



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

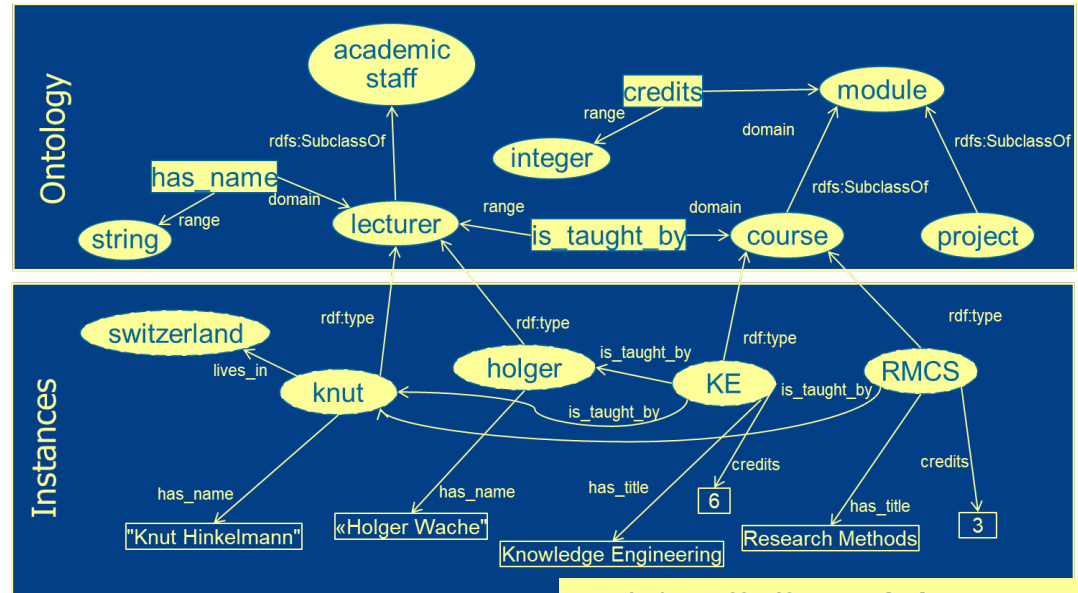
Knowledge Engineering = Modelling

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" .
  
```

General-purpose Modelling vs. Conceptual Modelling

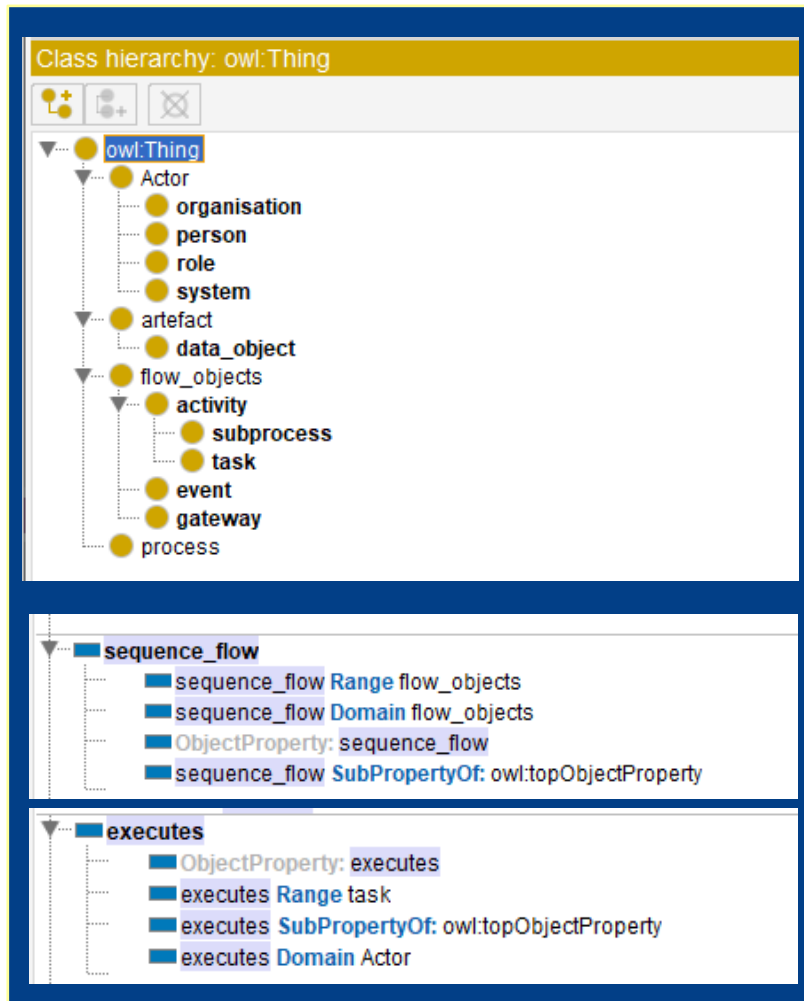
Modelling

Representing all relevant aspects of a domain in a defined language.

- **Ontology languages** like RDFS are called **general-purpose** modelling languages, because they can be used to represent knowledge about any domain
- **Conceptual modeling** creates models using **predefined** concepts that are specific for a domain
Conceptual modeling languages are also called domain-specific modeling languages

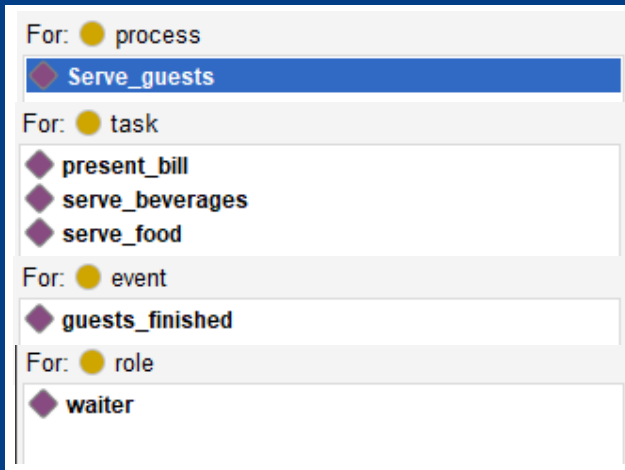
Example: Concepts and Instances for Process Modelling

Business Process Ontology (Metamodel):

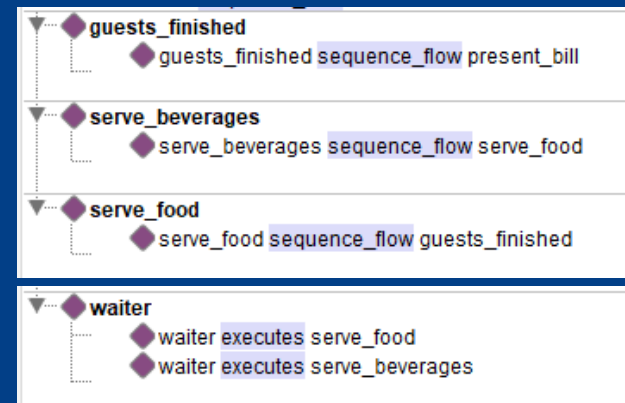


Process Model for Serve Guests

Instances:



Relations:



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

Modeling and Metamodeling

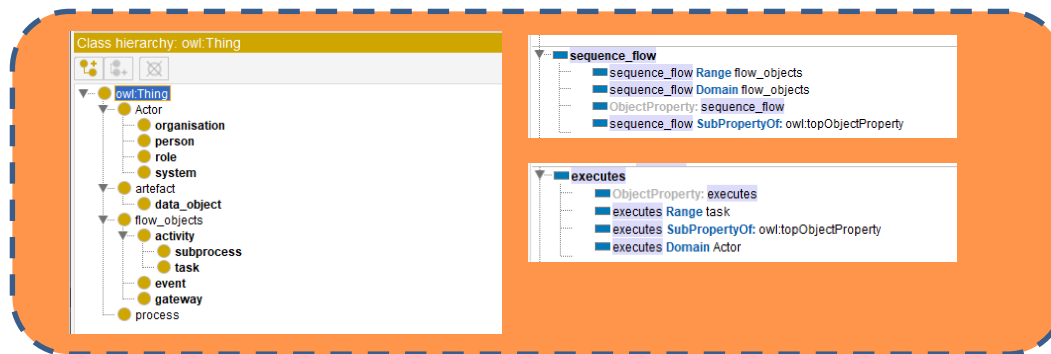
Conceptual Modeling consists of two phases

Metamodeling: Defining (domain-specific) concepts

Modeling: Creating models using these concepts



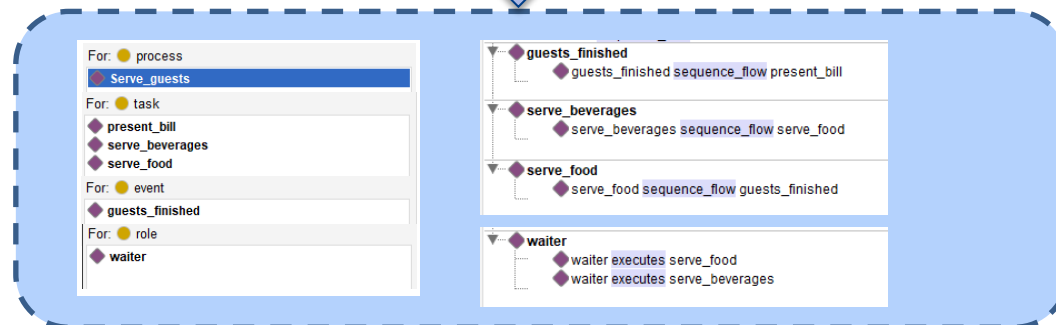
Metamodel Engineer



Meta-modeling



Modeler



Modeling

Strengths and Weaknesses of Conceptual Modelling

■ Strengths

◆ 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)

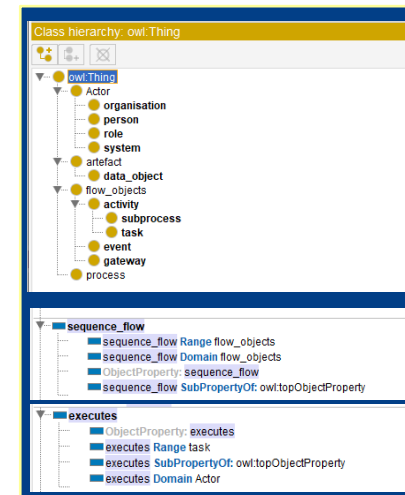
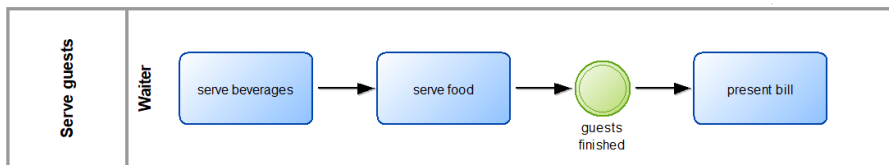
■ Weaknesses

◆ Restricted to a specific domain

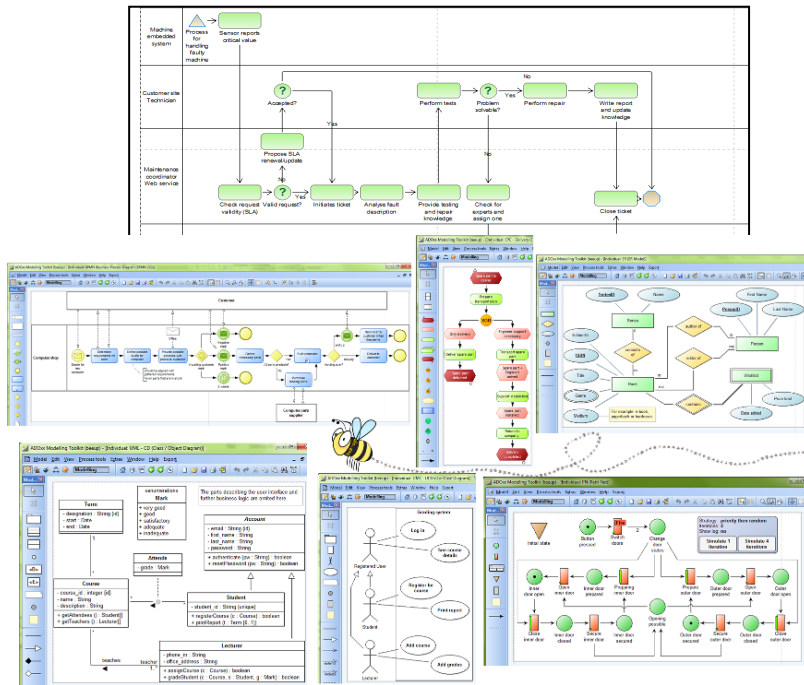
- Only what can be expressed with the modelling elements can be modeled

Knowledge Engineering using Graphical Models

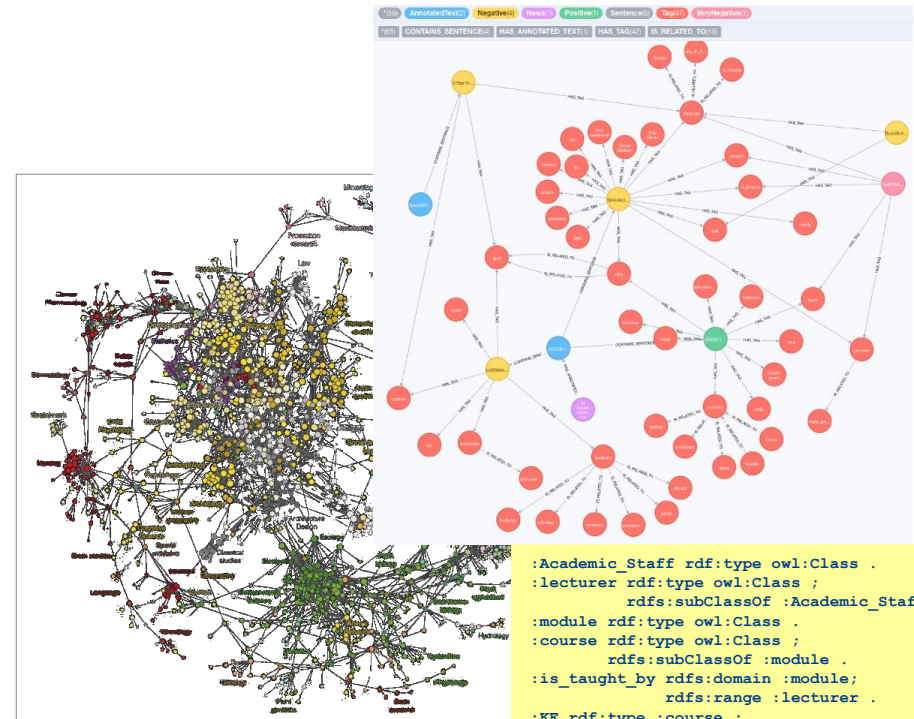
- Creating knowledge graphs is difficult for domain experts – it requires skills in modeling language
- Creating graphical models is more adequate for non-technical
- Objective: Create ontologies (knowledge graphs) from graphical models



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*

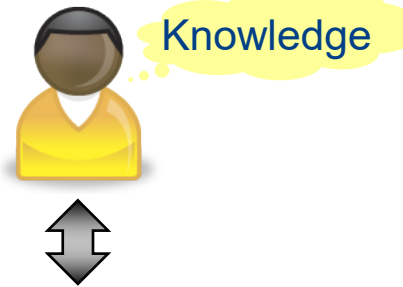
Agenda

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

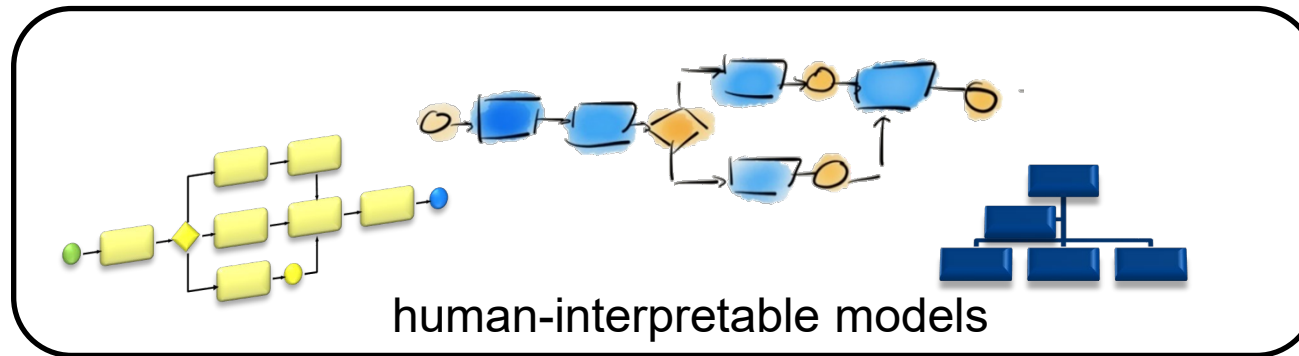
Conceptual Modeling and Metamodelling with Graphical Models

Graphical Models are appropriate for Humans

*Communication/
Analysis/
Decision Making*



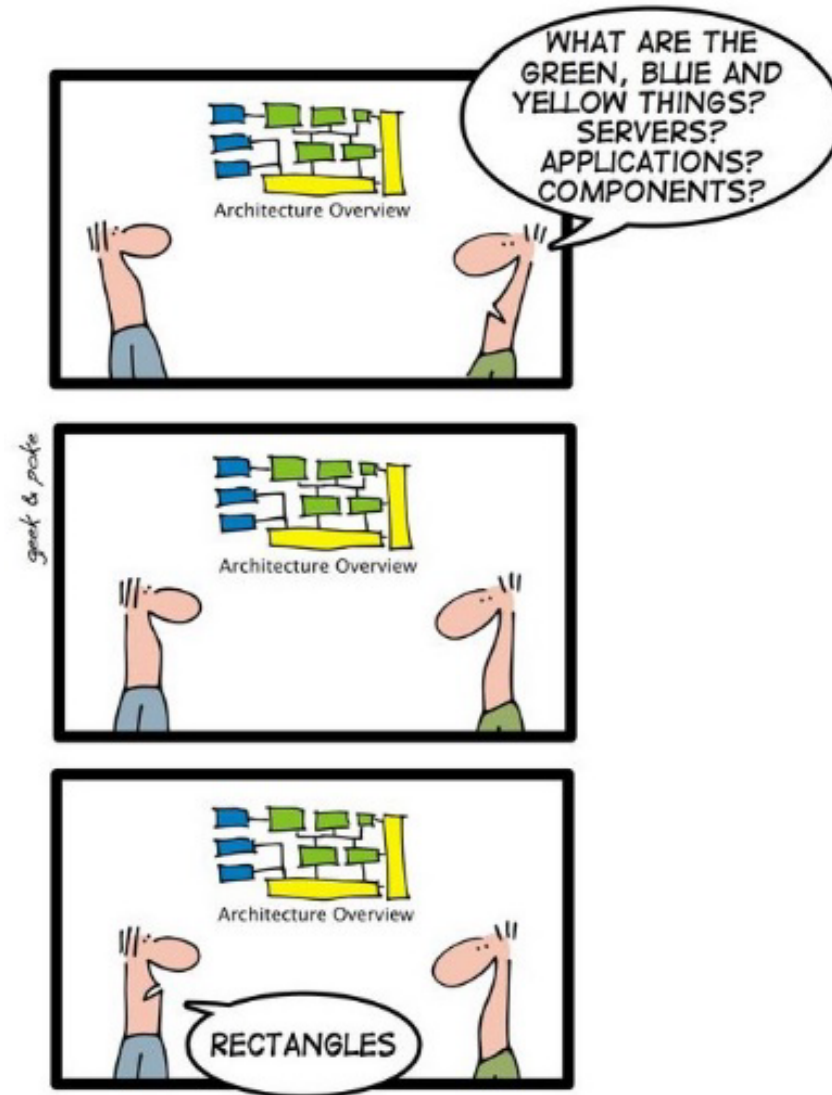
Models



Reality



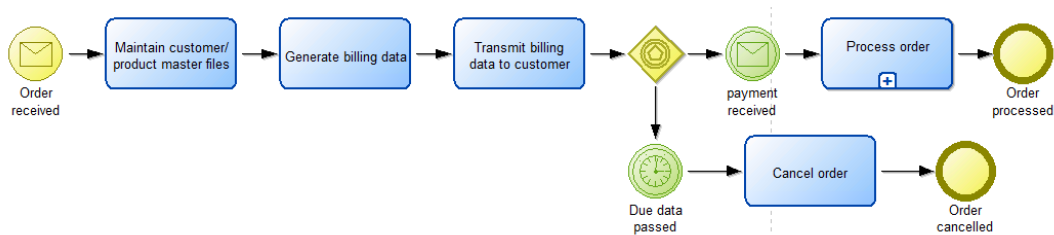
Interpretation of Models



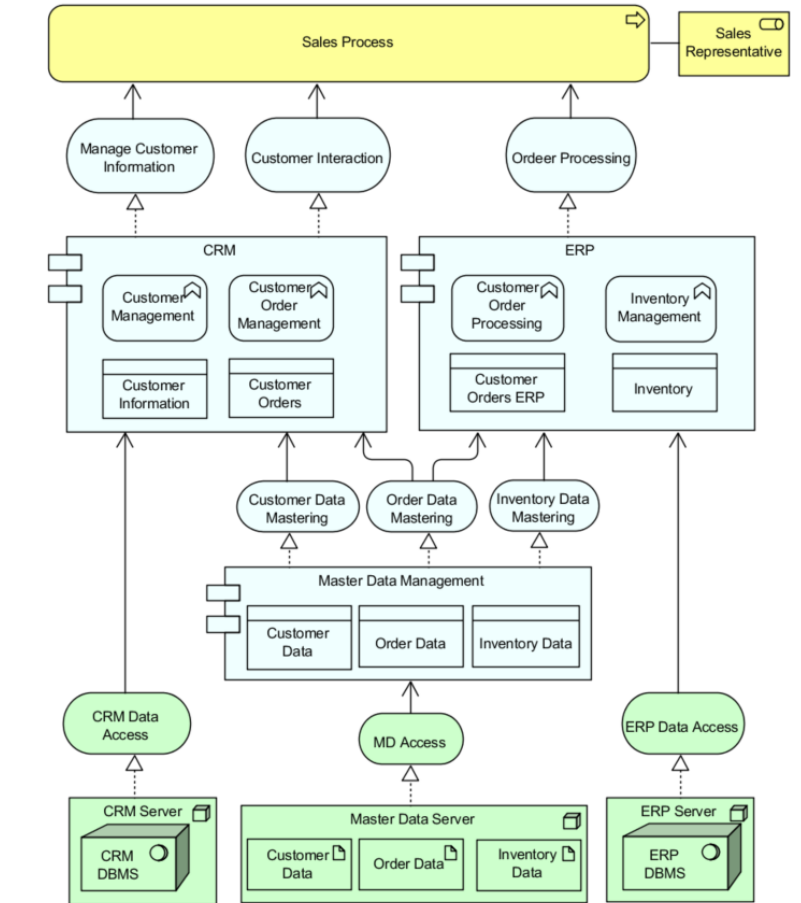
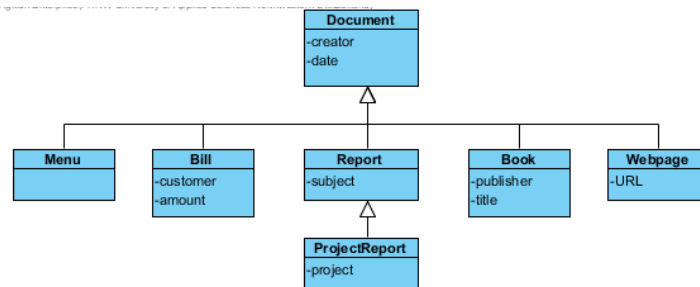
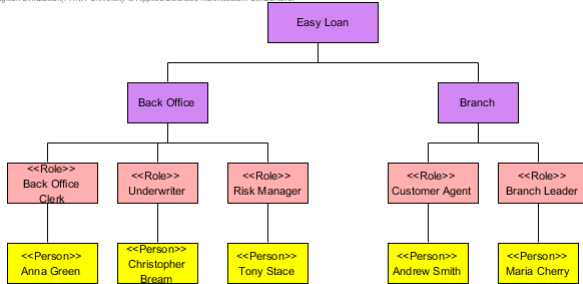
Models

- 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.

Enterprise Models using Domain-Specific Graphical Modeling



Agilan Simulacran/FHNW University of Applied Sciences Northwestern Switzerland

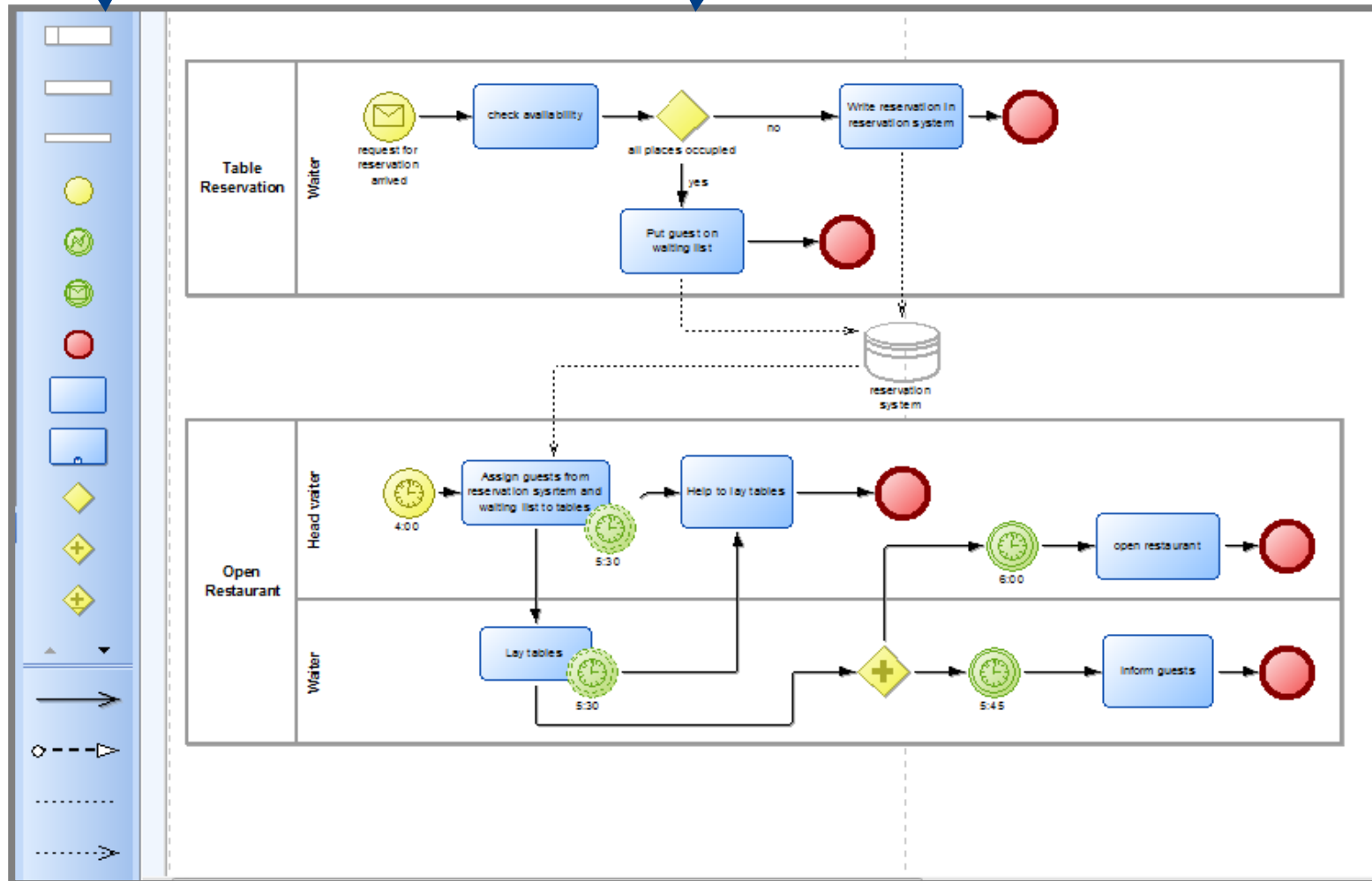


Domain-specific Graphical Modelling Languages

- Conceptual modeling with graphical models is typical for enterprise modelling
- The predefined concepts (elements and relationships) are specific for a domain
- Examples of conceptual modelling languages for enterprise modelling:
 - ◆ **BPMN** for business processes
 - Elements: task, event, gateway,
 - Relationships: sequence flow, message flow, association, ...
 - ◆ **ArchiMate** for enterprise architectures
 - Elements: process, actor, role, business object, ...
 - Relationships: uses, realizes, ...

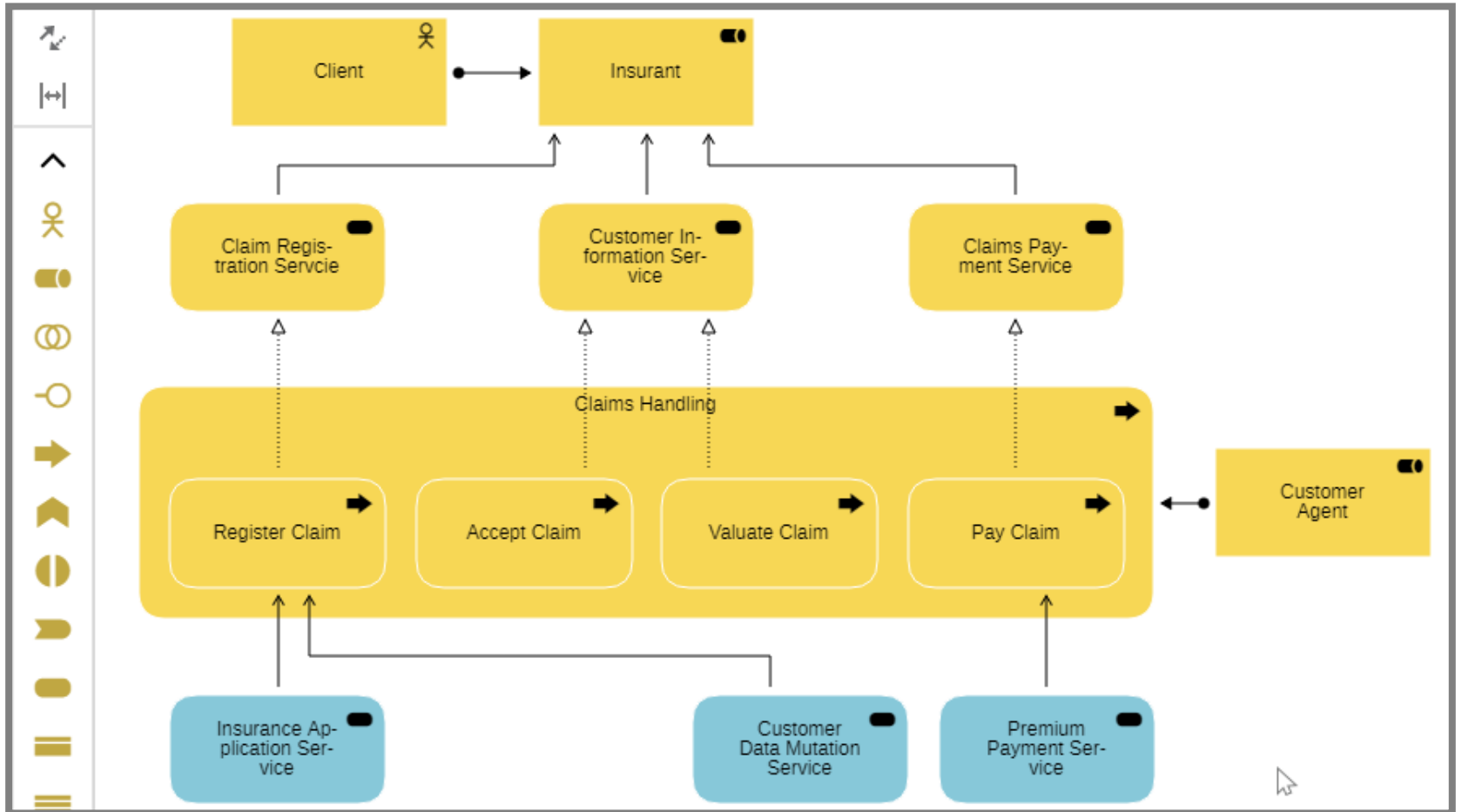
Modelling
Language

Model



Modelling Language

Model



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.

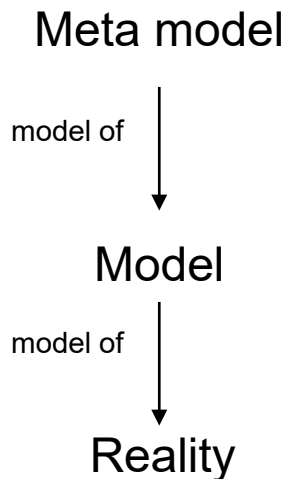
Meta Model

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

Modelling Language

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

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

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.

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

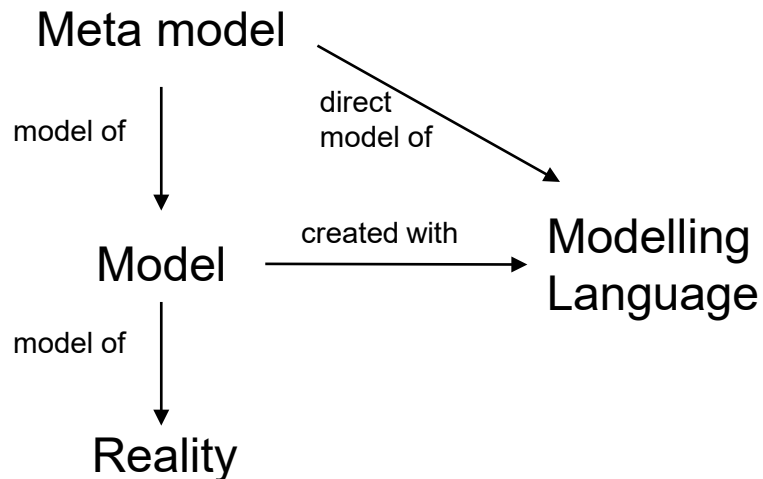


Illustration: Modeling Language for Business Processes

Meta model:

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

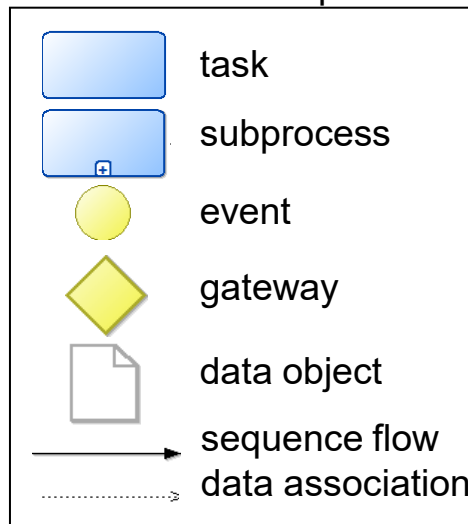


Illustration: Modeling Language for Business Processes

Metamodel:

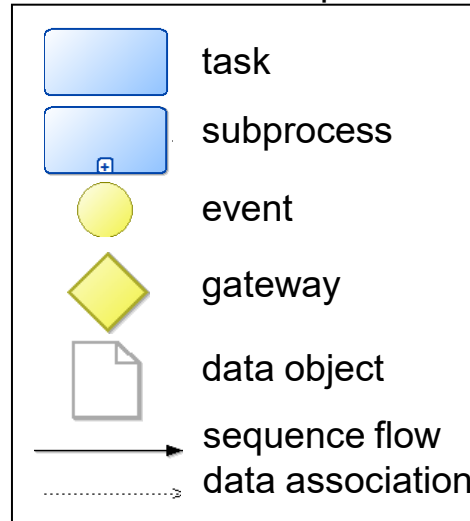
Concepts which can be used to create models.

Example: A process model consists of concepts for

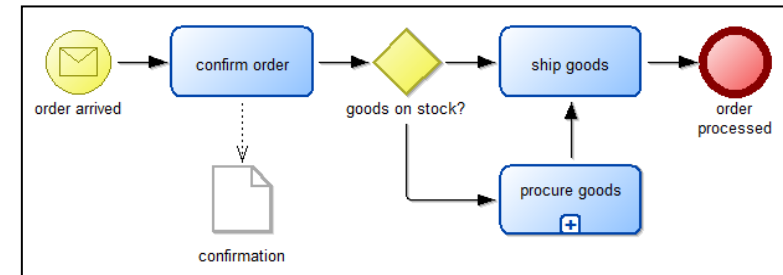
- Model elements: **event**, **task**, **subprocess**, **gateway**, **data object**
- Relationships: **sequence flow**, **data association**.

Modelling Language:

Notation/appearance of meta-model concept

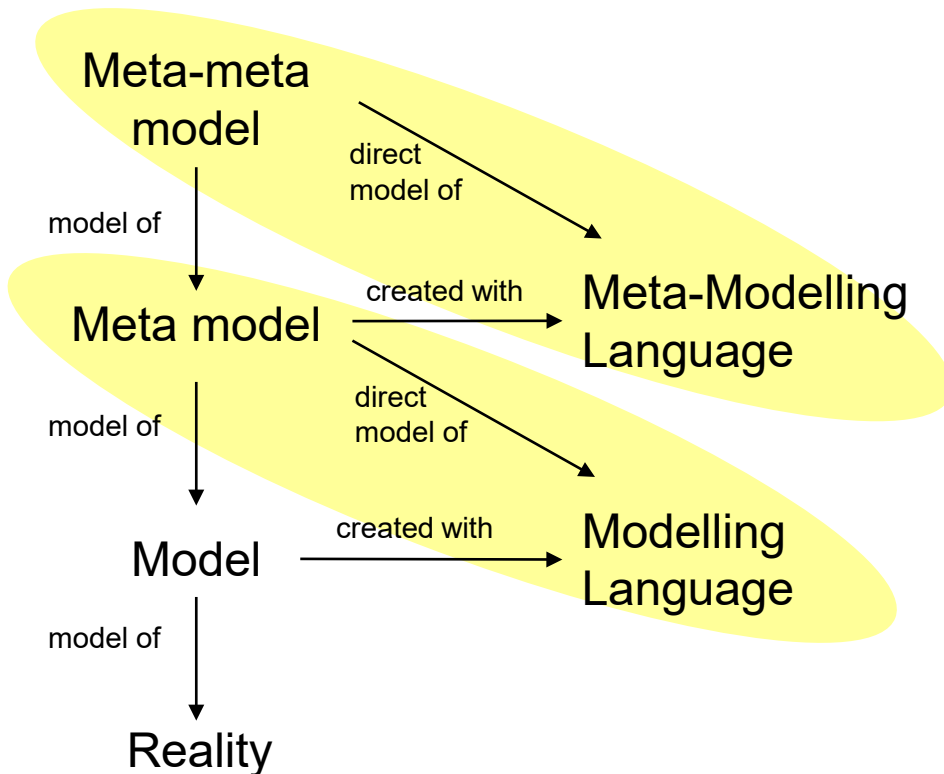


Model:



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»

Meta-meta model

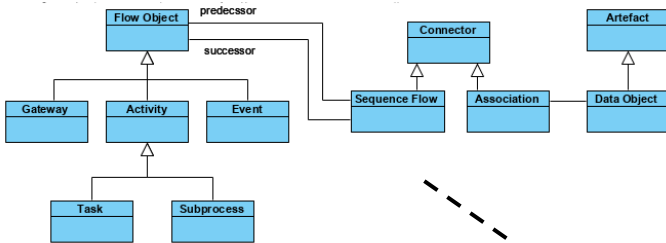


- The meta model must again be described in some language, which is specified in a meta-meta model
- A **meta-meta model** defines the concepts for describing a meta model
- Graphical models usually have to kinds of concepts
 - ◆ Modeling elements
 - ◆ Relationships
- Examples for meta-modeling languages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

Metamodels can be defined as Class Diagrams

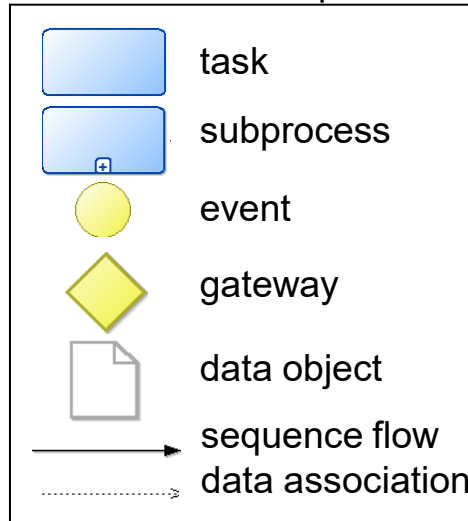
Metamodel:

Concepts which can be used to create models.

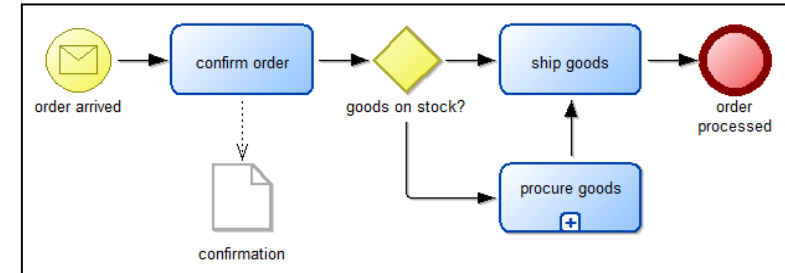


Modelling Language:

Notation/appearance of meta-model concept



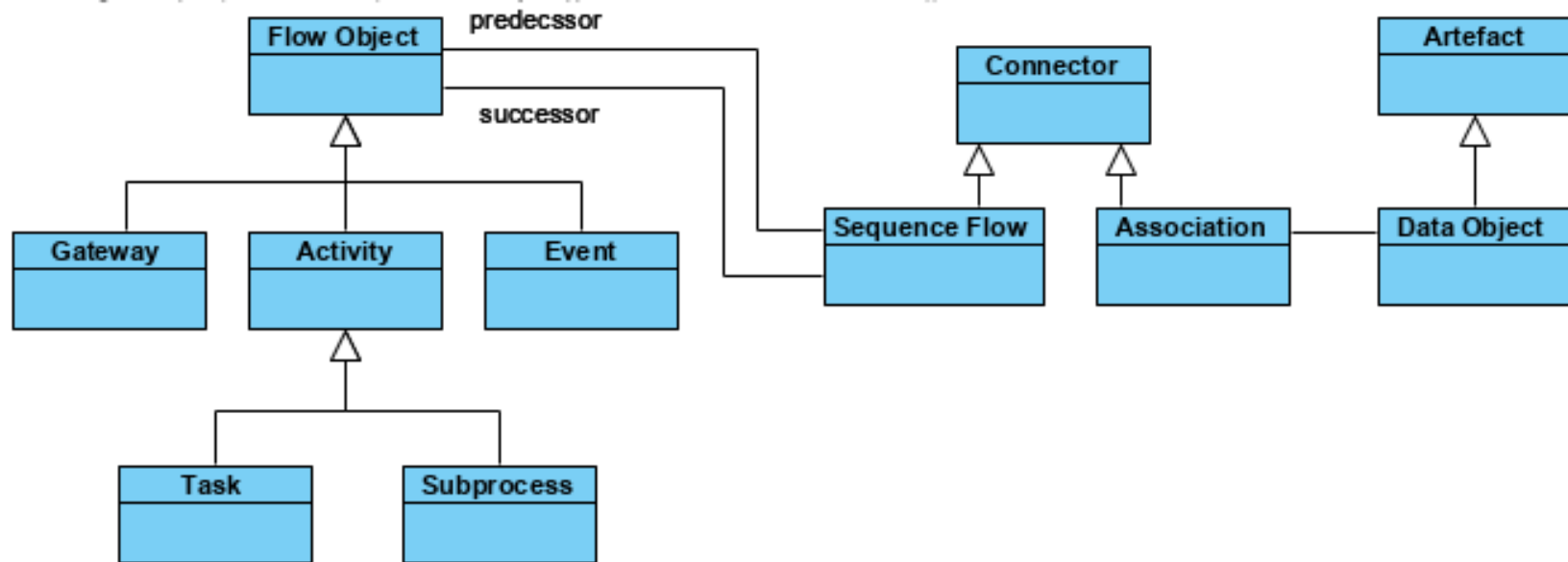
Model:



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»

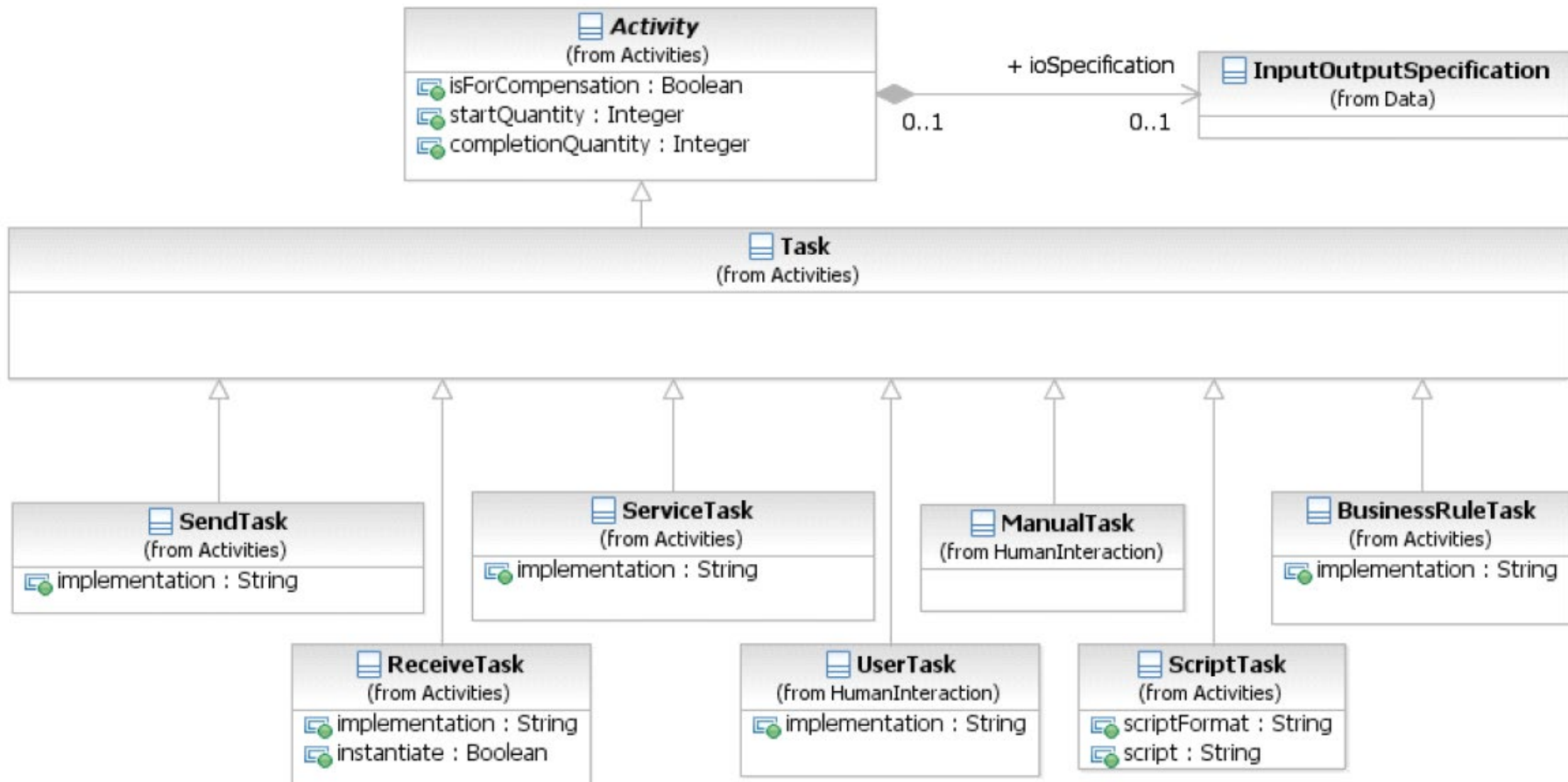
Metamodels can be defined as Class Diagrams

A Metamodeling language one can describe meta models
Metamodel corresponds to a knowledge base
Metamodels can be represented as class diagrams



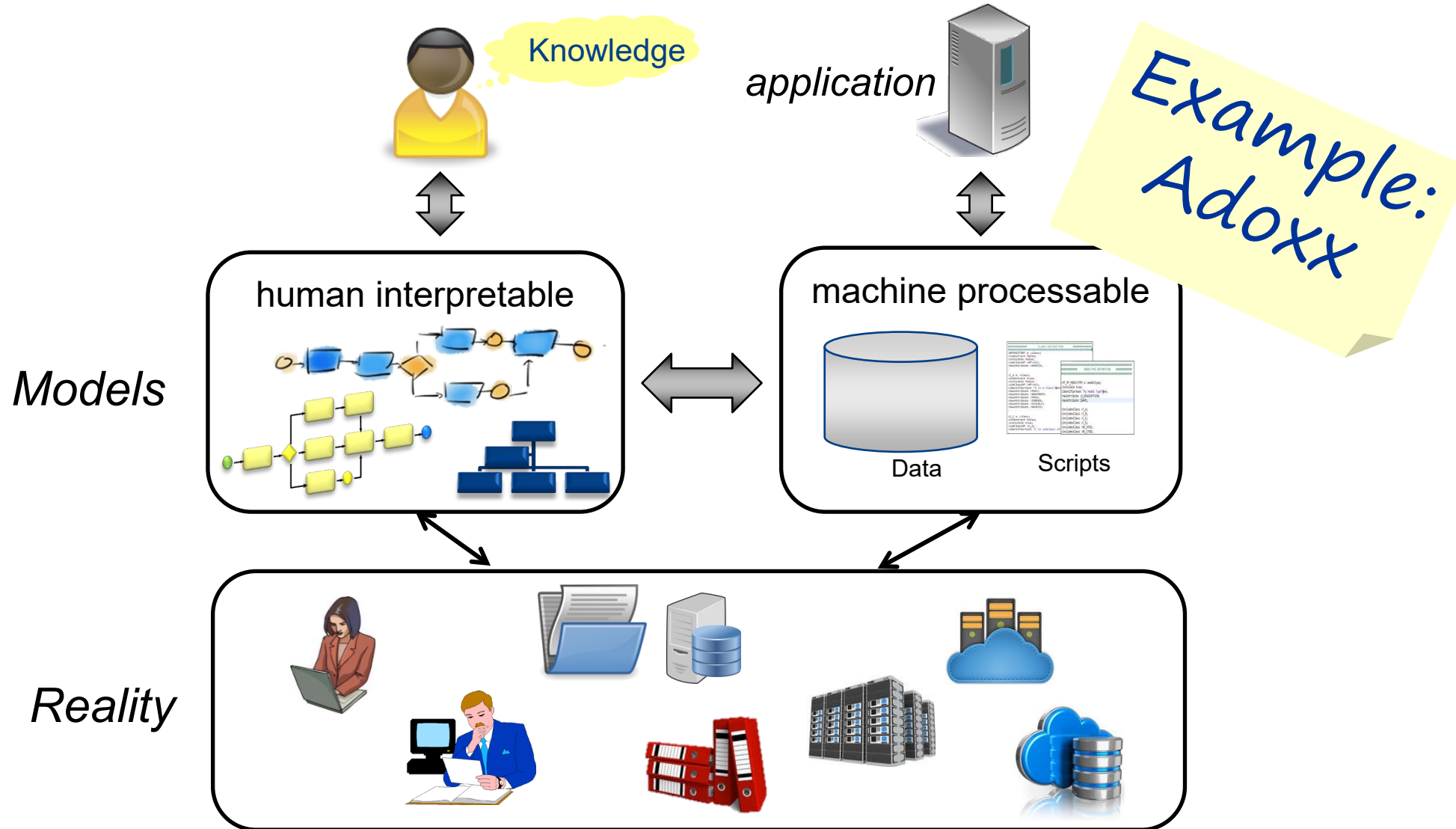
(UML Class diagrams were originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

Subset of the BPMN Metamodel as UML Class Diagram



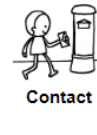
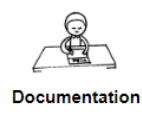
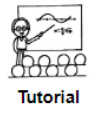
Metamodelling with ADOxx

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










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

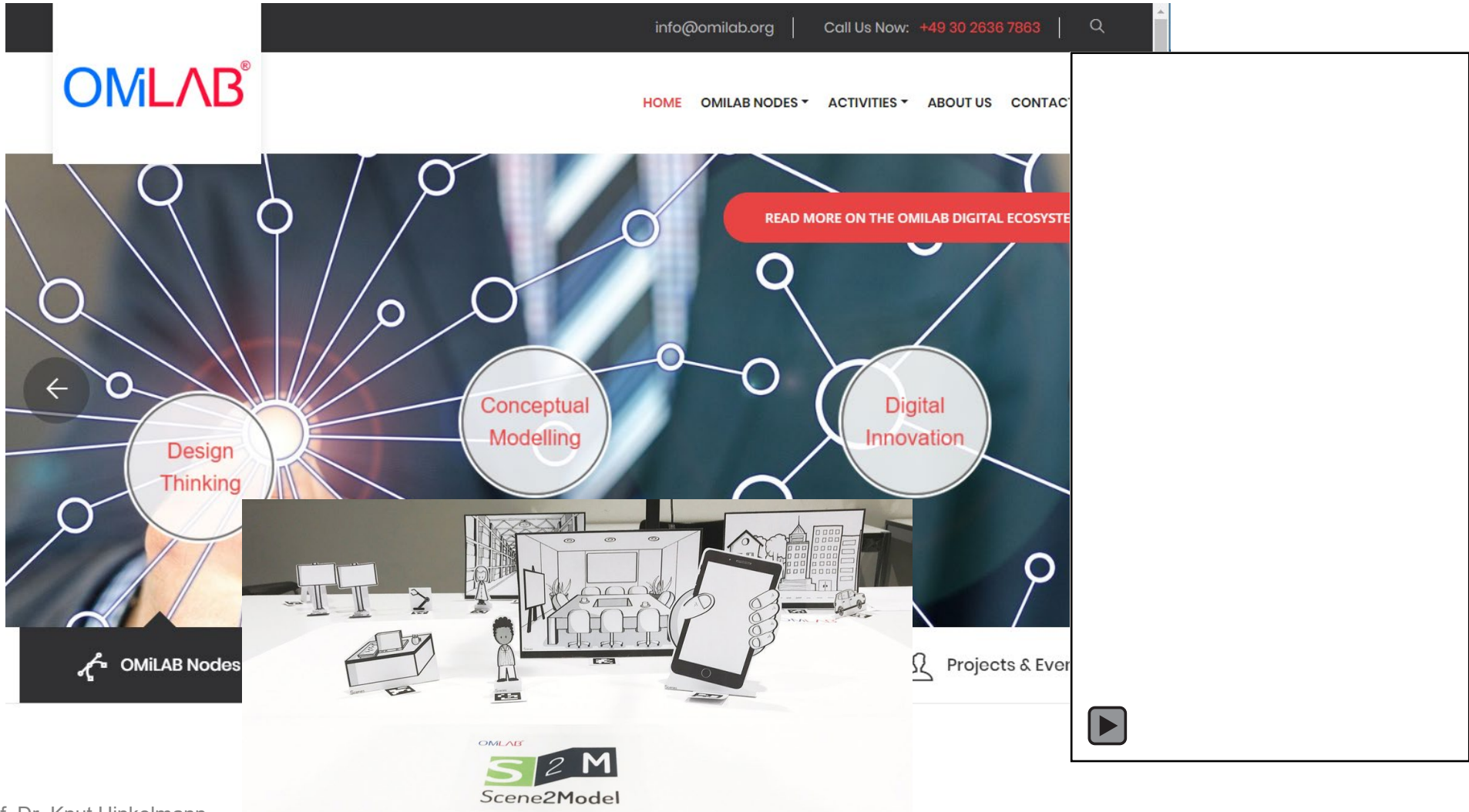


ADOxx Training Team
March 2020

Mar 28, 2020

OMiLAB – A Conceptual Modelling Community

ADOxx is the basis for OMiLAB

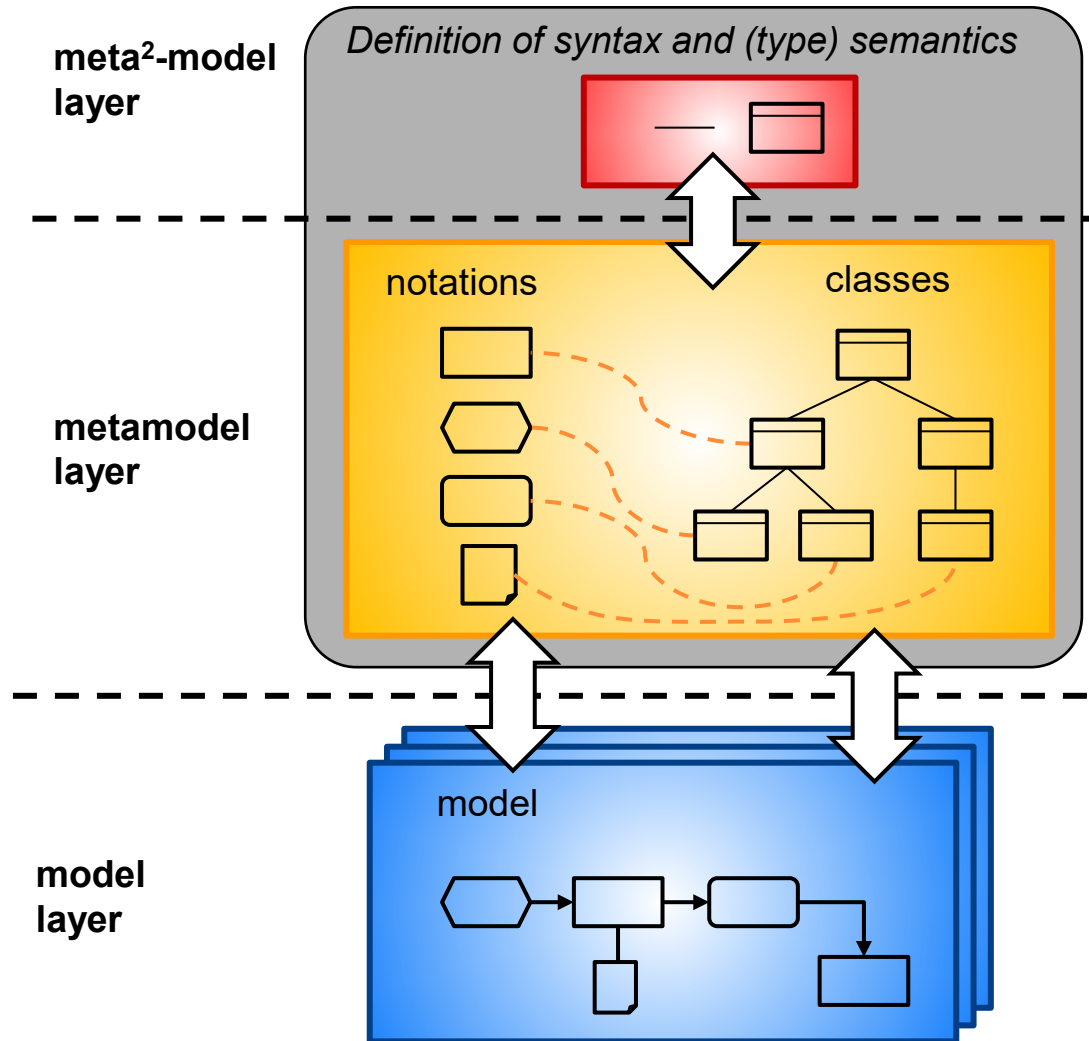


The screenshot shows the OMiLAB website interface. At the top, there is a navigation bar with the contact information: info@omilab.org and **Call Us Now: +49 30 2636 7863**. The OMiLAB logo is prominently displayed on the left. The main navigation menu includes **HOME**, **OMILAB NODES**, **ACTIVITIES**, **ABOUT US**, and **CONTACT**. The central banner features a network diagram with three highlighted nodes: **Design Thinking**, **Conceptual Modelling**, and **Digital Innovation**. A red button on the right of the banner says **READ MORE ON THE OMILAB DIGITAL ECOSYSTEM**. Below the banner, there are three main sections: **OMiLAB Nodes** (with a network icon), **Projects & Events** (with a person icon), and a central illustration of a modern office environment with a person holding a smartphone. At the bottom, the **Scene2Model** logo is visible, featuring the letters 'S2M' in a stylized font.

The ADOxx Environment

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

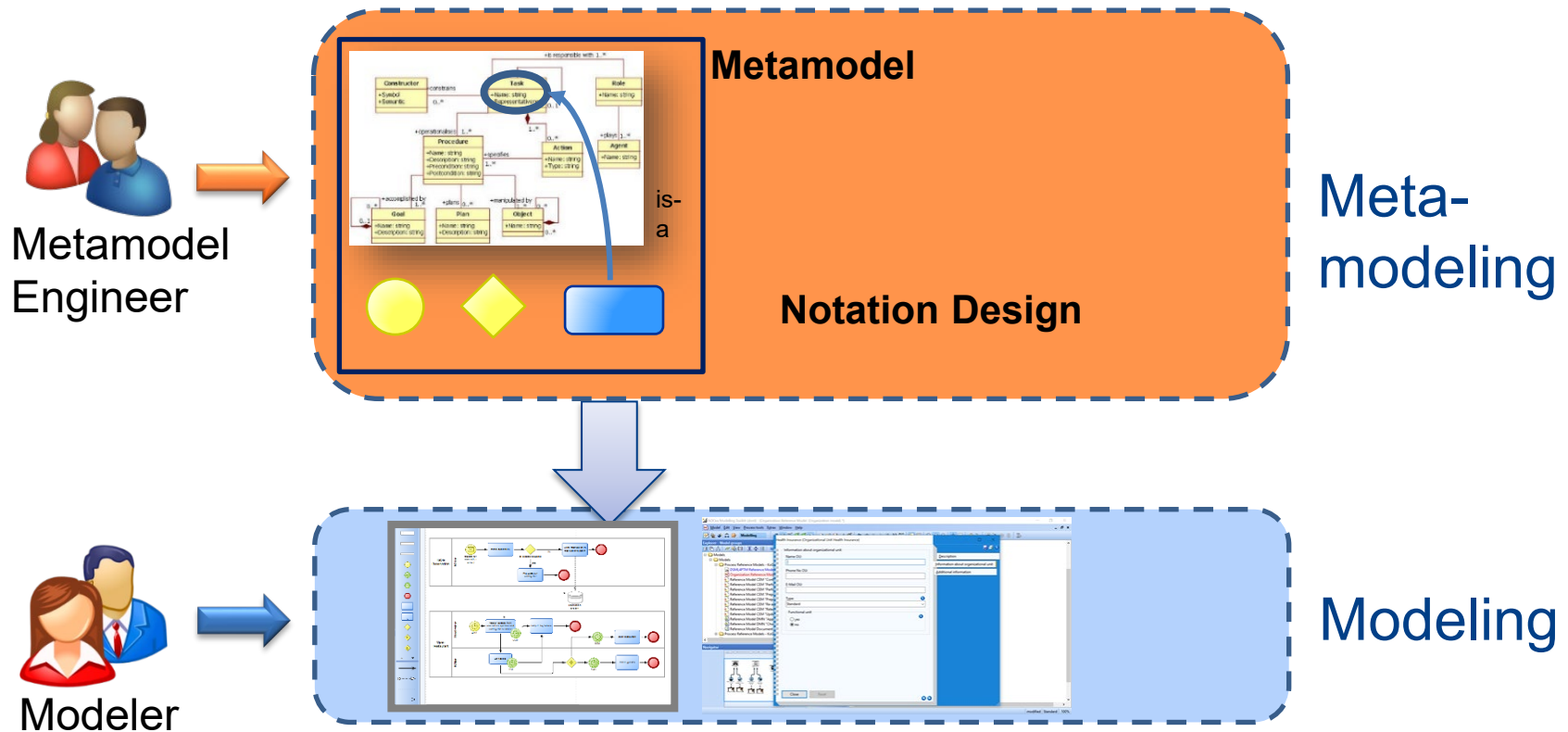
Modeling Environment



ADOxx Development Toolkit

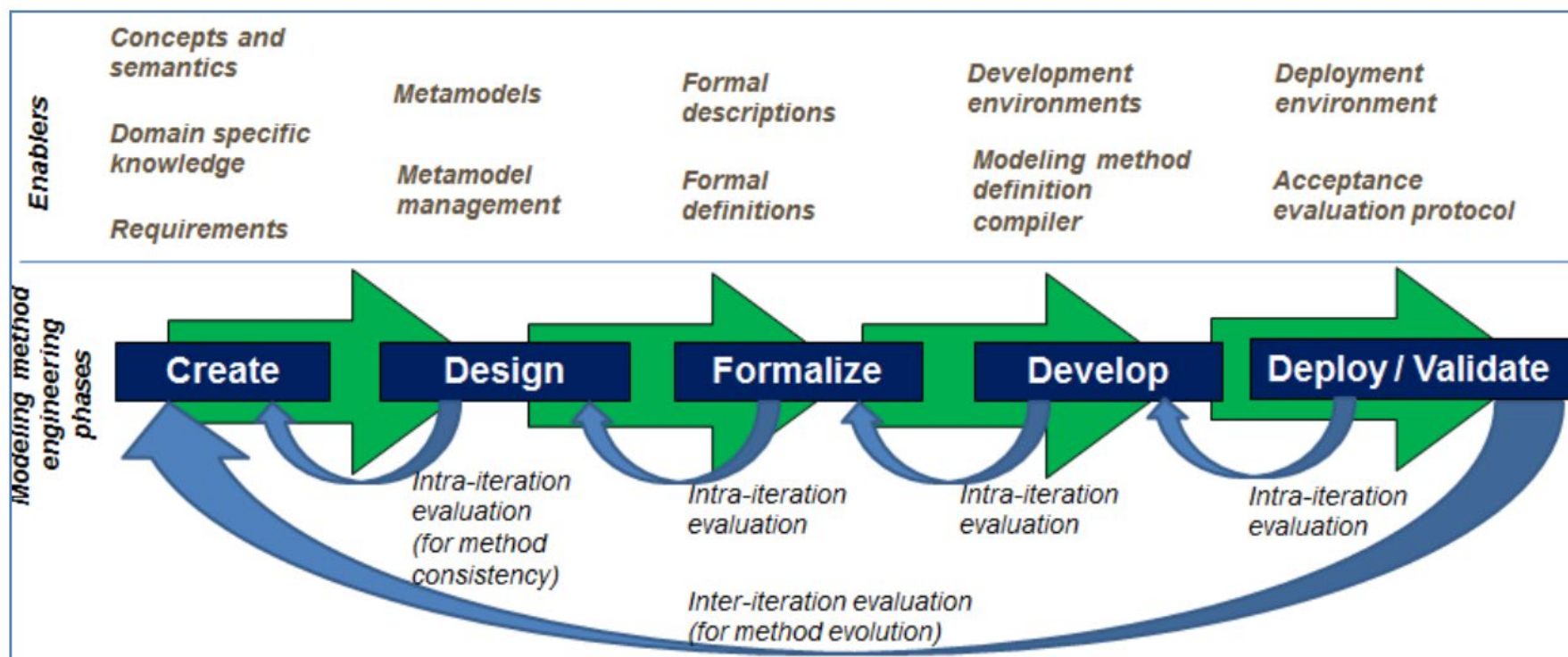
ADOxx Modeling Toolkit

Modeling and Metamodeling



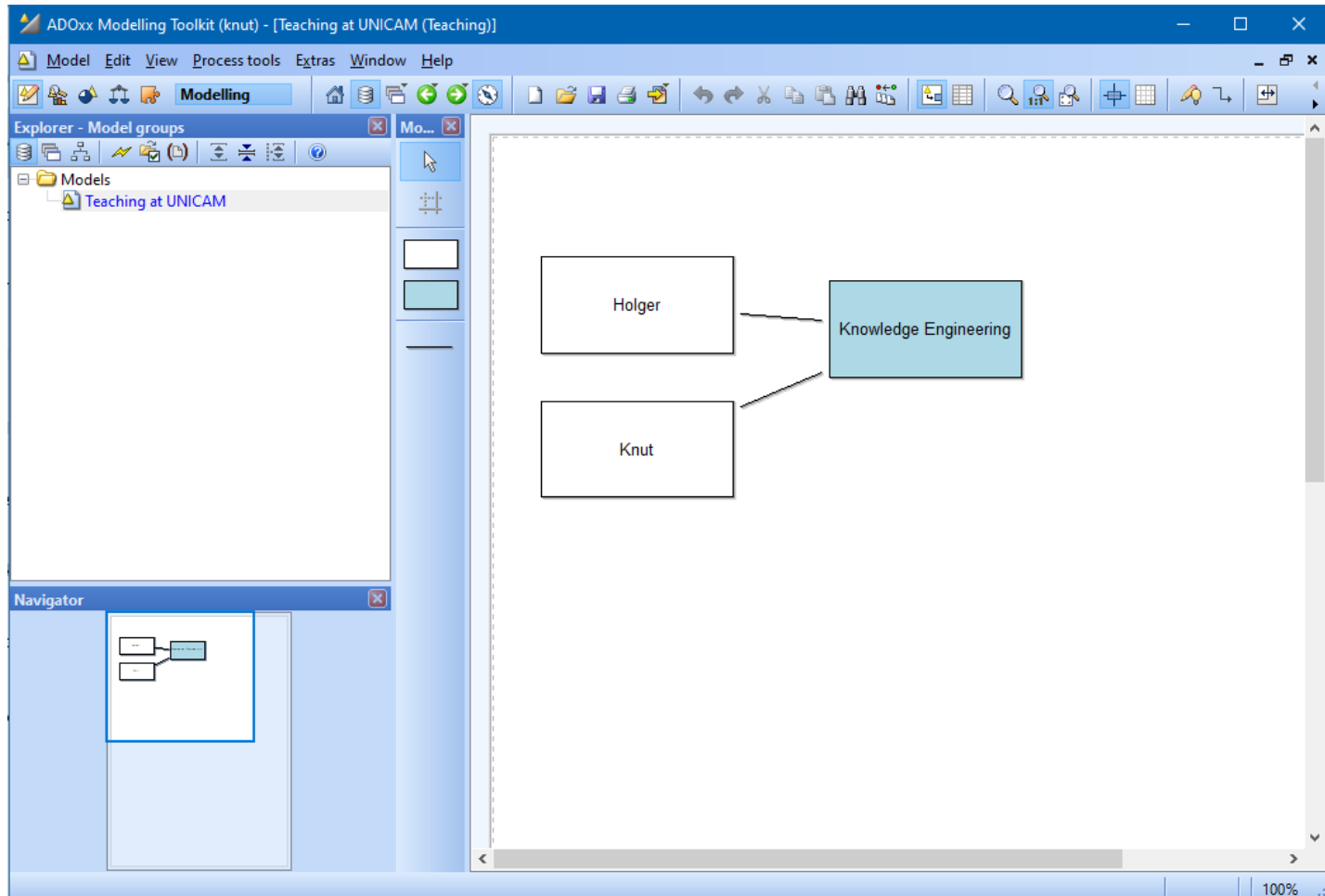
The AMME LifeCycle

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:

ADOxx login

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

ADOxx Experimentation
Platform
Modelling Toolkit

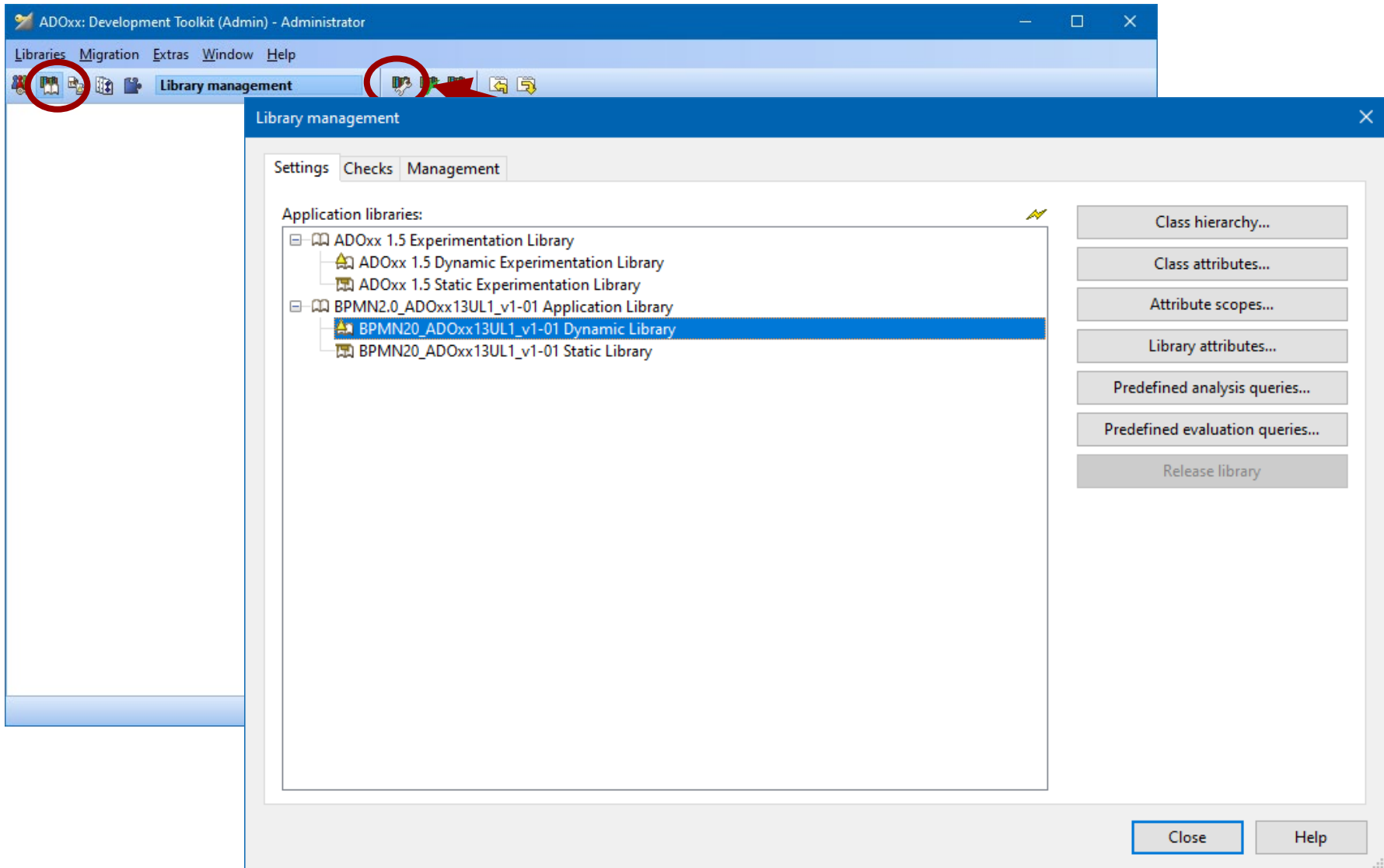
© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

User name:

Password:

Database name:

Metamodelling with ADOxx



The screenshot displays the ADOxx: Development Toolkit (Admin) - Administrator interface. The main window has a menu bar with 'Libraries', 'Migration', 'Extras', 'Window', and 'Help'. Below the menu bar is a toolbar with several icons. Two red circles highlight the 'Library management' icon and the 'Library management' button. The 'Library management' dialog box is open, showing a tree view of application libraries. The tree view is expanded to show the following structure:

- ADOxx 1.5 Experimentation Library
 - ADOxx 1.5 Dynamic Experimentation Library
 - ADOxx 1.5 Static Experimentation Library
- BPMN2.0_ADOxx13UL1_v1-01 Application Library
 - BPMN20_ADOxx13UL1_v1-01 Dynamic Library
 - BPMN20_ADOxx13UL1_v1-01 Static Library

The 'BPMN20_ADOxx13UL1_v1-01 Dynamic Library' is selected. To the right of the tree view are several buttons: 'Class hierarchy...', 'Class attributes...', 'Attribute scopes...', 'Library attributes...', 'Predefined analysis queries...', 'Predefined evaluation queries...', and 'Release library'. At the bottom of the dialog box are 'Close' and 'Help' buttons.

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

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, e.g. BPMN,

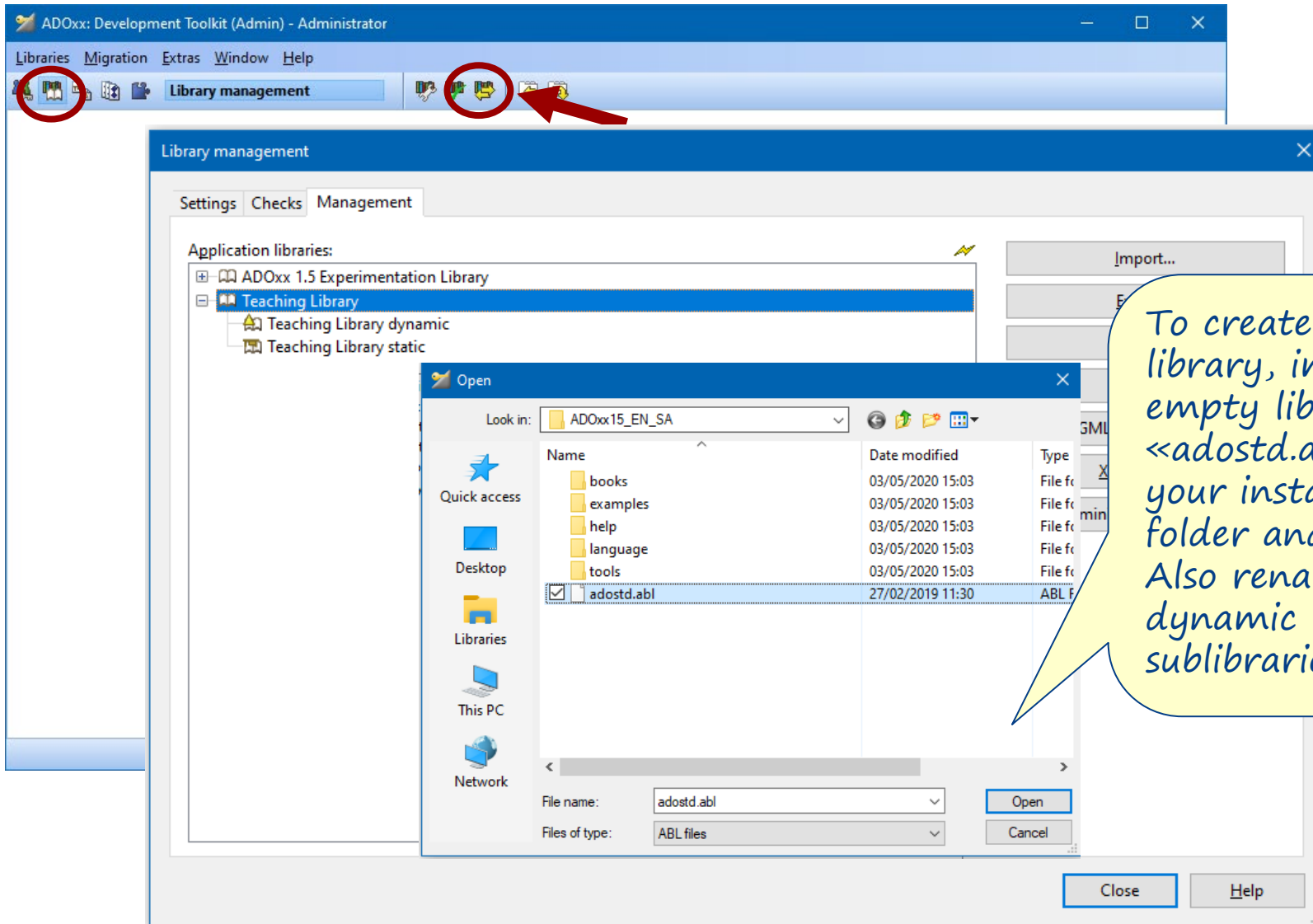
Tweets by @ADOxxORG

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 top menu bar includes 'Libraries', 'Migration', 'Extras', 'Window', and 'Help'. The 'Library management' button is highlighted with a red circle. A red arrow points to the 'Import...' button in the 'Library management' window. The 'Library management' window shows a tree view of application libraries, with 'Teaching Library' selected. An 'Open' dialog box is open, showing the contents of the 'ADOxx15_EN_SA' folder, with 'adostd.abl' selected. The 'File name' field in the 'Open' dialog is set to 'adostd.abl' and the 'Files of type' is set to '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

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

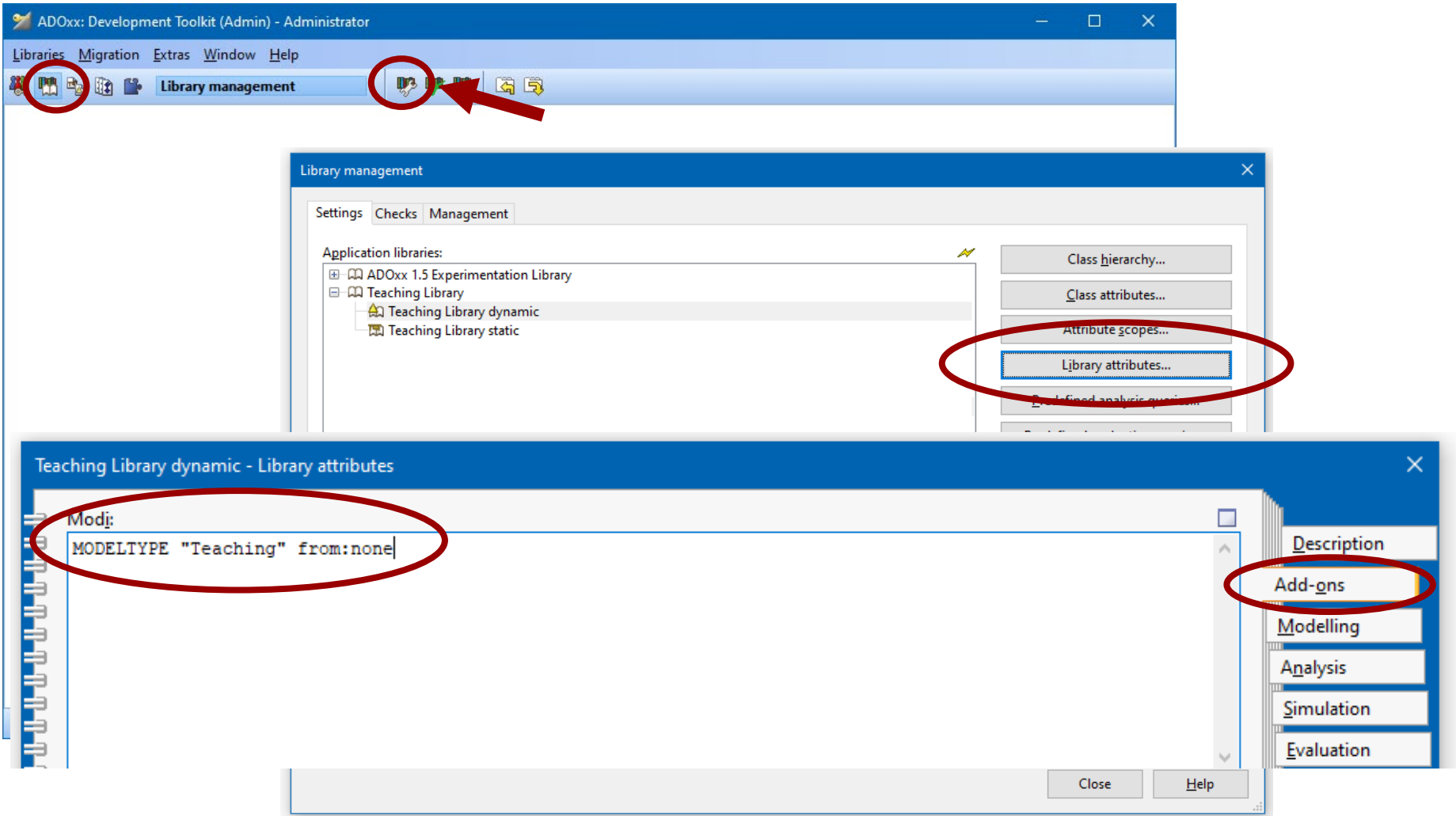
Files of type: ABL files

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

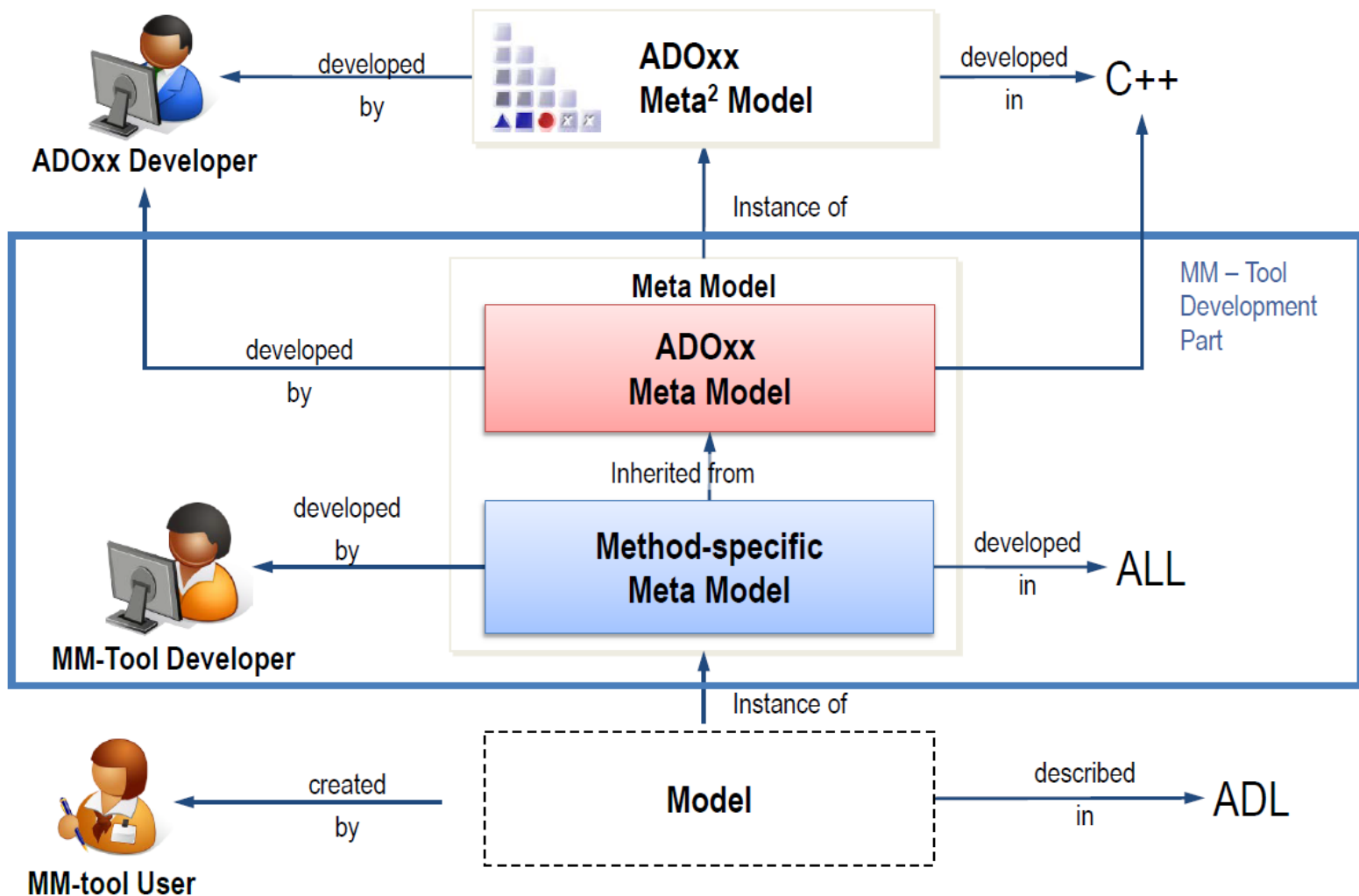


The screenshot illustrates the process of configuring library attributes in the ADOxx Development Toolkit. It consists of three overlapping windows:

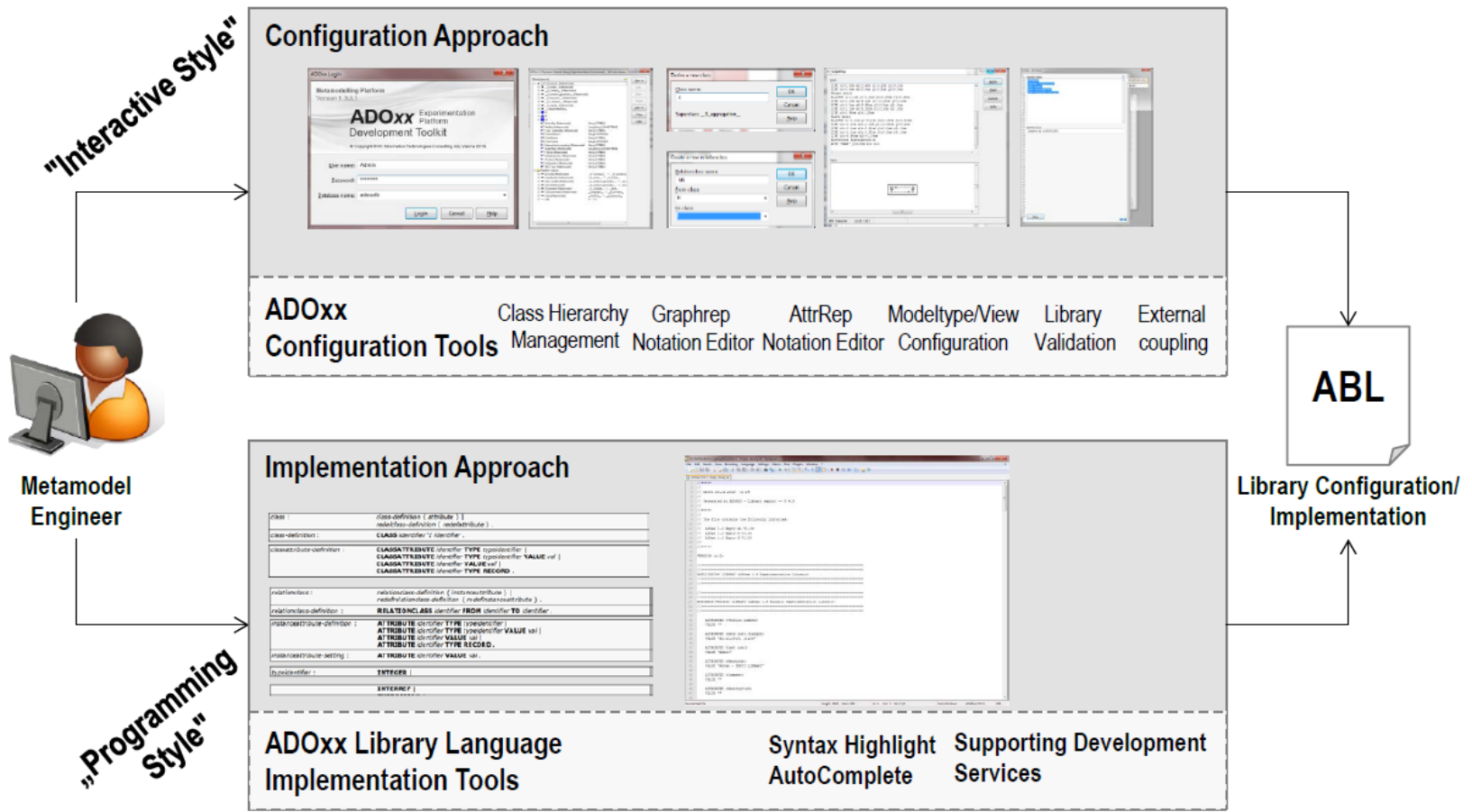
- ADOxx: Development Toolkit (Admin) - Administrator:** The main application window with a menu bar (Libraries, Migration, Extras, Window, Help) and a toolbar. A red circle highlights the 'Library management' icon in the toolbar, and a red arrow points to it.
- Library management:** A sub-window with 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 of the window.
- 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.

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

Meta Modelling Platforms Hierarchy in ADOxx

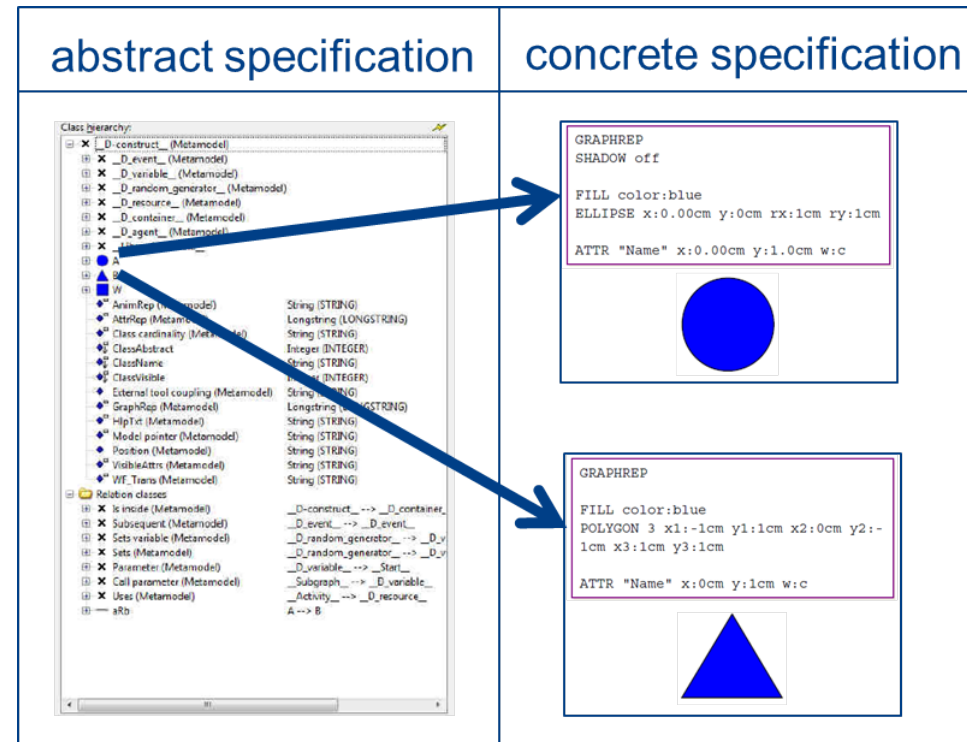


Development Approaches in ADOxx – Configuration and Implementation



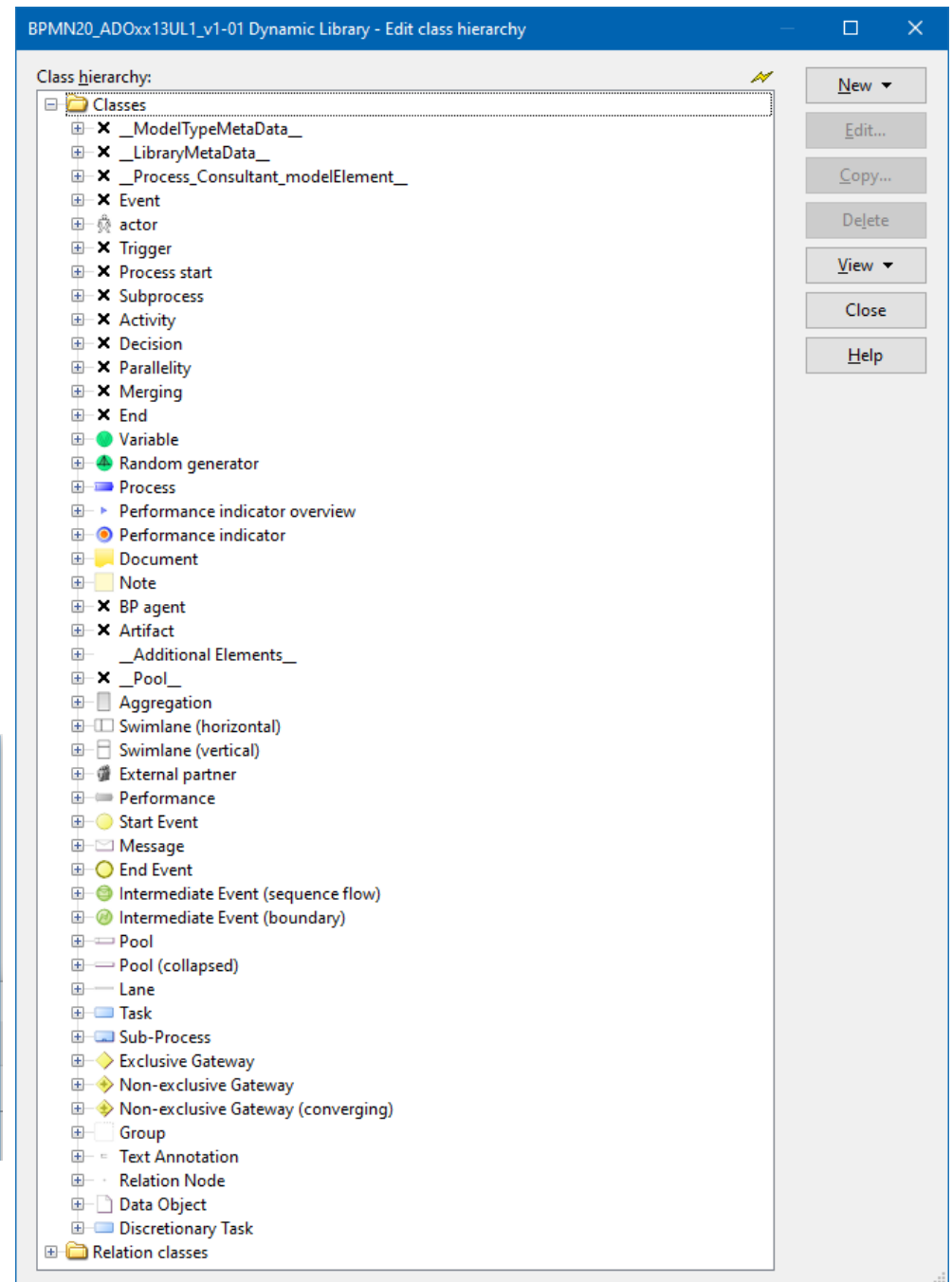
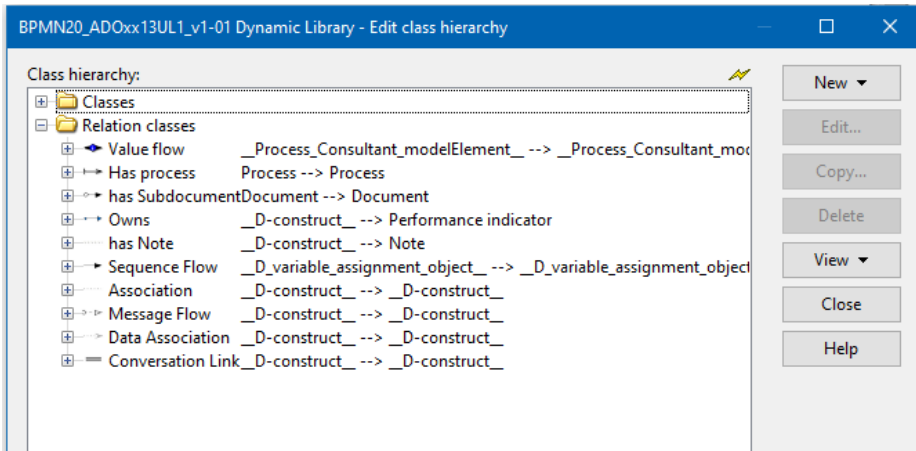
Metamodel and Modeling Language in ADOxx

- 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



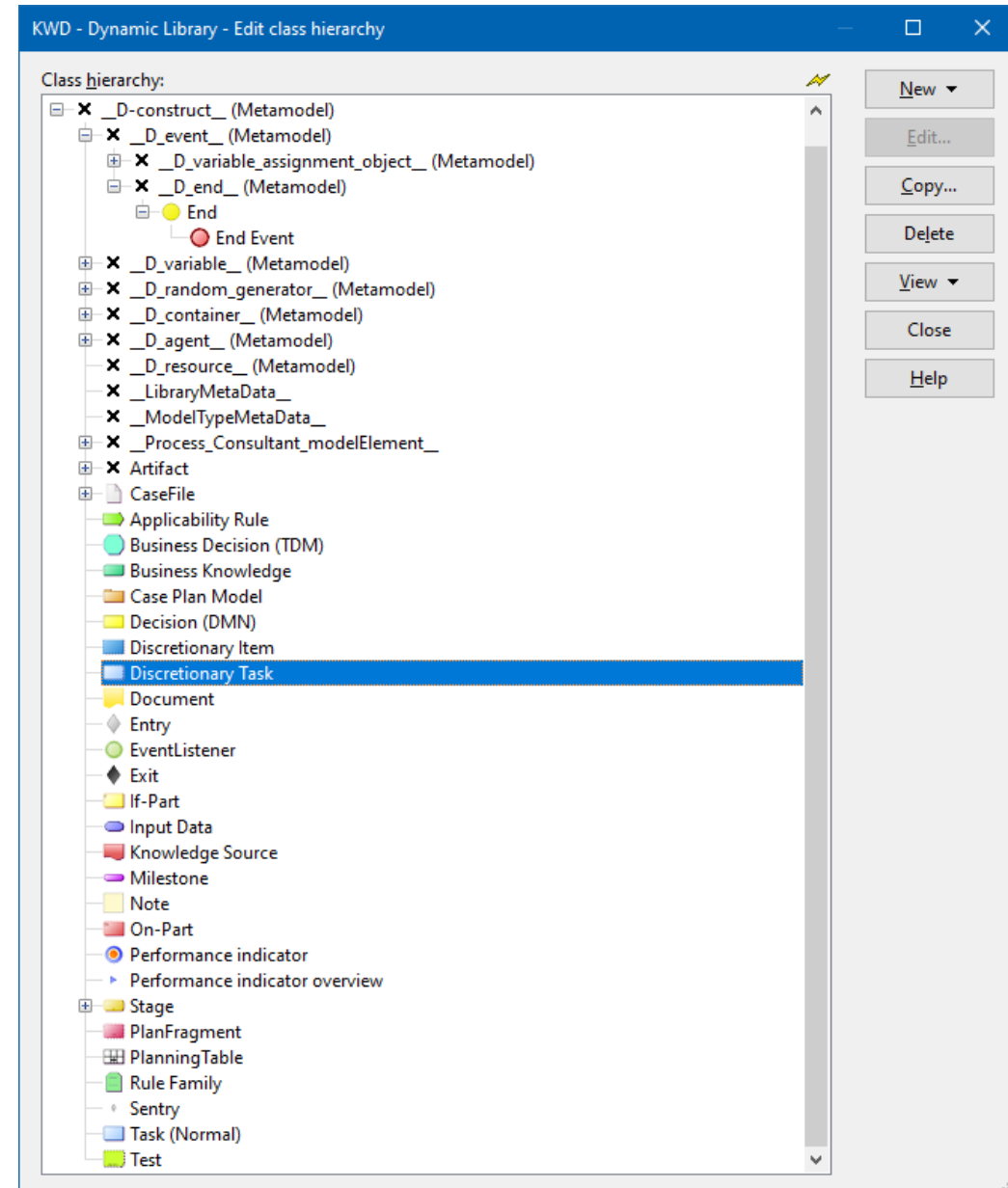
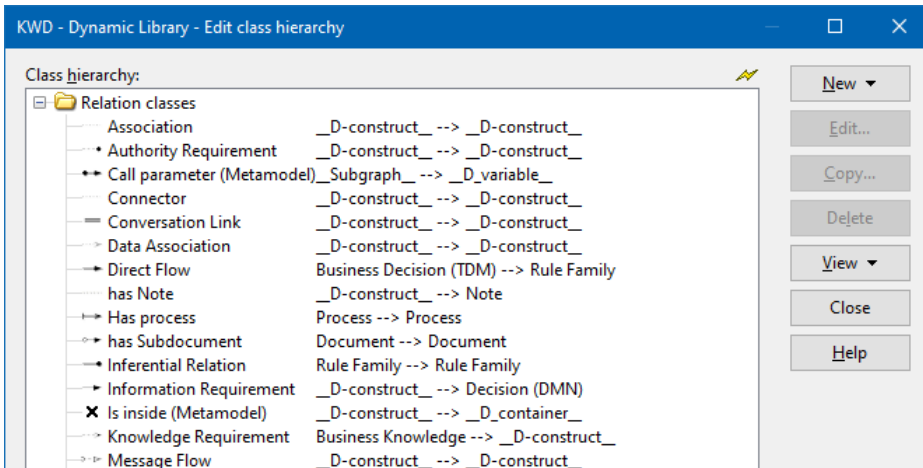
Class Hierarchies

- ADOxx distinguishes
 - ◆ Classes
 - ◆ Relation classes

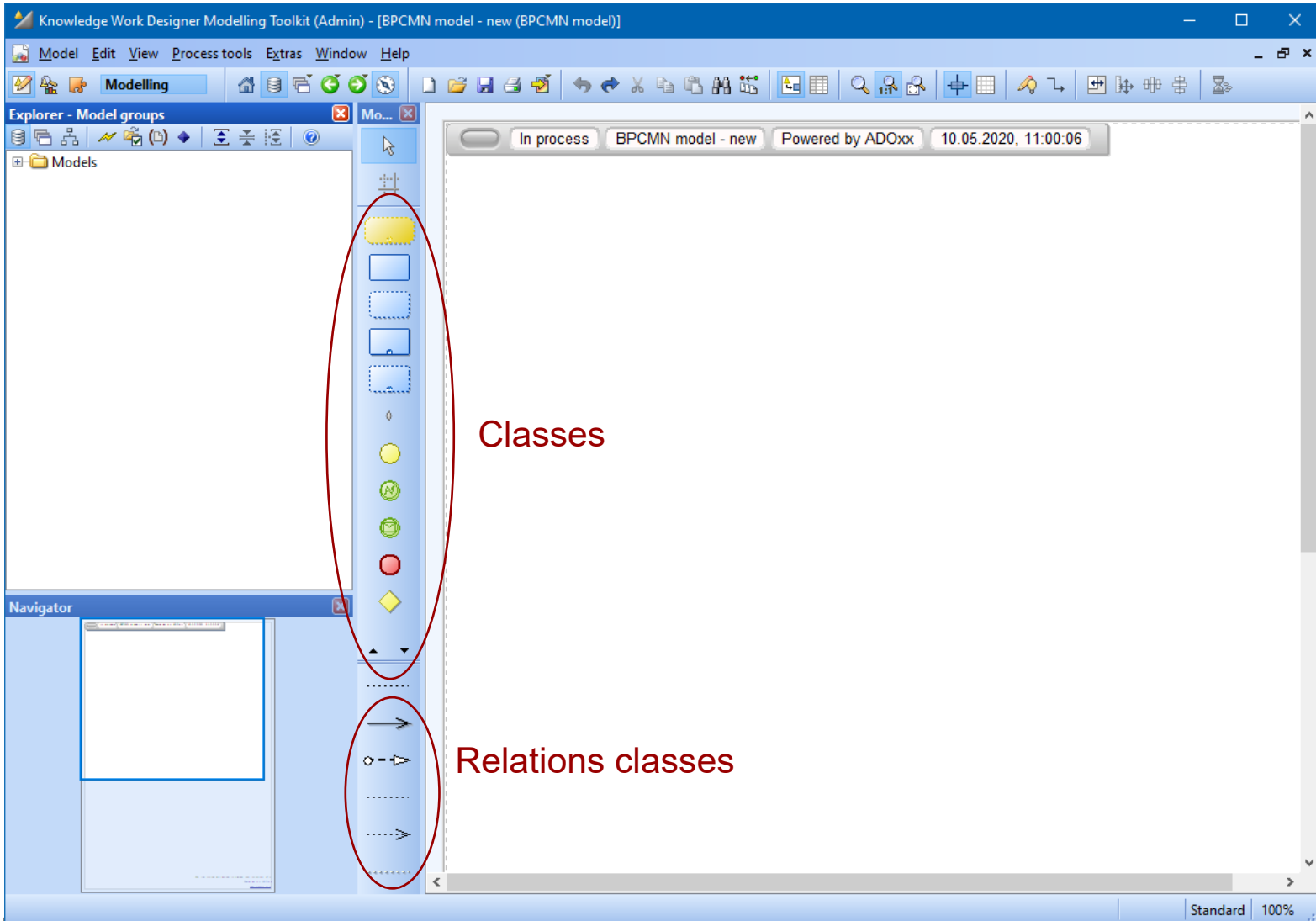


Class Hierarchies

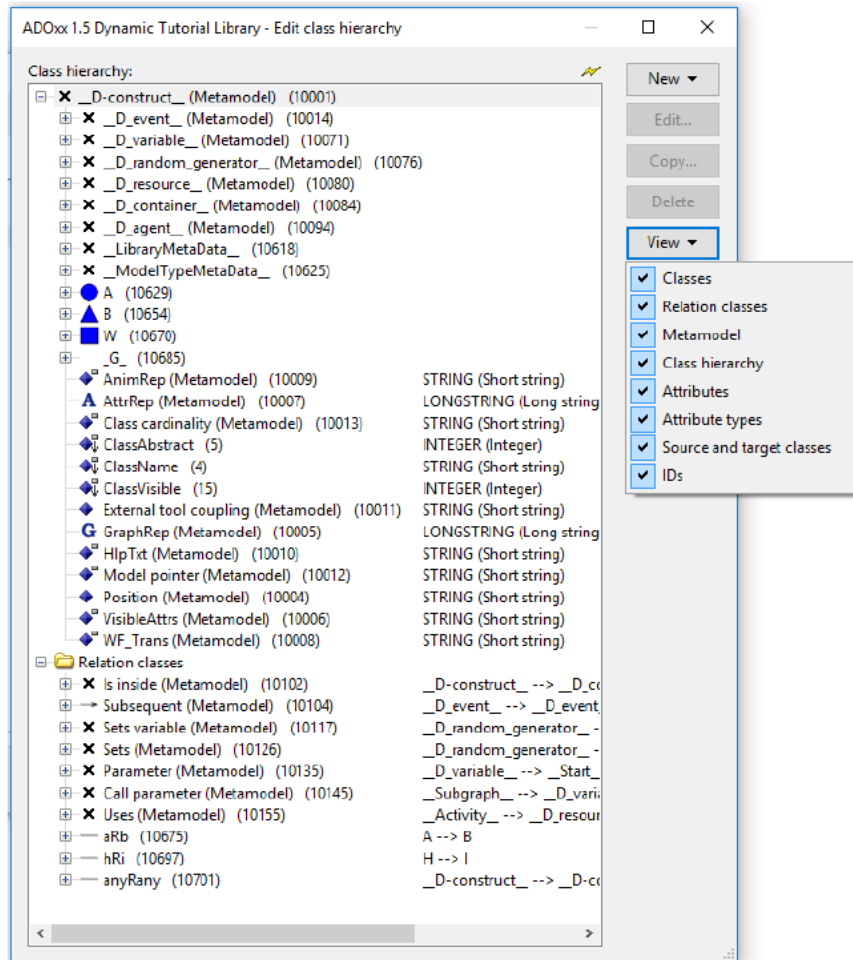
- ADOxx distinguishes
 - ◆ Classes
 - ◆ Relation classes



Appearance of Classes in the Modelling Toolkit



Views of the Class Hierarchy



ADOxx 1.5 Dynamic Tutorial Library - Edit class hierarchy

Class hierarchy:

- [-] X __D_construct__ (Metamodel) (10001)
- [-] X __D_event__ (Metamodel) (10014)
- [-] X __D_variable__ (Metamodel) (10071)
- [-] X __D_random_generator__ (Metamodel) (10076)
- [-] X __D_resource__ (Metamodel) (10080)
- [-] X __D_container__ (Metamodel) (10084)
- [-] X __D_agent__ (Metamodel) (10094)
- [-] X __LibraryMetaData__ (10618)
- [-] X __ModelTypeMetaData__ (10625)
- [-] A (10629)
- [-] B (10654)
- [-] W (10670)
- [-] G_ (10685)
- [-] AnimRep (Metamodel) (10009) STRING (Short string)
- [-] AttrRep (Metamodel) (10007) LONGSTRING (Long string)
- [-] Class cardinality (Metamodel) (10013) STRING (Short string)
- [-] ClassAbstract (5) INTEGER (Integer)
- [-] ClassName (4) STRING (Short string)
- [-] ClassVisible (15) INTEGER (Integer)
- [-] External tool coupling (Metamodel) (10011) STRING (Short string)
- [-] GraphRep (Metamodel) (10005) LONGSTRING (Long string)
- [-] HlpTxt (Metamodel) (10010) STRING (Short string)
- [-] Model pointer (Metamodel) (10012) STRING (Short string)
- [-] Position (Metamodel) (10004) STRING (Short string)
- [-] VisibleAttrs (Metamodel) (10006) STRING (Short string)
- [-] WF_Trans (Metamodel) (10008) STRING (Short string)
- [+] Relation classes
 - [-] X Is inside (Metamodel) (10102) __D_construct__ --> __D-co
 - [-] Subsequent (Metamodel) (10104) __D_event__ --> __D_event
 - [-] X Sets variable (Metamodel) (10117) __D_random_generator__ -
 - [-] X Sets (Metamodel) (10126) __D_random_generator__ -
 - [-] X Parameter (Metamodel) (10135) __D_variable__ --> __Start_
 - [-] X Call parameter (Metamodel) (10145) __Subgraph__ --> __D_vari
 - [-] X Uses (Metamodel) (10155) __Activity__ --> __D_resour
 - [-] aRb (10675) A --> B
 - [-] hRi (10697) H --> I
 - [-] anyRany (10701) __D_construct__ --> __D-co

View

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

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.

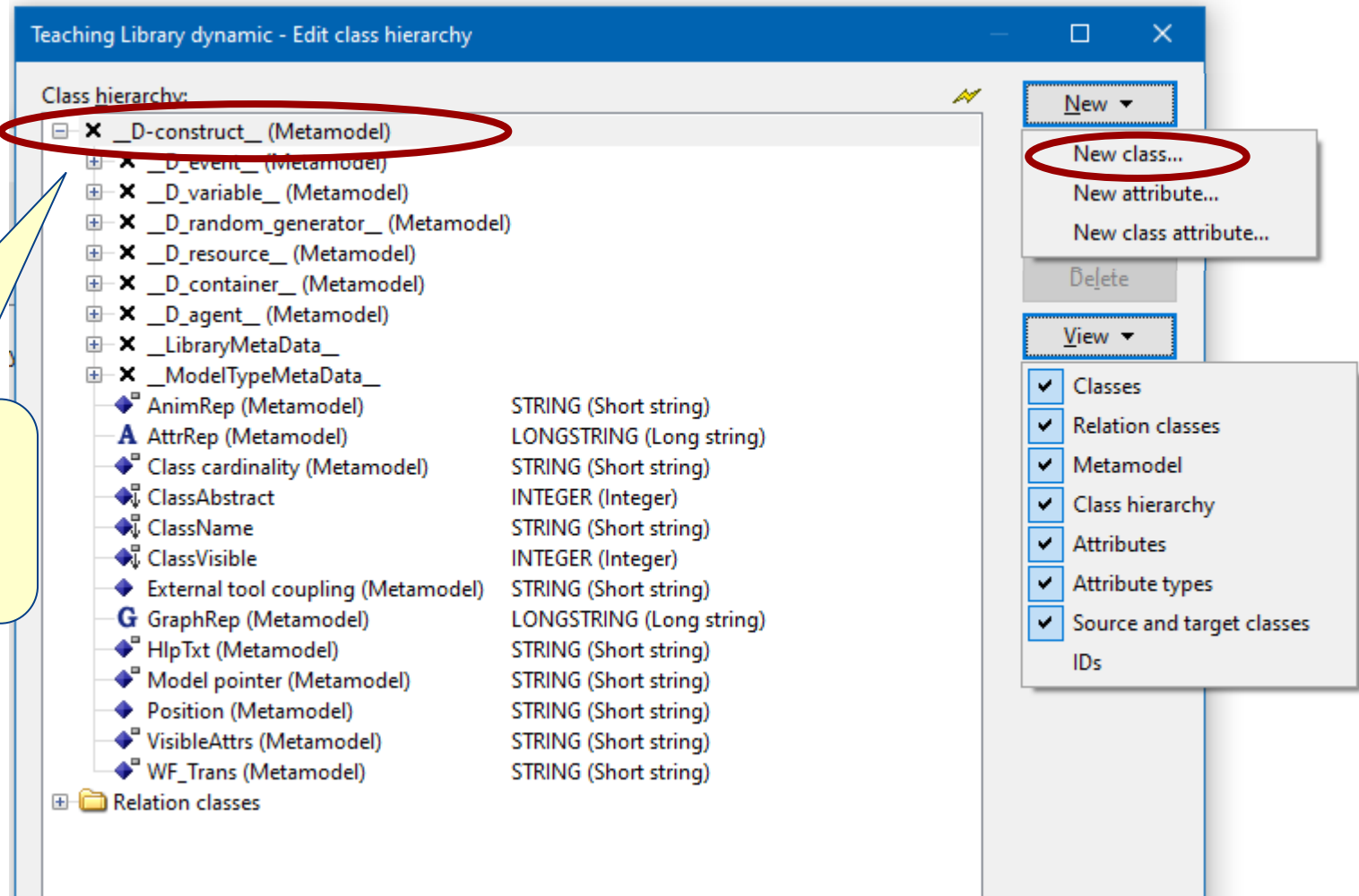
IDs

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

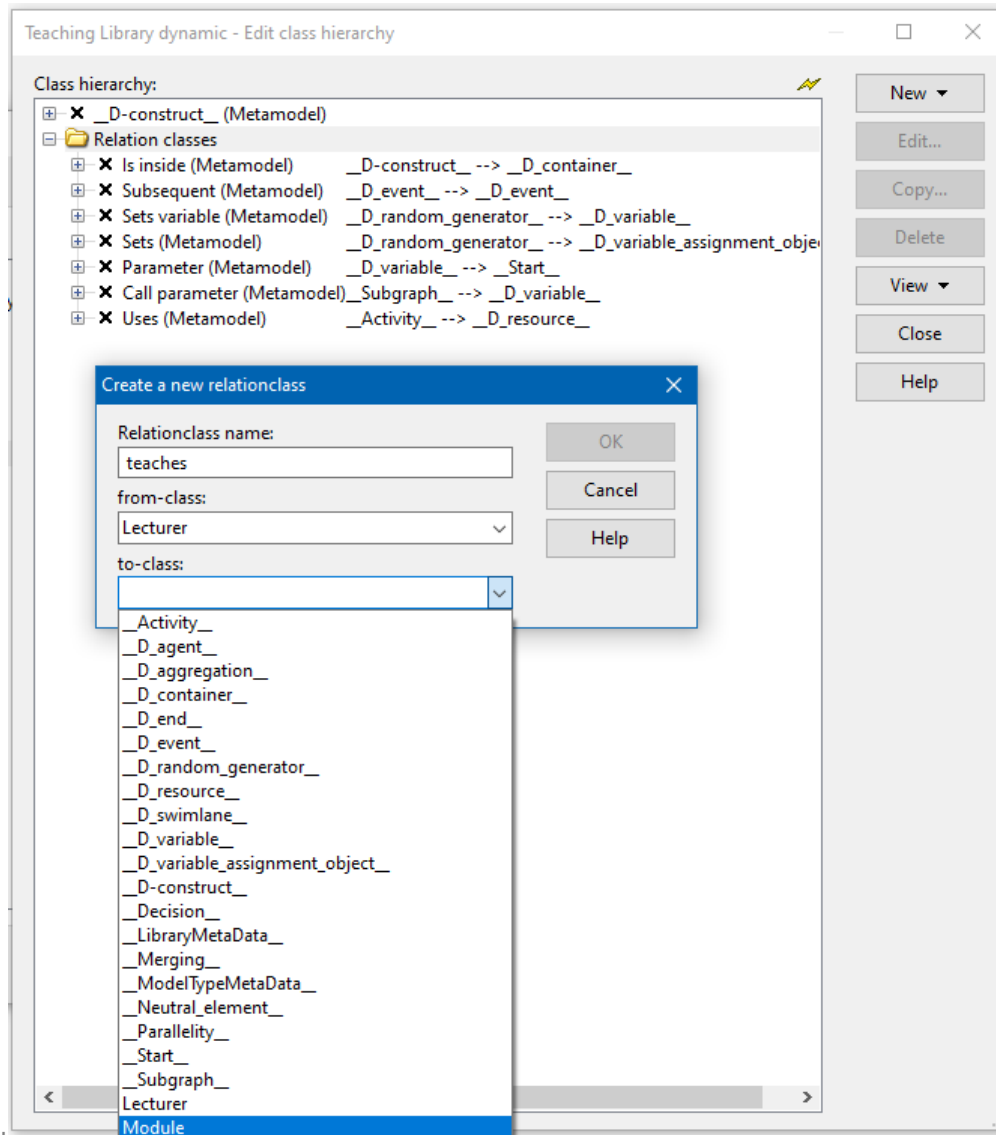
Class hierarchy:

- [-] 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»

Attributes

■ Kinds of Attributes

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

BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

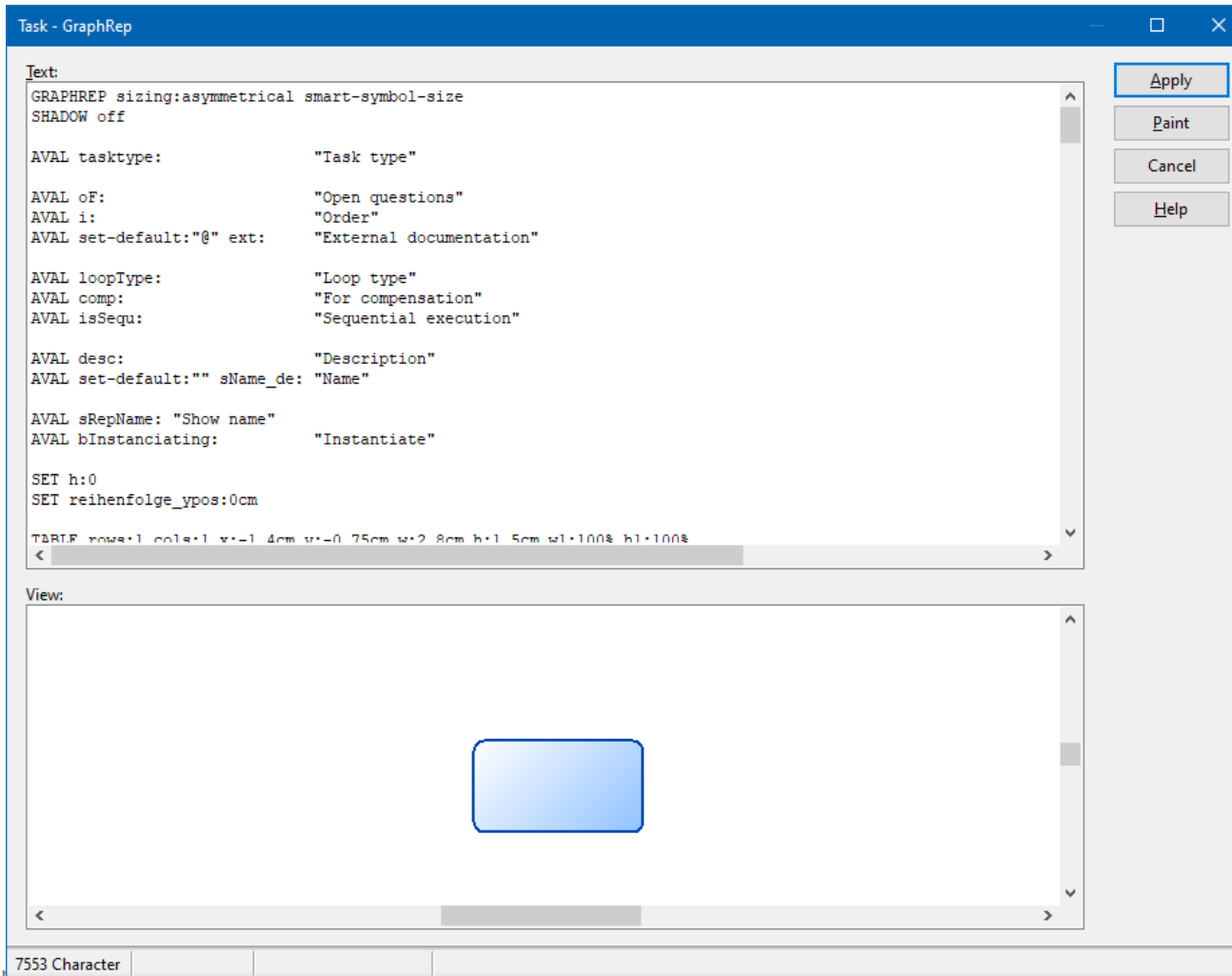
Class hierarchy:

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

Special Attribute GraphRep

GraphRep: A script language for the graphical representation



The screenshot shows a window titled "Task - GraphRep" with a blue header bar. The main area is divided into two sections: "Text" and "View".

Text:

```
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%
```

View:

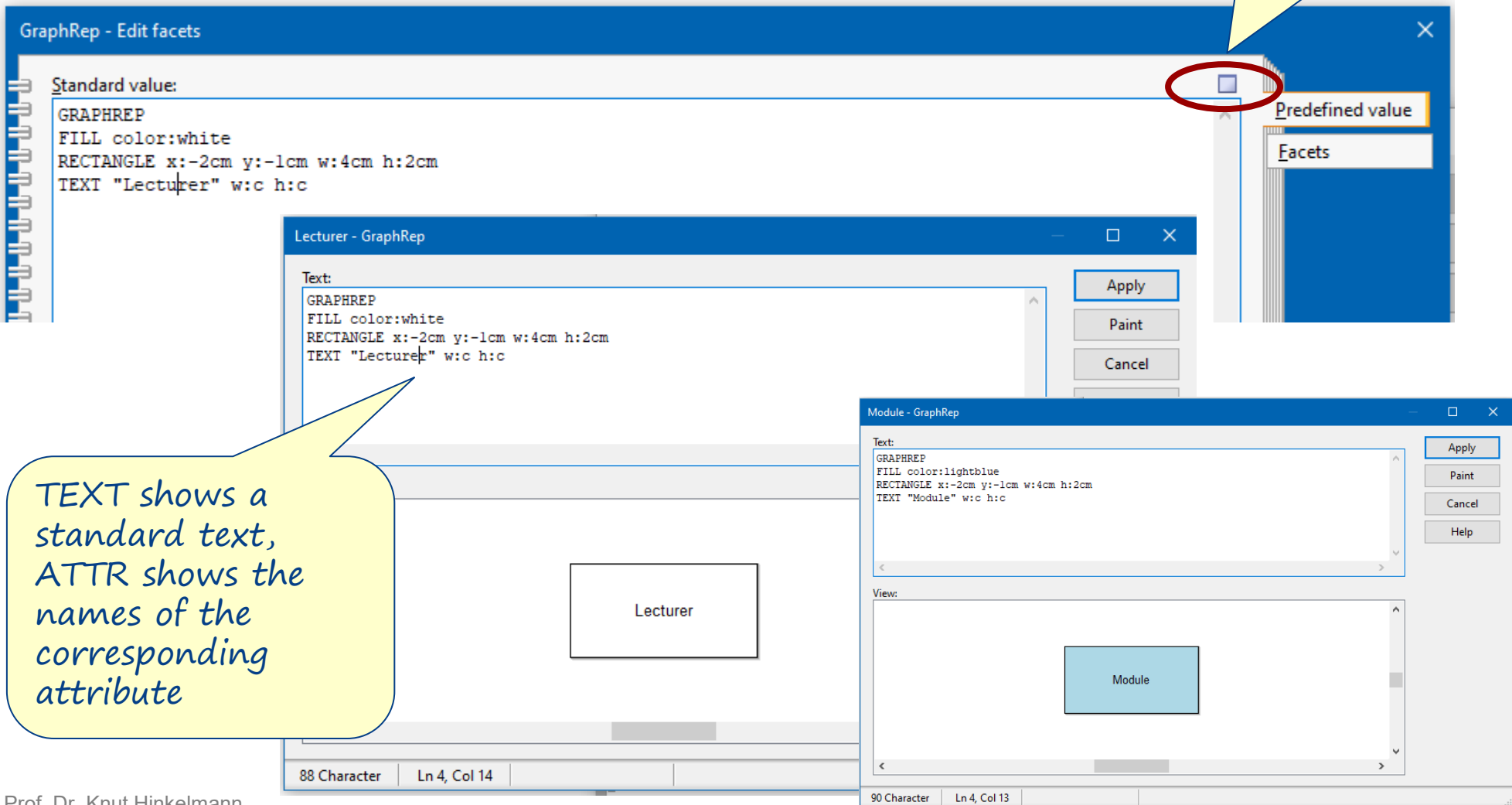
The view area shows a single blue rounded rectangle centered on a white background.

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 shows "7553 Character".

Defining a GraphRep

With the help button you can define and test the graphics



GraphRep - Edit facets

Standard value:

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

Lecturer - GraphRep

Text:

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

Module - GraphRep

Text:

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

View:

88 Character Ln 4, Col 14

90 Character Ln 4, Col 13

TEXT shows a standard text, ATTR shows the names of the corresponding attribute

ADOxx GraphRep Repository

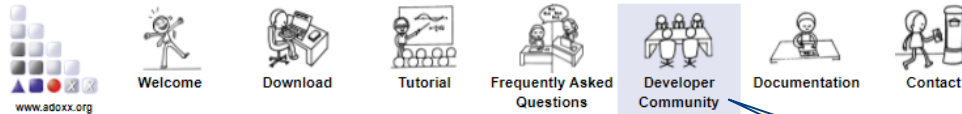
GRAPHREP COLLECTION



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](#)

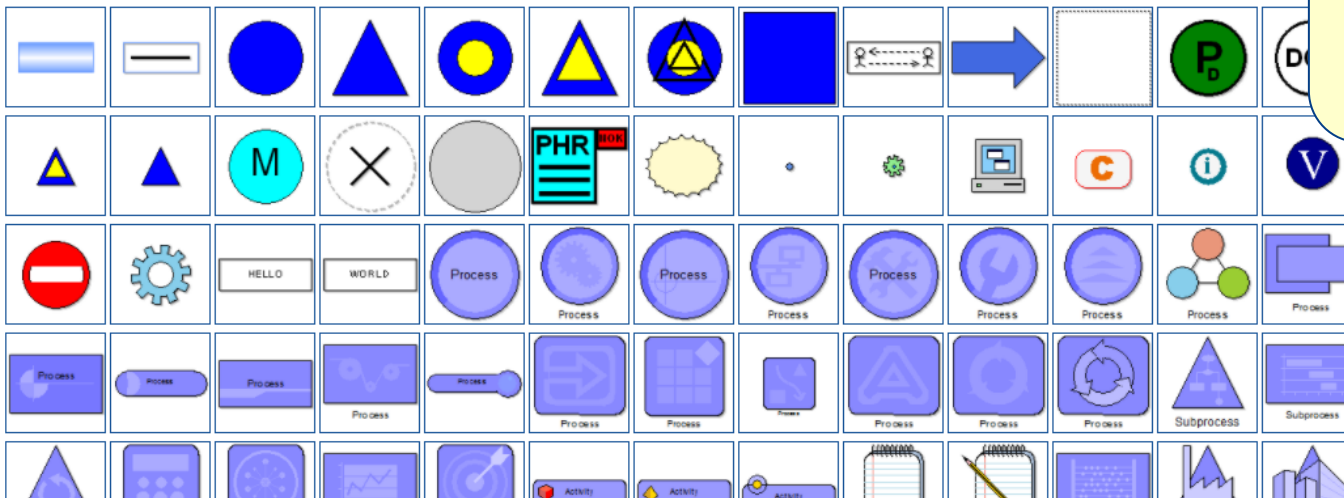
ADOxx GraphRep Repository

(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

GraphRep Elements

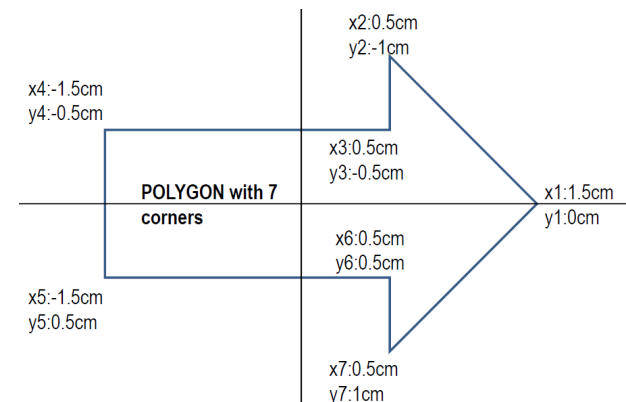
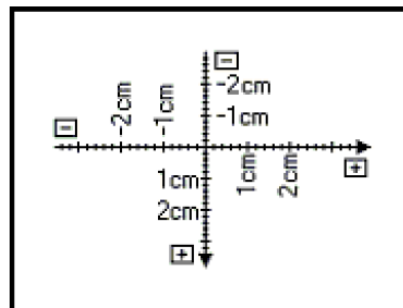
GraphRep Elements

- 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

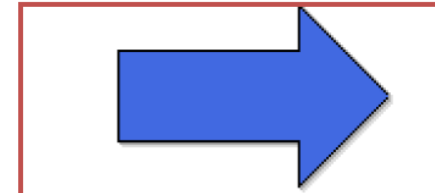
ATTR "Name" x:0.00cm y:1.0cm w:c
    
```



```

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
    
```

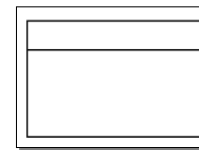


In case attribute name is available, it is shown here

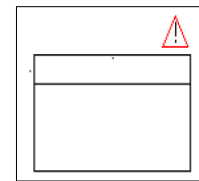
Conditional Representation

```

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

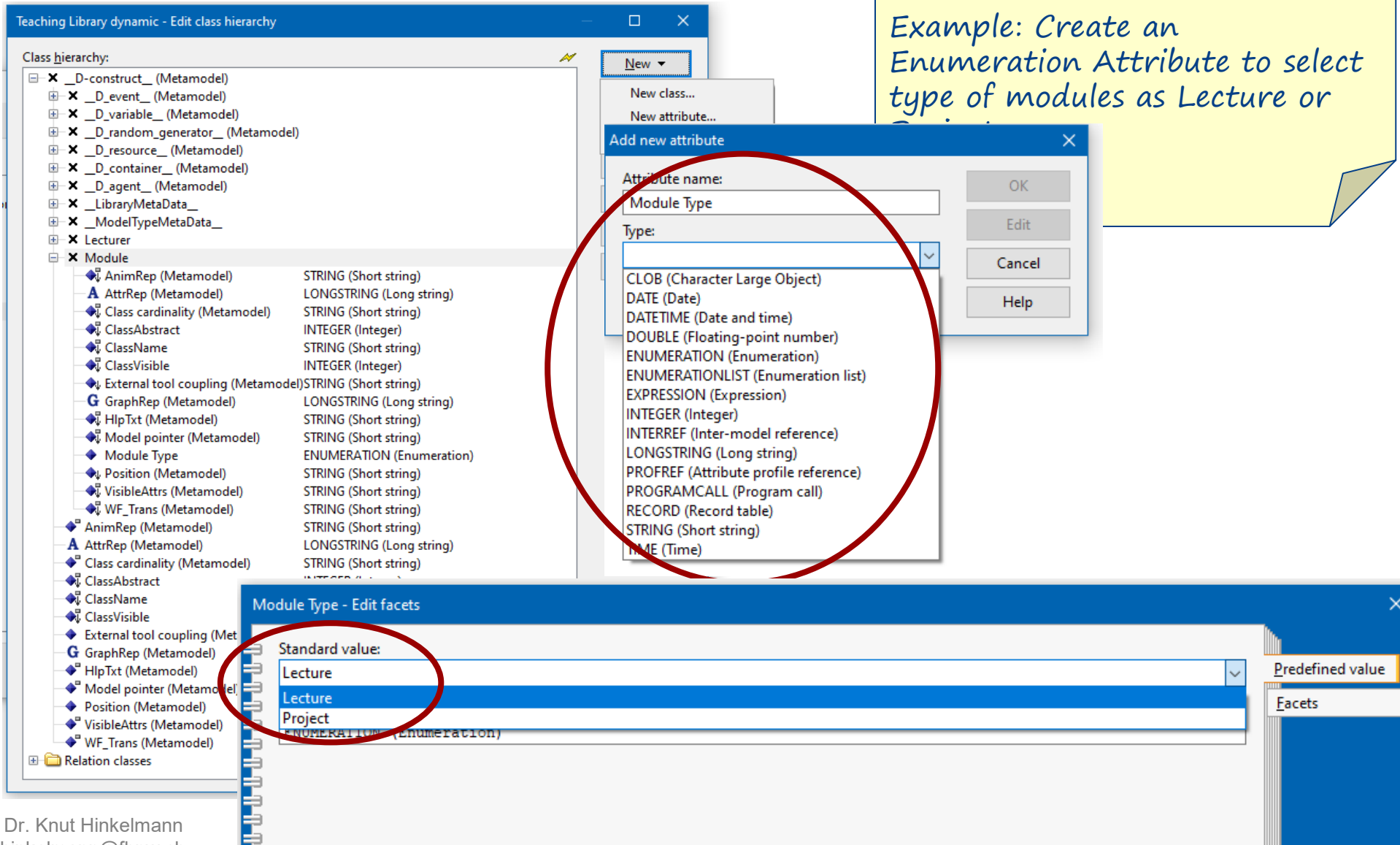


Condition not fulfilled

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

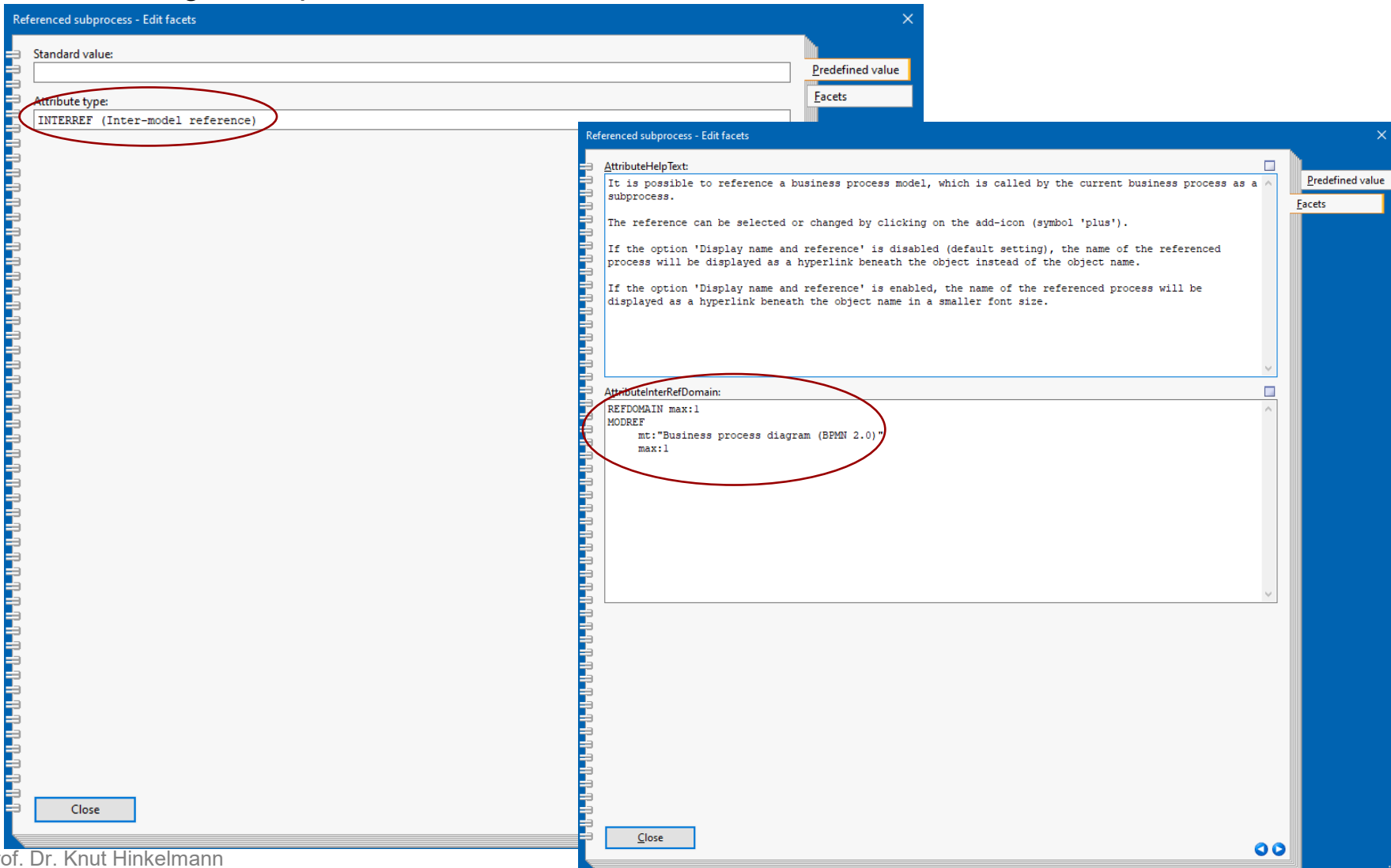
ENUMERATION (Enumeration)

Predefined value

Facets

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]

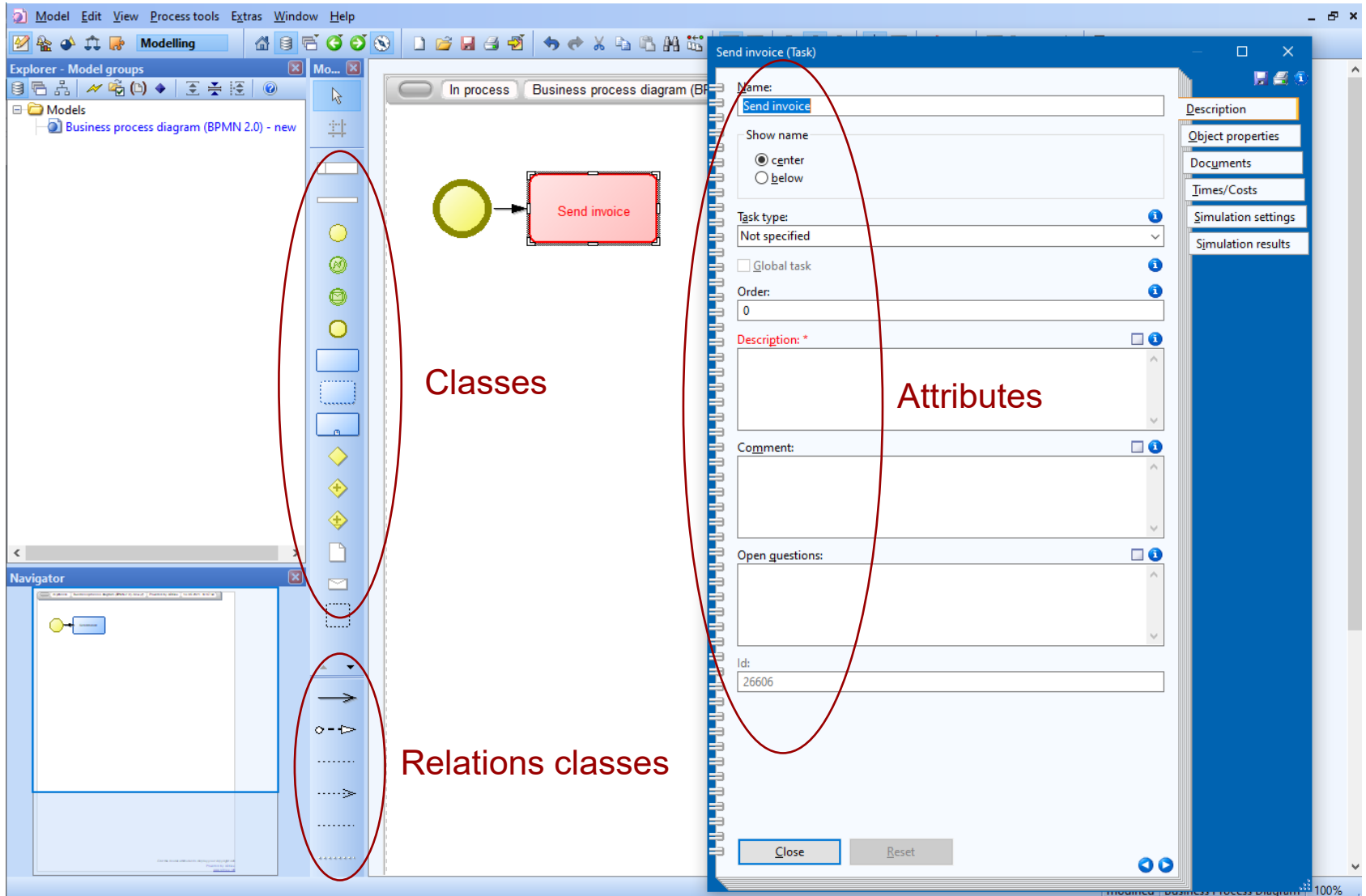
The bottom dialog box has the following fields:

- 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
```

Appearance of Classes in the Modelling Toolkit



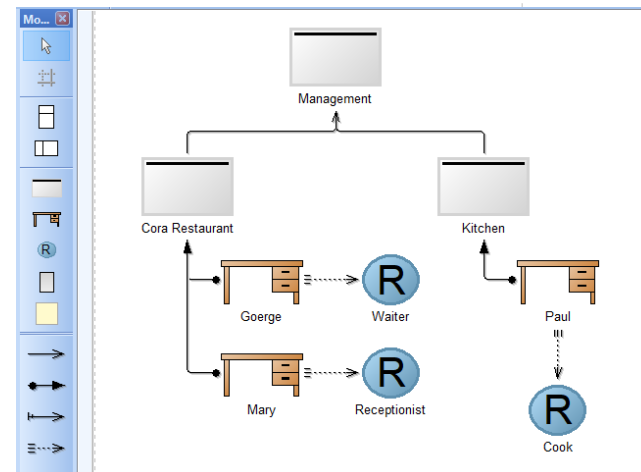
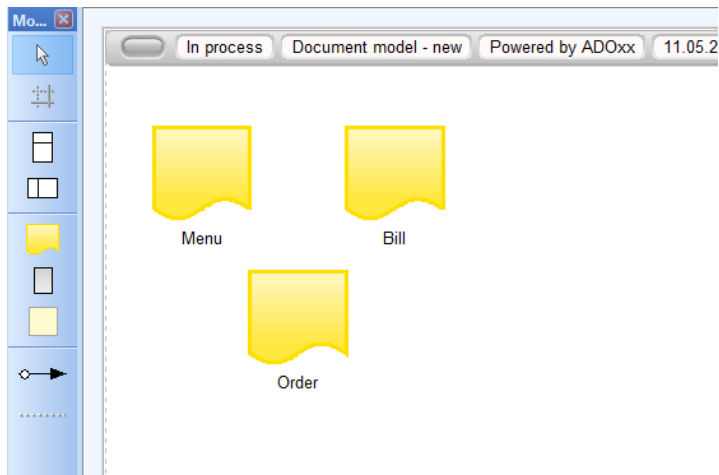
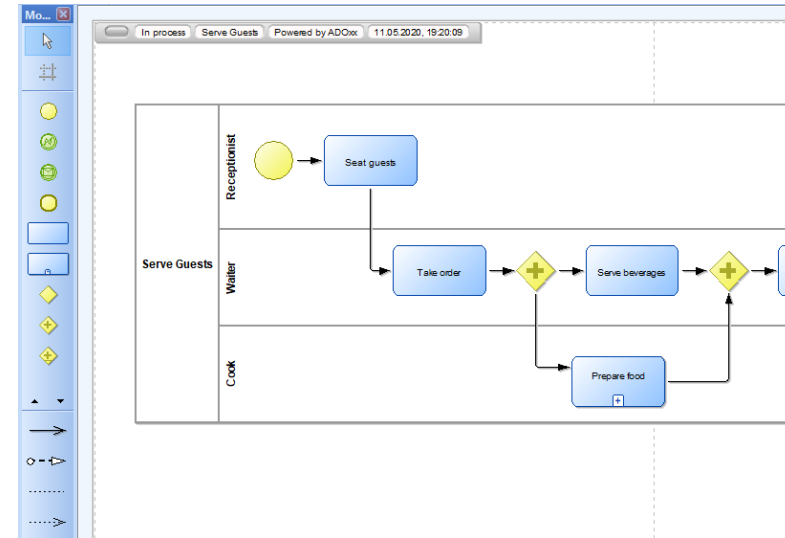
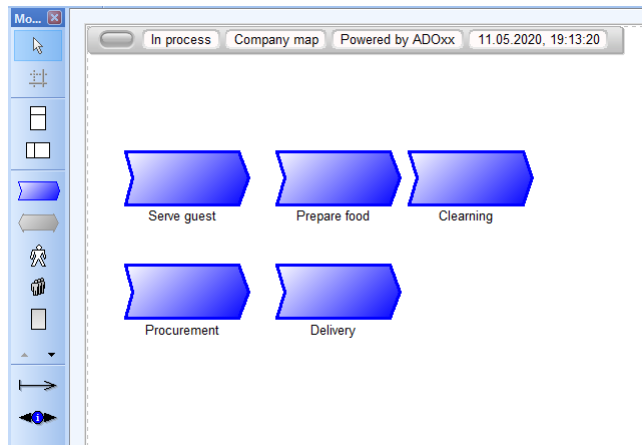
The screenshot displays the Modelling Toolkit interface. The main workspace shows a BPMN diagram with a yellow circle connected to a red rectangular task labeled "Send invoice". The left sidebar contains a palette of modeling elements, with a red oval highlighting a group of elements labeled "Classes" and another red oval highlighting a group of elements labeled "Relations classes". The right sidebar shows the "Send invoice (Task)" properties window, with a red oval highlighting the "Name" field (containing "Send invoice") and the "Attributes" section, which includes fields for "Show name", "Task type", "Order", "Description", "Comment", and "Open questions".

AttrRep

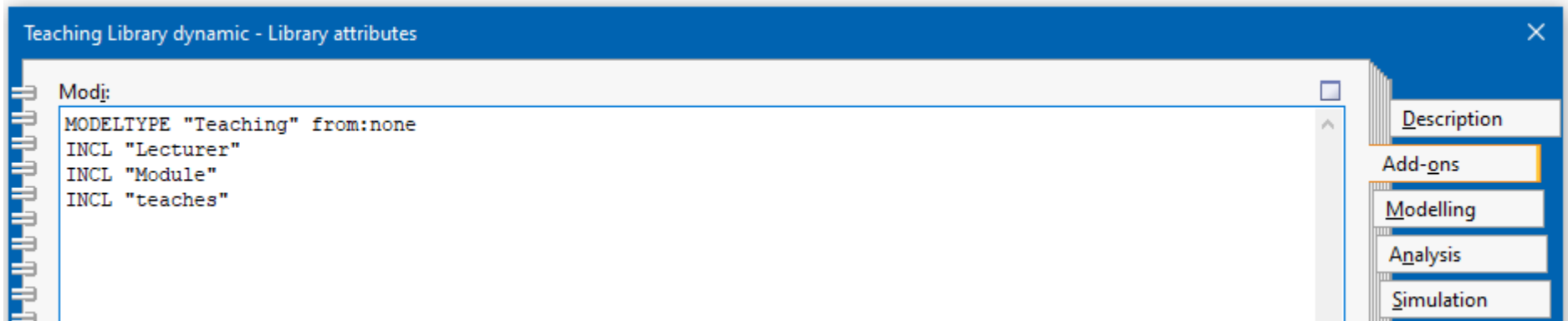
The class attribute „AttrRep“ controls the structure of the ADOxx-Notebook.



Model Types: Representation Views on the Knowledge



Example

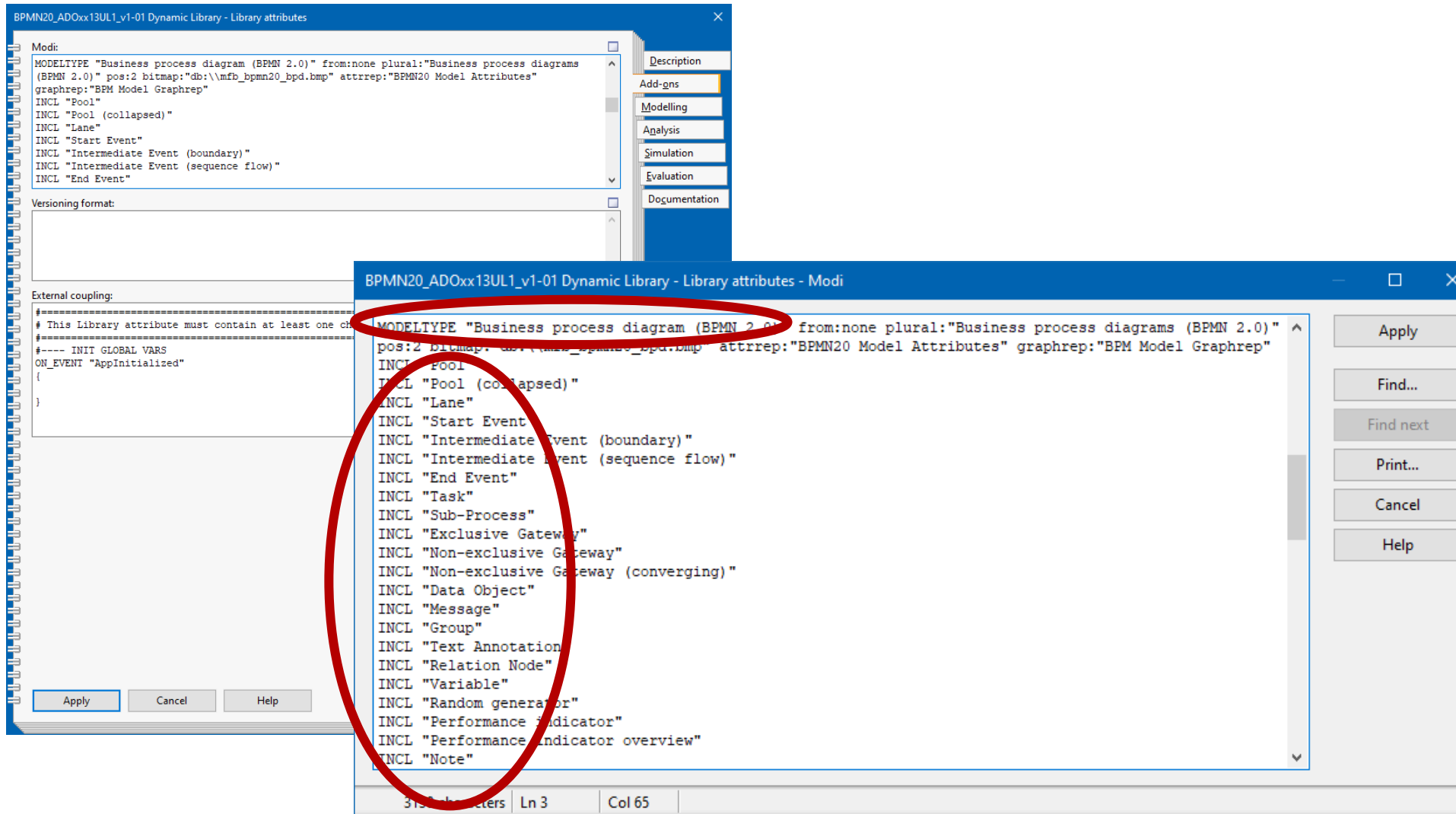


Teaching Library dynamic - Library attributes

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

Description
Add-ons
Modelling
Analysis
Simulation

Classes are assigned to Model Types



The image shows a screenshot of a software interface for defining a dynamic library. The main window is titled "BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes". It has a left pane with a tree view showing "Modi" and "Versioning format". The "Modi" pane is expanded, showing a list of BPMN elements: "Business process diagram (BPMN 2.0)", "Pool", "Pool (collapsed)", "Lane", "Start Event", "Intermediate Event (boundary)", "Intermediate Event (sequence flow)", and "End Event". The right pane shows a list of categories: "Description", "Add-gns", "Modelling", "Analysis", "Simulation", "Evaluation", and "Documentation".

A secondary dialog box, titled "BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes - Modi", is overlaid on the main window. It shows a list of BPMN elements with their corresponding model types. The first element, "Business process diagram (BPMN 2.0)", is circled in red. The list includes:


- MODELTYP "Business process diagram (BPMN 2.0)" from:none plural:"Business process diagrams (BPMN 2.0)" pos:2 bitmap:"db:\mf_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"

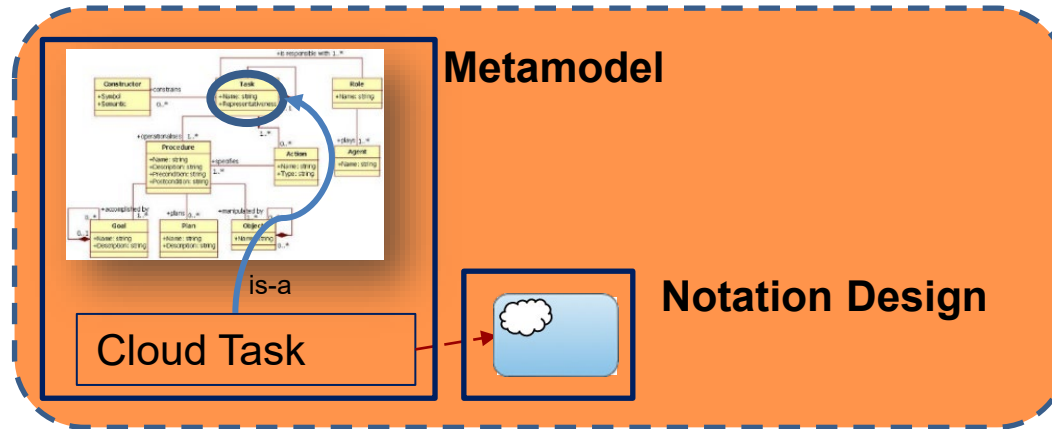
The dialog box also has "Apply", "Cancel", and "Help" buttons at the bottom. The status bar at the bottom of the dialog shows "315 characters Ln 3 Col 65".

Change of Metamodel

- Example: new task type Cloud Task



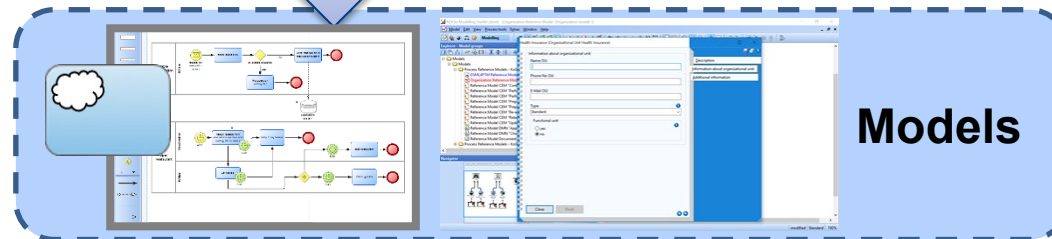

Metamodel
Engineer



Meta-
modeling

Feedback
Amendments
Improvements


Modeler



Modeling