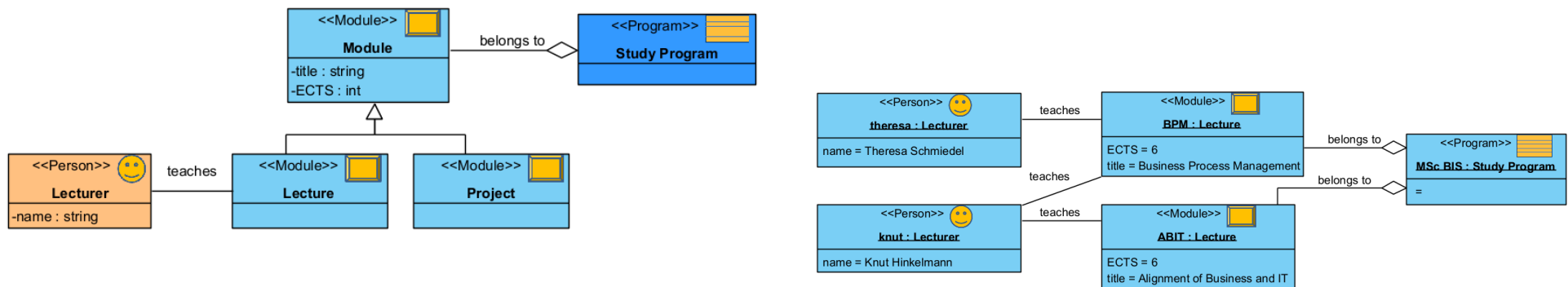
- A model consists of objects which are instances of these classes

Objects
(= model)



Customization of a General-purpose Modeling Language

- A Domain-specific Modeling Language can be regarded as a Customization of a General-purpose Modeling Language
- Example: In the Visual Paradigm tool we can use stereotypes also to specialize UML class diagrams.
- We can define a new stereotype for a class and change color or add an icon
- Example: stereotypes for modules and lecturers



Strengths and Weaknesses of General-Purpose Modeling Languages

■ Strengths

◆ Applicability

- Can be used to represent everything
- Every model in the same language
- Low learning curve for the language

■ Weakness

◆ No guidance: Users have to ...

- determine how to structure a domain
- to identify relevant concepts

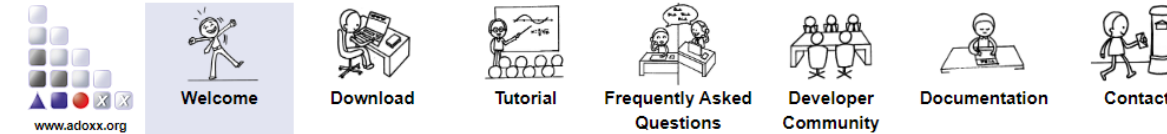
◆ Restricted reusability

- Different applications use different concepts

Creating a new Modeling Language: Metamodelling with ADOxx

adoxx.org – Download, Tutorials, Community

Sign In



ADOxx.org Welcome

ADOxx Event

ADOxx Training Days
25-27.03.2020 in Vienna

REGISTRATION REQUIRED!
Contact us at tutorial@adoxx.org

You can download conceptual modeling libraries from adoxx.org, e.g. BPMN,

Do you want to implement your modelling method on the open use metamodeling platform?
Get access to the open-use **ADOxx** Platform to get started.

DOWNLOAD

Do you want to realize model-value functionality?
Get access to the open-source **OLIVE** Microservice Framework - the **OMiLAB** Integrated Virtual Environment.

GET ACCESS

BPMN@ADOxx

UML@ADOxx

OWL@ADOxx

ER@ADOxx

Have a look at the following realization cases of modelling approaches from the research and industrial backgrounds to get your own development started.

Further usages of ADOxx are available at OMiLab/University of Vienna:
<http://www.omilab.org>

Tweets

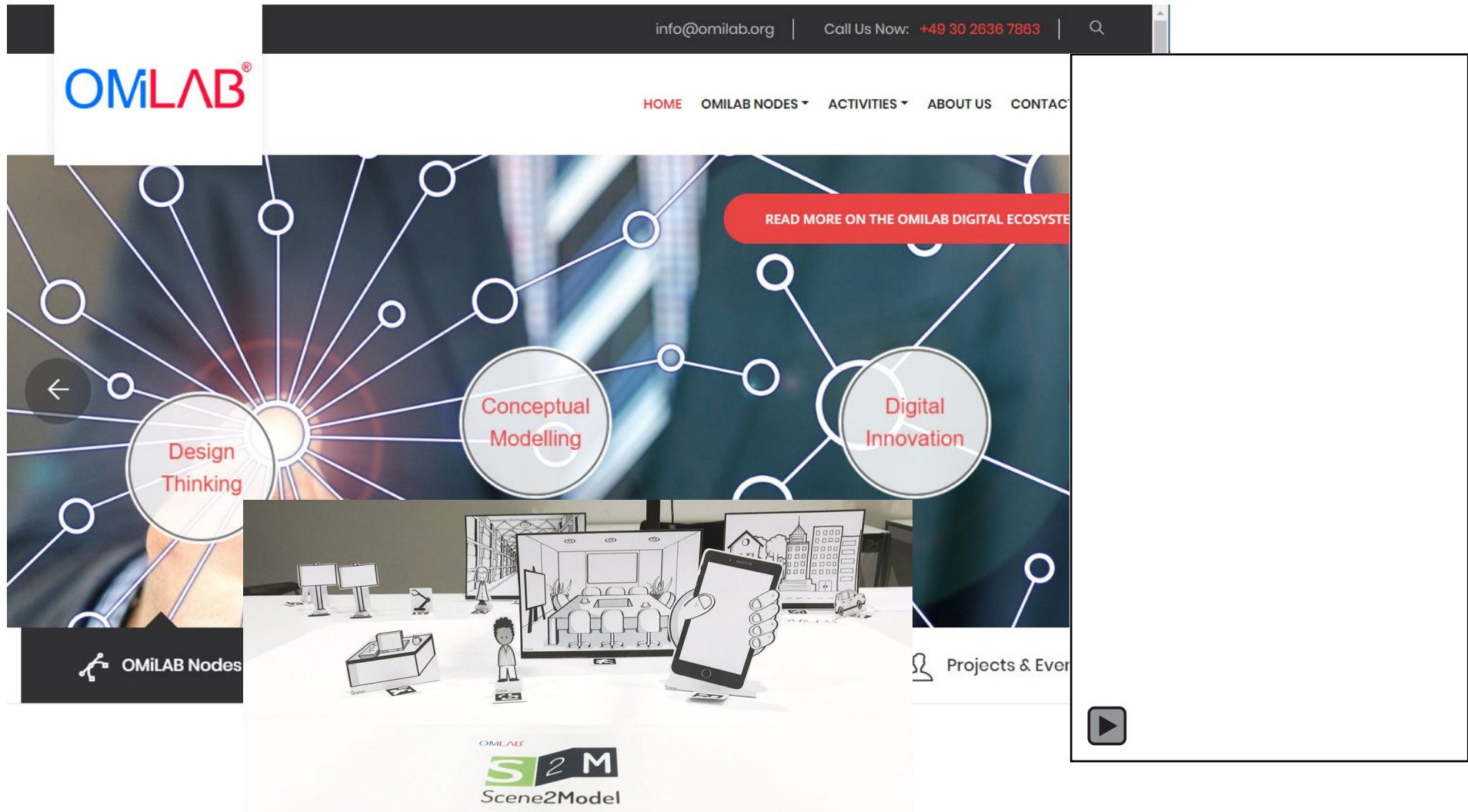
@ADOxxORG
Special times - a new mode of operation! Thank you all for joining three days of intense @ADOxxORG training in a virtual setting! #metamodelling #training

ADOxx Training Team
March 2020

Mar 28, 2020

OMiLAB – A Conceptual Modelling Community

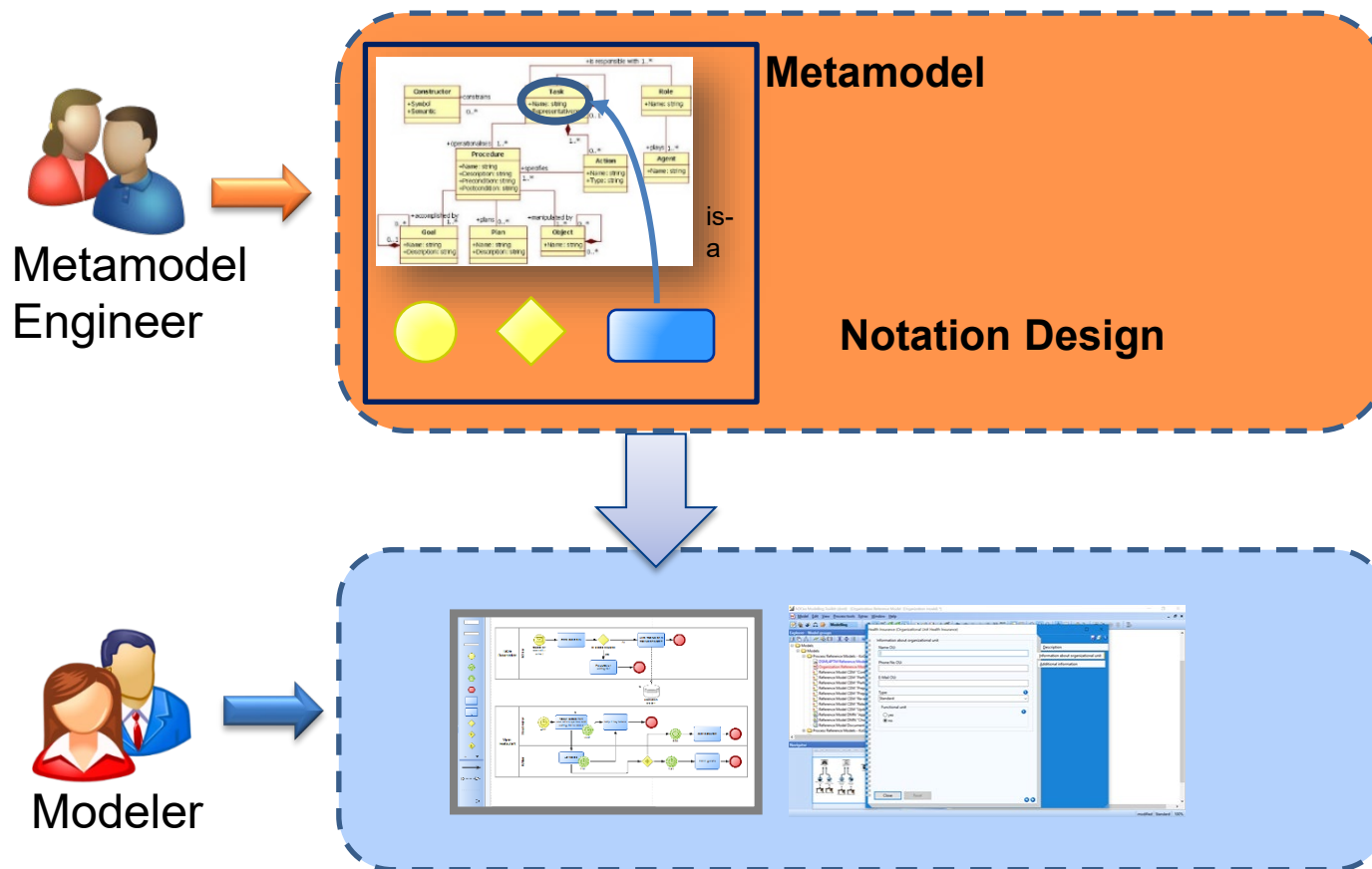
ADOxx is the basis for OMiLAB



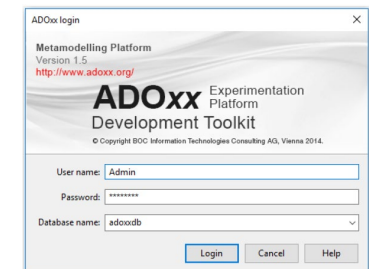
The ADOxx Environment

- ADOxx consists of ...
 - ◆ ADOxx Development Toolkit
 - Defining Modelling languages – Library Management
 - Administration of users, models, components
 - ◆ ADOxx Modelling Toolkit
 - Creating models

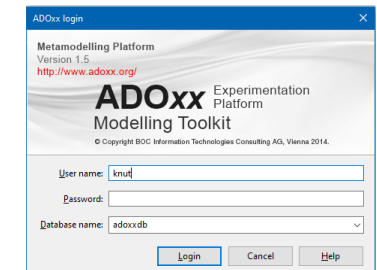
Modeling and Metamodeling



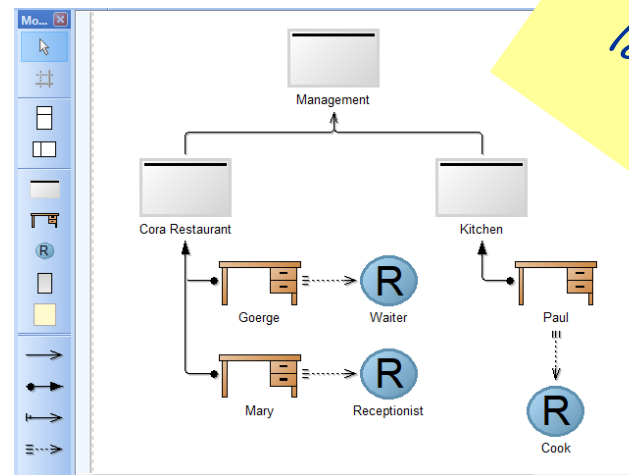
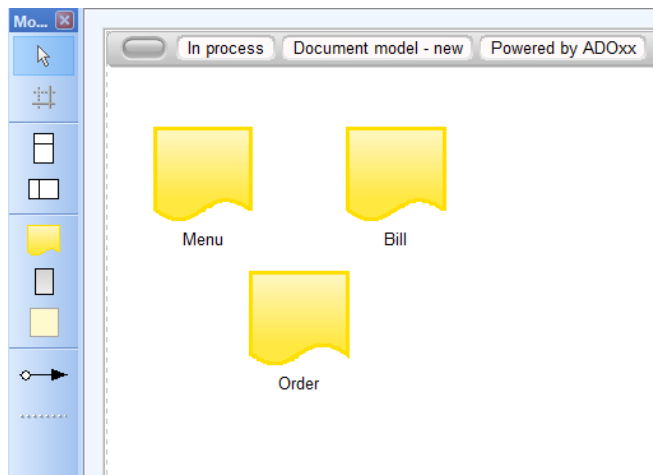
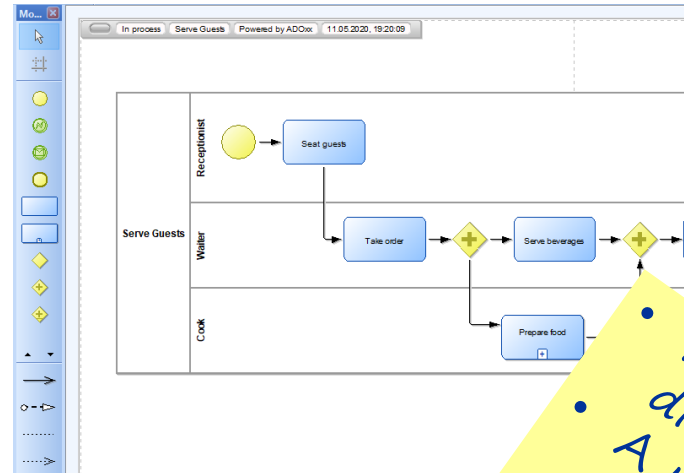
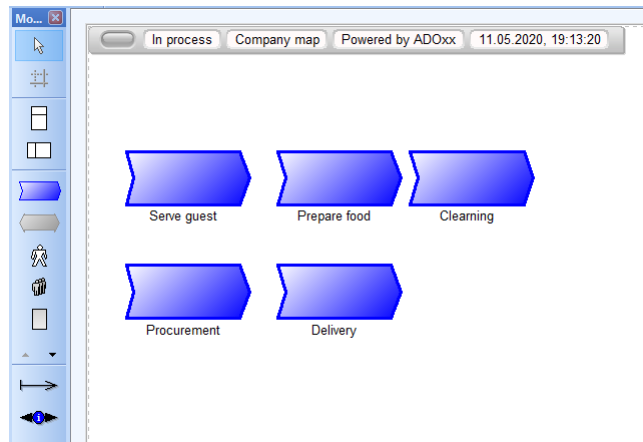
Meta-modeling



Modeling



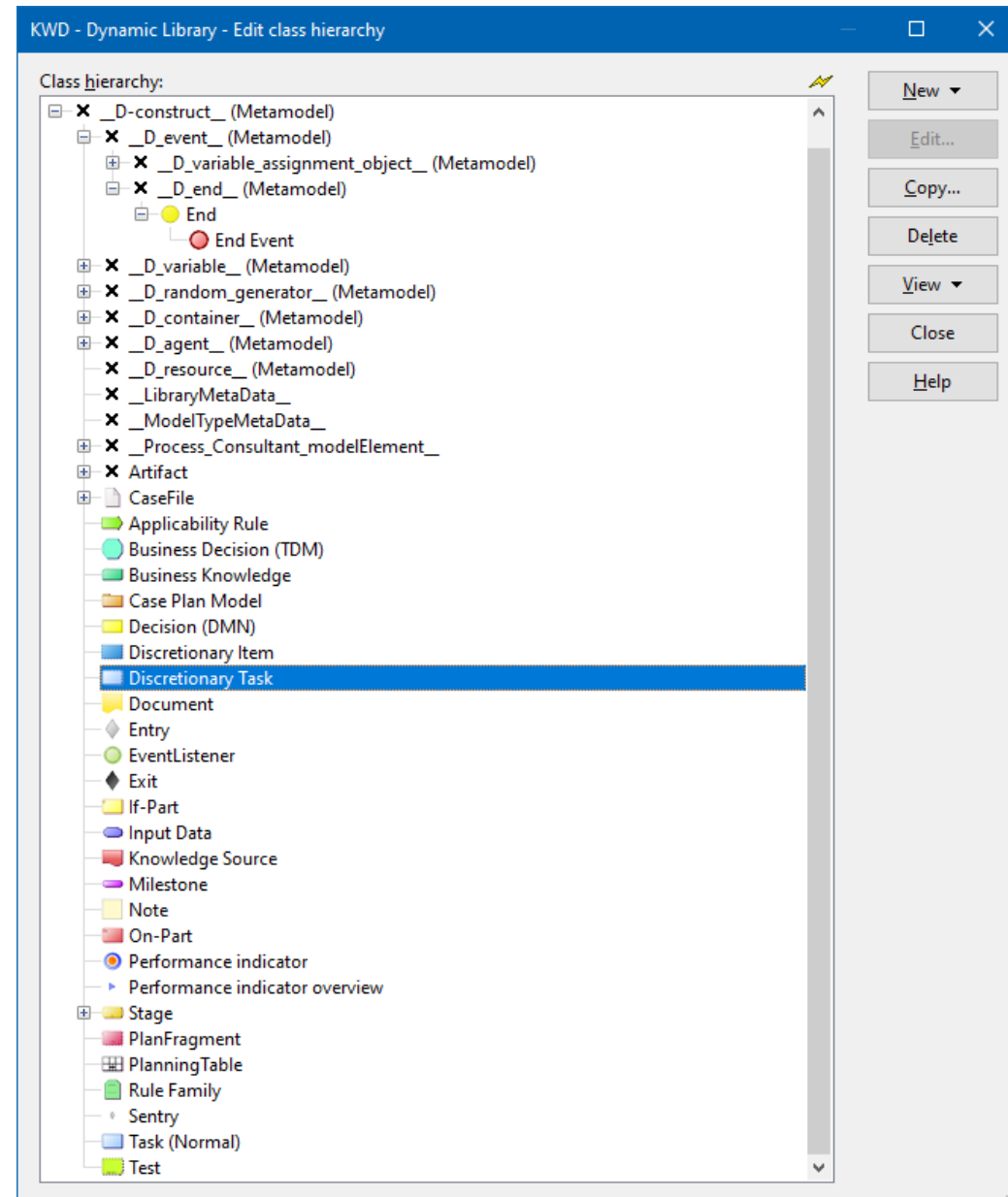
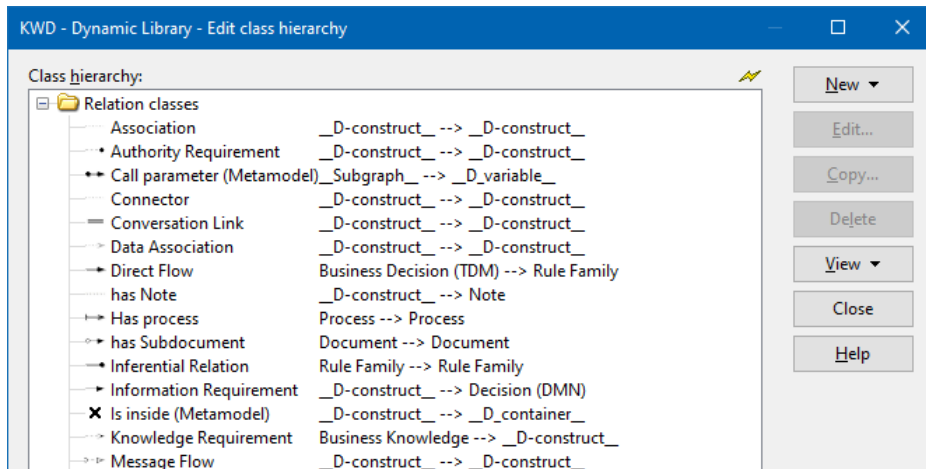
A Modeling Environment can consist of several Model Types



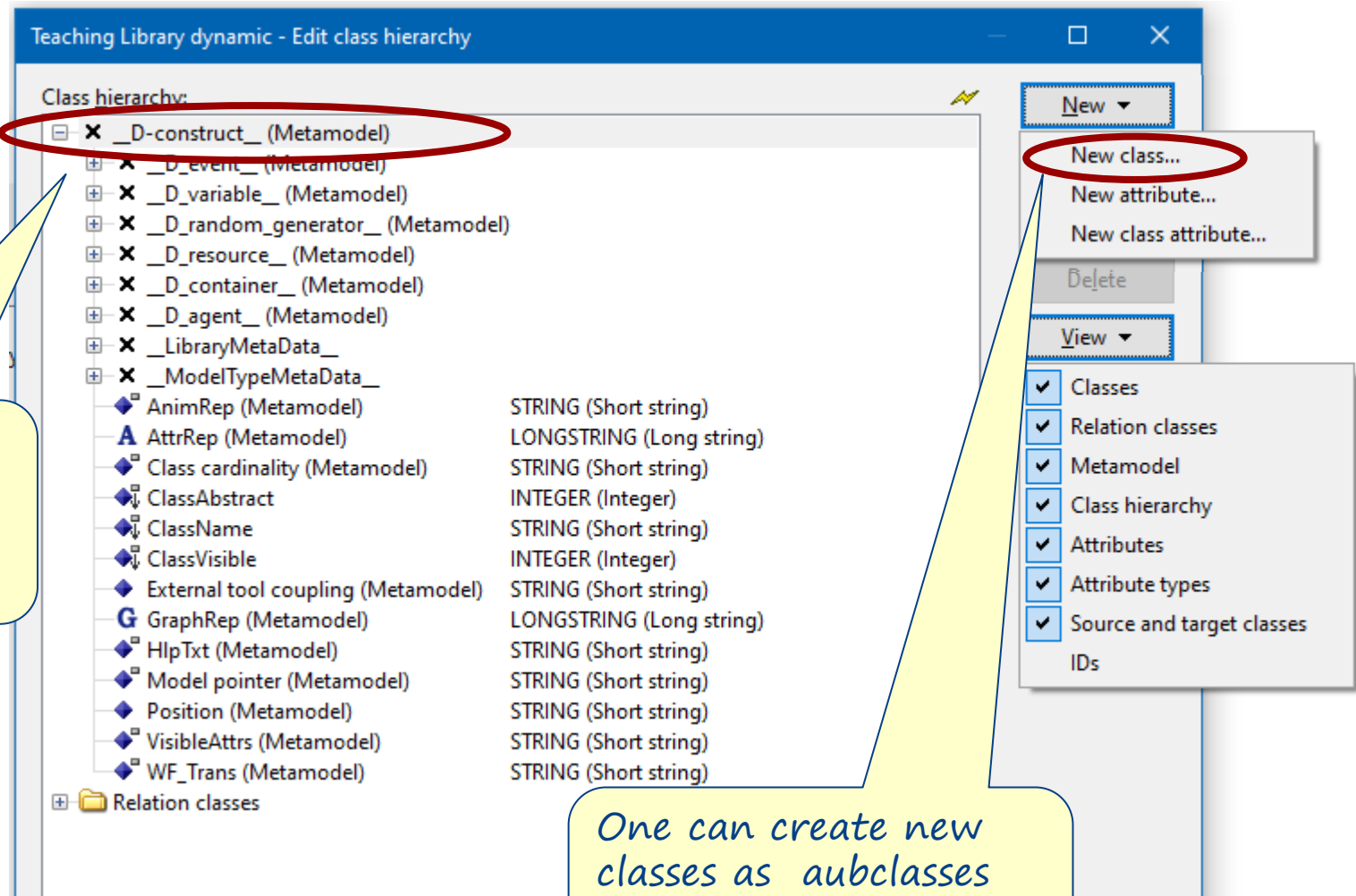
• Model Types represent different views
 • A model type can be regarded as a modeling language of its own.

Class Hierarchies

- ADOxx distinguishes
 - ◆ Classes
 - ◆ Relation classes



Creating new Classes



Teaching Library dynamic - Edit class hierarchy

Class hierarchy:

- ☒ **_D-construct_ (Metamodel)**
- ☒ **_D_event_ (Metamodel)**
- ☒ **_D_variable_ (Metamodel)**
- ☒ **_D_random_generator_ (Metamodel)**
- ☒ **_D_resource_ (Metamodel)**
- ☒ **_D_container_ (Metamodel)**
- ☒ **_D_agent_ (Metamodel)**
- ☒ **_LibraryMetaData_**
- ☒ **_ModelTypeMetaData_**
- ☒ **AnimRep (Metamodel)**
- ☒ **AttrRep (Metamodel)**
- ☒ **Class cardinality (Metamodel)**
- ☒ **ClassAbstract**
- ☒ **ClassName**
- ☒ **ClassVisible**
- ☒ **External tool coupling (Metamodel)**
- ☒ **GraphRep (Metamodel)**
- ☒ **HlpTxt (Metamodel)**
- ☒ **Model pointer (Metamodel)**
- ☒ **Position (Metamodel)**
- ☒ **VisibleAttrs (Metamodel)**
- ☒ **WF_Trans (Metamodel)**
- ☒ **Relation classes**

STRING (Short string)
 LONGSTRING (Long string)
 STRING (Short string)
 INTEGER (Integer)
 STRING (Short string)
 INTEGER (Integer)
 STRING (Short string)
 LONGSTRING (Long string)
 STRING (Short string)
 STRING (Short string)
 STRING (Short string)
 STRING (Short string)
 STRING (Short string)

New ▾

- New class...**
- New attribute...
- New class attribute...

Delete

View ▾

- ☒ Classes
- ☒ Relation classes
- ☒ Metamodel
- ☒ Class hierarchy
- ☒ Attributes
- ☒ Attribute types
- ☒ Source and target classes
- IDs

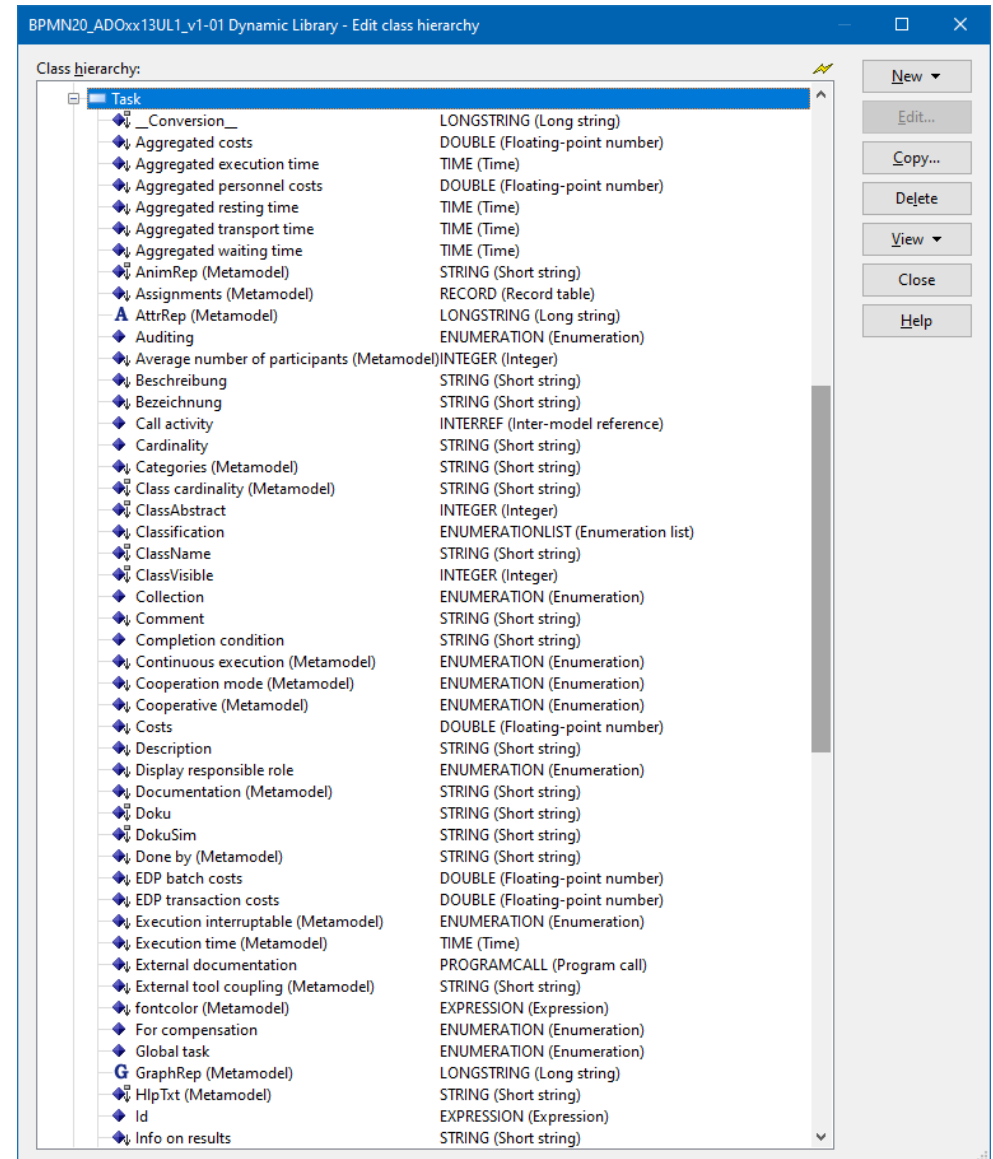
There are predefined abstract classes which have specific functionalit

One can create new classes as subclasses for predefined classes and relations classes

Attributes

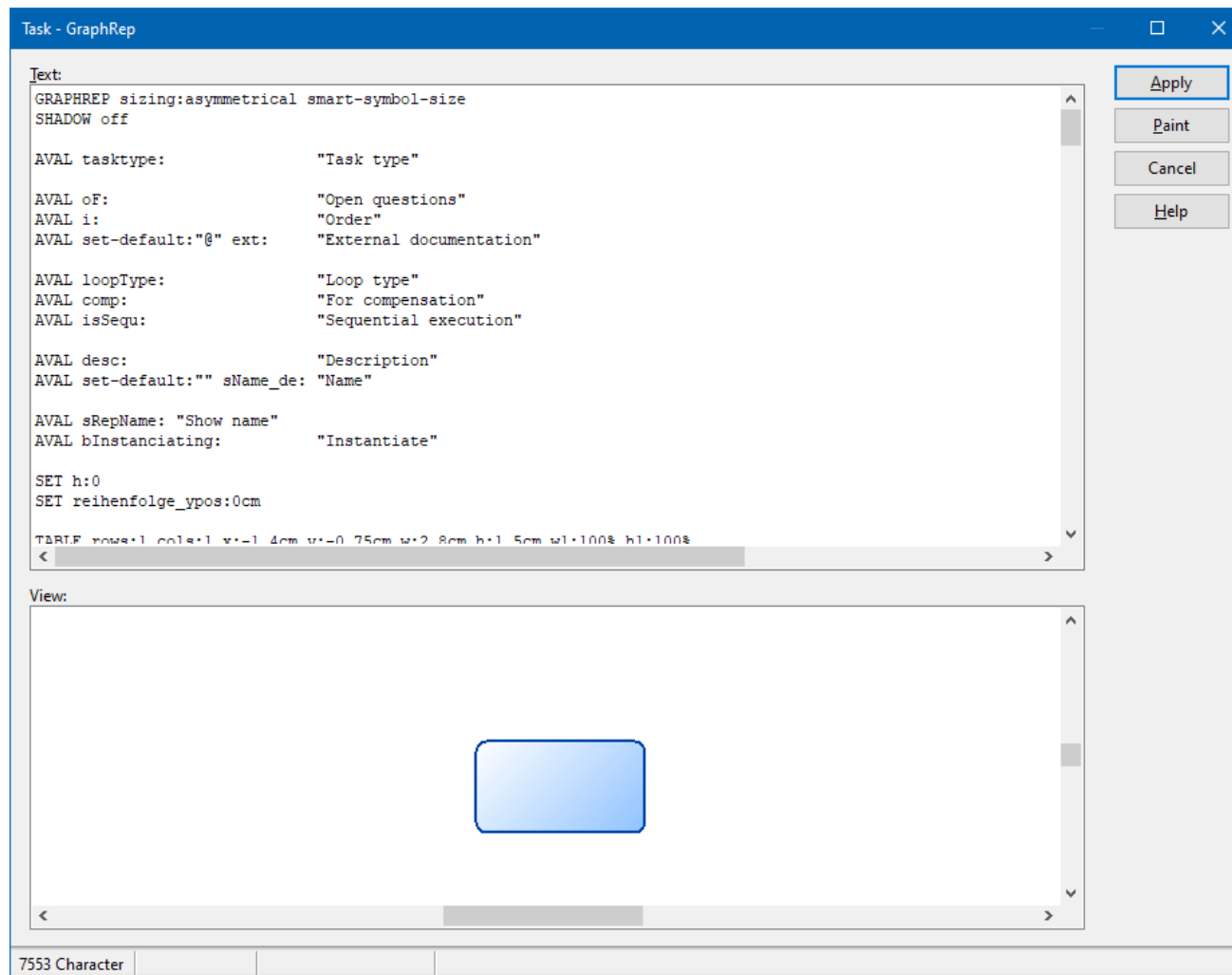
■ Classes and Relations have Attributes

- ◆ Properties of Elements
- ◆ Graphical Representation
- ◆ References



Special Attribute GraphRep

GraphRep: A script language for the graphical representation



References

Referencing a another model, e.g. a subprocess

Referenced subprocess - Edit facets

Standard value:

Attribute type:

INTERREF (Inter-model reference)

Predefined value

Facets

Close

Referenced subprocess - Edit facets

AttributeHelpText:

It is possible to reference a business process model, which is called by the current business process as a subprocess.

The reference can be selected or changed by clicking on the add-icon (symbol 'plus').

If the option 'Display name and reference' is disabled (default setting), the name of the referenced process will be displayed as a hyperlink beneath the object instead of the object name.

If the option 'Display name and reference' is enabled, the name of the referenced process will be displayed as a hyperlink beneath the object name in a smaller font size.

AttributeInterRefDomain:

REFDOMAIN max:1

MODREF

mt:"Business process diagram (BPMN 2.0)"

max:1

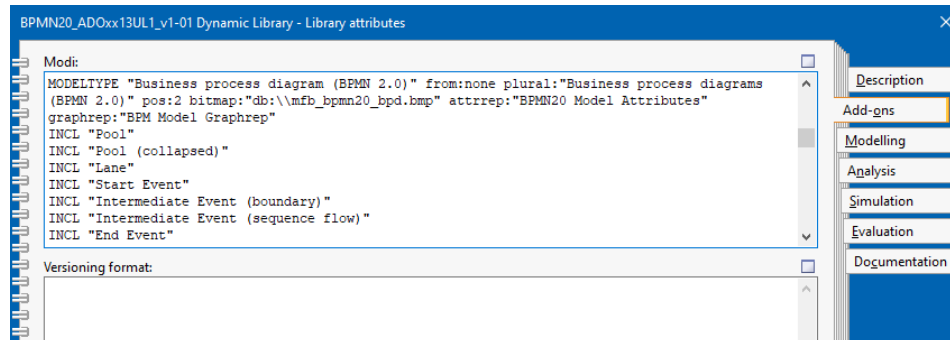
Predefined value

Facets

Close



Classes and Relationss are assigned to Model Types



BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes - Modi

MODELTYPE "Business process diagram (BPMN 2.0)" from:none plural:"Business process diagrams (BPMN 2.0)"
pos:2 bitmap:"db:\\mfb_bpmn20_bpd.bmp" attrrep:"BPMN20 Model Attributes" graphrep:"BPM Model Graphrep"

INCL "Pool"

INCL "Pool (collapsed)"

INCL "Lane"

INCL "Start Event"

INCL "Intermediate Event (boundary)"

INCL "Intermediate Event (sequence flow)"

INCL "End Event"

INCL "Task"

INCL "Sub-Process"

INCL "Exclusive Gateway"

INCL "Non-exclusive Gateway"

INCL "Non-exclusive Gateway (converging)"

INCL "Data Object"

INCL "Message"

INCL "Group"

INCL "Text Annotation"

INCL "Relation Node"

INCL "Variable"

INCL "Random generator"

INCL "Performance Indicator"

INCL "Performance Indicator overview"

INCL "Note"

Apply

Find...

Find next

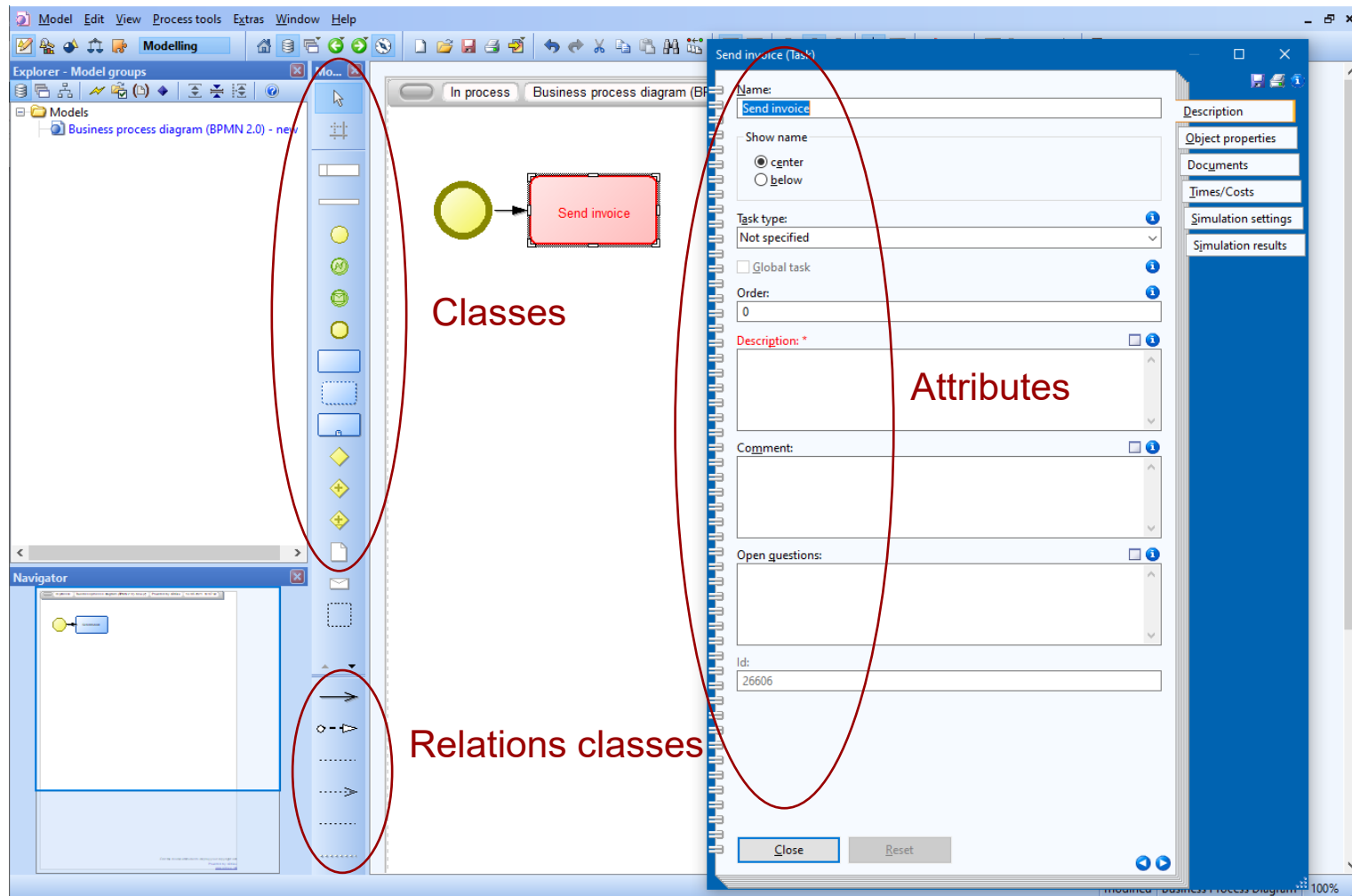
Print...

Cancel

Help

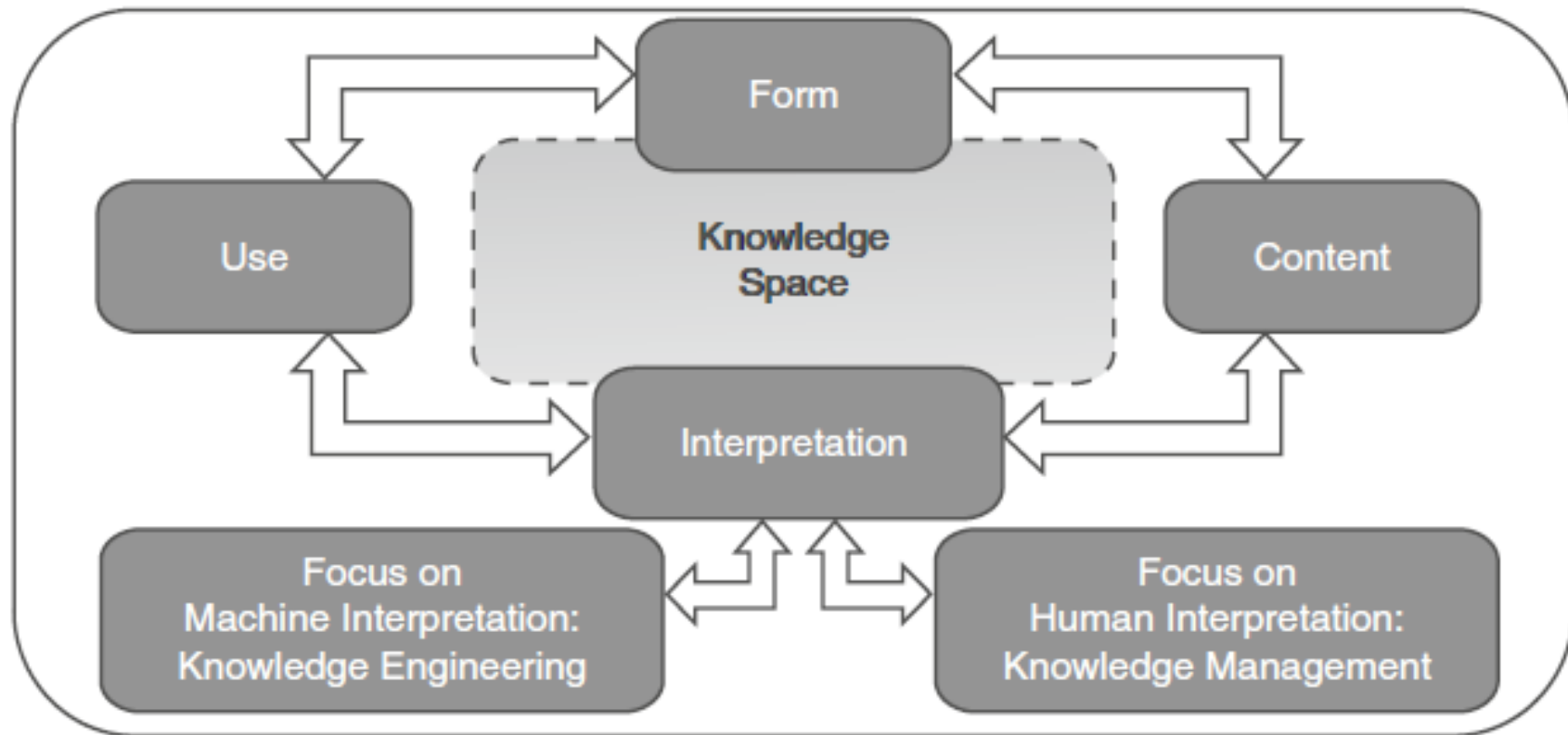


Appearance of Classes and Attributes in the Modelling Toolkit



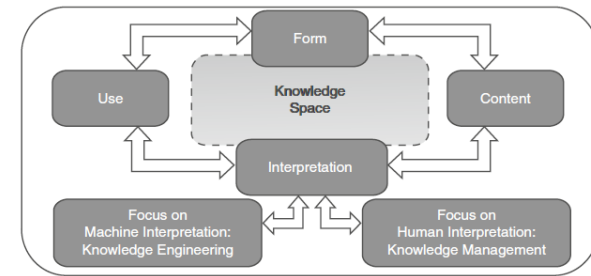
Knowledge in Models

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



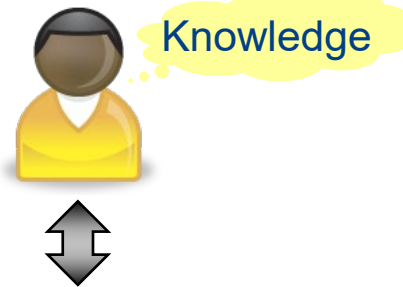
Content: Instantiation of concepts



- **Use:** Stakeholders and their concerns determine the relevant subset of the knowledge
- **Form:** Syntax and semantic of *predefined concepts*.
- **Content:** *Instantiation* of predefined 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

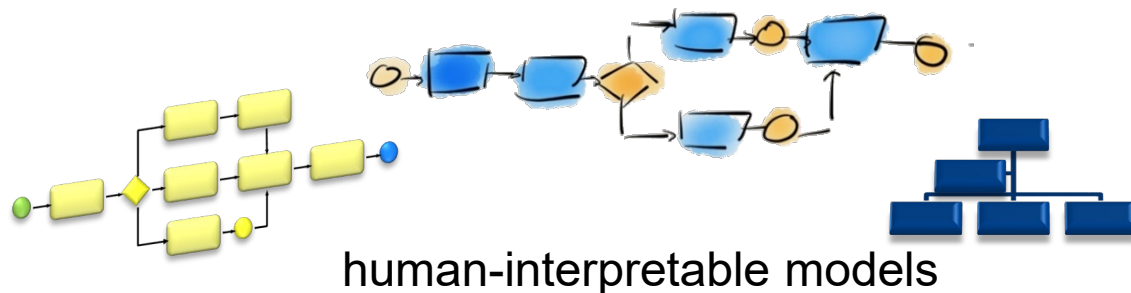
Graphical Models are appropriate for Humans

*Communication/
Analysis/
Decision Making*

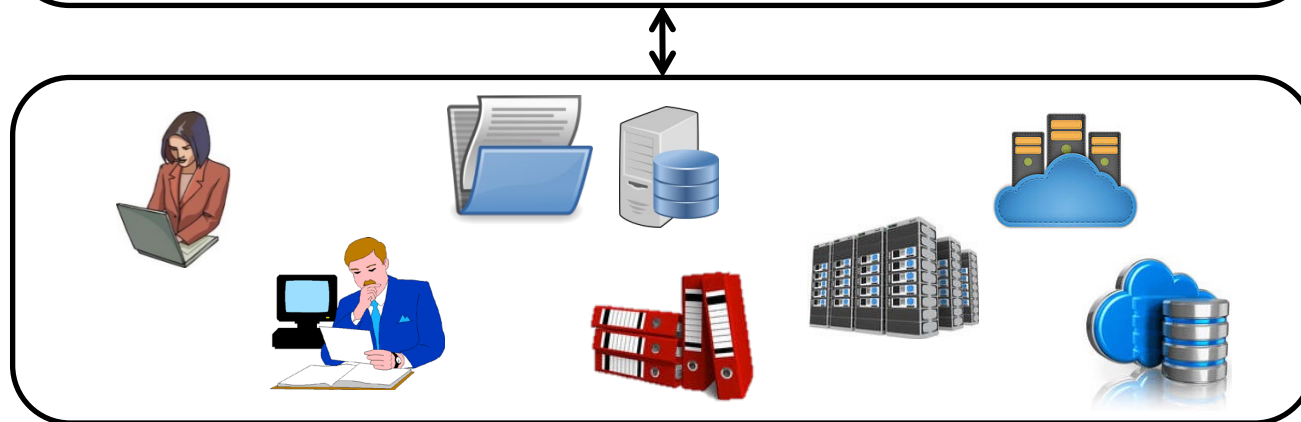


*Examples:
Visio
Powerpoint*

Models



Reality



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 Component
- ◆ Domain: «ERP System» is business software

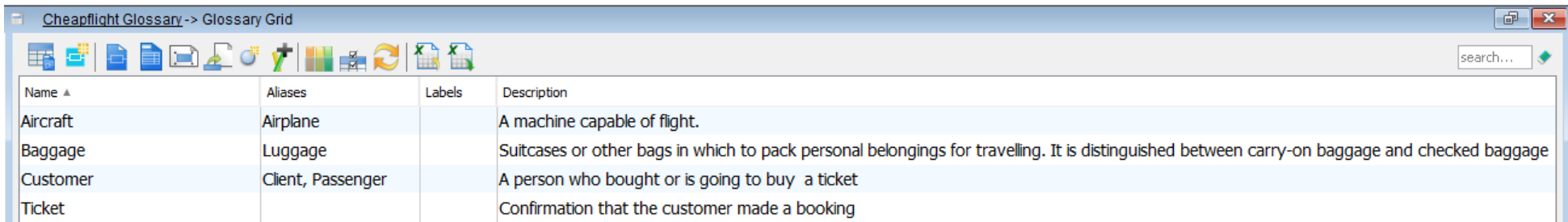


- ◆ Model element: Task
- ◆ Domain: «Cook pasta» is about preparing food

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

Glossary

- A glossary is a collection of terms and their definitions.
- Example: Glossary in Visual Paradigm
 - ◆ Defining the terms used in the labels of models or in Business Rules



Name ▲	Aliases	Labels	Description
Aircraft	Airplane		A machine capable of flight.
Baggage	Luggage		Suitcases or other bags in which to pack personal belongings for travelling. It is distinguished between carry-on baggage and checked baggage
Customer	Client, Passenger		A person who bought or is going to buy a ticket
Ticket			Confirmation that the customer made a booking