



# *Conceptual Modelling*

*Knut Hinkelmann*



## Model

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

There can be different kind of models, e.g.

- logical models
- conceptual model
- graphical model
- textual description
- mathematical model
- physical model

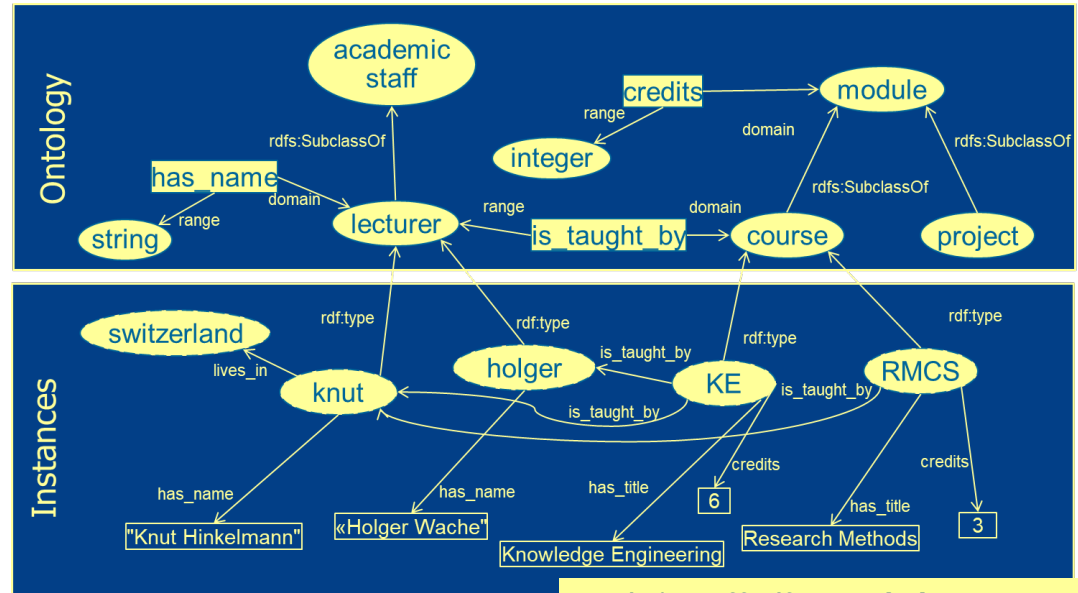
# Knowledge Engineering = Modelling

A Knowledge Base is a representation of reality

Reality



Model



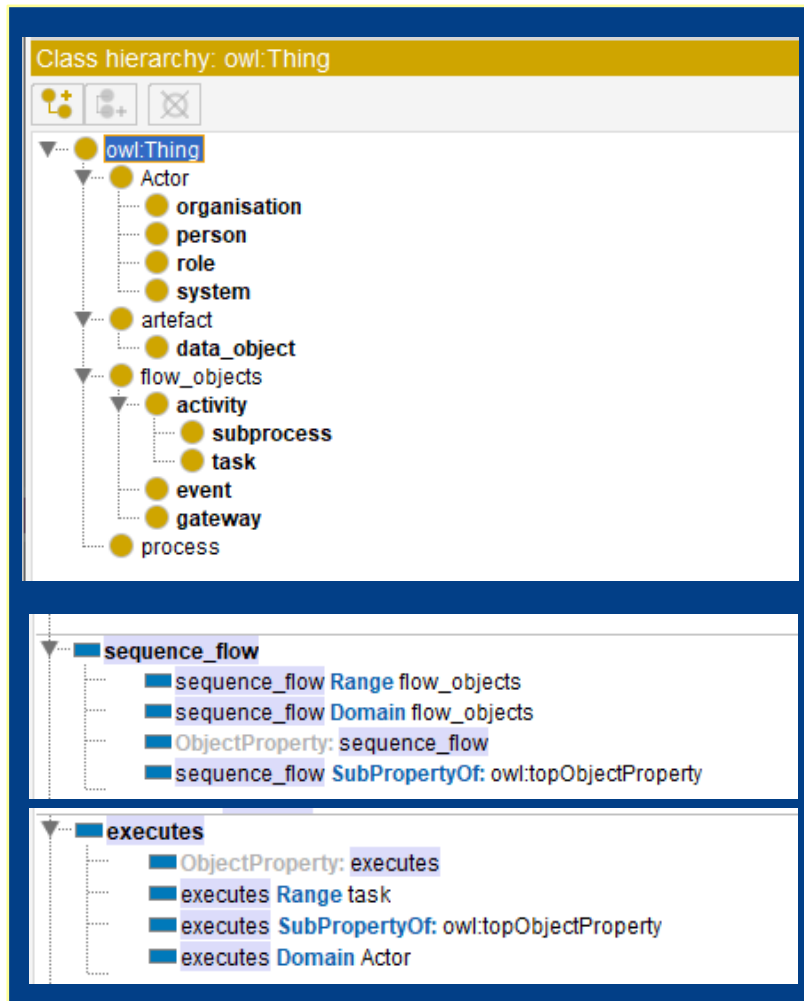
```

:Academic_Staff rdf:type owl:Class .
:lecturer rdf:type owl:Class ;
            rdfs:subClassOf :Academic_Staff .
:module rdf:type owl:Class .
:course rdf:type owl:Class ;
        rdfs:subClassOf :module .
:is_taught_by rdfs:domain :module;
             rdfs:range :lecturer .
:KE rdf:type :course ;
   :is_taught_by :knut ;
   :credits 6 ;
   :title "Knowledge Engineering" .
:knut rdf:type :lecturer ;
     :name "Knut Hinkelmann" .

```

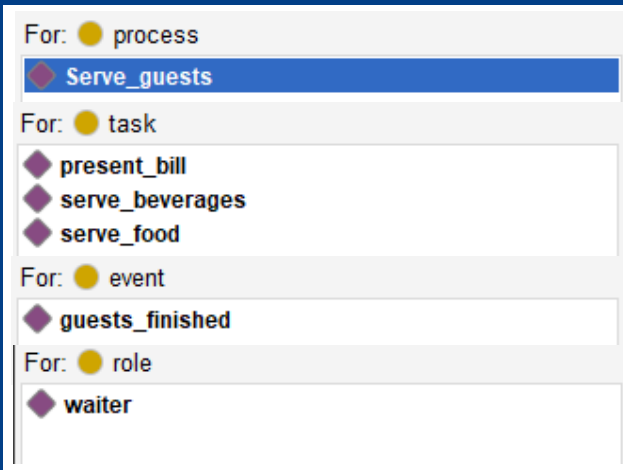
# Example: Concepts and Instances for Process Modelling

## Business Process Ontology (Metamodel):

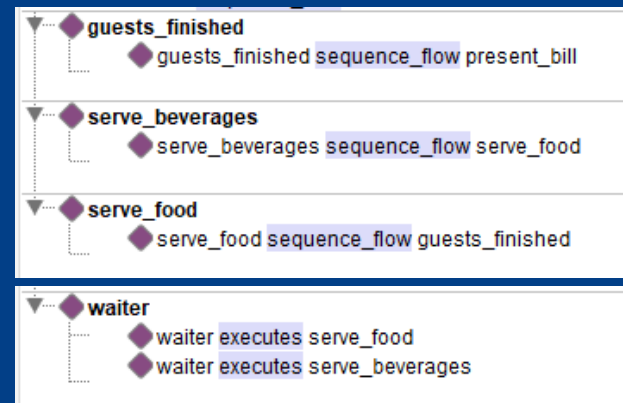


## Process Model for Serve Guests

### Instances:

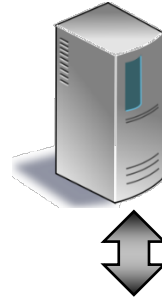


### Relations:



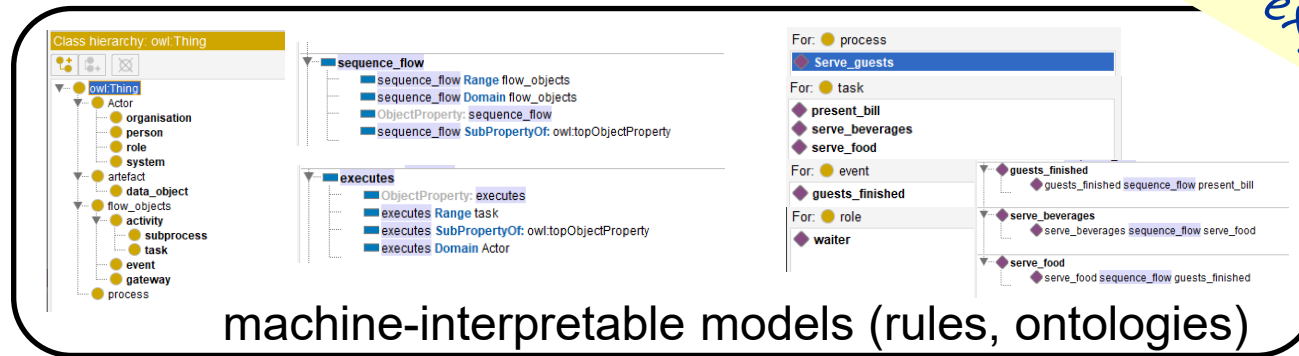
# Knowledge Engineering = Modelling

Reasoning/  
Decision Making



*Creating knowledge graphs requires skills in modeling language – difficult for domain experts*

Models



Reality

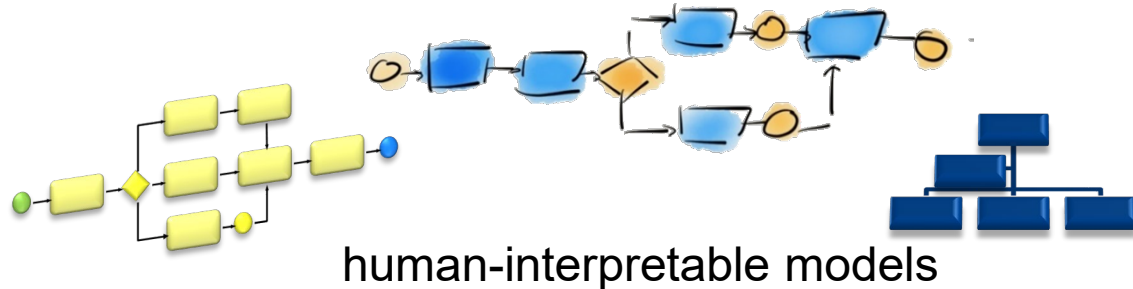


# Graphical Models are appropriate for Humans

*Communication/  
Analysis/  
Decision Making*



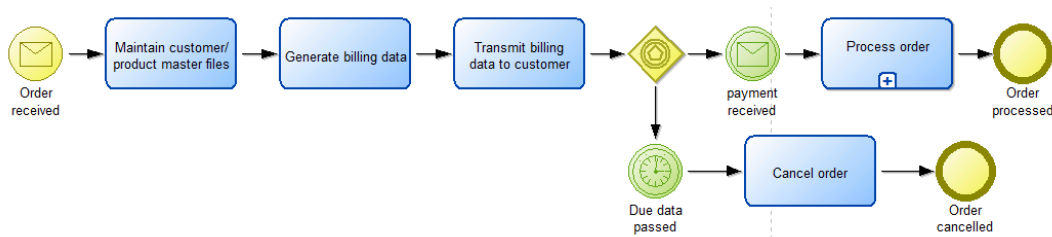
*Models*



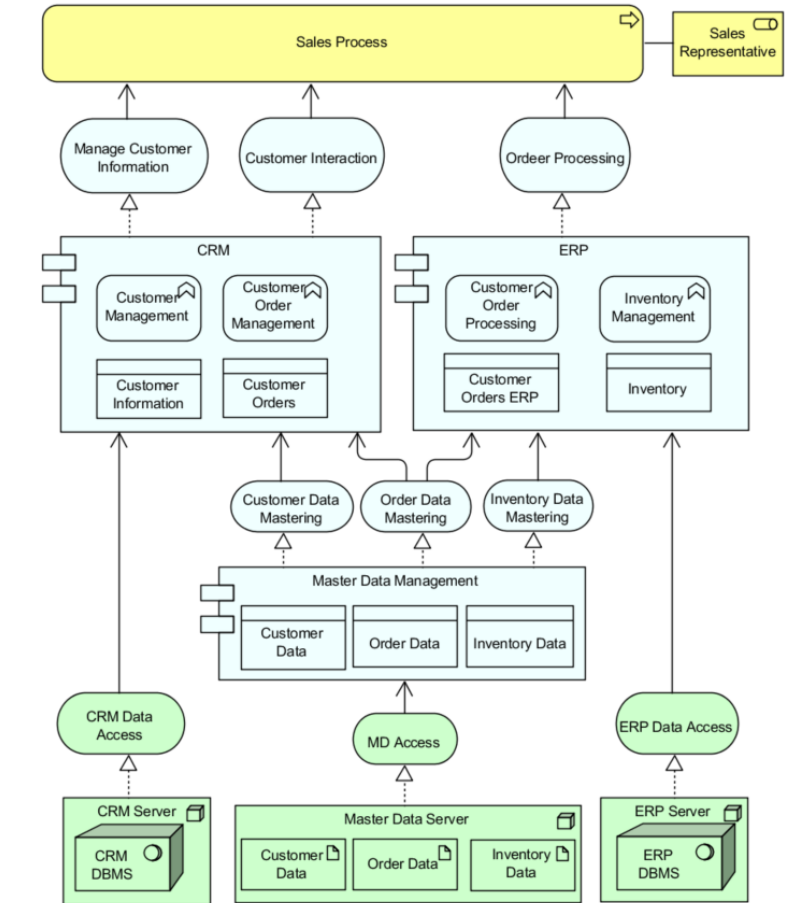
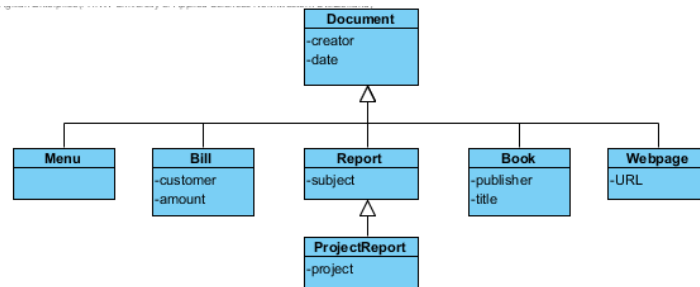
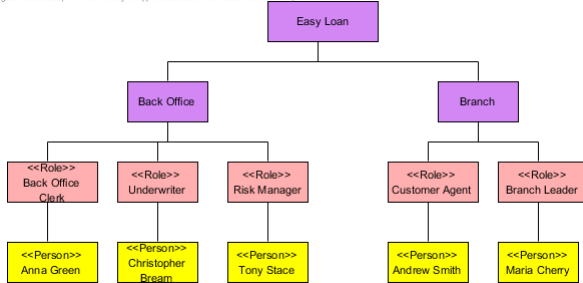
*Reality*



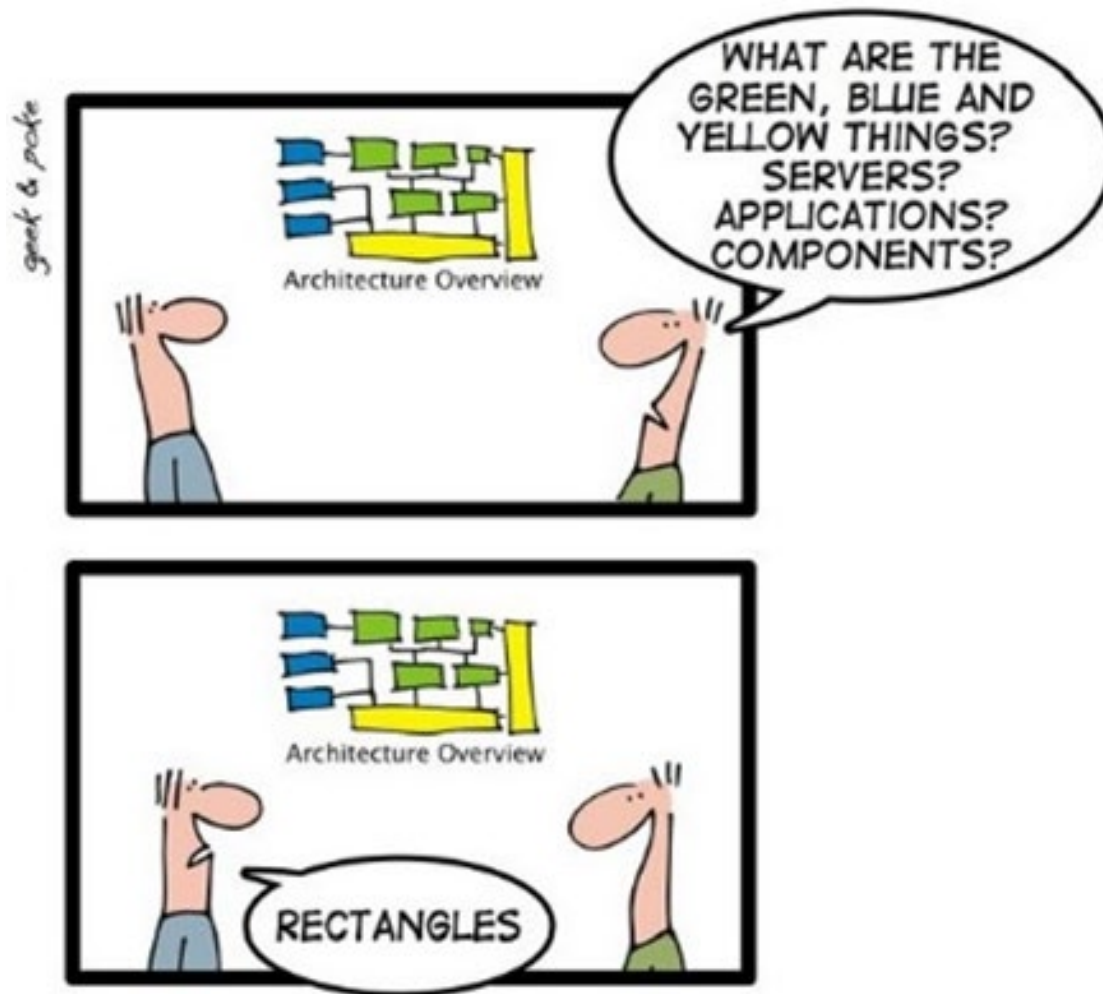
# Enterprise Models using Domain-Specific Graphical Modeling



Agilan Simulacran/FHNW University of Applied Sciences Northwestern Switzerland



# Drawing



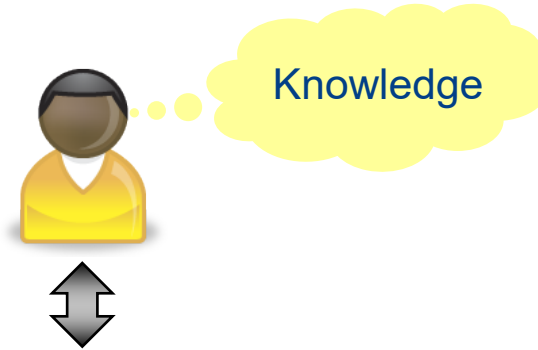


# Models

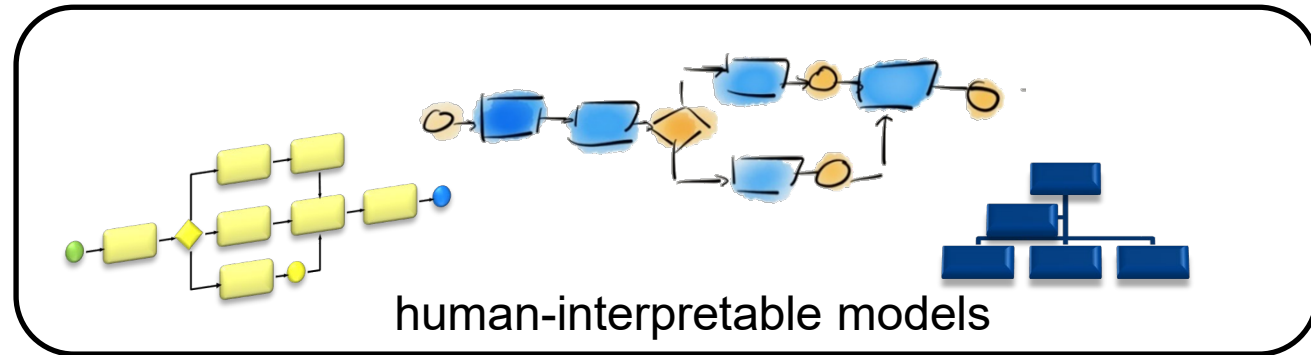
- Models are not mere pictures; rather, they
  - ◆ provide a precise, meaningful description that can be visualized in different ways for different stakeholders;
  - ◆ can also be used to analyze the impact of changes, cost, risk, security, compliance and other relevant KPIs.

# Humans use Knowledge to interpret Models

*Communication/  
Analysis/  
Decision Making*



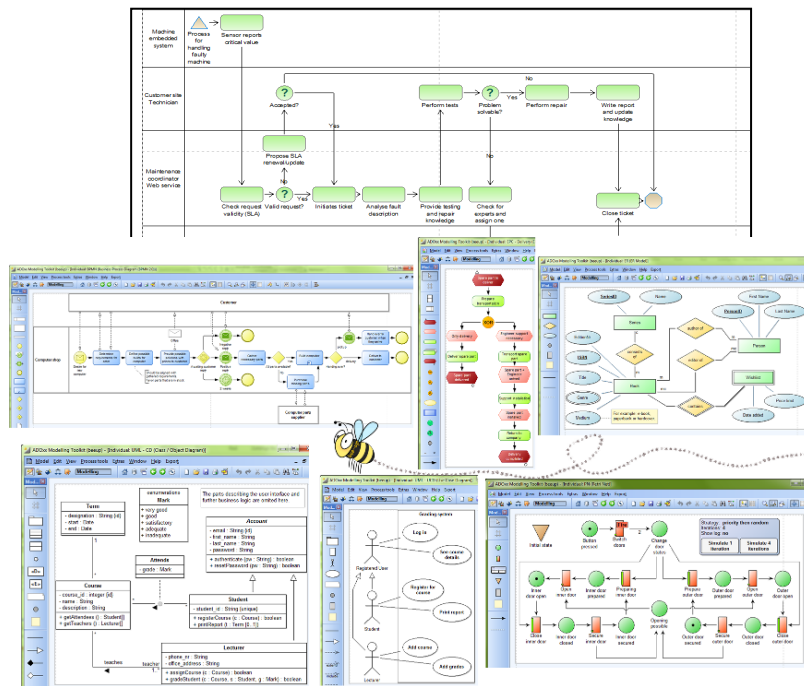
*Models*



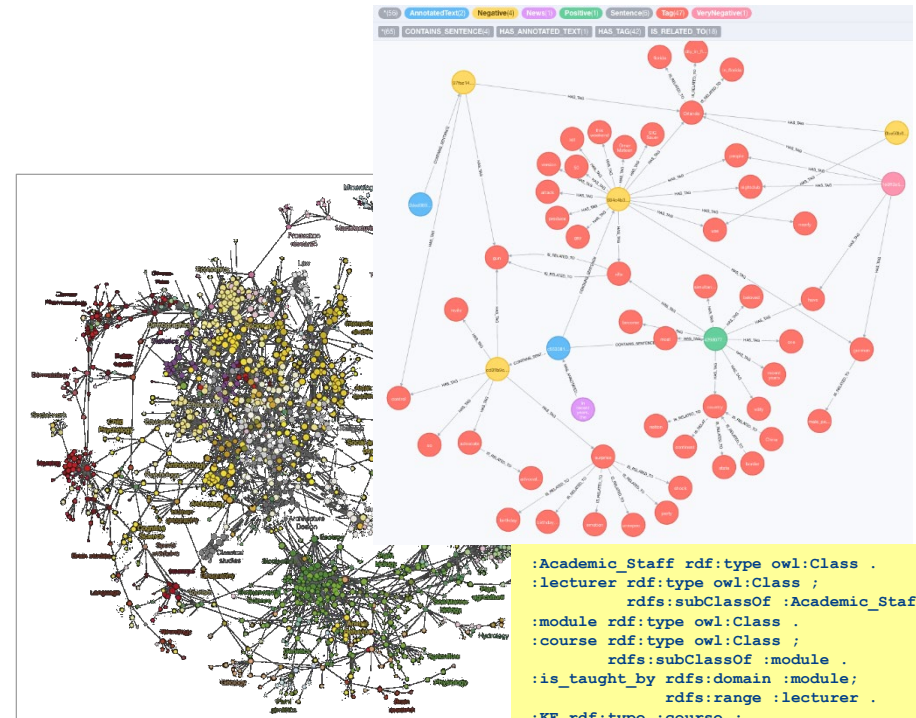
*Reality*



# Conceptual Graphical Models



# Knowledge Graphs



```

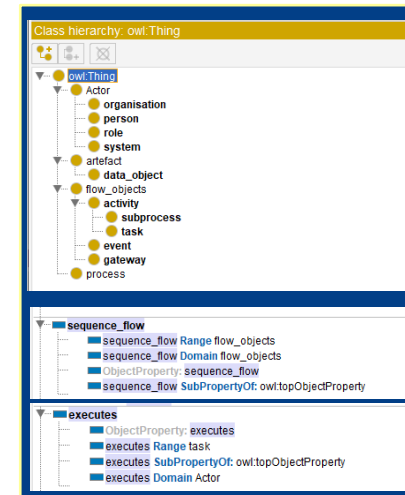
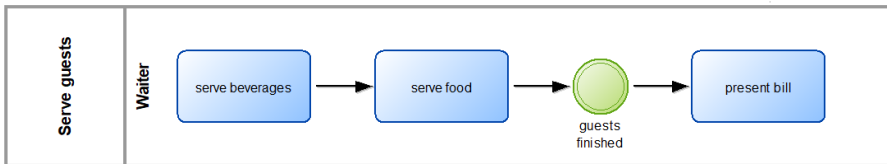
:Academic_Staff rdf:type owl:Class .
:lecturer rdf:type owl:Class ;
              rdfs:subClassOf :Academic_Staff .
:module rdf:type owl:Class .
:course rdf:type owl:Class ;
         rdfs:subClassOf :module .
:is_taught_by rdfs:domain :module ;
              rdfs:range :lecturer .

:KE rdf:type :course ;
    :is_taught_by :knut ;
    :credits 6 ;
    :title "Knowledge Engineering" .
:knut rdf:type :lecturer ;
      :name "Knut Hinkelmann" .
    
```

Modeling using predefined *concepts*

# Objective: Combining Knowledge Graphs and Graphical Modeling

- Creating knowledge graphs is difficult for domain experts – it requires skills in modeling language
- Creating graphical models is more adequate for domain experts
- Objective: Create ontologies (knowledge graphs) from graphical models



# *Conceptual Modelling*

## **Conceptual Modelling**


Creating models using predefined concepts.

Conceptual Modeling consists of two phases

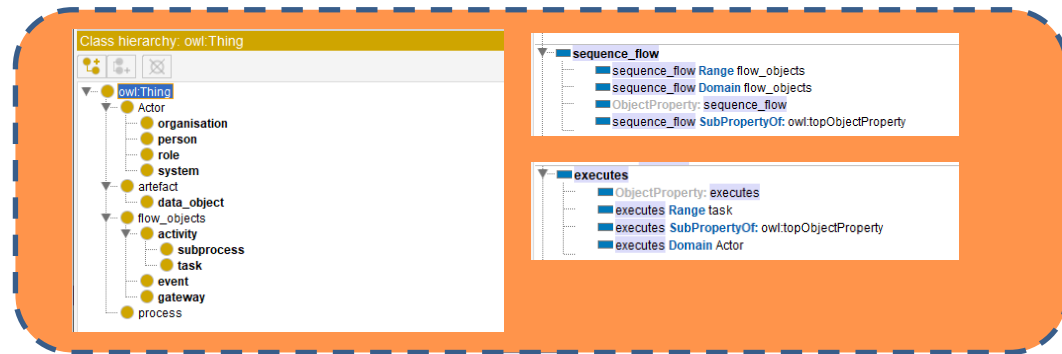
Metamodeling: Defining (domain-specific) concepts

Modeling: Creating models by instantiating these concepts


# Conceptual Modeling using Ontologies



Metamodel Engineer



Meta-modeling

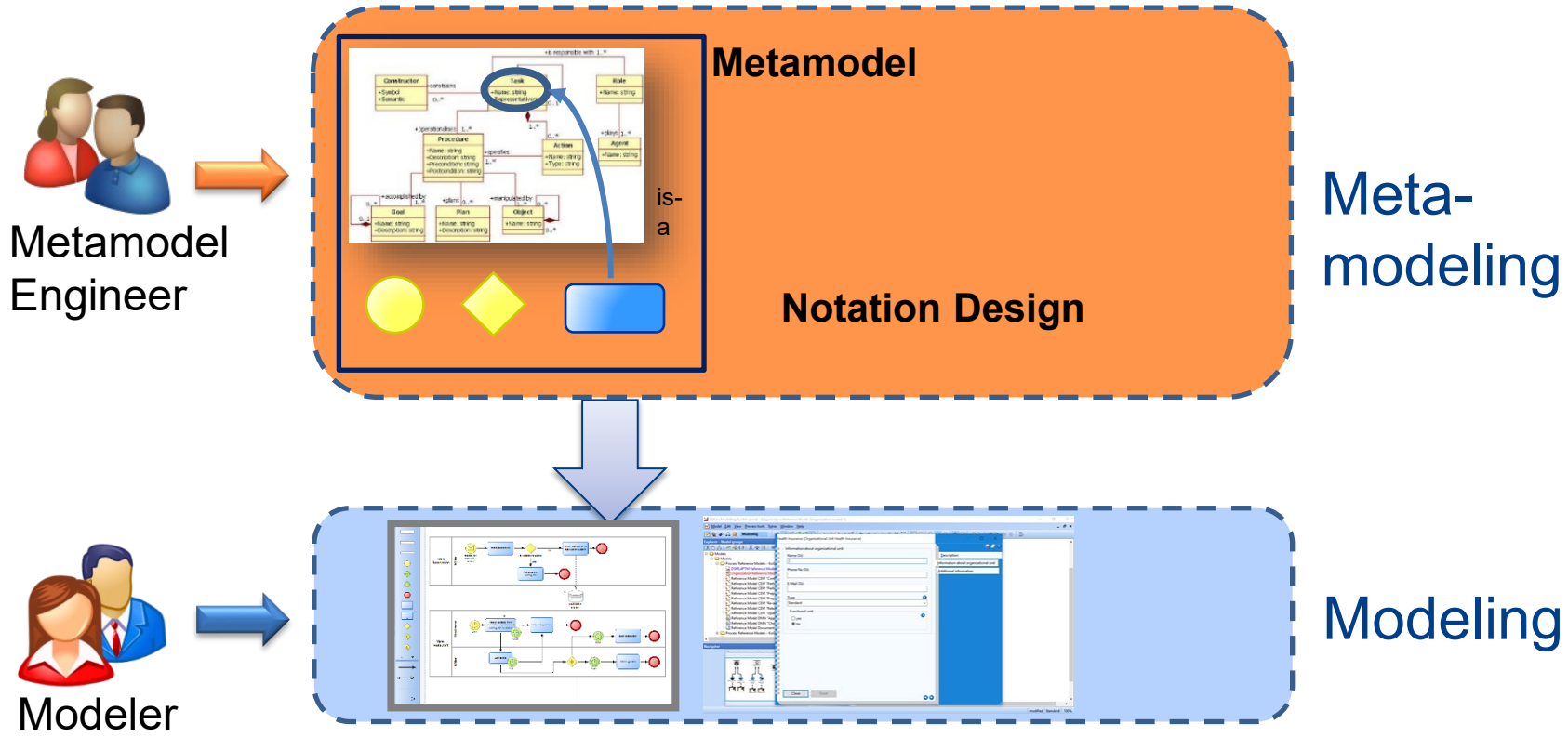


Modeler



Modeling

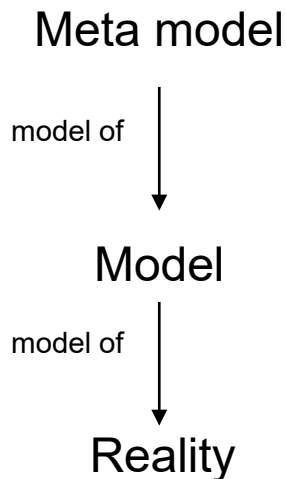
# Conceptual Modeling with Graphical Models



## Meta-model

### Meta Model

The specification of the domain-specific concepts that can be used for modeling



- 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



# Concepts for Business Process Models

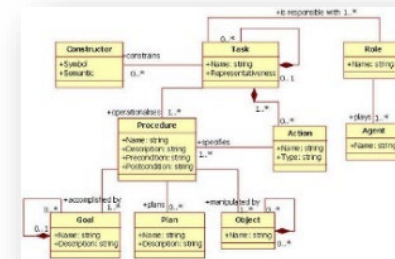
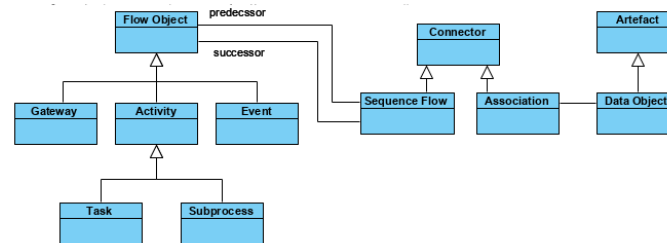
## Metamodel:

Concepts which can be used to create models.

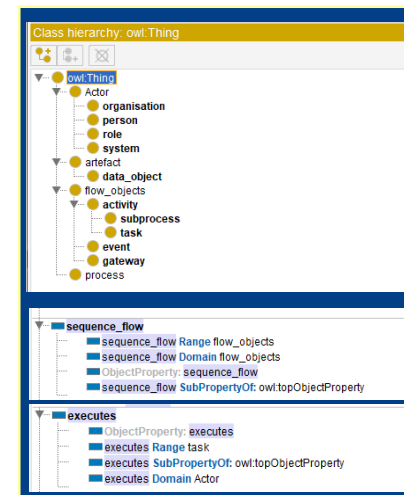
Example: A process model consists of concepts for

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

Metamodel represented as a class diagram or data model:



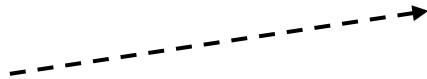
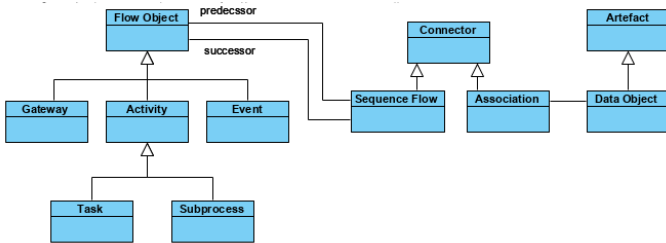
Metamodel represented as a knowledge graph:



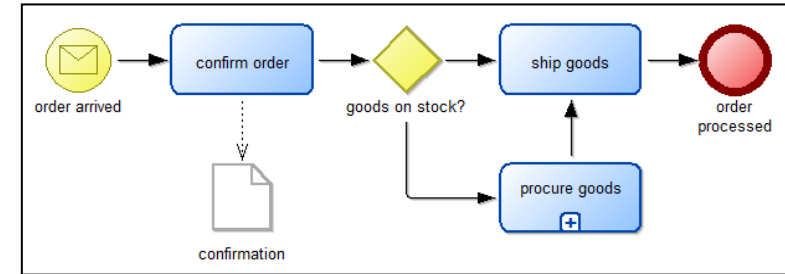
# A Model is the Instantiation of the Metamodel

## Metamodel:

Concepts which can be used to create models.



## Model:

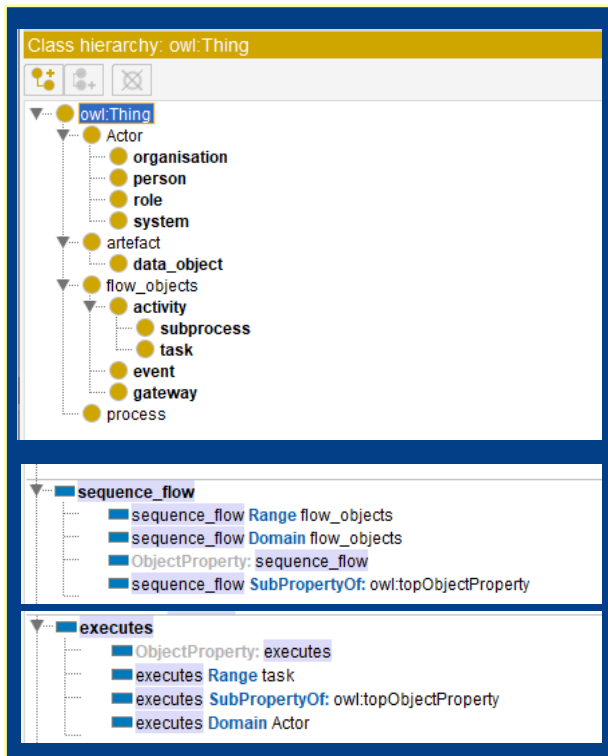


*A model contains instances of the concepts defined in the meta-model. The object „confirm order“ represents a real entity; it is an instance of the concept «task»*

# A Model is the Instantiation of the Metamodel

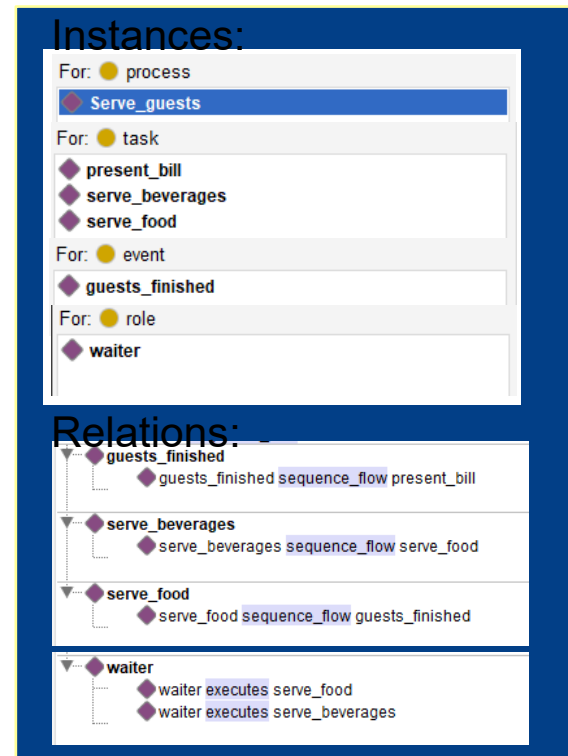
## Metamodel:

Concepts which can be used to create models.



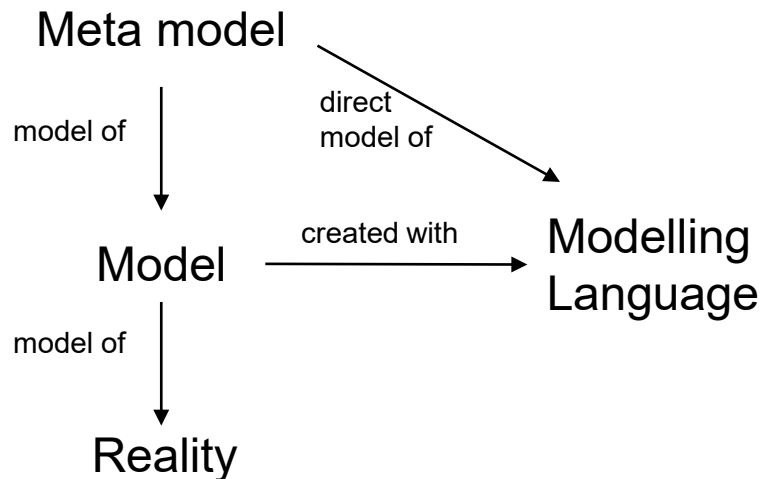
## Model:

A model contains instances of the concepts defined in the meta-model.



# Modelling Language

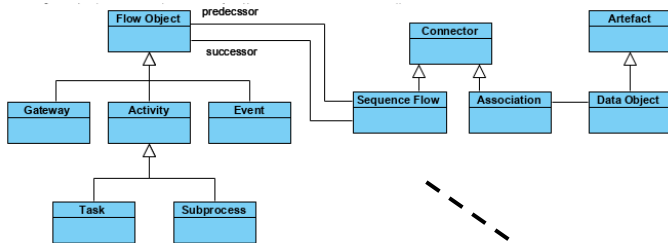
- A **modelling language** specifies the notation for the concepts, from which a model can be made.
- There are different kinds of notations
  - ◆ For graphical models the notation consists of *visualization* of the concepts
  - ◆ Textual models consist of words
  - ◆ Mathematical models use symbols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



# Modeling Language extends Metamodel with Notation

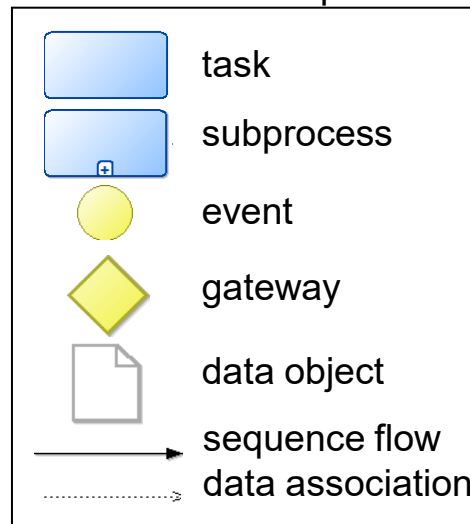
## Metamodel:

Concepts which can be used to create models.

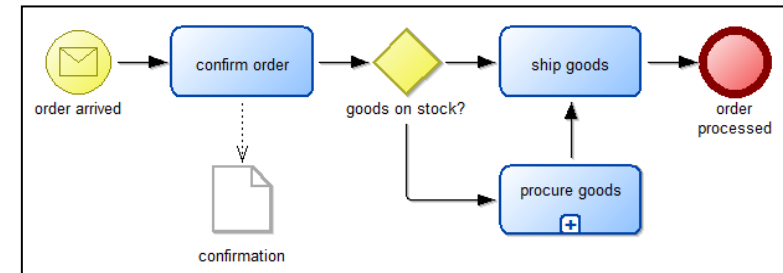


## Modelling Language:

Notation/appearance of meta-model concept



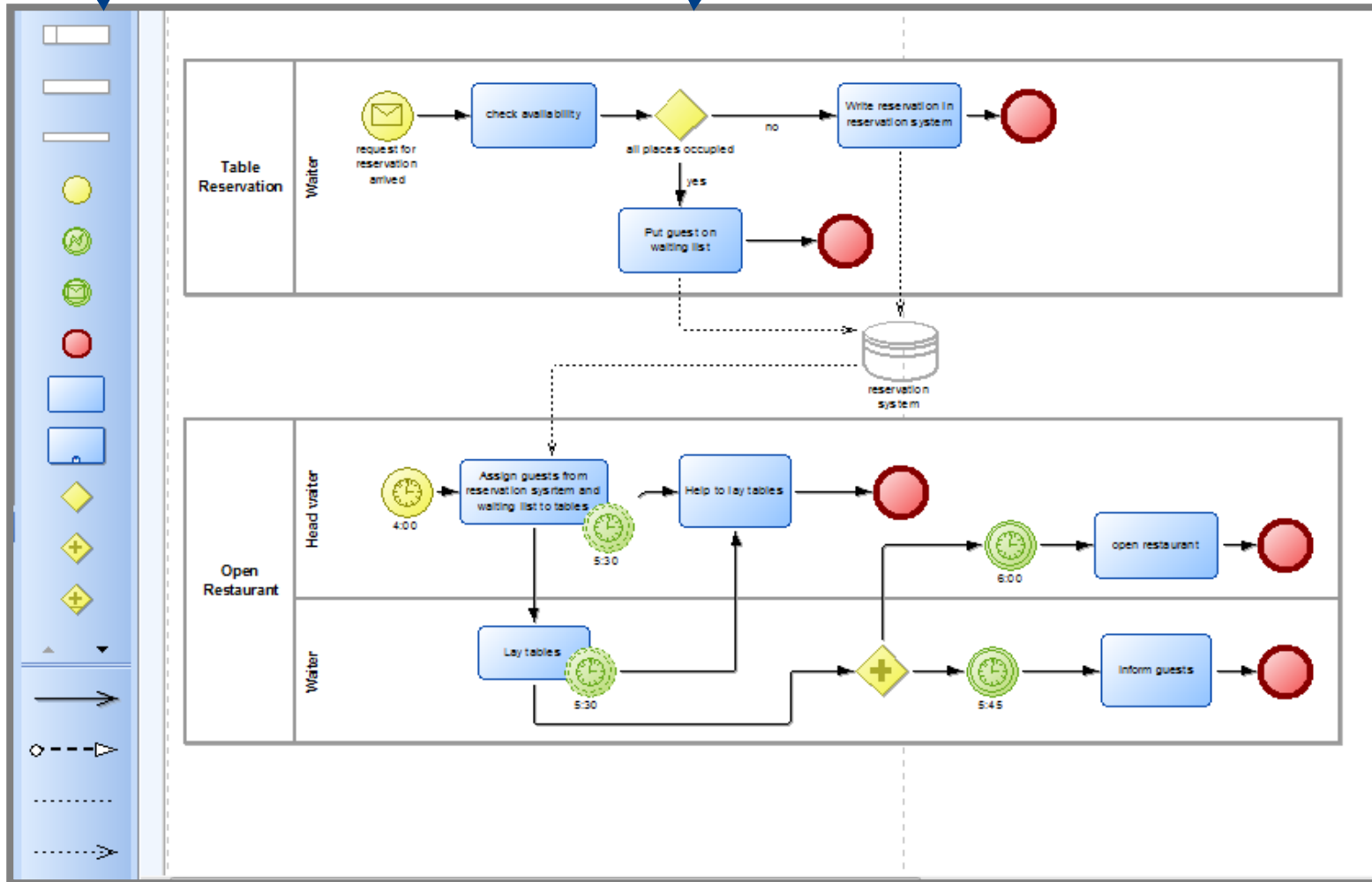
## Model:



*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»*

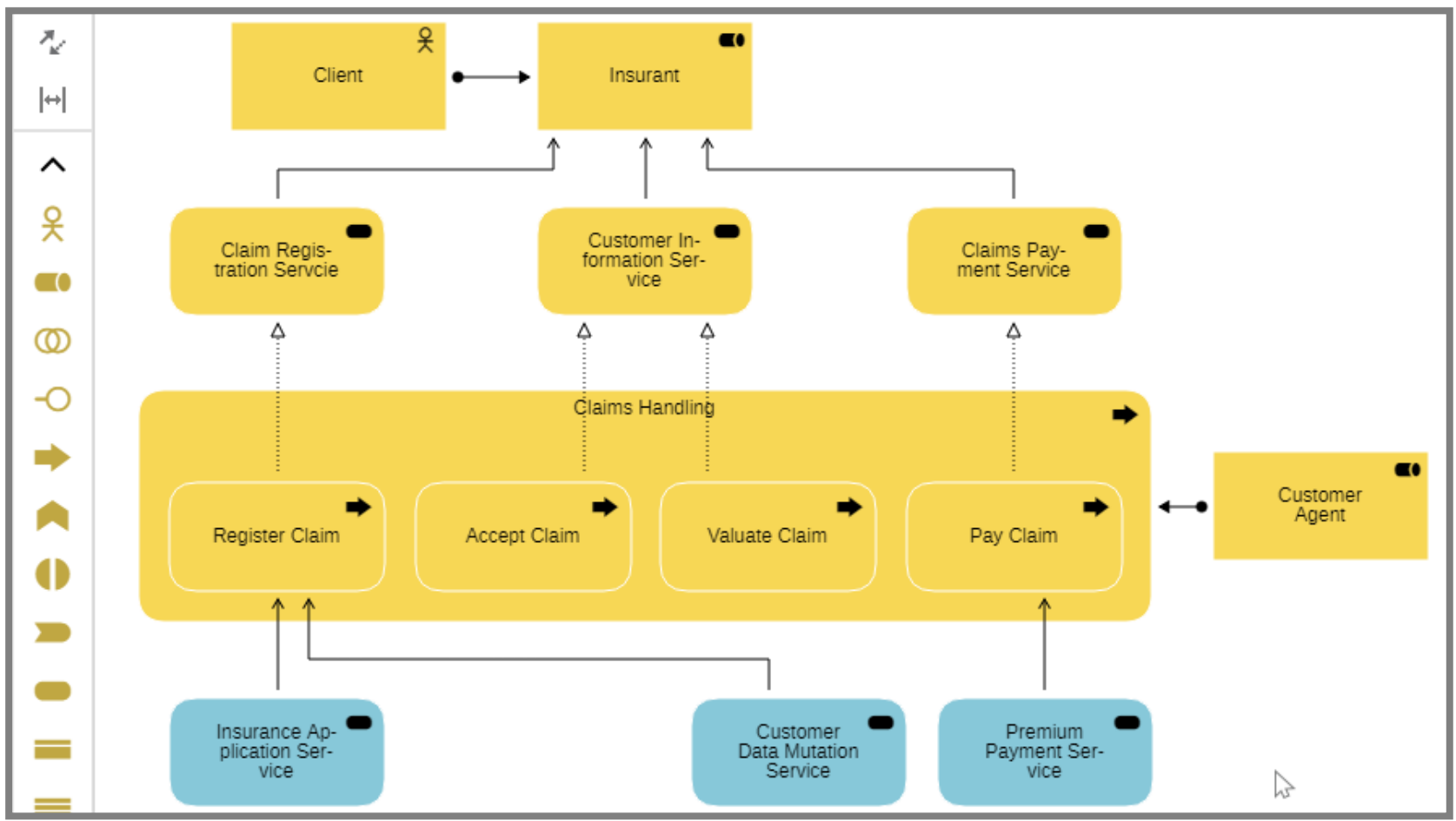
Modelling Language

Model



Modelling Language

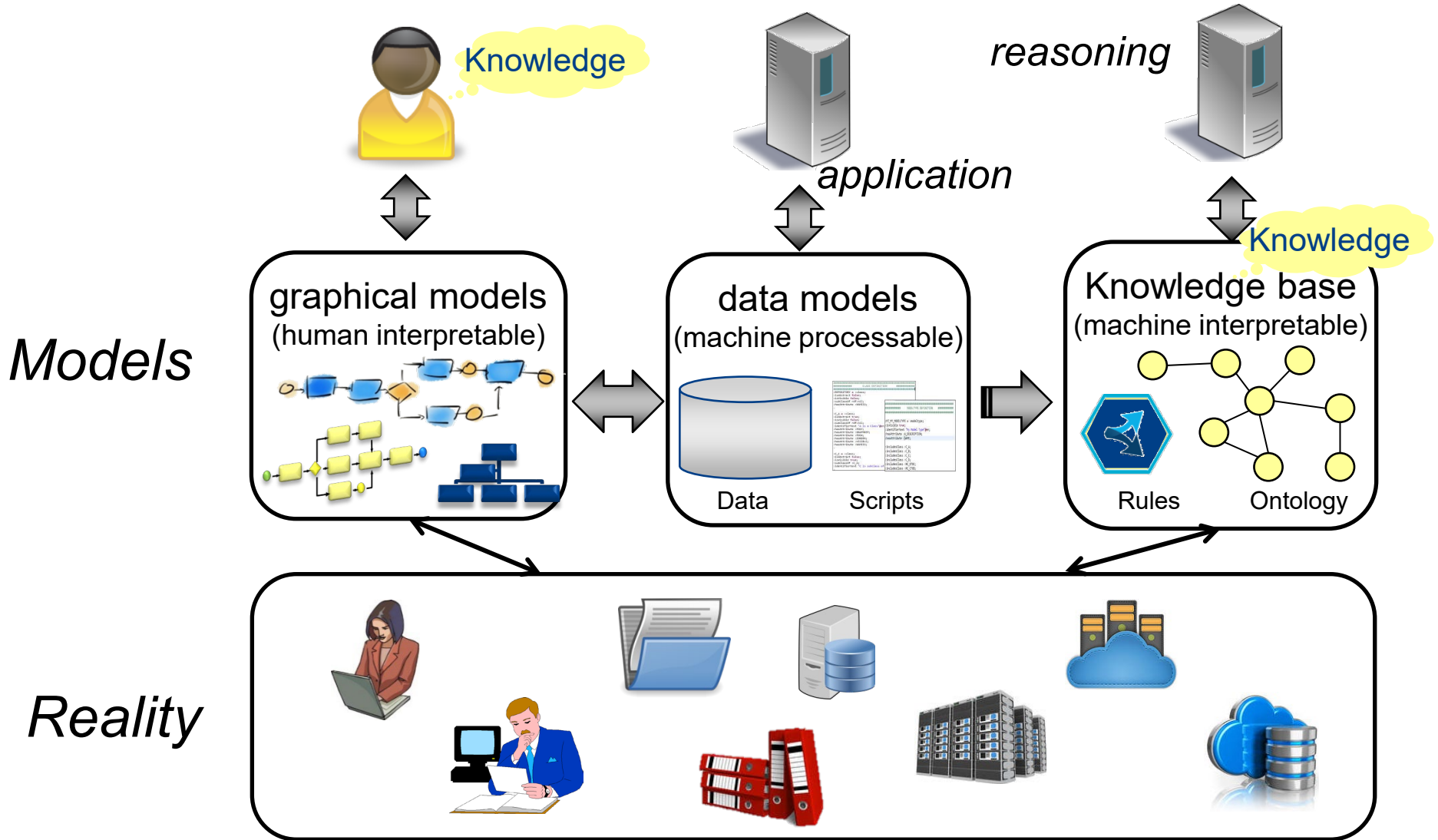
Model



# *Semantic Lifting*

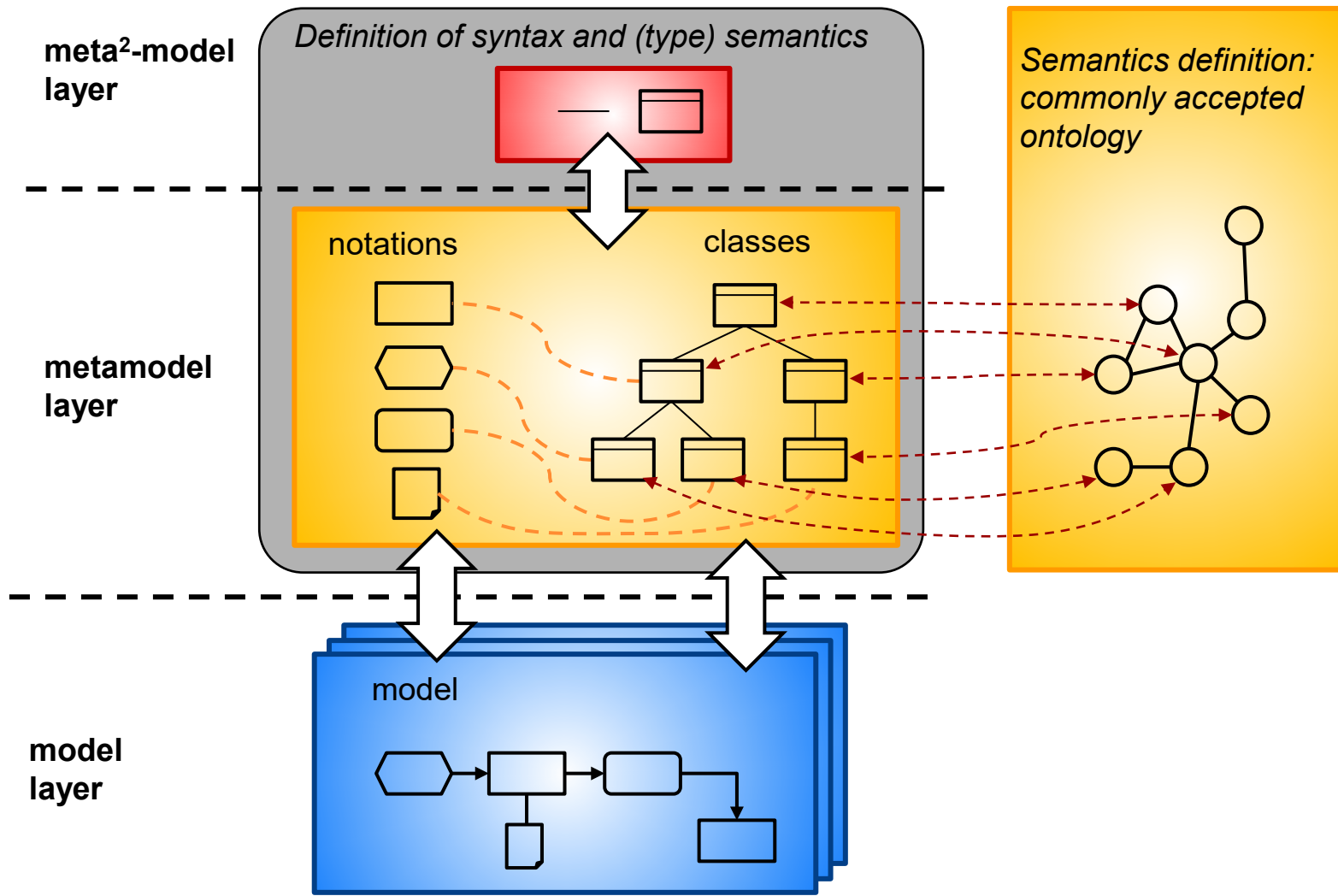


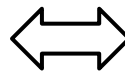

# Semantic Lifting: Map Models into an Ontology



# Modelling Environment

# Ontology



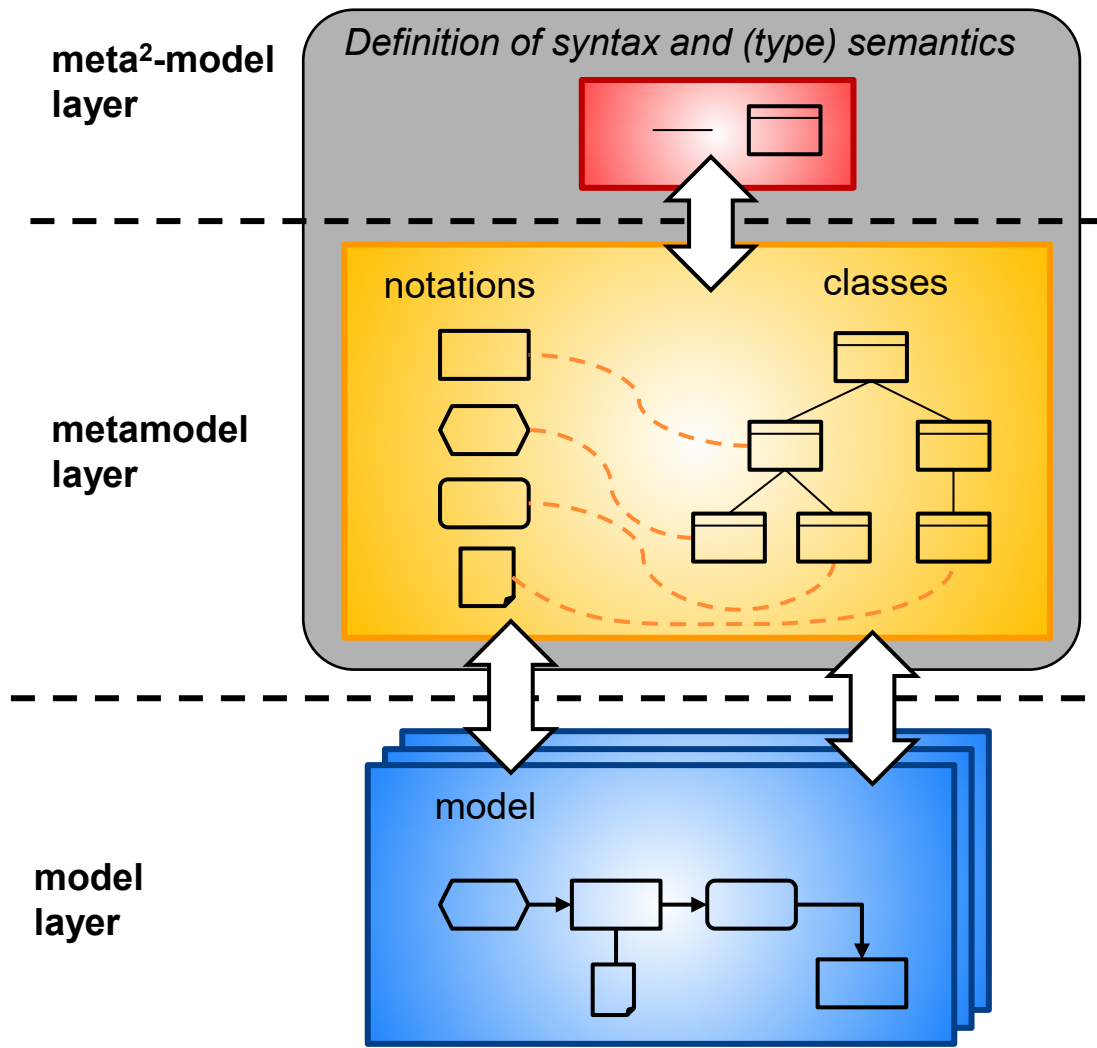
 **linguistic metamodelling:** *syntax, structure, type semantics*  
 **ontological metamodelling (lifting):** *explication of type semantics*

# Agenda

- Conceptual Modeling with graphical models (today)
- Combining graphical modeling with knowledge graphs (next lecture)

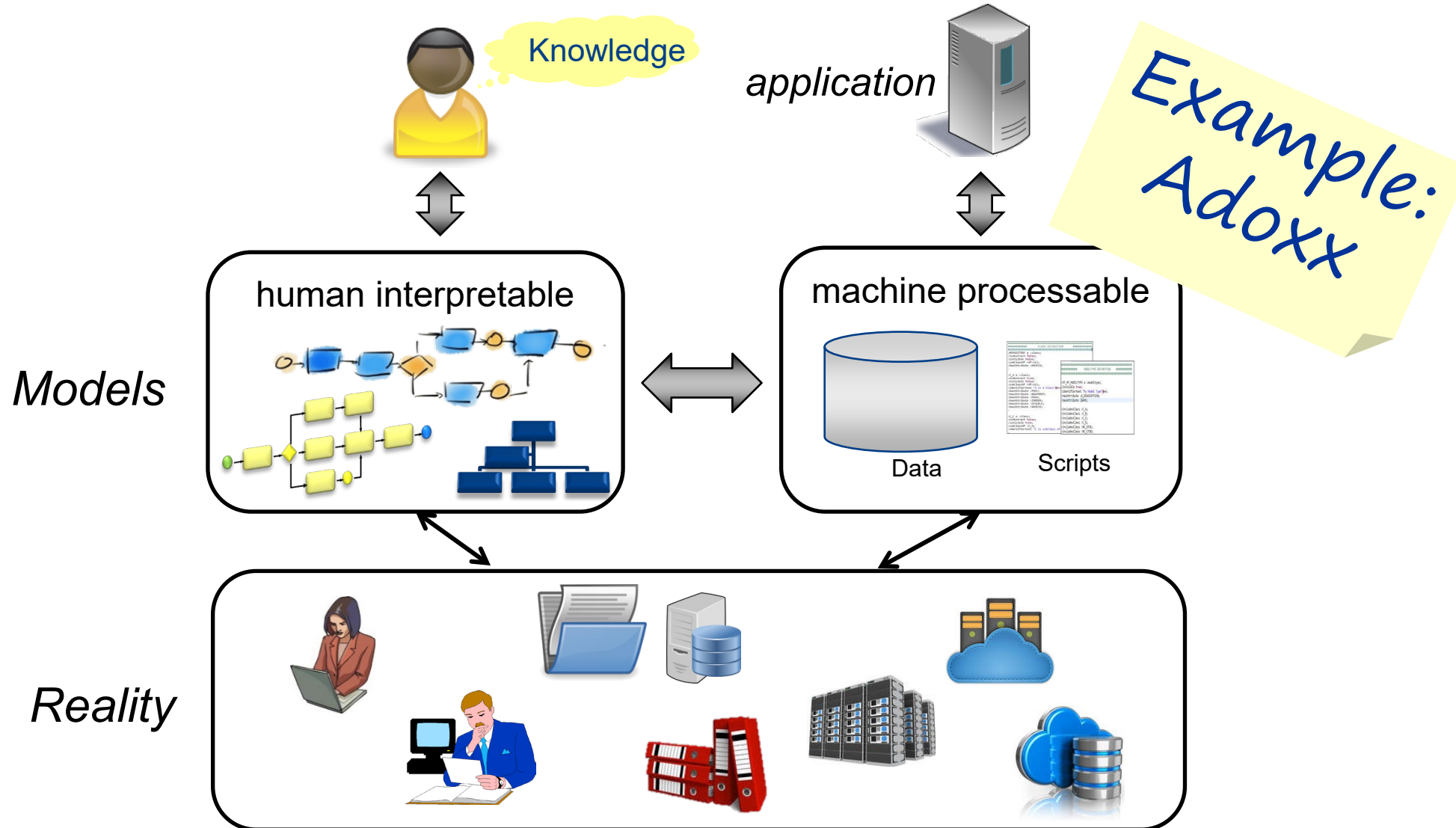
# *Metamodelling with ADOxx*

# Modelling Environment



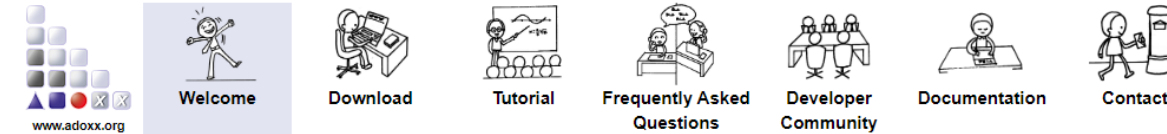
**linguistic metamodeling: syntax, structure, type semantics**

# Graphical Models represented in a Database



# 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](mailto:tutorial@adoxx.org)




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 by @ADOxxORG

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



Mar 28, 2020

# OMiLAB – A Conceptual Modelling Community

ADOxx is the basis for OMiLAB



The screenshot shows the OMiLAB website interface. At the top, there is a navigation bar with the contact information: [info@omilab.org](mailto:info@omilab.org) and a phone number: **Call Us Now: +49 30 2636 7863**. The main header features the **OMiLAB** logo and a menu with items: **HOME**, **OMILAB NODES**, **ACTIVITIES**, **ABOUT US**, and **CONTACT**. The main content area is a large graphic with a network of white circles and lines on a dark blue background. Three prominent circles are labeled: **Design Thinking**, **Conceptual Modelling**, and **Digital Innovation**. A red button on the right says **READ MORE ON THE OMILAB DIGITAL ECOSYSTEM**. Below the graphic, there are three navigation buttons: **OMiLAB Nodes**, **Projects & Events**, and **Scene2Model**. The **Scene2Model** logo is also visible at the bottom center.

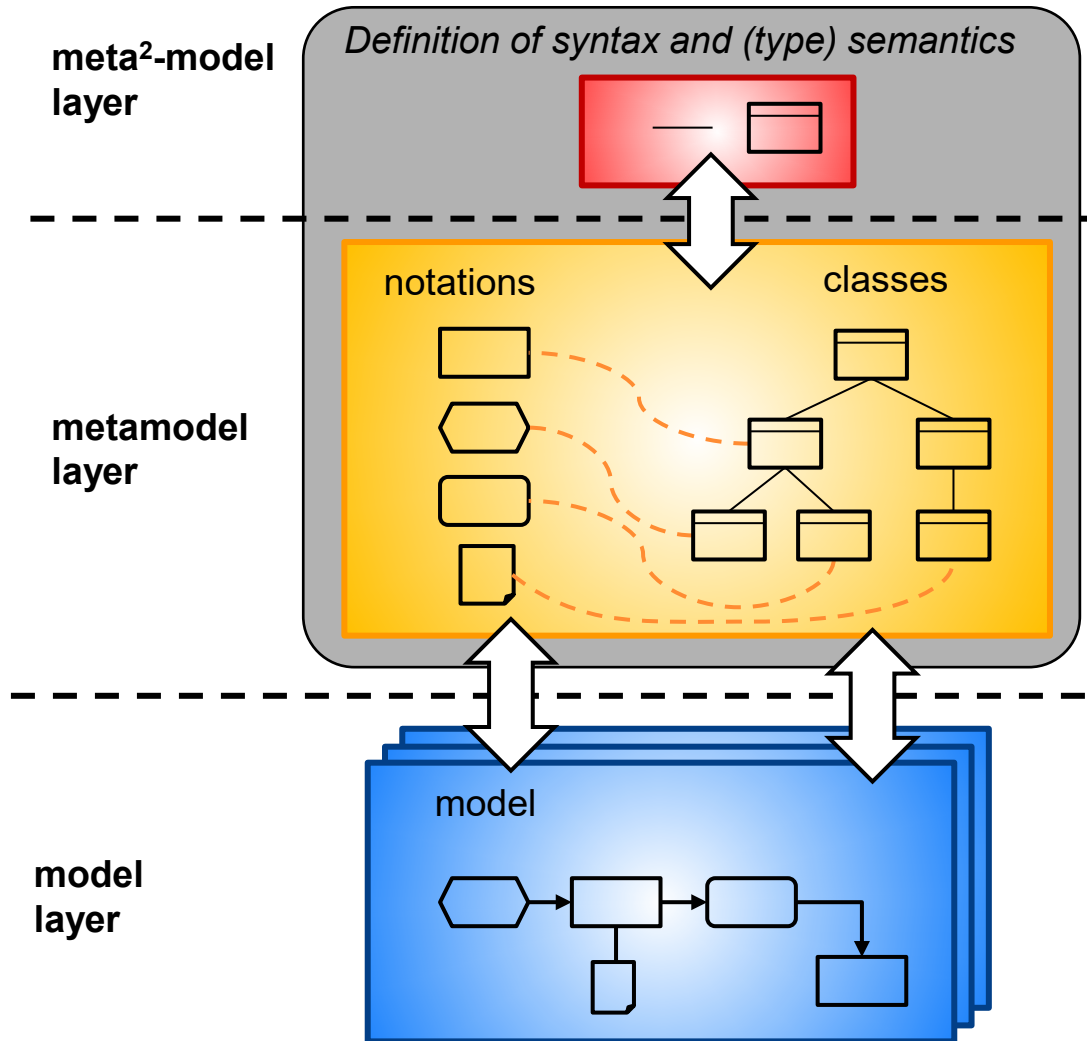




# The ADOxx Environment

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

# Modeling Environment



ADOxx Development Toolkit

ADOxx Modeling Toolkit

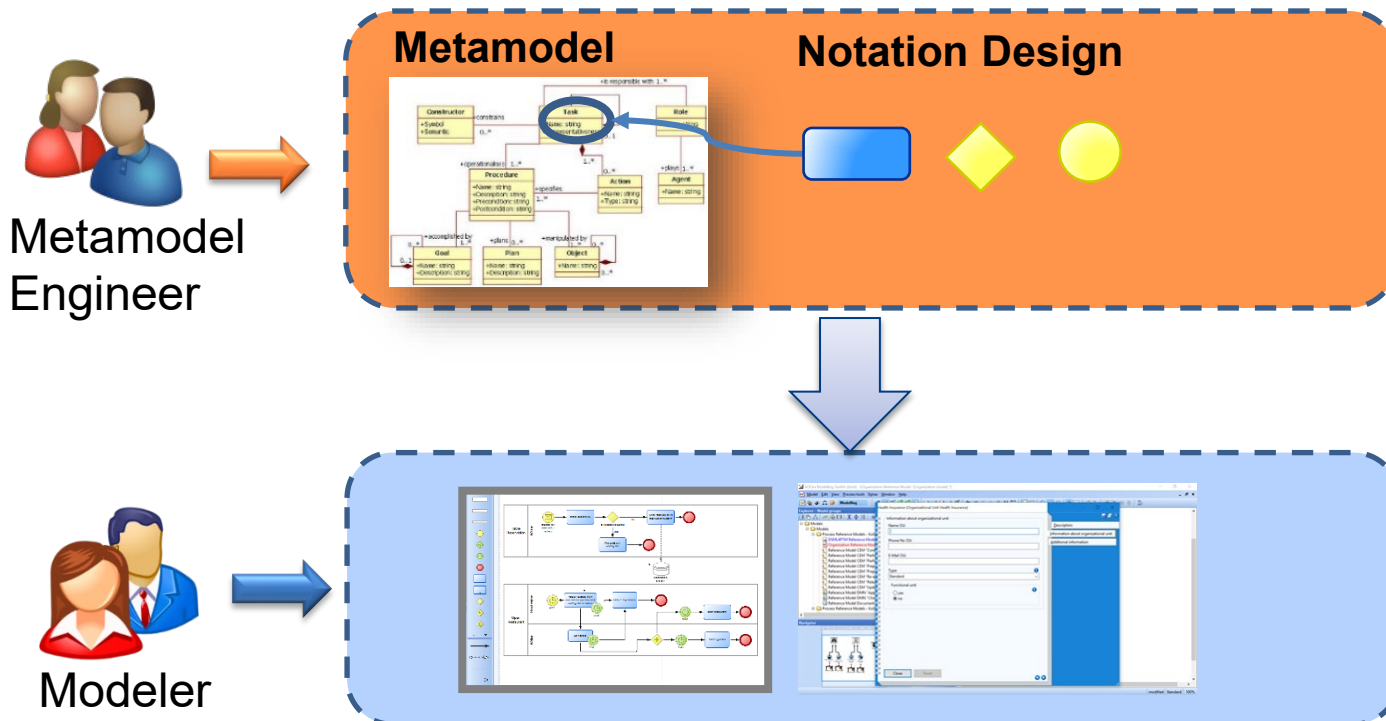
# Modeling and Metamodeling

Conceptual Modeling consists of two phases

Metamodeling: Defining (domain-specific) concepts

Modeling: Creating models using these concepts

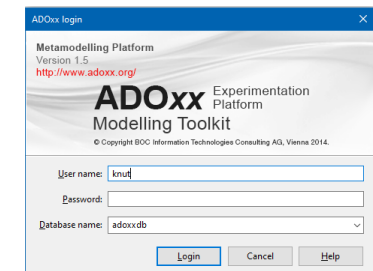
ADOxx has specific tools for each of these phases






## Meta-modeling

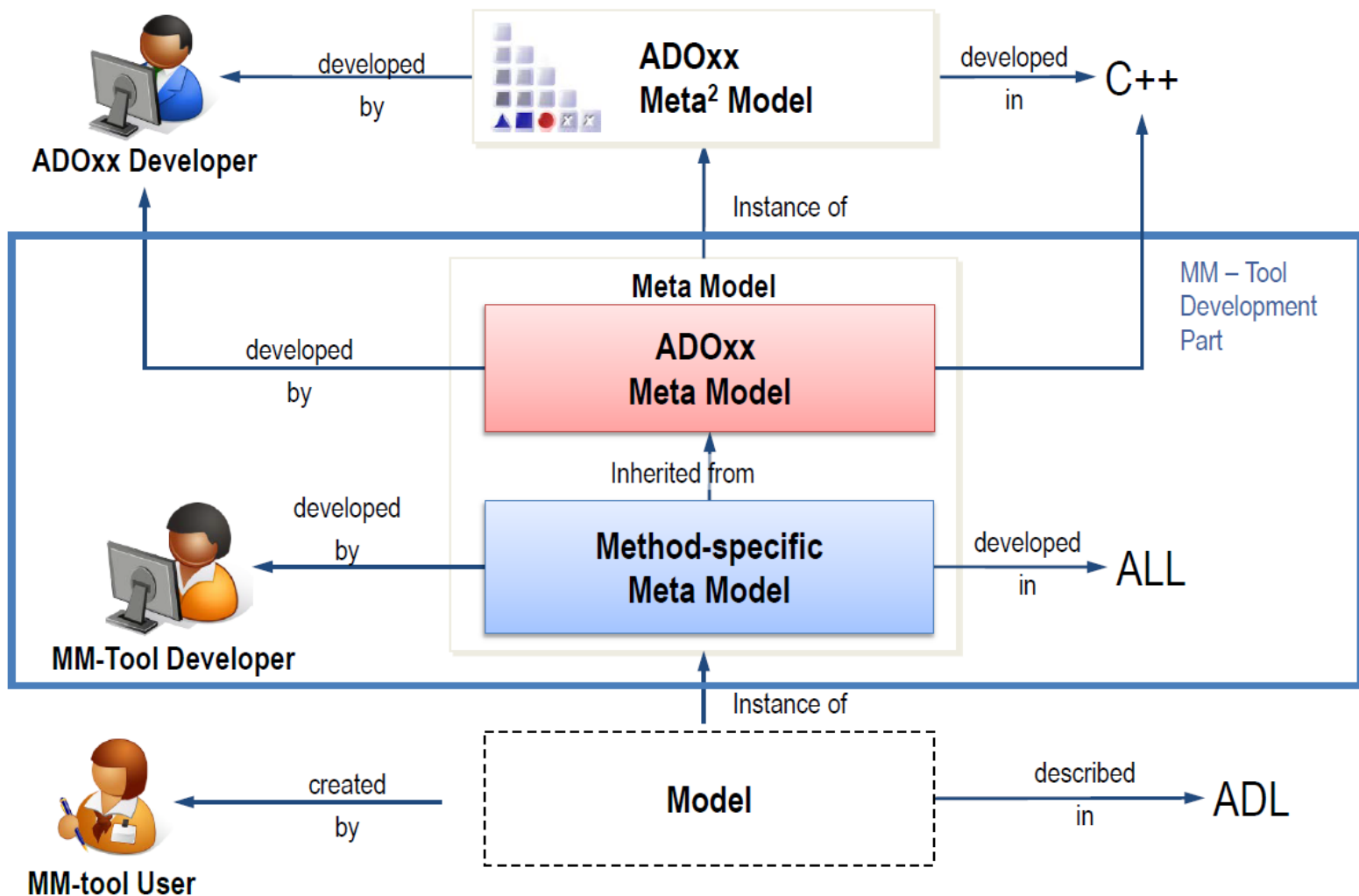


## Modeling



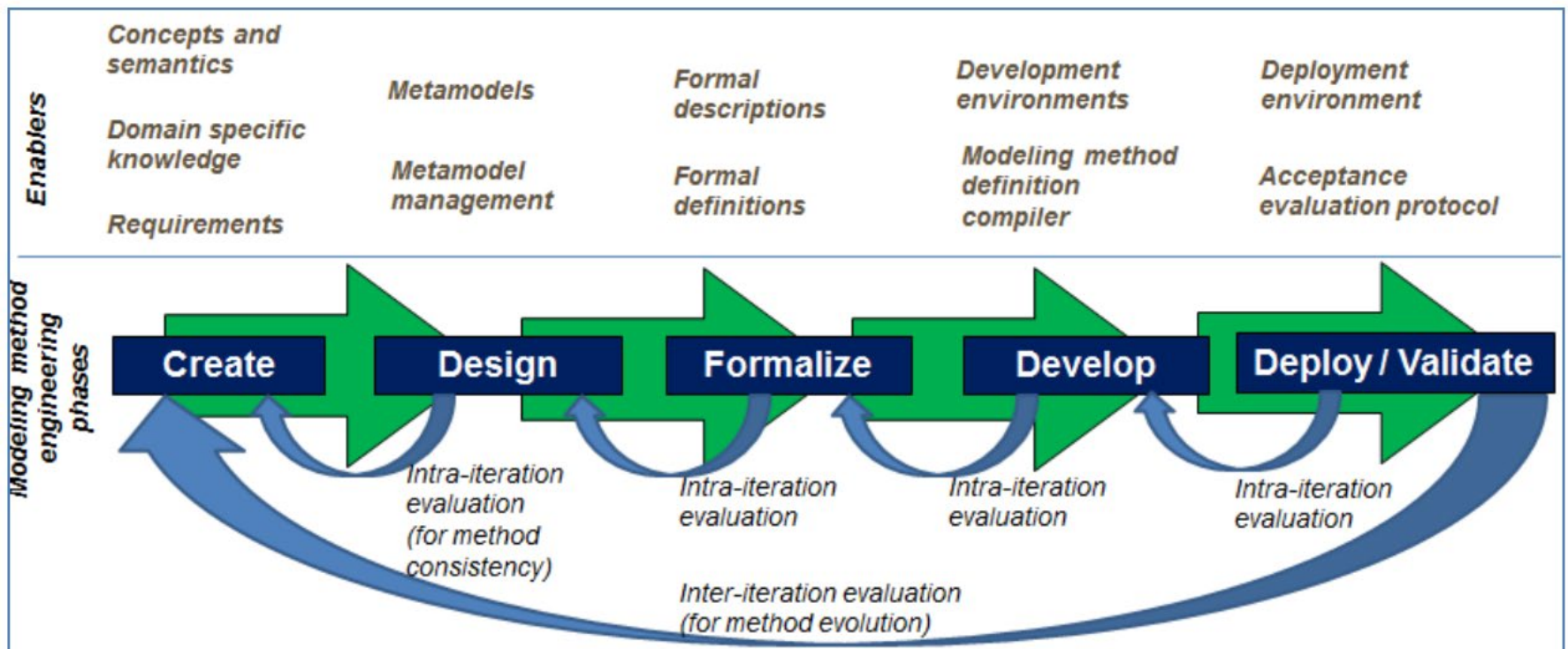
Identified Roles	Major Tasks	Required Skills	Cases
 <p>MM-tool User</p>	<p>Modelling Domain Knowledge</p>	<p>Domain Knowledge Method Knowledge</p>	<p>Established modelling tools</p> <p>Agile development of modelling tool in parallel to modelling tool usage</p> <p>...</p>
 <p>MM-Tool Developer</p>	<p>Developing an Meta Modelling Tool</p>	<p>Domain Knowledge Method Knowledge Platform Knowledge</p>	<p>Agile development of ADOxx platform in parallel to modelling method development</p>
 <p>ADOxx Developer</p>	<p>Implementation of tool specific and ADOxx functionality</p>	<p>Platform Knowledge ADOxx Technology Skills</p>	<p>Agile development of ADOxx platform in parallel to modelling method development</p>

# Meta Modelling Platforms Hierarchy in ADOxx



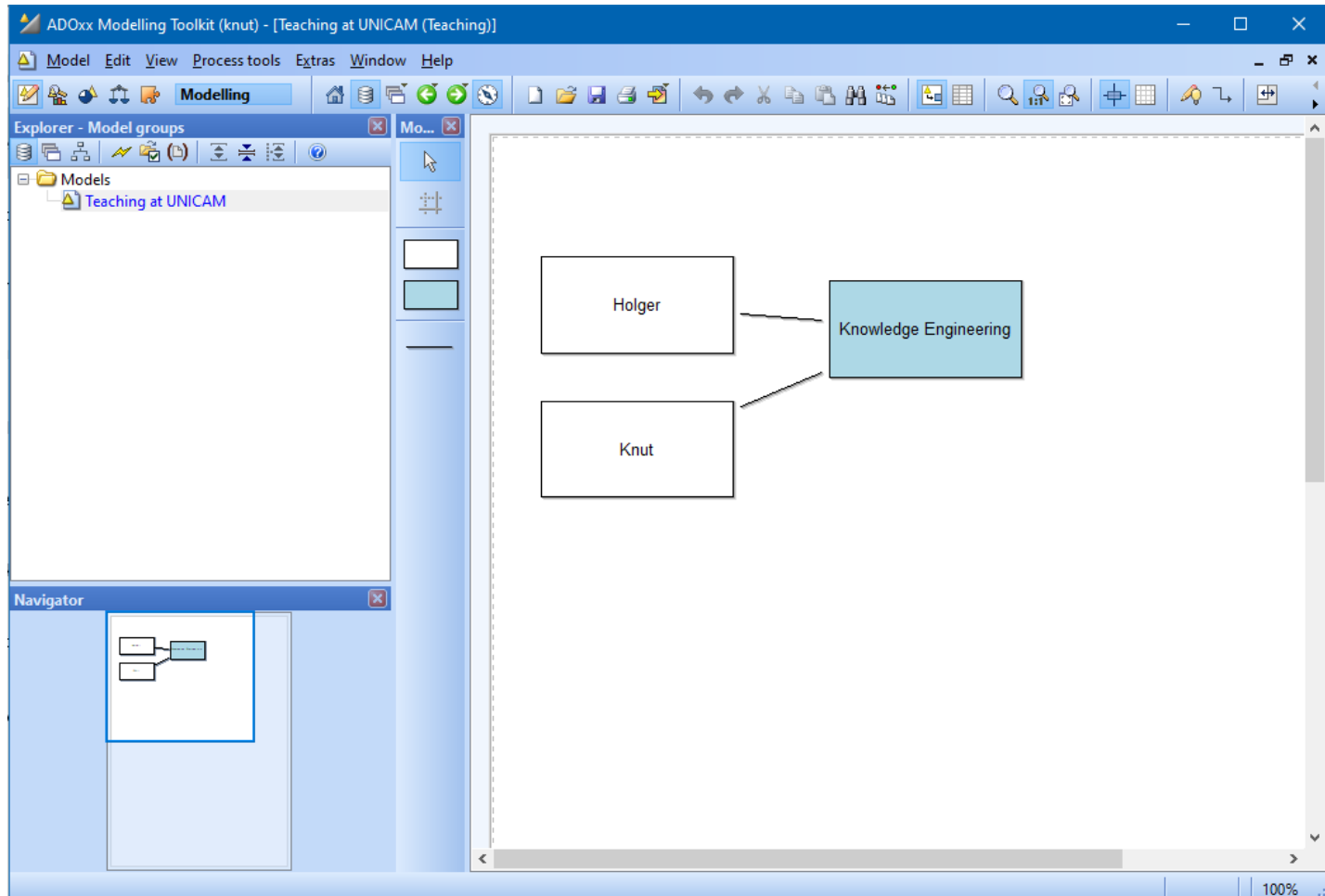
# The AMME LifeCycle

## Agile Modeling Method Engineering



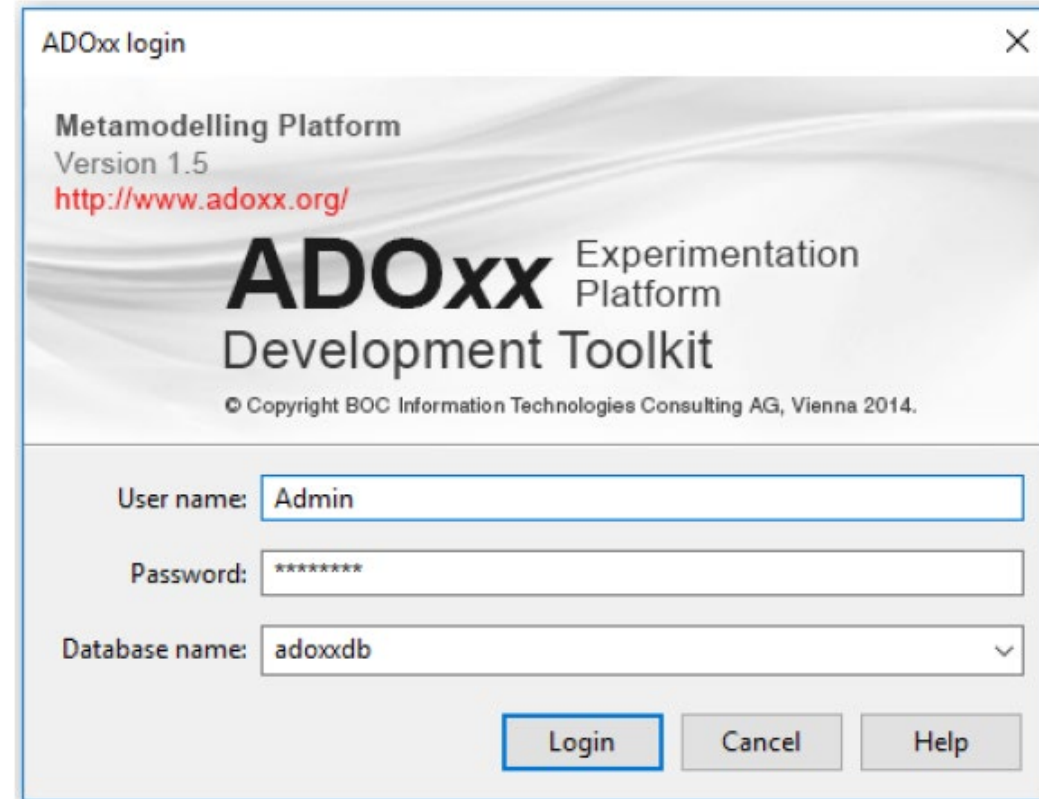
(Karagiannis 2015)

# Example: Create a Modeling Language for Teaching



# Development Toolkit

- Start Development Toolkit
- Login
  - ◆ Username: Admin
  - ◆ Password: password
  - ◆ DB: adoxxdb  
(or the one you created during installation)



ADOxx login

Metamodelling Platform  
Version 1.5  
<http://www.adoxx.org/>

**ADOxx** Experimentation Platform  
Development Toolkit

© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

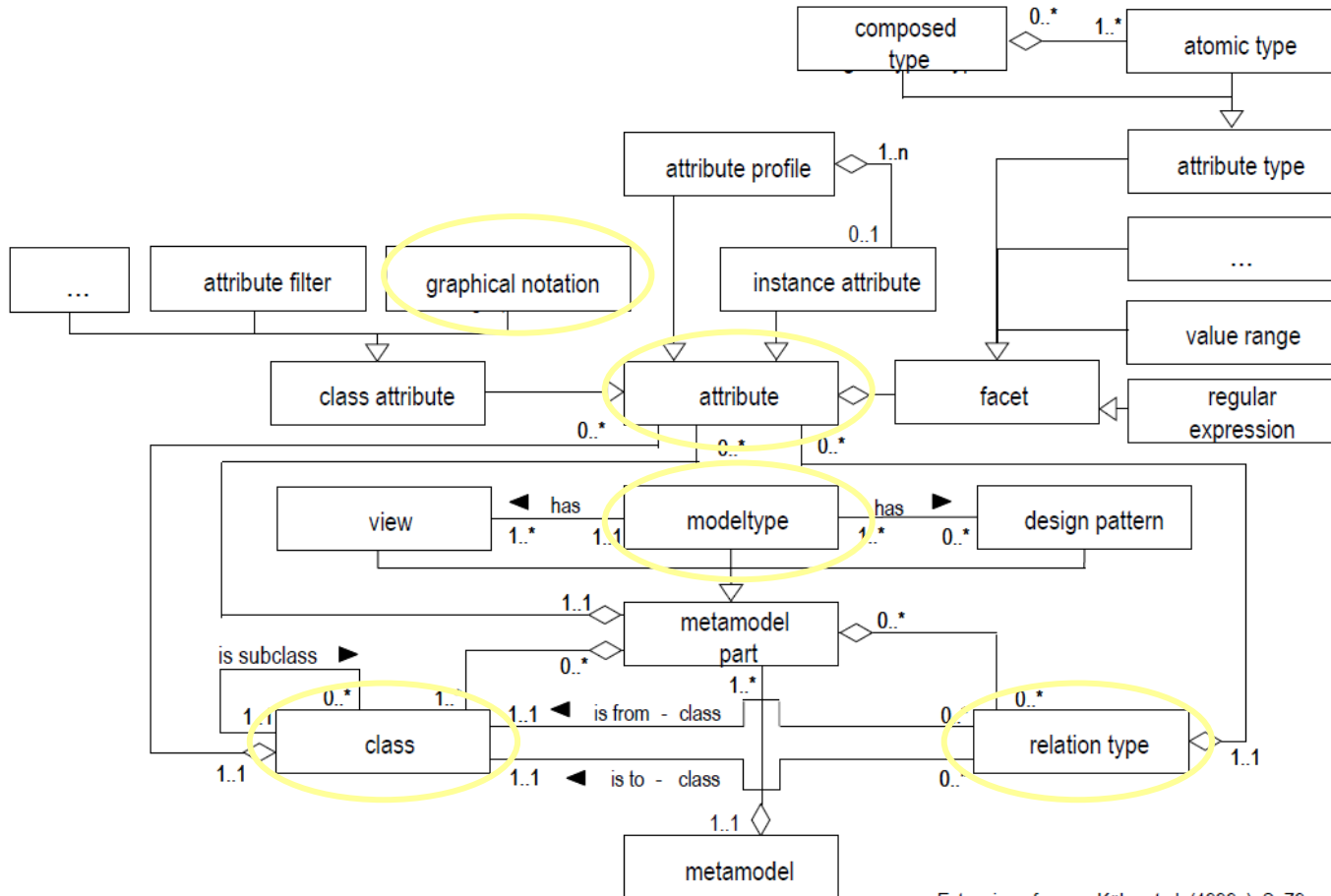
User name:

Password:

Database name:



# Meta<sup>2</sup> Model: Meta Model of Meta Modelling Language



Extension of: Kühn et al. (1999a), S. 79

# Import Modeling Language Libraries

ADOxx: Development Toolkit (Admin) - Administrator

Libraries Migration Extras Window Help

Library management

Welcome Download Tutorial Frequently Asked Questions Developer Community Documentation Contact

ADOxx.org Welcome

ADOxx Event

ADOxx Training Days  
25-27.03.2020 in Vienna

REGISTRATION REQUIRED!  
Contact us at [tutorial@adoxx.org](mailto:tutorial@adoxx.org)

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** Microservices Framework - the OMLAB 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 OMLab/University of Vienna: <http://www.omlab.org>

You can download conceptual modeling libraries from [adoxx.org](http://adoxx.org), e.g. BPMN,

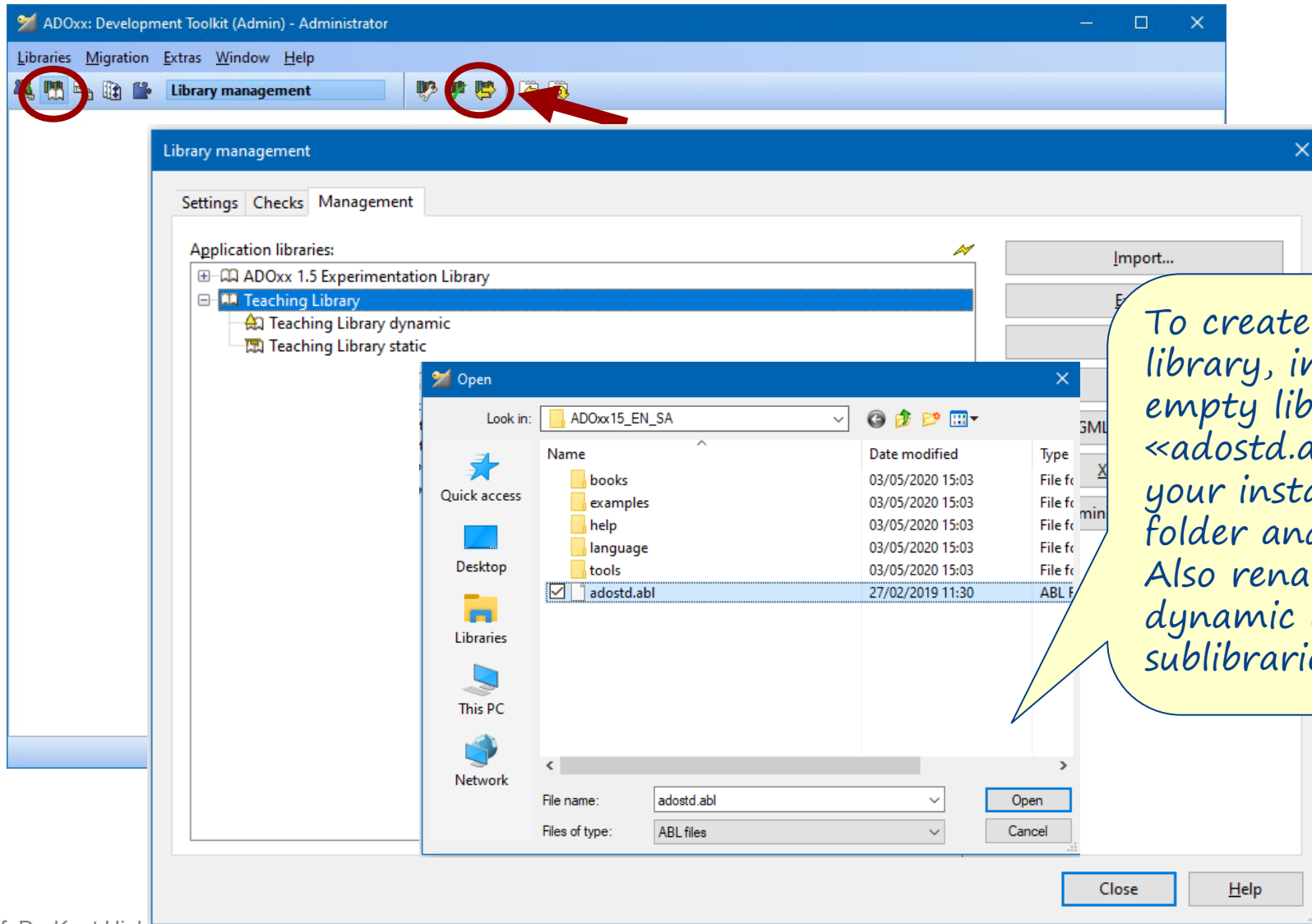
Tweets by @ADOxxORG

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

ADOxx Training Team  
March 2020

Mar 28, 2020

# Create a new Modeling Language Library



The screenshot shows the ADOxx Development Toolkit interface. The main window is titled "ADOxx: Development Toolkit (Admin) - Administrator". The "Library management" menu is highlighted, and a red circle is drawn around the "Library management" icon in the toolbar. A red arrow points to the "Import..." button in the "Library management" dialog box. The "Library management" dialog box shows the "Application libraries" list with "Teaching Library" selected. An "Open" dialog box is open, showing the "Look in:" path as "ADOxx15\_EN\_SA" and the file "adostd.abl" selected. The "File name:" field is "adostd.abl" and the "Files of type:" is "ABL files".

ADOxx: Development Toolkit (Admin) - Administrator

Libraries Migration Extras Window Help

Library management

Library management

Settings Checks Management

Application libraries:

- ADOxx 1.5 Experimentation Library
- Teaching Library
  - Teaching Library dynamic
  - Teaching Library static

Open

Look in: ADOxx15\_EN\_SA

Name	Date modified	Type
books	03/05/2020 15:03	File folder
examples	03/05/2020 15:03	File folder
help	03/05/2020 15:03	File folder
language	03/05/2020 15:03	File folder
tools	03/05/2020 15:03	File folder
adostd.abl	27/02/2019 11:30	ABL file

File name: adostd.abl

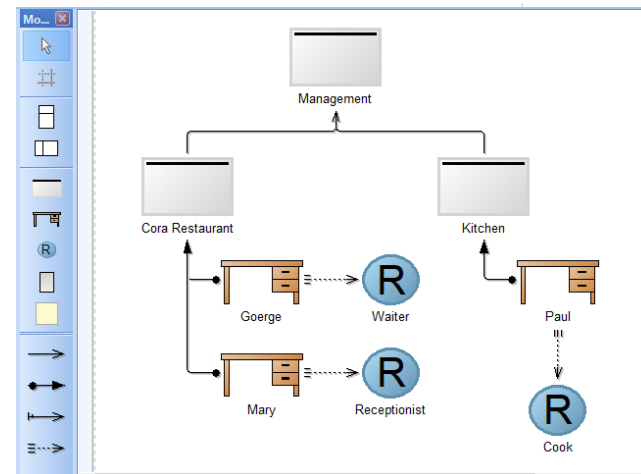
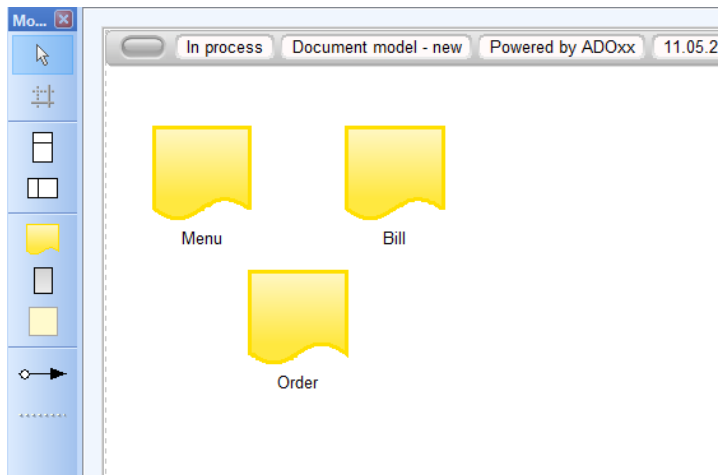
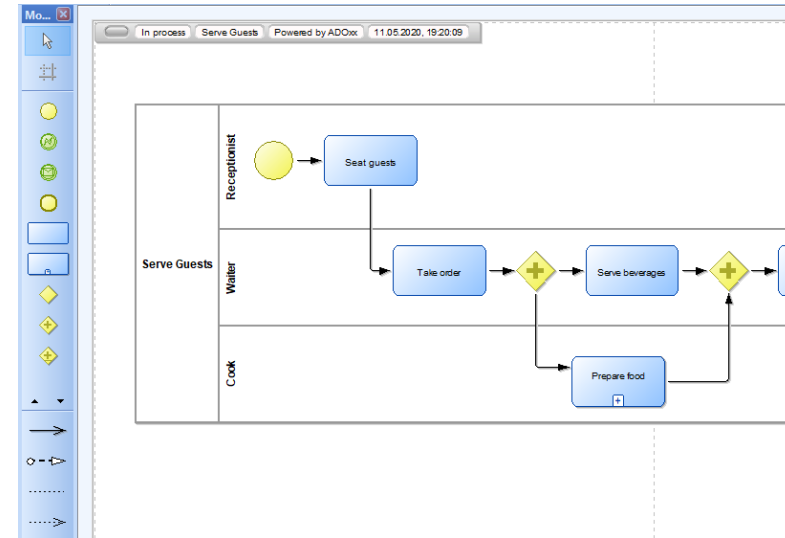
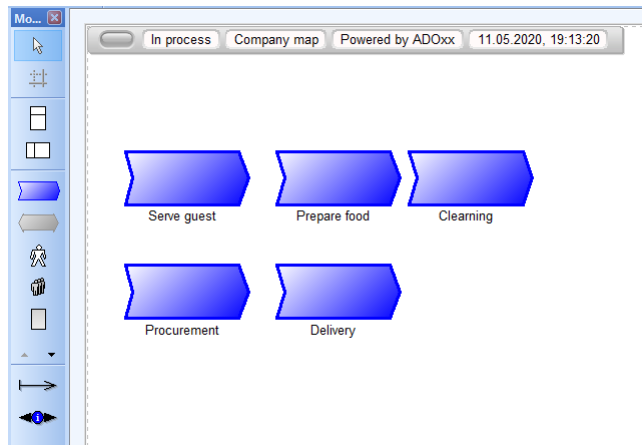
Files of type: ABL files

Open Cancel

Close Help

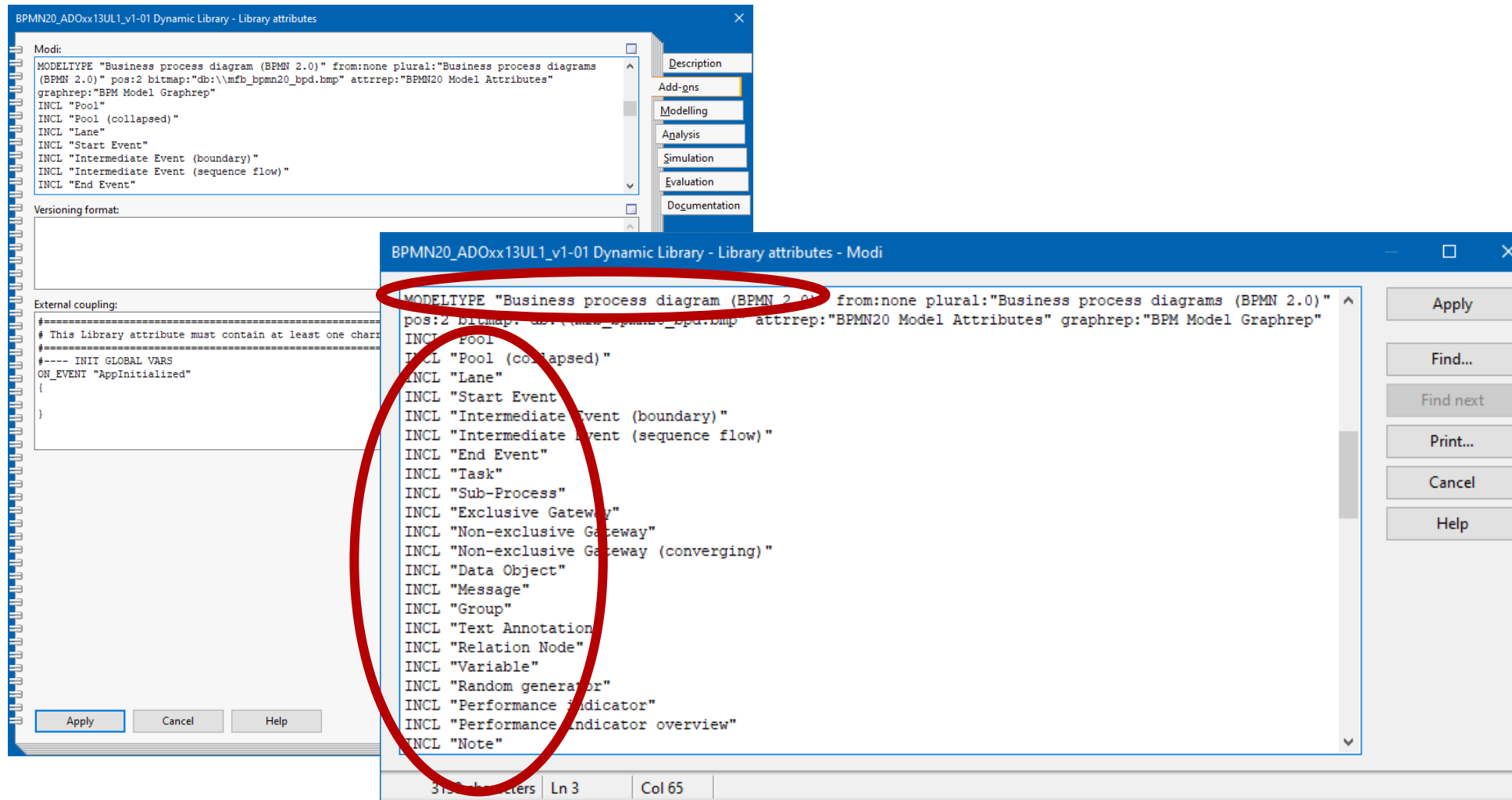
To create a new library, import the empty library <<adostd.abl>> from your installation folder and rename it. Also rename the dynamic and static sublibraries

# Model Types: Representing Views on the Knowledge



# Classes are assigned to Model Types

## Example: BPMN



The screenshot displays two overlapping windows from a software application. The background window, titled "BPMN20\_ADOxx13UL1\_v1-01 Dynamic Library - Library attributes", shows a list of model types under the "Modi:" section. The foreground window, titled "BPMN20\_ADOxx13UL1\_v1-01 Dynamic Library - Library attributes - Modi", shows a detailed view of the "Business process diagram (BPMN 2.0)" model type. A red circle highlights the "Modi:" section in the foreground window, which lists various BPMN elements assigned to this model type.

**Modi:**

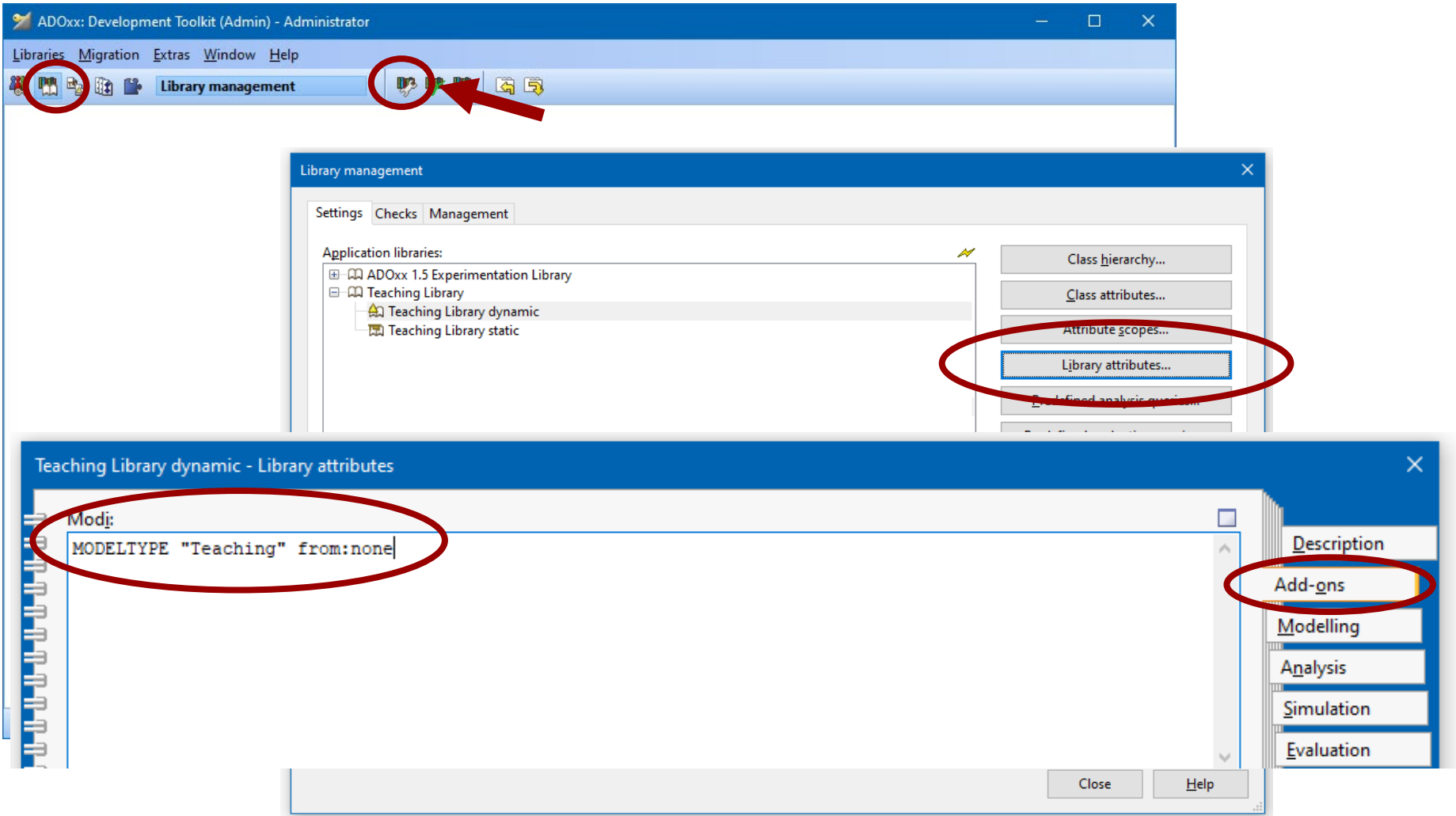
```
MODELTYPE "Business process diagram (BPMN 2.0)" from:none plural:"Business process diagrams (BPMN 2.0)" pos:2 bitmap:"db:\mfb_bpmn20_bpd.bmp" attrrep:"BPMN20 Model Attributes" graphrep:"BPM Model Graphrep"
INCL "Pool"
INCL "Pool (collapsed)"
INCL "Lane"
INCL "Start Event"
INCL "Intermediate Event (boundary)"
INCL "Intermediate Event (sequence flow)"
INCL "End Event"
```

**External coupling:**

```
# This Library attribute must contain at least one char...
#---- INIT GLOBAL VARS
ON_EVENT "AppInitialized"
{
}
```

**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"
```

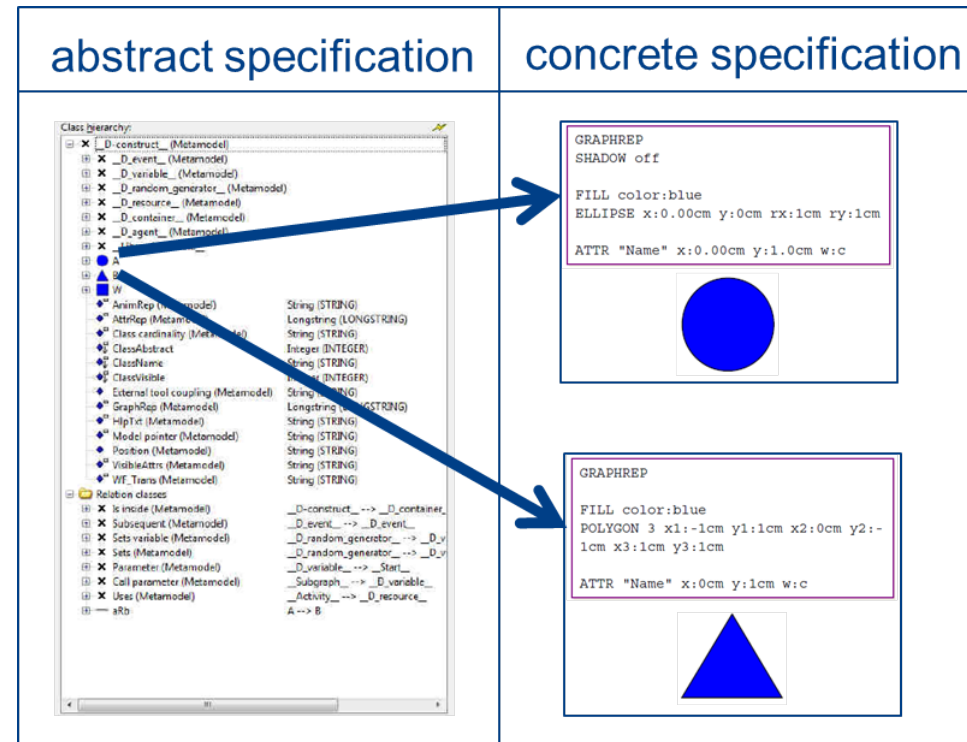


The screenshot illustrates the process of configuring library attributes in the ADOxx Development Toolkit. It consists of three overlapping windows:

- ADOxx: Development Toolkit (Admin) - Administrator:** The main application window with a menu bar (Libraries, Migration, Extras, Window, Help) and a toolbar. A red circle highlights the 'Library management' icon in the toolbar, and a red arrow points to it.
- Library management:** A sub-window with tabs for Settings, Checks, and Management. Under the 'Management' tab, a tree view shows 'Application libraries' including 'Teaching Library dynamic'. A red circle highlights the 'Library attributes...' button on the right side.
- Teaching Library dynamic - Library attributes:** A configuration window for the selected library. The 'Modj:' field contains the text `MODELTYPE "Teaching" from:none`, which is circled in red. On the right, a list of categories is shown, with 'Add-ons' circled in red. Other categories include Description, Modelling, Analysis, Simulation, and Evaluation. 'Close' and 'Help' buttons are at the bottom.

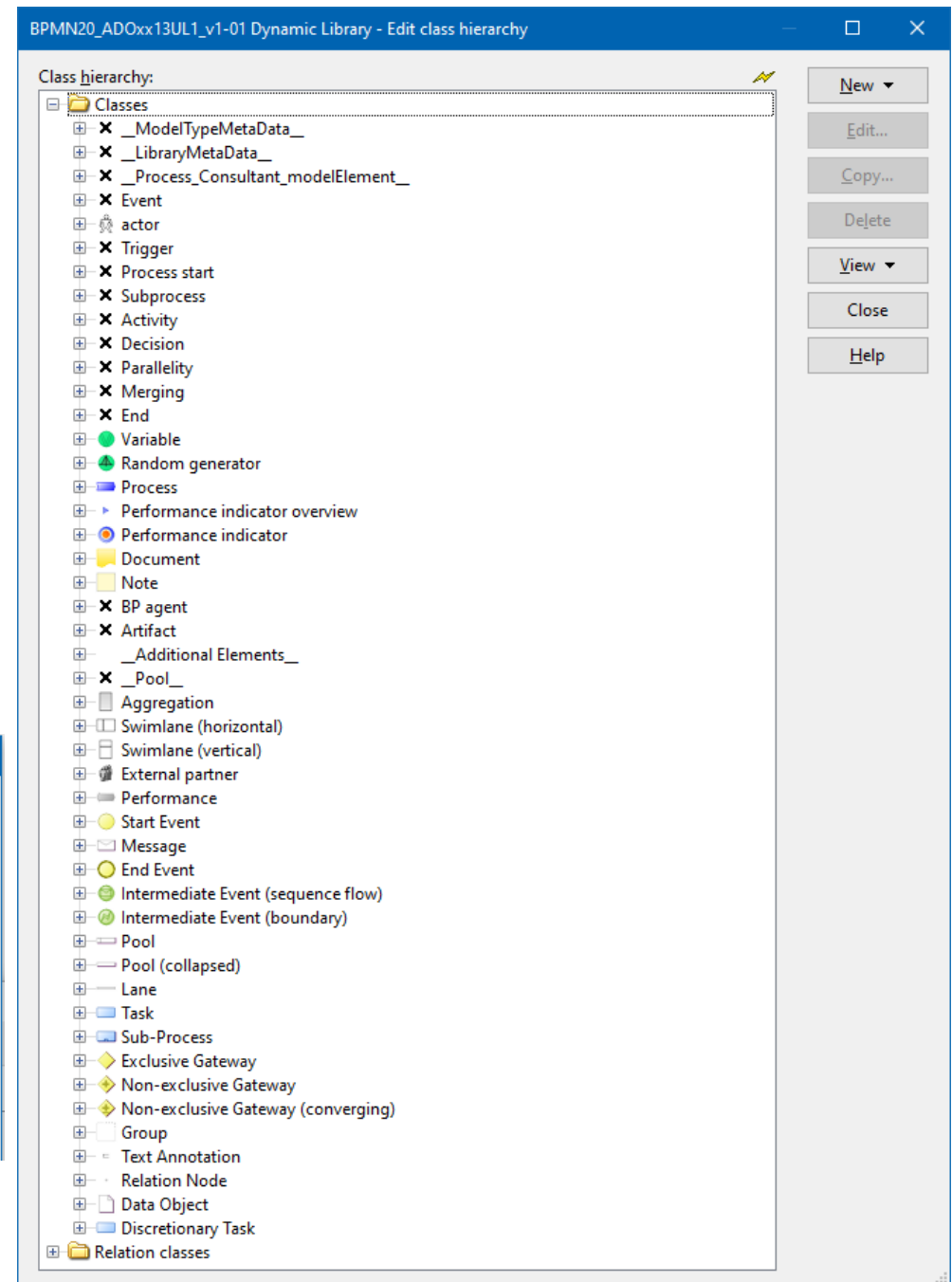
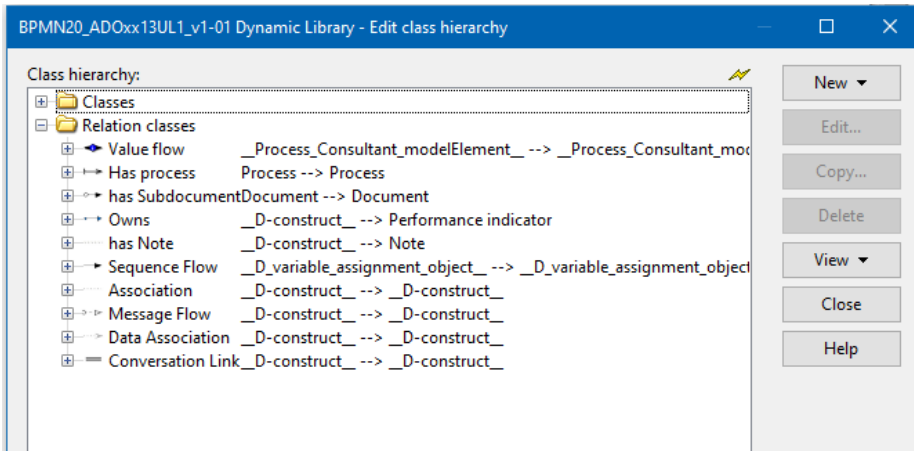
# Metamodel and Modeling Language in ADOxx

- The meta model of a model language is defined by
  - ◆ Classes of elements and relations
  - ◆ Class hierarchy
  - ◆ Attributes of the elements
- The notation is defined by
  - ◆ special attribute GraphRep



# Class Hierarchies

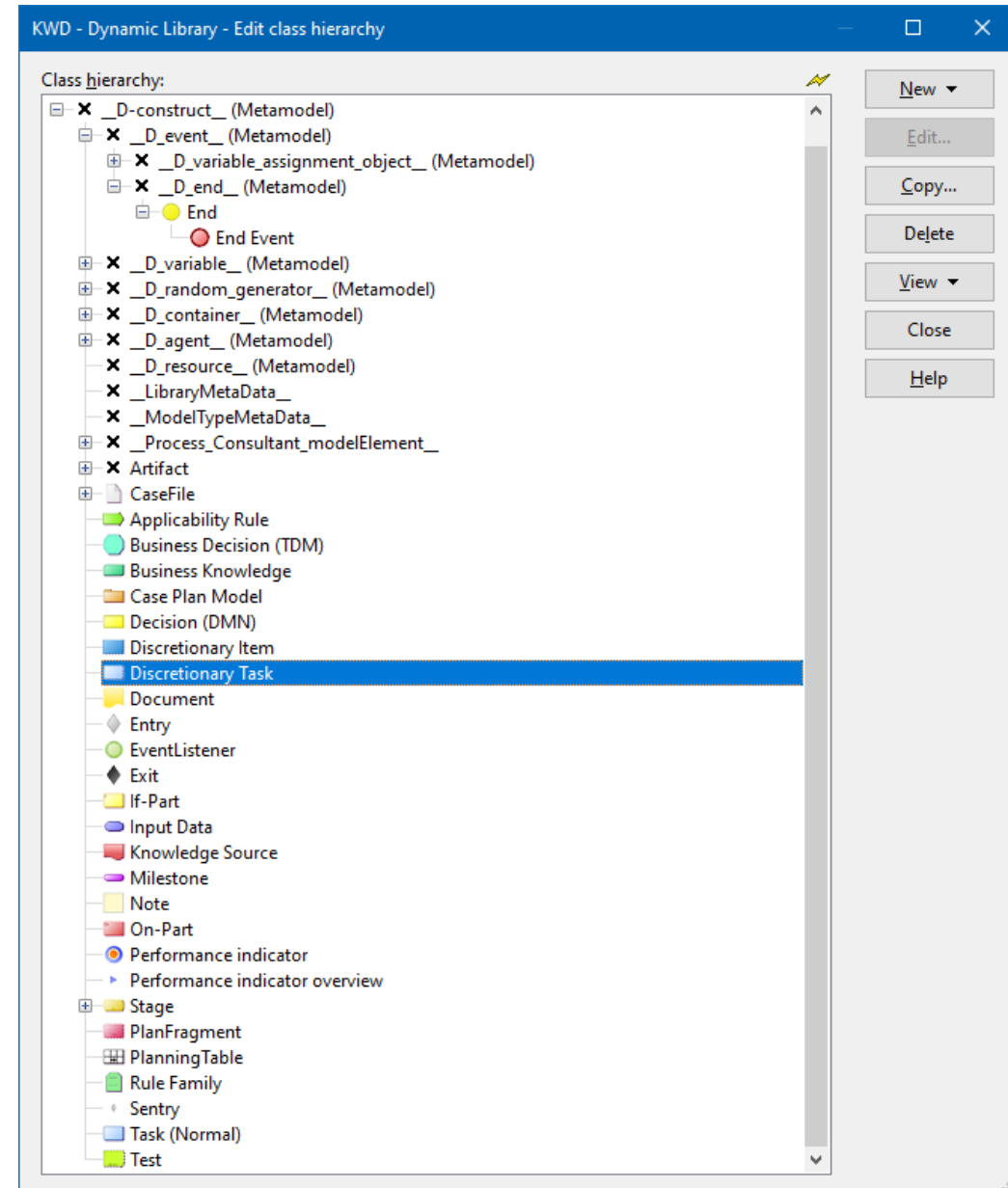
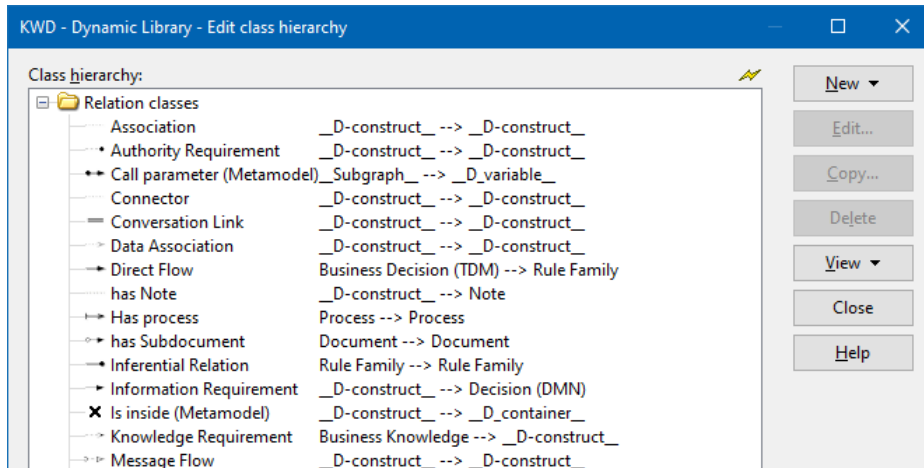
- ADOxx distinguishes
  - ◆ Classes
  - ◆ Relation classes



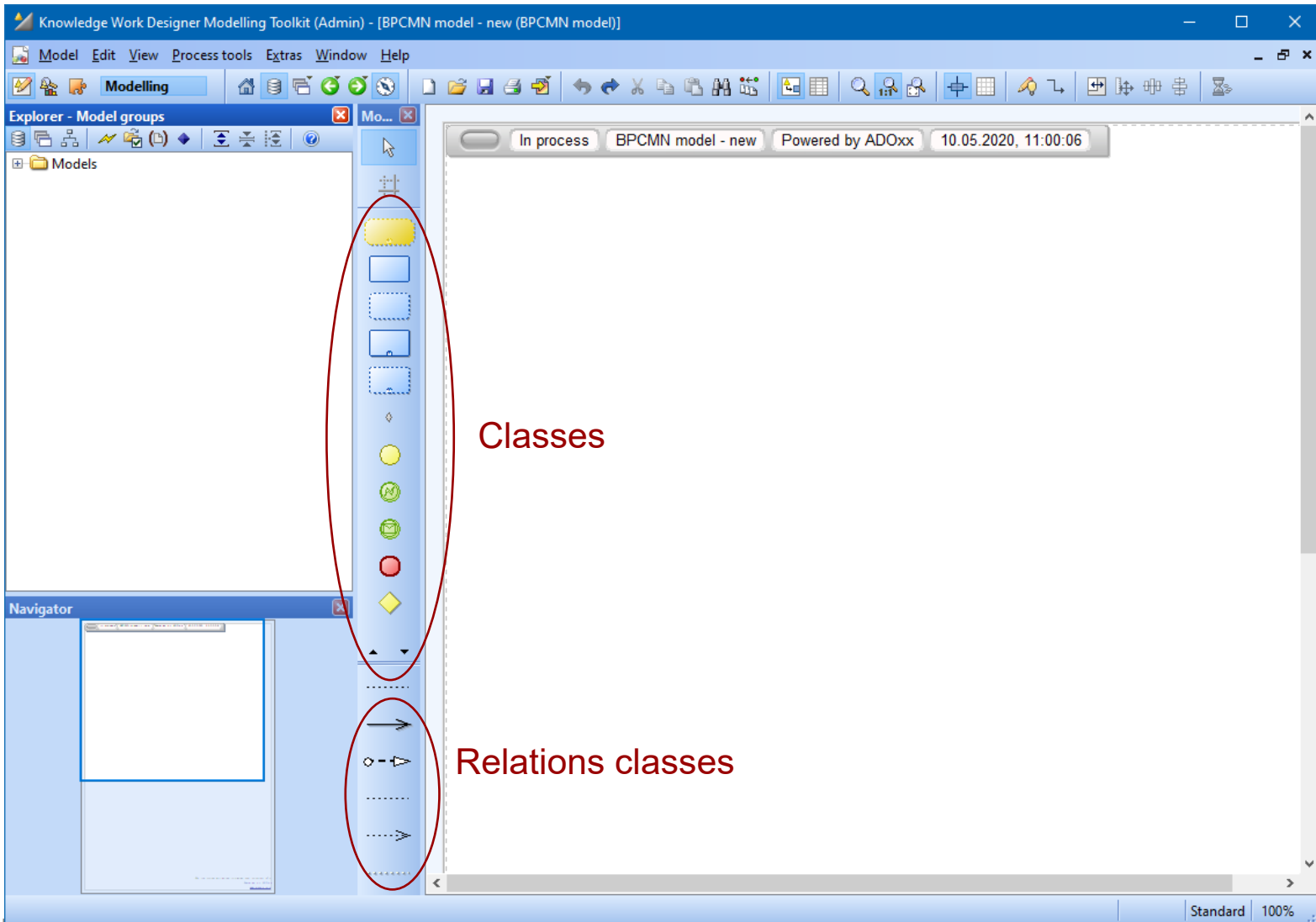


# Class Hierarchies

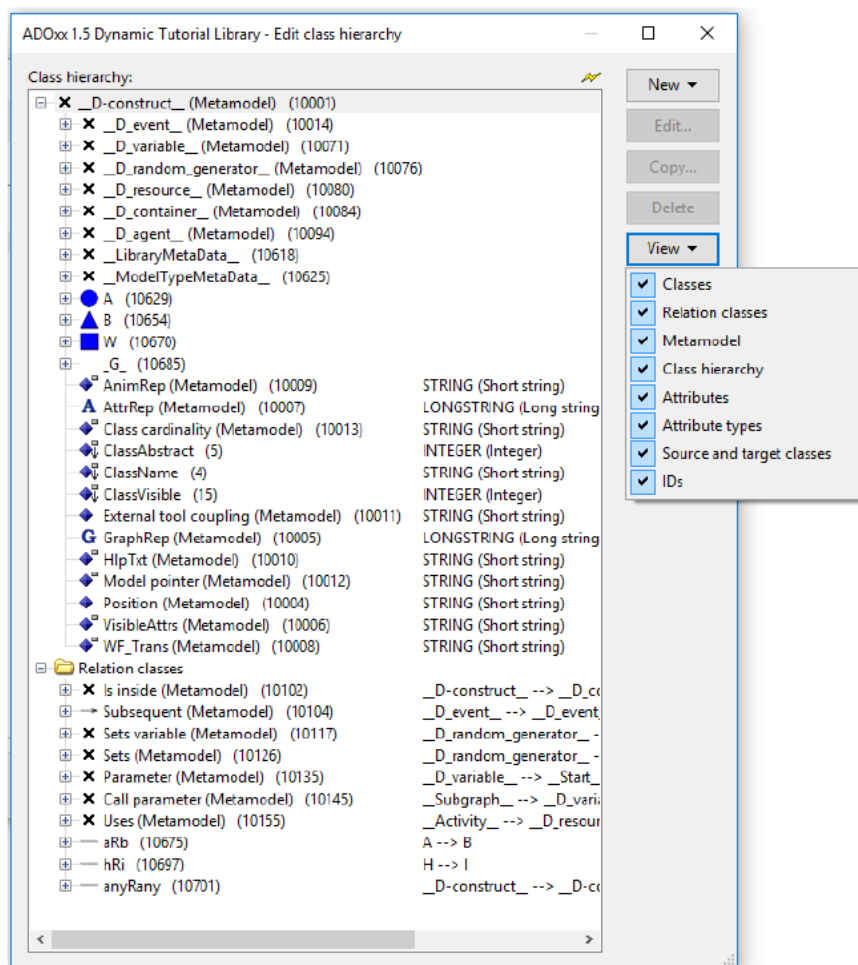
- ADOxx distinguishes
  - ◆ Classes
  - ◆ Relation classes



# Appearance of Classes in the Modelling Toolkit



# Views of the Class Hierarchy



## Classes

All visible classes will be shown

## Relation classes

All available relation classes will be shown

## Metamodel

All classes will be shown

## Class hierarchy

All classes will be shown with their inheritance in a hierarchy

## Attributes

The attributes of the (relation-)classes will be shown

## Attribute types

The type of each attribute will be shown







## Source- and Target-classes

Shows the endpoints for each relation class, i.e. between which classes it can be used.

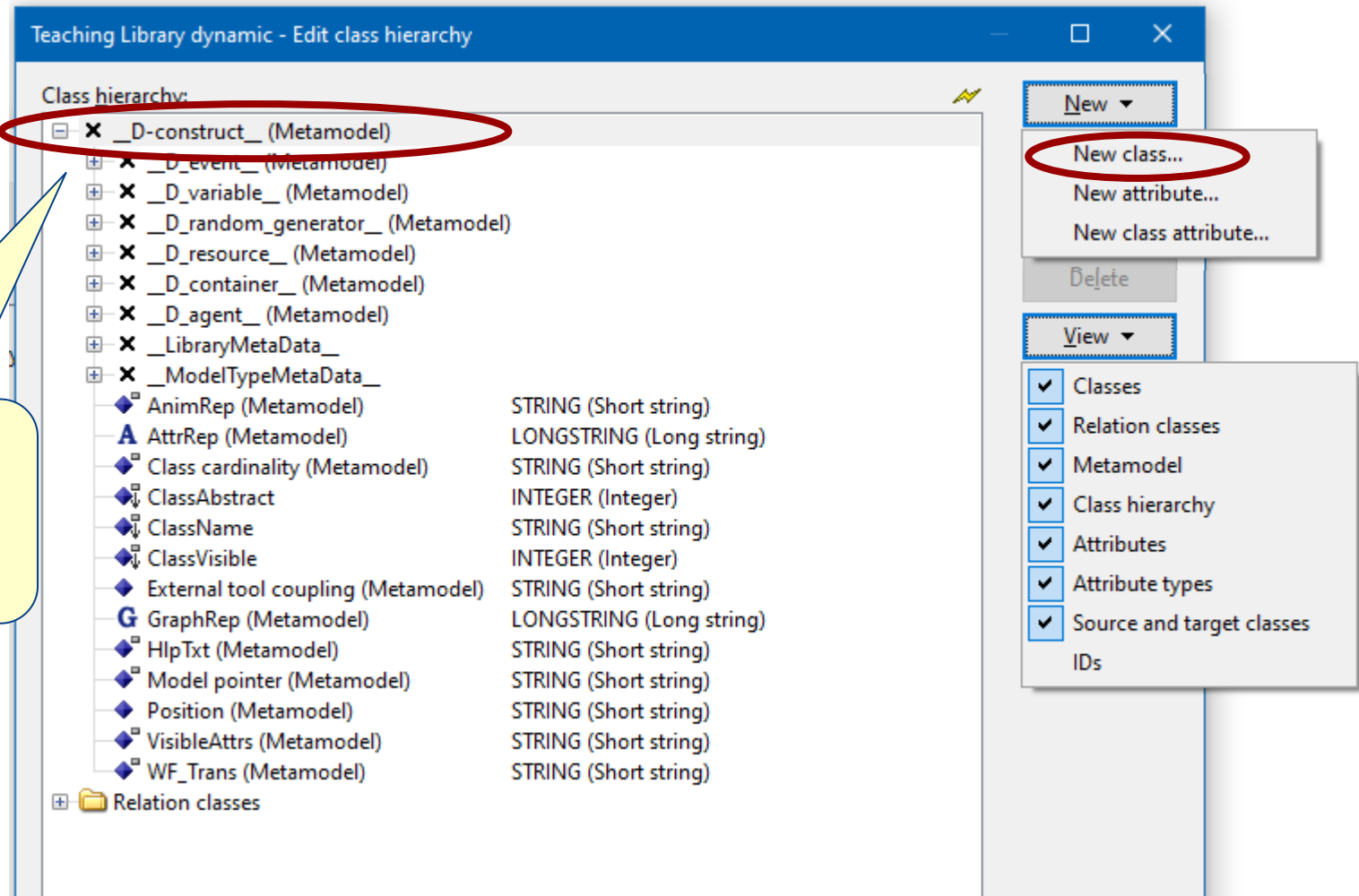
## IDs

Shows ID numbers of classes and attributes

# Icons in Class Hierarchy

-  **Class** (the icon shows the graphical definition of the object and can therefore vary)
-  **Class** (without a graphical definition)
-  **Attribute**
-  **Attribute** (inherited from another class)
-  **Class attribute**
-  **Class attribute** (inherited from another class)

# 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)** STRING (Short string)
- AttrRep (Metamodel)** LONGSTRING (Long string)
- Class cardinality (Metamodel)** STRING (Short string)
- ClassAbstract** INTEGER (Integer)
- ClassName** STRING (Short string)
- ClassVisible** INTEGER (Integer)
- External tool coupling (Metamodel)** STRING (Short string)
- GraphRep (Metamodel)** LONGSTRING (Long string)
- HlpTxt (Metamodel)** STRING (Short string)
- Model pointer (Metamodel)** STRING (Short string)
- Position (Metamodel)** STRING (Short string)
- VisibleAttrs (Metamodel)** STRING (Short string)
- WF\_Trans (Metamodel)** STRING (Short string)
- Relation classes**

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 functionality

# New Classes for Lecturer and Module

Teaching Library dynamic - Edit class hierarchy

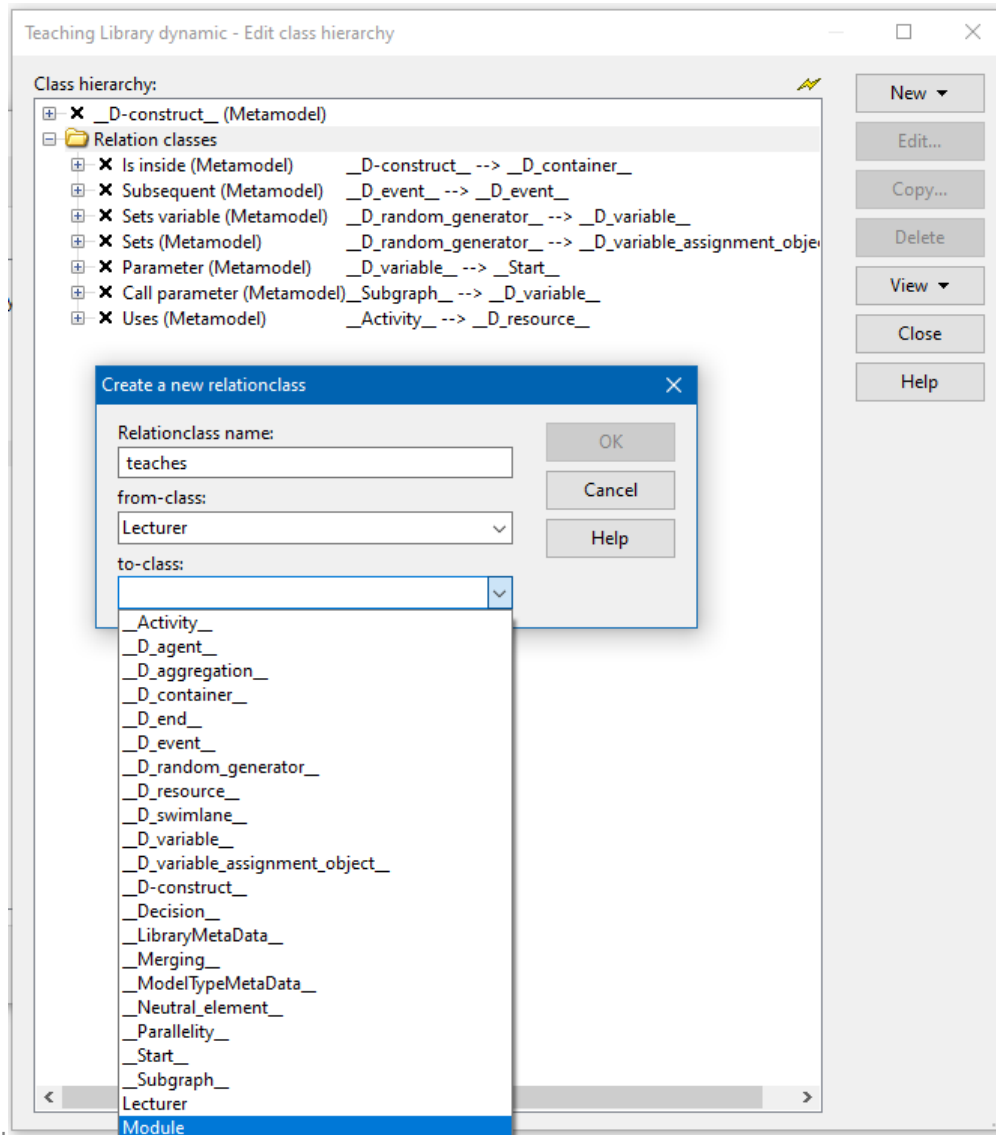
Class hierarchy:

- [-] X \_D-construct\_ (Metamodel)
  - [+] X \_D\_event\_ (Metamodel)
  - [+] X \_D\_variable\_ (Metamodel)
  - [+] X \_D\_random\_generator\_ (Metamodel)
  - [+] X \_D\_resource\_ (Metamodel)
  - [+] X \_D\_container\_ (Metamodel)
  - [+] X \_D\_agent\_ (Metamodel)
  - [+] X \_LibraryMetaData\_
  - [+] X \_ModelTypeMetaData\_
  - [+] X Lecturer
    - [-] AnimRep (Metamodel) STRING (Short string)
    - [+] AttrRep (Metamodel) LONGSTRING (Long string)
    - [-] Class cardinality (Metamodel) STRING (Short string)
    - [-] ClassAbstract INTEGER (Integer)
    - [-] ClassName STRING (Short string)
    - [-] ClassVisible INTEGER (Integer)
    - [-] External tool coupling (Metamodel) STRING (Short string)
    - [+] GraphRep (Metamodel) LONGSTRING (Long string)
    - [-] HlpTxt (Metamodel) STRING (Short string)
    - [-] Model pointer (Metamodel) STRING (Short string)
    - [-] Position (Metamodel) STRING (Short string)
    - [-] VisibleAttrs (Metamodel) STRING (Short string)
    - [-] WF\_Trans (Metamodel) STRING (Short string)
  - [+] X Module
    - [-] AnimRep (Metamodel) STRING (Short string)
    - [+] AttrRep (Metamodel) LONGSTRING (Long string)
    - [-] Class cardinality (Metamodel) STRING (Short string)
    - [-] ClassAbstract INTEGER (Integer)
    - [-] ClassName STRING (Short string)
    - [-] ClassVisible INTEGER (Integer)
    - [-] External tool coupling (Metamodel) STRING (Short string)
    - [+] GraphRep (Metamodel) LONGSTRING (Long string)
    - [-] HlpTxt (Metamodel) STRING (Short string)
    - [-] Model pointer (Metamodel) STRING (Short string)
    - [-] Position (Metamodel) STRING (Short string)
    - [-] VisibleAttrs (Metamodel) STRING (Short string)
    - [-] WF\_Trans (Metamodel) STRING (Short string)
    - [-] AnimRep (Metamodel) STRING (Short string)
    - [+] AttrRep (Metamodel) LONGSTRING (Long string)
    - [-] Class cardinality (Metamodel) STRING (Short string)

Buttons: New, Edit..., Copy..., Delete, View, Close, Help

New classes, e.g. «Lecturer» and «Module» can be defined as subclasses of D-construct, if no specific functionality is needed. They inherit the attributes of the superclass

# Defining a new Relation



Teaching Library dynamic - Edit class hierarchy

Class hierarchy:

- [-] X \_D-construct\_ (Metamodel)
- [-] Relation classes
  - [-] X Is inside (Metamodel) \_D-construct\_ --> \_D\_container\_
  - [-] X Subsequent (Metamodel) \_D\_event\_ --> \_D\_event\_
  - [-] X Sets variable (Metamodel) \_D\_random\_generator\_ --> \_D\_variable\_
  - [-] X Sets (Metamodel) \_D\_random\_generator\_ --> \_D\_variable\_assignment\_obje
  - [-] X Parameter (Metamodel) \_D\_variable\_ --> \_Start\_
  - [-] X Call parameter (Metamodel) \_Subgraph\_ --> \_D\_variable\_
  - [-] X Uses (Metamodel) \_Activity\_ --> \_D\_resource\_

Create a new relationclass

Relationclass name: teaches

from-class: Lecturer

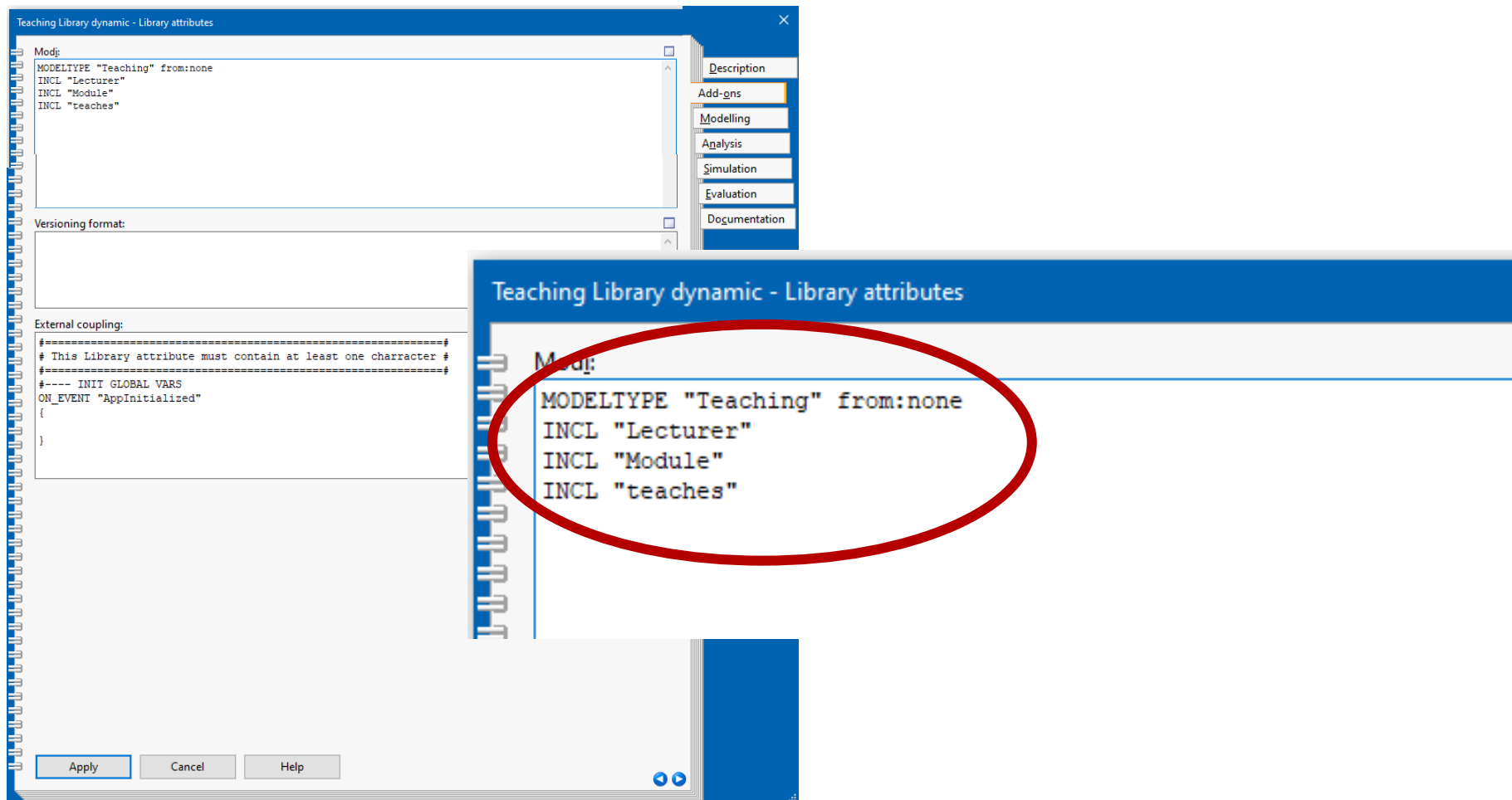
to-class:

- \_Activity\_
- \_D\_agent\_
- \_D\_aggregation\_
- \_D\_container\_
- \_D\_end\_
- \_D\_event\_
- \_D\_random\_generator\_
- \_D\_resource\_
- \_D\_swimlane\_
- \_D\_variable\_
- \_D\_variable\_assignment\_object\_
- \_D-construct\_
- \_Decision\_
- \_LibraryMetaData\_
- \_Merging\_
- \_ModelTypeMetaData\_
- \_Neutral\_element\_
- \_Parallellity\_
- \_Start\_
- \_Subgraph\_
- Lecturer
- Module

Buttons: New, Edit..., Copy..., Delete, View, Close, Help

Example: A new relation «teaches» for elements from class «Lecturer» to class «Module»

# Classes and Relations are assigned to Model Types



Teaching Library dynamic - Library attributes

```
Mod:  
MODELTYPE "Teaching" from:none  
INCL "Lecturer"  
INCL "Module"  
INCL "teaches"
```

Versioning format:

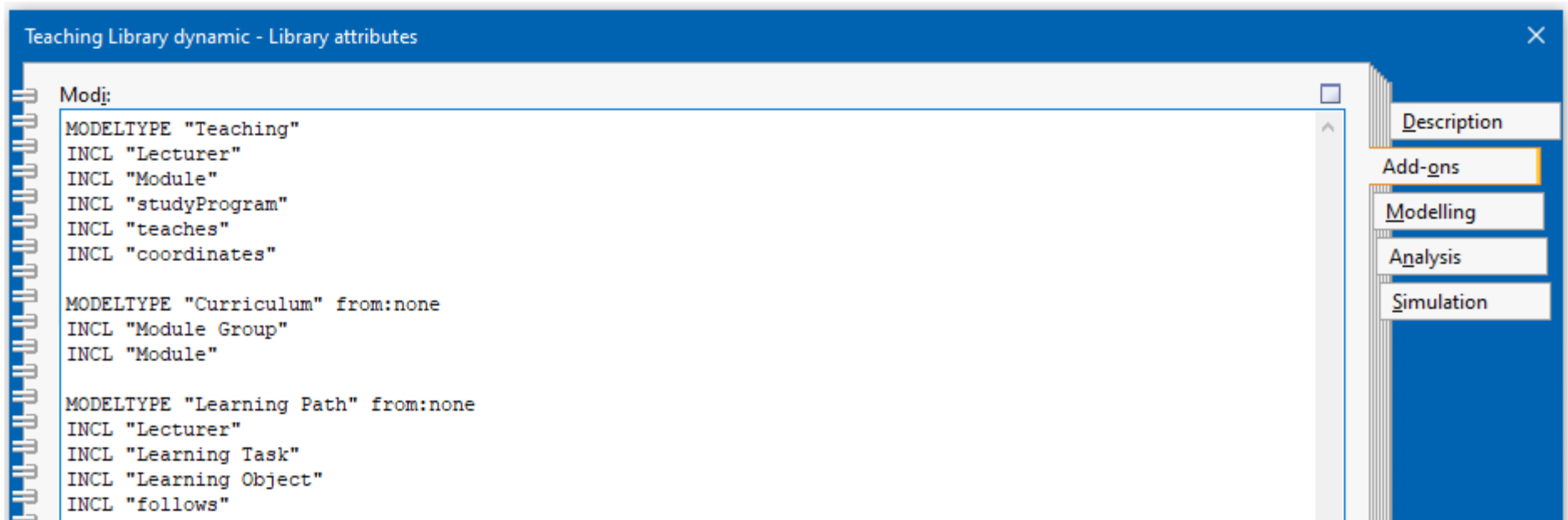
External coupling:

```
#-----  
# This Library attribute must contain at least one character #  
#-----  
#---- INIT GLOBAL VARS  
ON_EVENT "AppInitialized"  
{  
}
```

Apply Cancel Help



# Example with several Model Types



Teaching Library dynamic - Library attributes

Modj:

```
MODELTYPE "Teaching"  
INCL "Lecturer"  
INCL "Module"  
INCL "studyProgram"  
INCL "teaches"  
INCL "coordinates"  
  
MODELTYPE "Curriculum" from:none  
INCL "Module Group"  
INCL "Module"  
  
MODELTYPE "Learning Path" from:none  
INCL "Lecturer"  
INCL "Learning Task"  
INCL "Learning Object"  
INCL "follows"
```

Navigation menu:

- Description
- Add-ons
- Modelling
- Analysis
- Simulation

# Attributes

## ■ Kinds of Attributes

- ◆ Properties of Models
- ◆ Graphical Representation
- ◆ References

BPMN20\_ADOxx13UL1\_v1-01 Dynamic Library - Edit class hierarchy

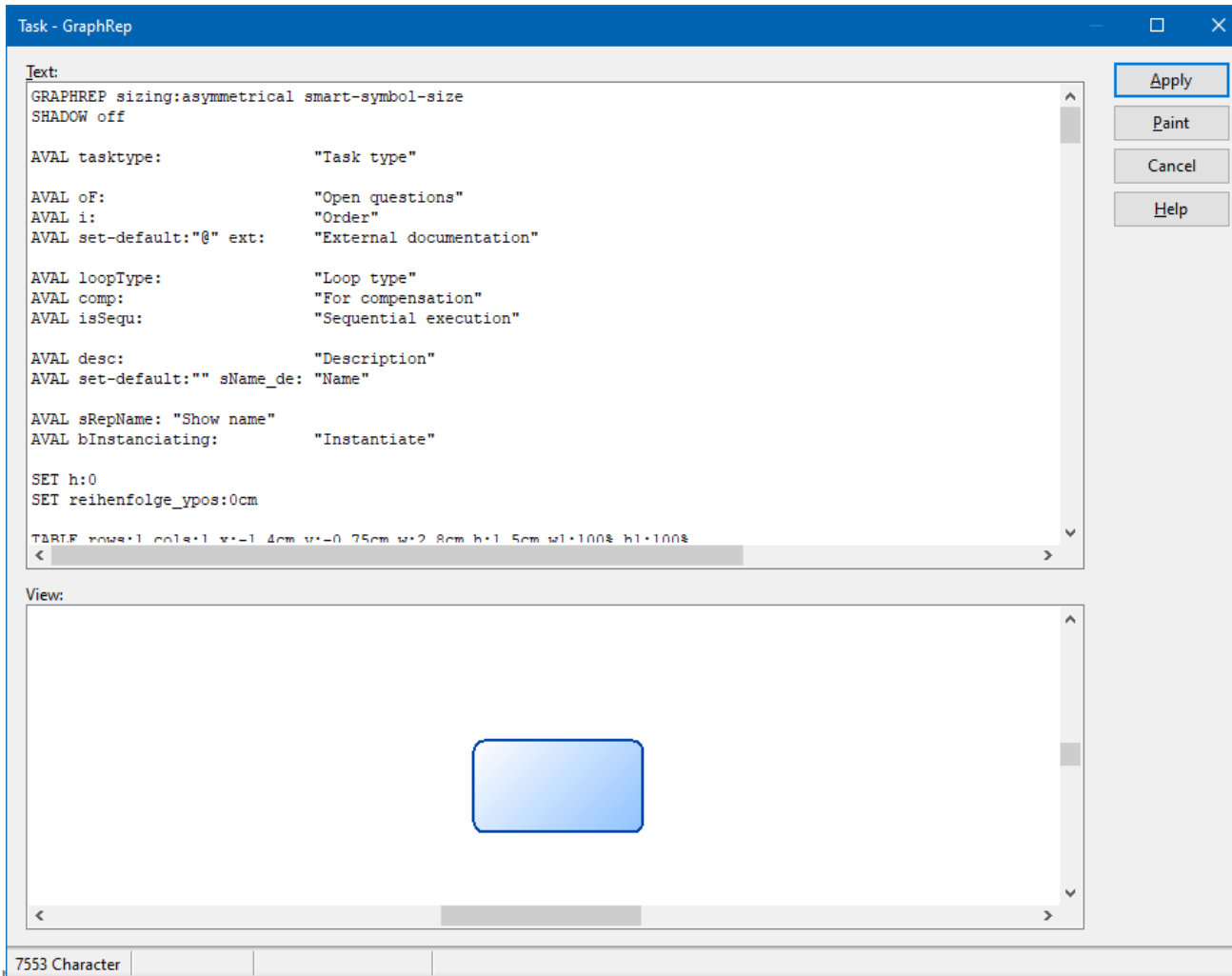
Class hierarchy:

Task	
↳ __Conversion__	LONGSTRING (Long string)
↳ Aggregated costs	DOUBLE (Floating-point number)
↳ Aggregated execution time	TIME (Time)
↳ Aggregated personnel costs	DOUBLE (Floating-point number)
↳ Aggregated resting time	TIME (Time)
↳ Aggregated transport time	TIME (Time)
↳ Aggregated waiting time	TIME (Time)
↳ AnimRep (Metamodel)	STRING (Short string)
↳ Assignments (Metamodel)	RECORD (Record table)
↳ AttrRep (Metamodel)	LONGSTRING (Long string)
↳ Auditing	ENUMERATION (Enumeration)
↳ Average number of participants (Metamodel)	INTEGER (Integer)
↳ Beschreibung	STRING (Short string)
↳ Bezeichnung	STRING (Short string)
↳ Call activity	INTERREF (Inter-model reference)
↳ Cardinality	STRING (Short string)
↳ Categories (Metamodel)	STRING (Short string)
↳ Class cardinality (Metamodel)	STRING (Short string)
↳ ClassAbstract	INTEGER (Integer)
↳ Classification	ENUMERATIONLIST (Enumeration list)
↳ ClassName	STRING (Short string)
↳ ClassVisible	INTEGER (Integer)
↳ Collection	ENUMERATION (Enumeration)
↳ Comment	STRING (Short string)
↳ Completion condition	STRING (Short string)
↳ Continuous execution (Metamodel)	ENUMERATION (Enumeration)
↳ Cooperation mode (Metamodel)	ENUMERATION (Enumeration)
↳ Cooperative (Metamodel)	ENUMERATION (Enumeration)
↳ Costs	DOUBLE (Floating-point number)
↳ Description	STRING (Short string)
↳ Display responsible role	ENUMERATION (Enumeration)
↳ Documentation (Metamodel)	STRING (Short string)
↳ Doku	STRING (Short string)
↳ DokuSim	STRING (Short string)
↳ Done by (Metamodel)	STRING (Short string)
↳ EDP batch costs	DOUBLE (Floating-point number)
↳ EDP transaction costs	DOUBLE (Floating-point number)
↳ Execution interruptable (Metamodel)	ENUMERATION (Enumeration)
↳ Execution time (Metamodel)	TIME (Time)
↳ External documentation	PROGRAMCALL (Program call)
↳ External tool coupling (Metamodel)	STRING (Short string)
↳ fontcolor (Metamodel)	EXPRESSION (Expression)
↳ For compensation	ENUMERATION (Enumeration)
↳ Global task	ENUMERATION (Enumeration)
↳ GraphRep (Metamodel)	LONGSTRING (Long string)
↳ HlpTxt (Metamodel)	STRING (Short string)
↳ Id	EXPRESSION (Expression)
↳ Info on results	STRING (Short string)

Buttons: New, Edit..., Copy..., Delete, View, Close, Help

# Modelling Language: Special Attribute GraphRep

GraphRep: A script language for the graphical representation



The screenshot shows a software window titled "Task - GraphRep". The window is divided into two main sections: "Text" and "View".

The "Text" section contains a script defining graphical elements and their attributes:

```
GRAPHREP sizing:asymmetrical smart-symbol-size
SHADOW off

AVAL tasktype:          "Task type"

AVAL oF:                "Open questions"
AVAL i:                 "Order"
AVAL set-default:"@" ext: "External documentation"

AVAL loopType:         "Loop type"
AVAL comp:             "For compensation"
AVAL isSequ:          "Sequential execution"

AVAL desc:             "Description"
AVAL set-default:"" sName_de: "Name"

AVAL sRepName: "Show name"
AVAL bInstanciating:  "Instantiate"

SET h:0
SET reihenfolge_ypos:0cm

TABLE rows:1 cols:1 x:-1.4cm y:-0.75cm w:2.8cm h:1.5cm w1:100% h1:100%
```

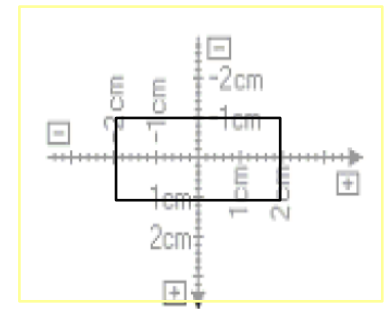
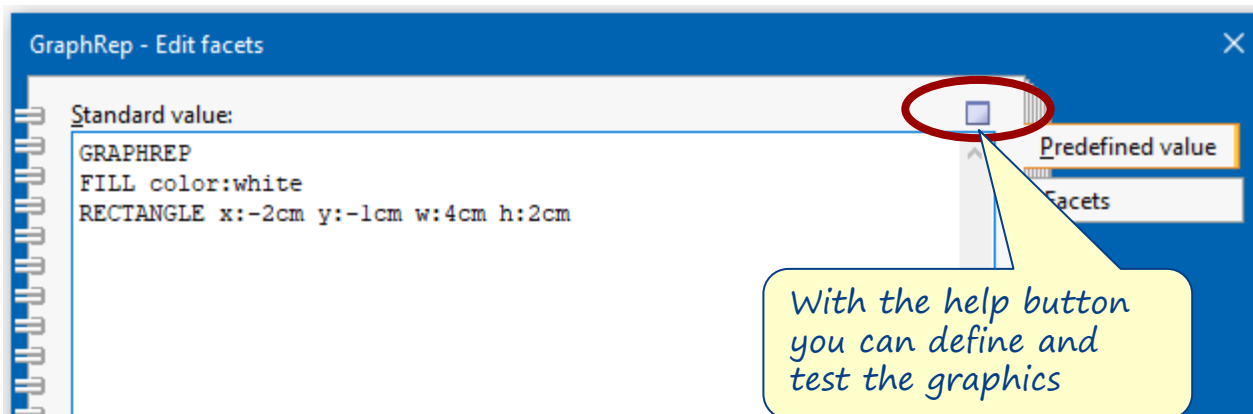
The "View" section shows a graphical representation of a single rounded rectangular box with a blue gradient fill and a blue border, centered in the view area.

On the right side of the window, there are four buttons: "Apply", "Paint", "Cancel", and "Help".

At the bottom of the window, a status bar indicates "7553 Character".

# Defining a GraphRep

- GraphRep allows to draw elements and display texts
- This example draws a white rectangle
  - ◆ It starts at the top right corner, is 2cm left from the center (x:-2cm) and 1 cm above the center (y:-1cm)
  - ◆ It is 4cm wide and 2 cm high (w:4cm h:2cm)

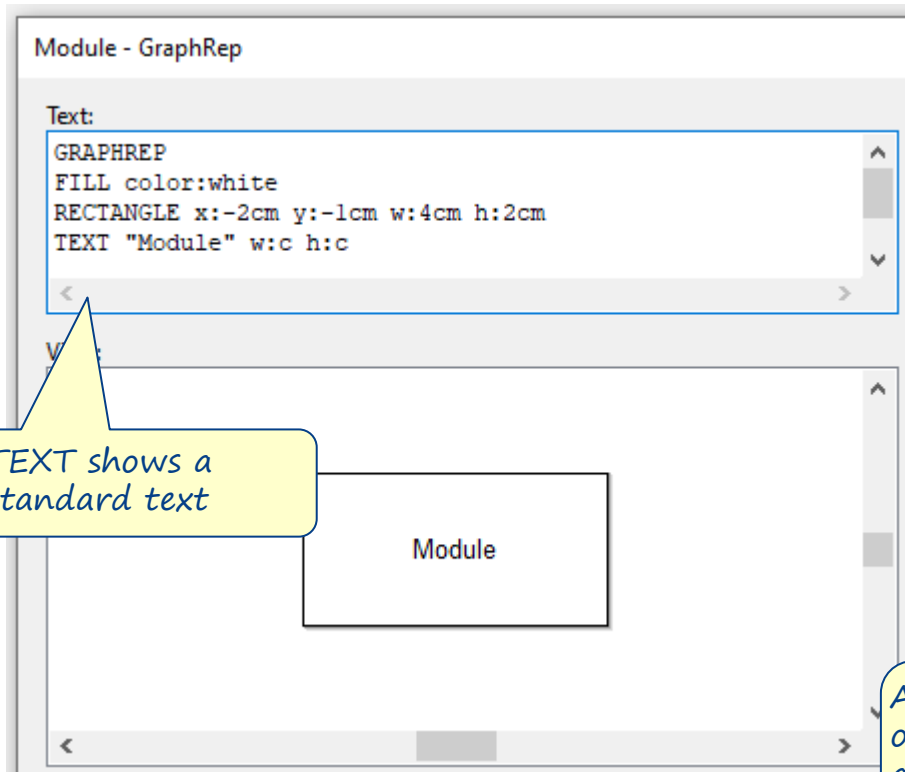


# Defining a GraphRep with Text

Module - GraphRep

```
Text:
GRAPHREP
FILL color:white
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
TEXT "Module" w:c h:c
```

Module



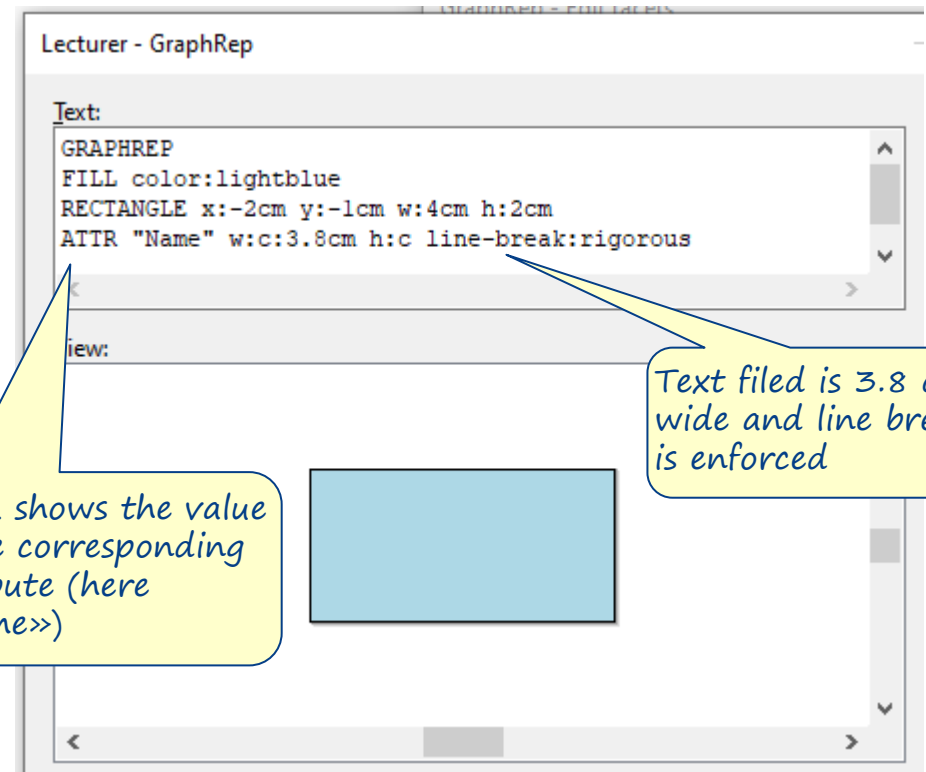
TEXT shows a standard text

ATTR shows the value of the corresponding attribute (here <<Name>>)

Lecturer - GraphRep

```
Text:
GRAPHREP
FILL color:lightblue
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous
```

iew:



Text filed is 3.8 cm wide and line break is enforced

# GraphRep Elements

## GraphRep Elements

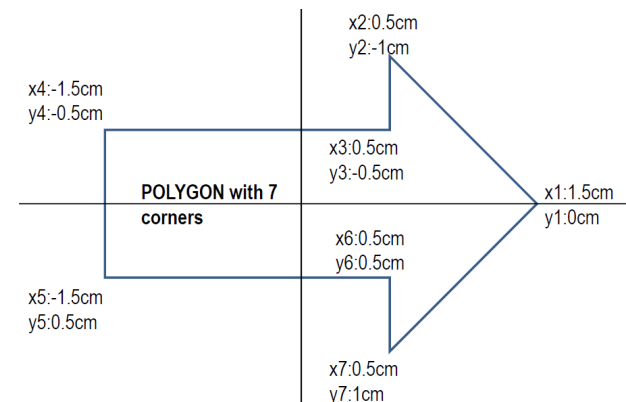
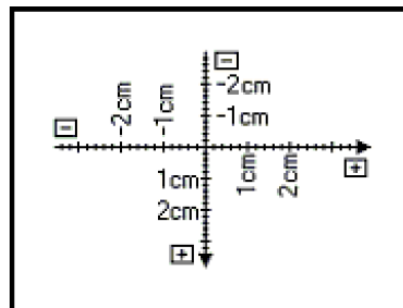
- Types of elements
  - ◆ Style elements
  - ◆ Shape elements
  - ◆ Variable assigning elements
  - ◆ Context elements
  - ◆ Control elements

```

Edge | Start | Middle | End |
Pen | Fill | Shadow | Stretch | Map | Font |
ClipRect | ClipRoundRect | ClipPoly | ClipEllipse | ClipOff |
Point | Line | PolyLine | Arc | Bezier | Curve |
Rectangle | RoundRect | Polygon | Ellipse | Pie |
BeginPath | MoveTo | LineTo | BezierTo |
EndPath | DrawPath |
Compound | Bitmap | GradientRect | GradientTri |
Text | Attr | Hotspot |
Set | Aval | Table | TextBox | AttrBox | BitmapInfo |
IfStatement | WhileStatement |
ForNumStatement | ForTokenStatement | Execute.

```

- Elements are placed on x-y-axes



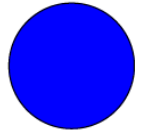
# GraphRep Examples

```

GRAPHREP
SHADOW off

FILL color:blue
ELLIPSE x:0.00cm y:0cm rx:1cm ry:1cm

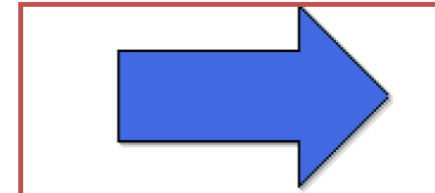
ATTR "Name" x:0.00cm y:1.0cm w:c
    
```



```

GRAPHREP
FILL color:royalblue
POLYGON 7 x1:1.5cm y1:0cm x2:0.5cm
y2:-1cm x3:0.5cm y3:-0.5cm x4:-1.5cm
y4:-0.5cm x5:-1.5cm y5:0.5cm
x6:0.5cm y6:0.5cm x7:0.5cm y7:1cm

ATTR "Name" y:1.4cm w:c h:c
    
```

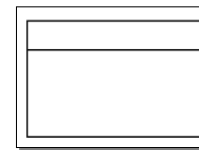


In case attribute name is available, it is shown here

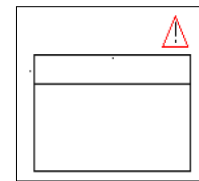
## Conditional Representation

```

GRAPHREP
AVAL set-default:"Modeling finished" b:"Status"
SHADOW off
FILL style:null
POLYGON 4 x1:-1.54cm y1:0.92cm x2:1.54cm y2:0.92cm
x3:1.54cm y3:-0.98cm x4:-1.54cm y4:-0.98cm
LINE x1:-1.54cm y1:-0.50cm x2:1.54cm y2:-0.50cm
IF (b = "Modeling not finished")
  LINE x1:1.25cm y1:-1.5cm x2:1.25cm y2:-1.3cm
  LINE x1:1.25cm y1:-1.22cm x2:1.25cm y2:-1.18cm
  PEN color:red
  POLYGON 3 x1:1cm y1:-1.1cm x2:1.25cm y2:-1.6cm
x3:1.50cm y3:-1.1cm
ENDIF
    
```



Condition fulfilled



Condition not fulfilled

# ADOxx GraphRep Repository

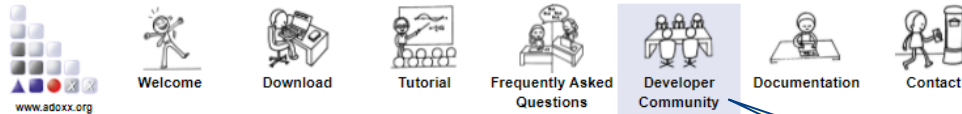
## GRAPHREP COLLECTION



A collection of implementation of graphical representation from different scenarios and projects are provided to the community as GRAPHREP code snippets.

As a community member, feel free to add, revise, modify, comment and rate the GRAPHREPs available in this repository.

USE



ADOxx.org > Developer Community > ADOxx Knowledge Base > ADOxx GraphRep Repository Wiki > ADOxx GraphRep Repository

FrontPage | Recent Changes | All Pages | Draft Pages  Search

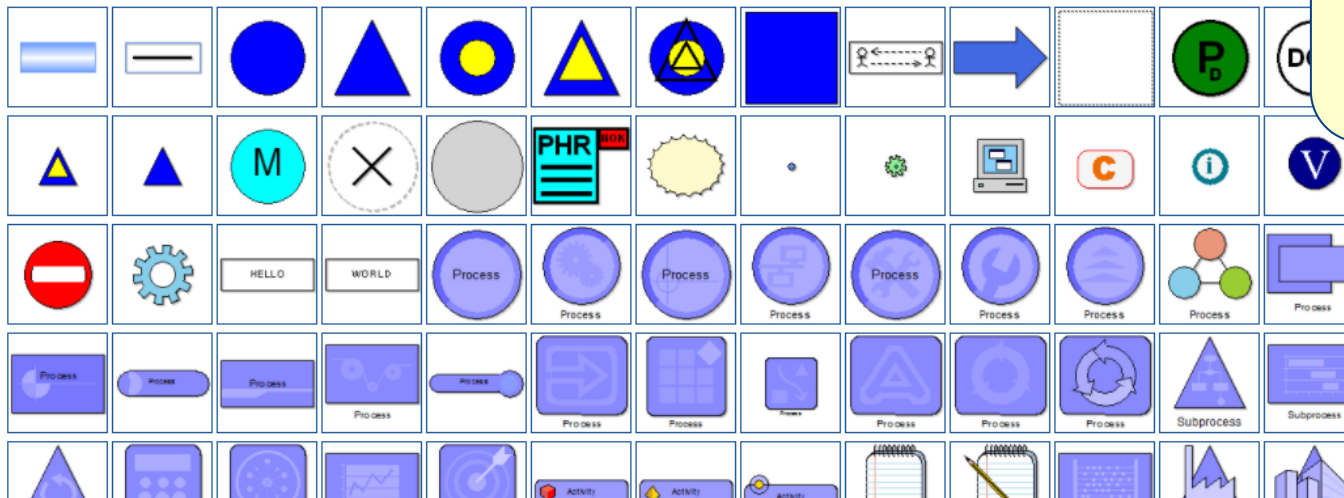
## ADOxx GraphRep Repository

(Redirected from FrontPage)

Tags: graphrep

The ADOxx GraphRep repository collects implementation of graphical representation from different scenarios and projects and provides them to the community. As a community member, feel free to add, revise, use, modify, comment and rate the GraphReps available in the repository.

### CLASSES



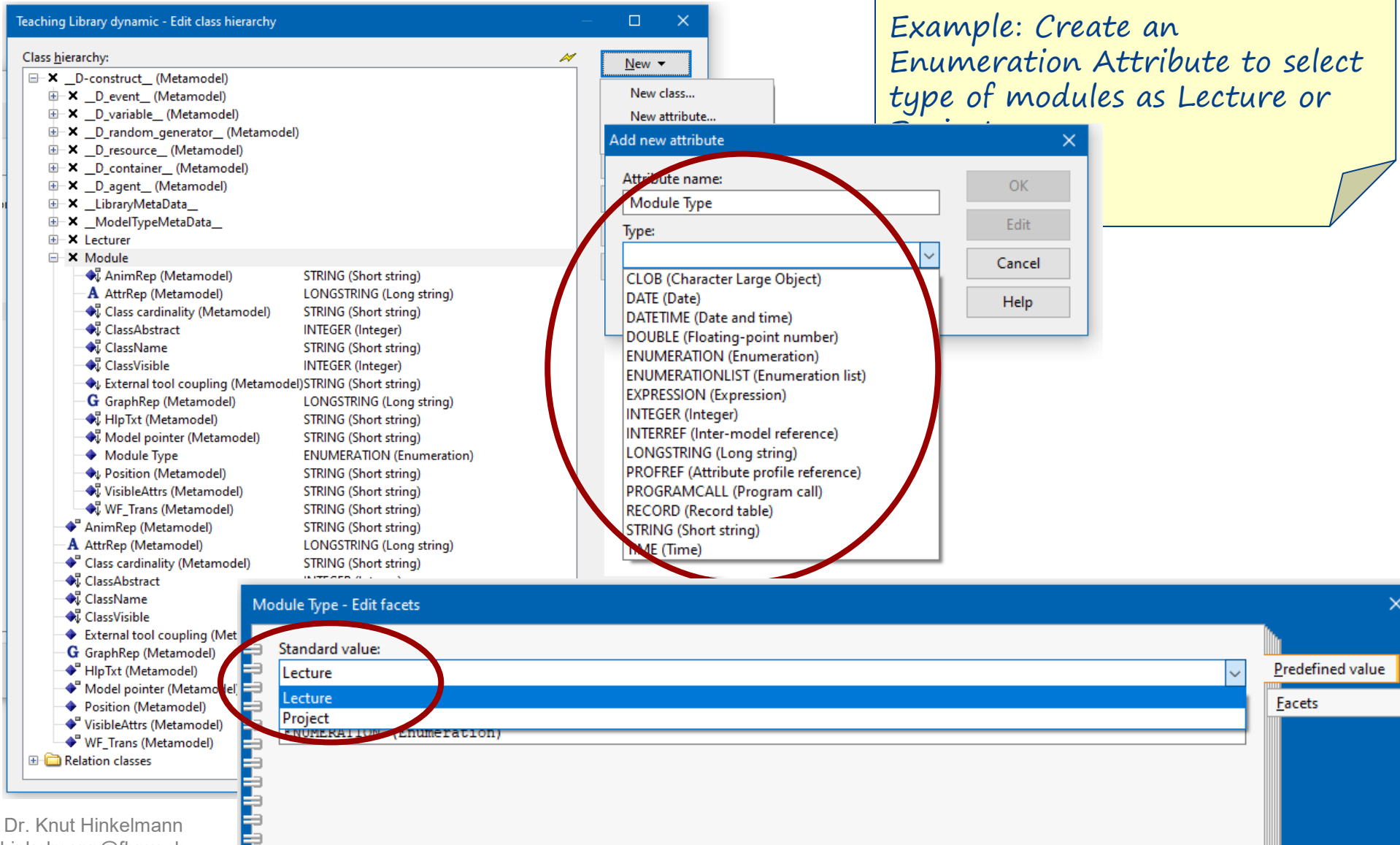
Examples of GraphReps can be found in the ADOxx Developer Community



# Defining a new Attribute

1. Select Class
2. Right Click or select «New Attribute ...»
3. Define Attribute

Example: Create an Enumeration Attribute to select type of modules as Lecture or



Teaching Library dynamic - Edit class hierarchy

Class hierarchy:

- [-] X \_D-construct\_ (Metamodel)
- [+] X \_D\_event\_ (Metamodel)
- [+] X \_D\_variable\_ (Metamodel)
- [+] X \_D\_random\_generator\_ (Metamodel)
- [+] X \_D\_resource\_ (Metamodel)
- [+] X \_D\_container\_ (Metamodel)
- [+] X \_D\_agent\_ (Metamodel)
- [+] X \_LibraryMetaData\_
- [+] X \_ModelTypeMetaData\_
- [+] X Lecturer
- [+] X Module
  - [-] AnimRep (Metamodel) STRING (Short string)
  - [+] AttrRep (Metamodel) LONGSTRING (Long string)
  - [-] Class cardinality (Metamodel) STRING (Short string)
  - [-] ClassAbstract INTEGER (Integer)
  - [-] ClassName STRING (Short string)
  - [-] ClassVisible INTEGER (Integer)
  - [-] External tool coupling (Metamodel) STRING (Short string)
  - [+] GraphRep (Metamodel) LONGSTRING (Long string)
  - [-] HlpTxt (Metamodel) STRING (Short string)
  - [-] Model pointer (Metamodel) STRING (Short string)
  - [-] Module Type ENUMERATION (Enumeration)
  - [-] Position (Metamodel) STRING (Short string)
  - [-] VisibleAttrs (Metamodel) STRING (Short string)
  - [-] WF\_Trans (Metamodel) STRING (Short string)
- [-] AnimRep (Metamodel) STRING (Short string)
- [+] AttrRep (Metamodel) LONGSTRING (Long string)
- [-] Class cardinality (Metamodel) STRING (Short string)
- [-] ClassAbstract
- [-] ClassName
- [-] ClassVisible
- [-] External tool coupling (Metamodel)
- [+] GraphRep (Metamodel)
- [-] HlpTxt (Metamodel)
- [-] Model pointer (Metamodel)
- [-] Position (Metamodel)
- [-] VisibleAttrs (Metamodel)
- [-] WF\_Trans (Metamodel)

Relation classes

New

New class...

New attribute...

Add new attribute

Attribute name: Module Type

Type: ENUMERATION (Enumeration)

OK

Edit

Cancel

Help

Module Type - Edit facets

Standard value: Lecture

Lecture

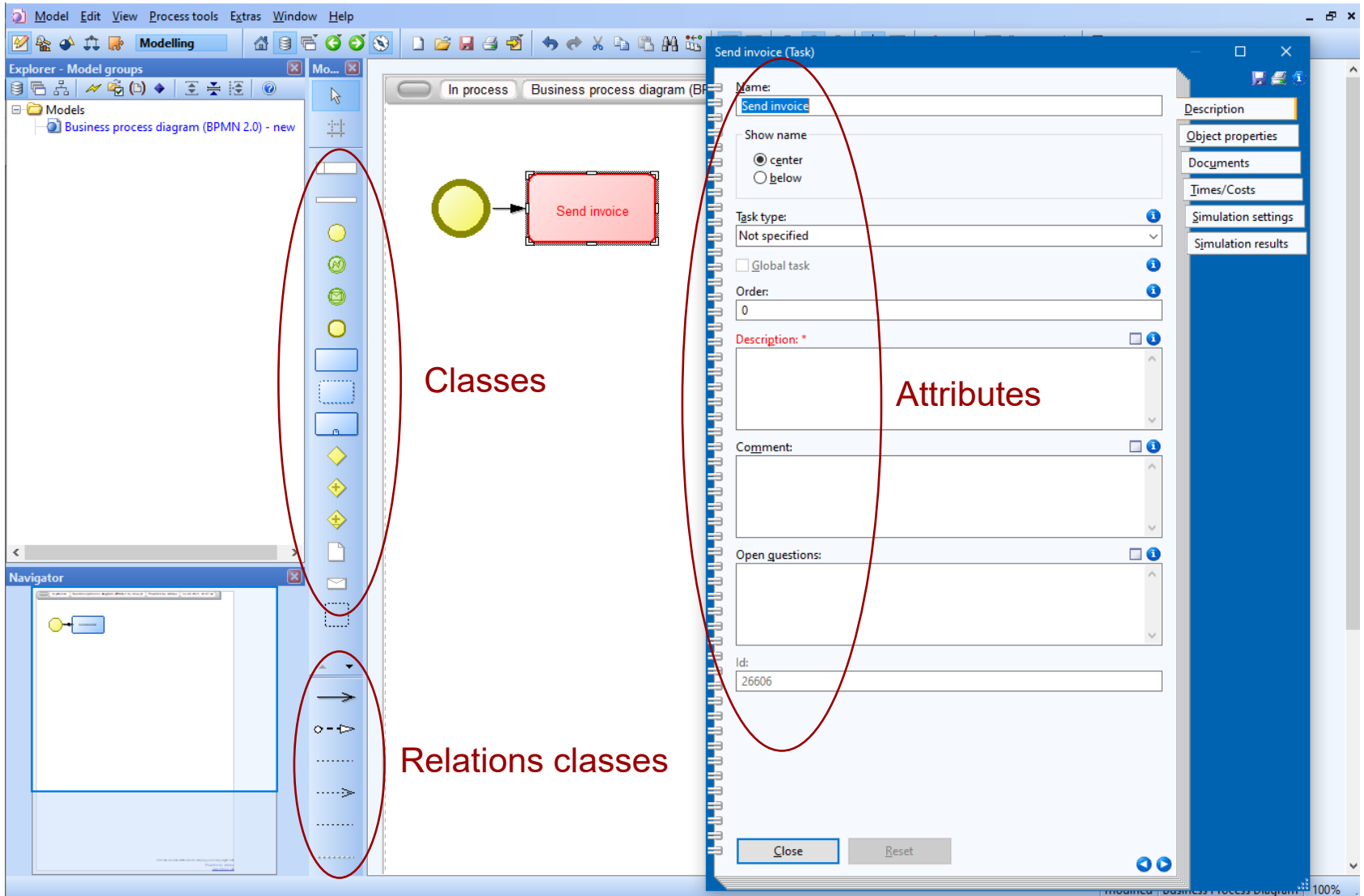
Project

ENUMERATION (Enumeration)

Predefined value

Facets

# Notebook: Adding Attribute Values



The screenshot displays the software interface for modeling a Business Process Diagram (BPMN 2.0). The main workspace shows a diagram with a yellow circle (Start event) connected to a red rounded rectangle (Task) labeled "Send invoice".

Three red ovals highlight specific areas:

- Classes:** The top section of the left toolbar, containing various BPMN symbols like start events, tasks, and end events.
- Relations classes:** The bottom section of the left toolbar, containing symbols for flow types such as start, intermediate, and end events.
- Attributes:** The "Send invoice (Task)" configuration panel on the right, which includes fields for Name, Show name, Task type, Order, Description, Comment, and Open questions.

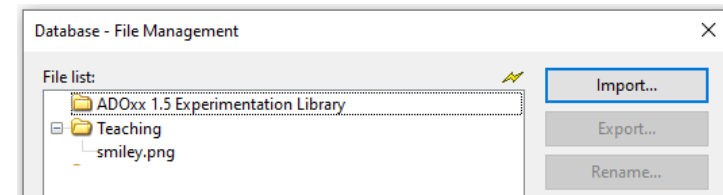
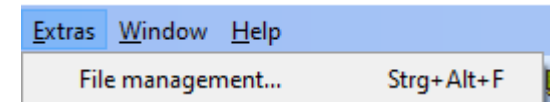
# AttrRep

The class attribute „AttrRep“ controls the structure of the ADOxx-Notebook.



# Adding Icons to GraphRep

- Icons can be imported into the System
  - ◆ Choose «File management ...» in Extras
  - ◆ Select the file and load into your library



- In GraphRep add the icon using the BITMAP

```
GRAPHREP
FILL color:white
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c h:c
BITMAP "db:\\smiley.png" x:1.4cm y:-0.9cm w:0.5cm h:0.5cm
```



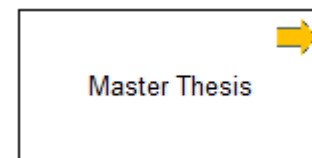
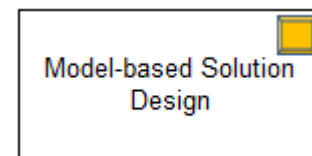
```
BITMAP "db:\\smiley.png" x:1.4cm y:-0.9cm w:0.5cm h:0.5cm
```

# Dynamic GraphRep

- Appearance of elements can be modified depending on values of attributes
- Example: Add an Icon for Module depending on the module type
  - ◆ The first line declares a variable «moduletype» using AVAL and assigns the value of the attribute «Module Type»
  - ◆ Add Icon if variable has specific value

```
AVAL moduletype: "Module Type"  
IF (moduletype = "Lecture")  
  BITMAP "db:\\lecture.png"    x:1.4cm y:-0.9cm w:0.5cm h:0.5cm  
ENDIF
```

```
AVAL moduletype: "Module Type"  
IF (moduletype = "Lecture")  
  BITMAP "db:\\lecture.png"    x:1.4cm y:-0.9cm w:0.5cm h:0.5cm  
ELSIF (moduletype = "Project")  
  BITMAP "db:\\yellowarrow.png"  x:1.4cm y:-0.9cm w:0.5cm h:0.5cm  
ENDIF
```



# Using Images for Visualization

- Instead of drawing a visualization with GRAPHREP, it is also possible to use images.



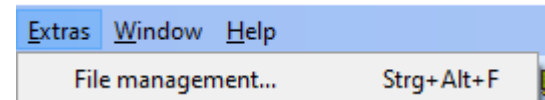
- This example uses a file graduation\_hat.png.

```
GRAPHREP
```

```
BITMAP "db:\\graduation_hat.png"    x:-2cm y:-1cm w:4cm h:1.6cm
```

```
ATTR "Name" y:1cm w:c:4cm h:c line-break:rigorous
```

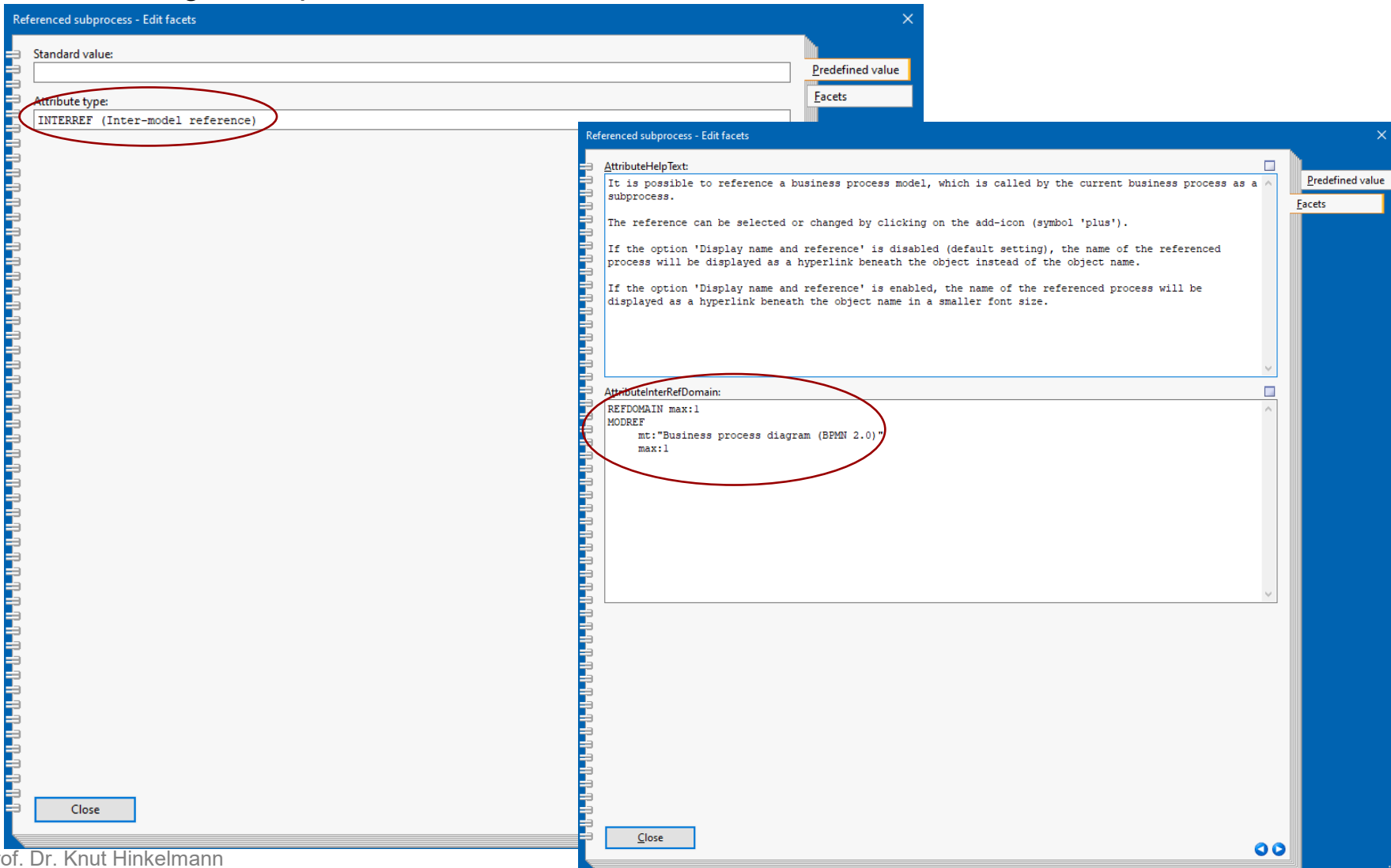
- ◆ The keyword «BITMAP» indicates the use of a file. (The file has to be uploaded using the file management.)



- ◆ Behind the name of the file there is the coordinate of the upper left corner (**x:-2cm**, **y:-1cm**) and the size (**w:4cm** **h:1.6cm**)
- ◆ The last line indicates that the name should be placed 1cm below the center, is 4cm wide and lines are broken if they are too long.

# References

## Referencing a Subprocess



The image shows two overlapping dialog boxes titled "Referenced subprocess - Edit facets".

The top dialog box has the following fields:

- Standard value: [Empty text box]
- Attribute type: INTERREF (Inter-model reference) [Circled in red]

The bottom dialog box has the following fields:

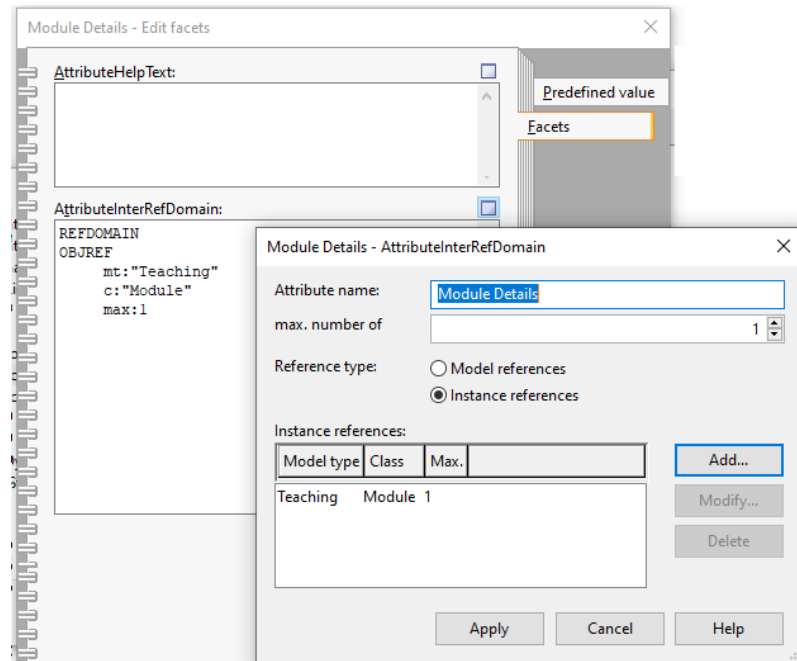
- AttributeHelpText: [Text area containing help text about referencing a business process model]
- AttributeInterRefDomain: [List of domains with values, circled in red]

The "AttributeInterRefDomain" field contains the following text:

```
REFDOMAIN max:1  
MODREF  
  mt:"Business process diagram (BPMN 2.0)"  
  max:1
```

# Reference

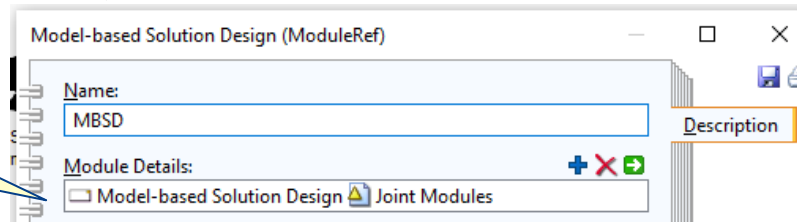
- References are used to make relations to another model or an element in another models
  - ◆ Create an attribute with data type «INTERREF»
  - ◆ In the facets specify whether it is a reference to a model or an element and determine the appropriate types



- Example:

- ◆ In a model of a curriculum create a class which contains a reference to the module description, which is modeled in another module

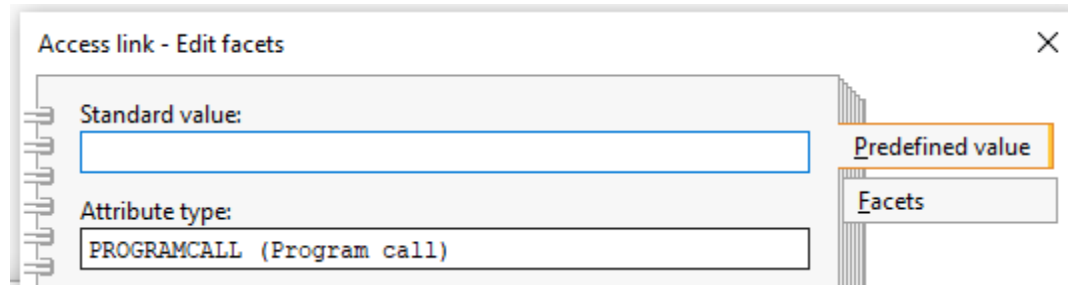
*Reference to the module "Model-based Solution Design" in the model "Joint Modules"*



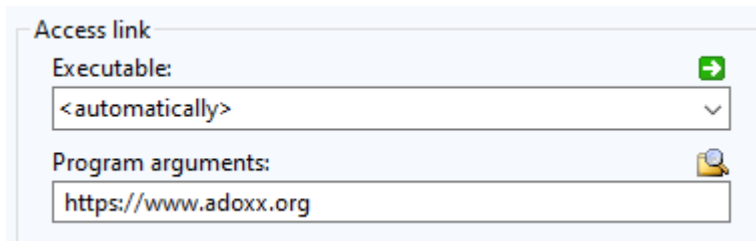


# Attributes for Documents, Websites, Applications

- Attributes of type PROGRAMCALL allow to provide links to document, websites or applications



- This is how it looks like in a notebook



## Clickable Links

- References from INTERREF or links to documents, website or applications can be made accessible directly from the model
- This GRAPHREP the displayed name of the element is a link to the value of the attribute Module Details

```
AVAL sname: "Name"
```

```
ATTR "Module Details" text:(sname) w:c h:c
```

- ◆ AVAL assigns the value of the attribute **Name** to the variable **sname**
- ◆ The second line displays the value of that variable and has as link the value of the attribute **Module Details**




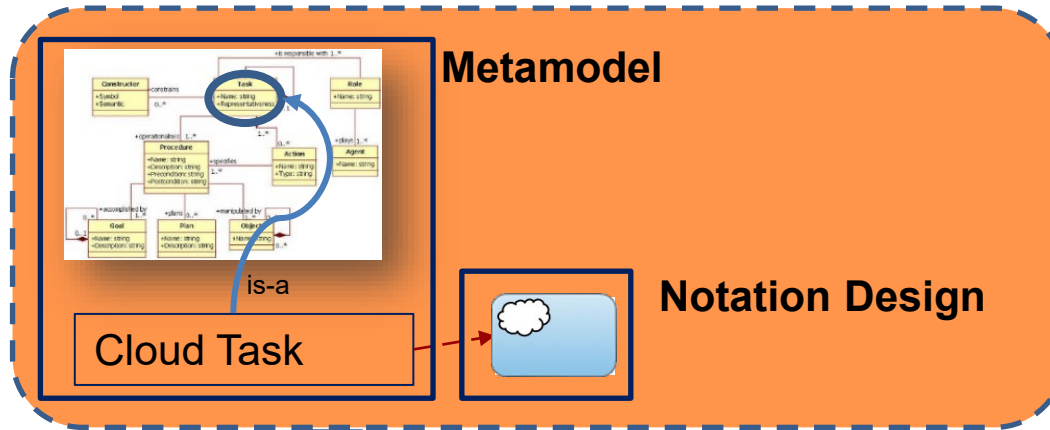
[Alignment of  
Business and IT](#)

# Change of Metamodel

- Example: new task type Cloud Task



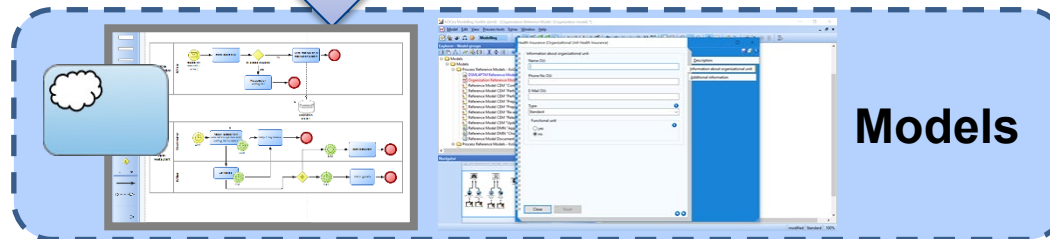
  
 Metamodel  
 Engineer



Meta-  
modeling

Feedback  
Amendments  
Improvements

  
 Modeler



Modeling