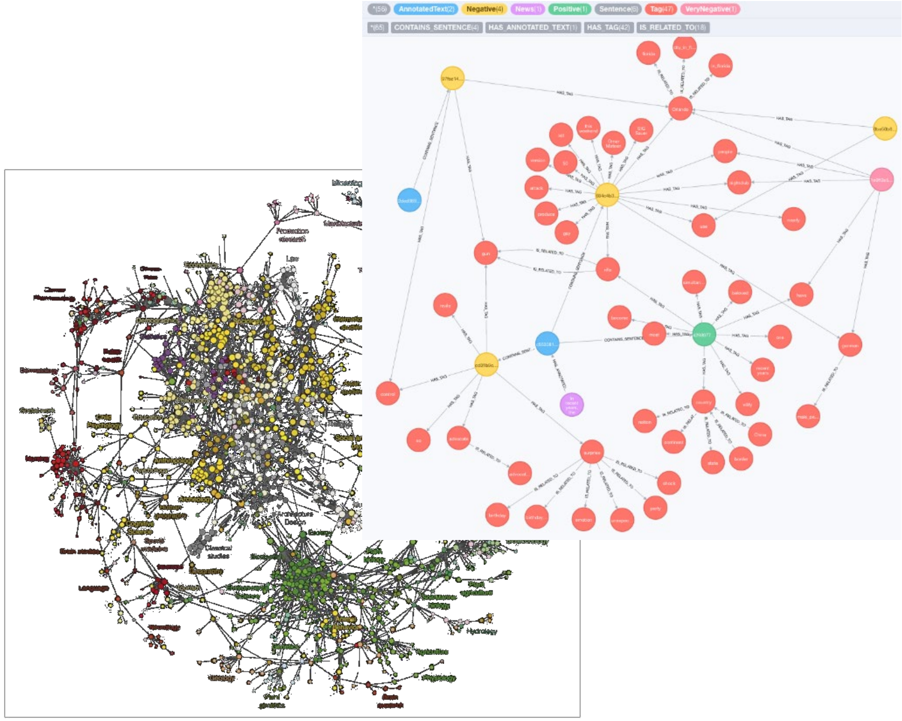
UNICAM
Università di Camerino
1336

Conceptual Modelling

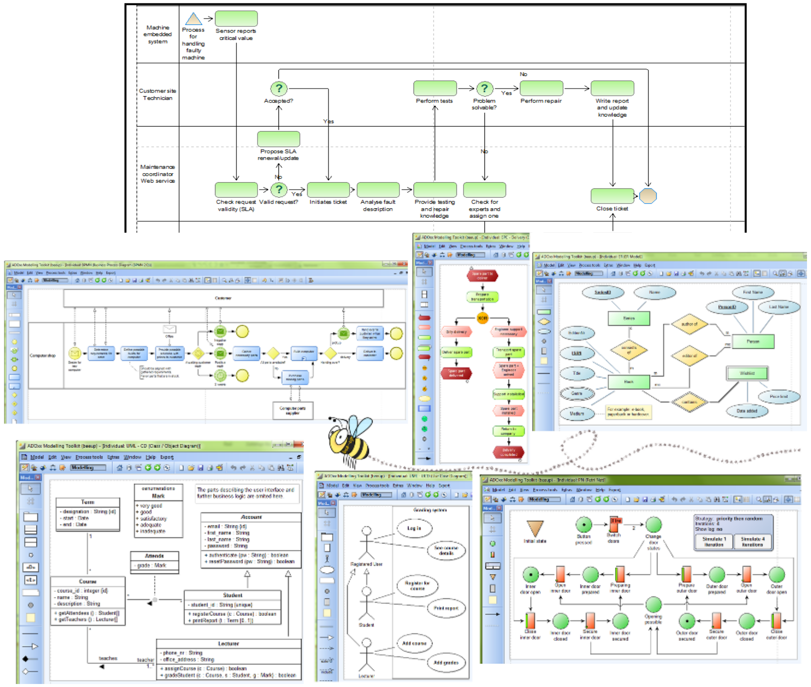
Knut Hinkelmann

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Knowledge Graphs

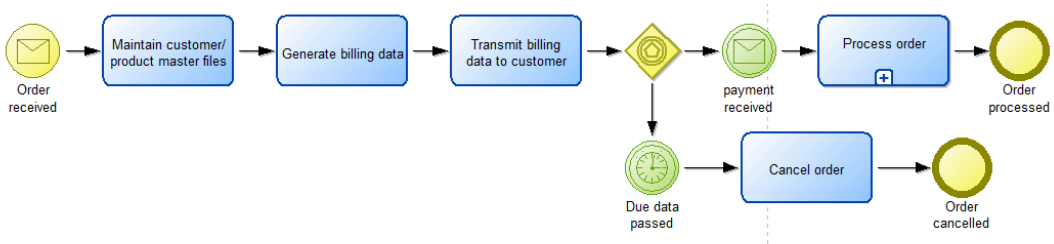


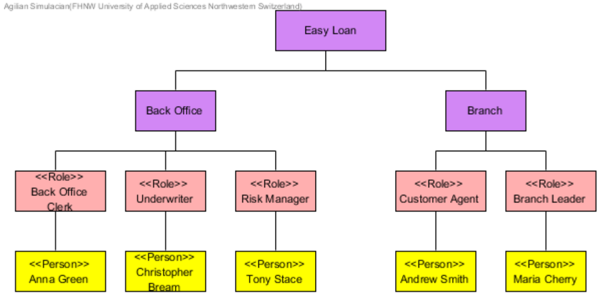
Conceptual (graphcial) Models

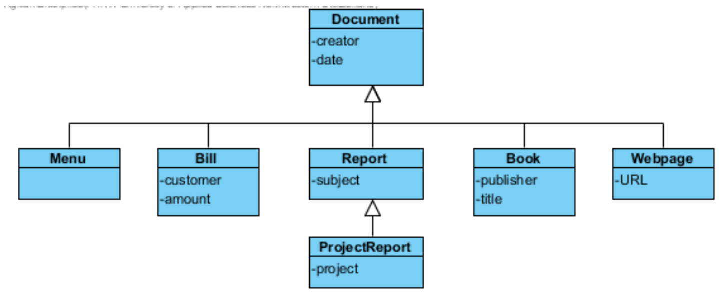

Machine embedded system
Process for handling faulty machine
Sensor reports critical value
Customer site Technician
Accepted?
Perform tests
Problem solvable?
Perform repair
Write report and update knowledge
Yes
No
Maintenance coordinator Web service
Propose SLA renewal/update
Check request validity (SLA)
Valid request?
Initiates ticket
Analyse fault description
Provide testing and repair knowledge
Check for experts and assign one
Close ticket

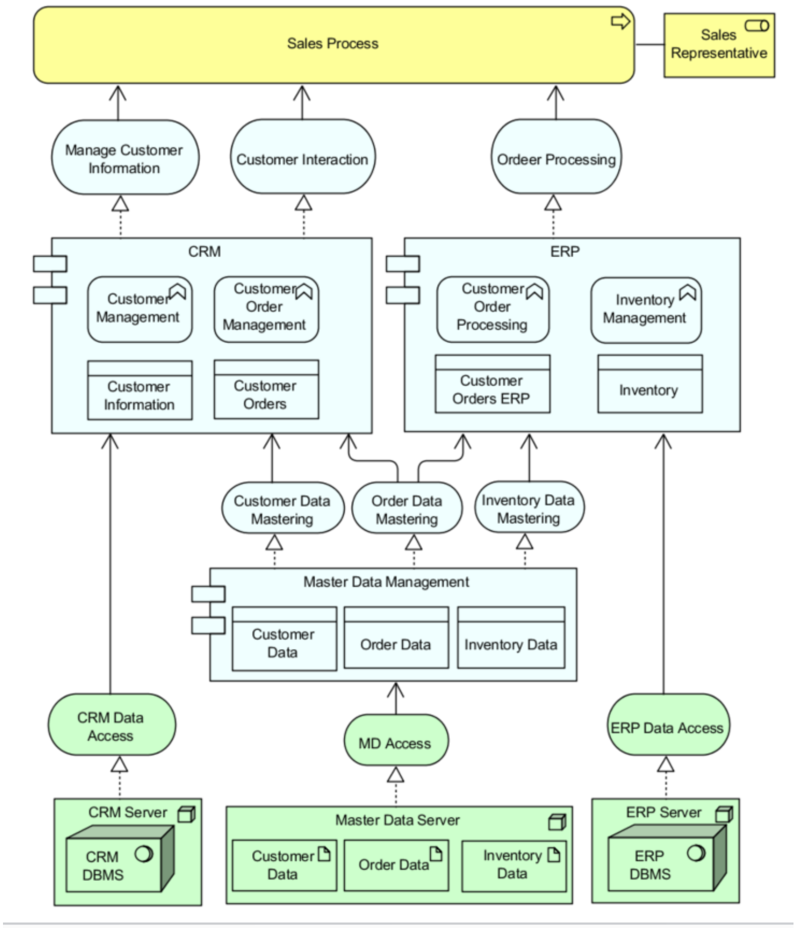
Modeling using predefined *concepts*

Enterprise Models









General-purpose Modelling Languages vs. Domain-specific Modelling Languages

- ***General-purpose*** modelling languages can be used to represent any kind of knowledge
  - Examples: Class diagrams, Knowledge graphs (RDFS)
  - Concepts: Classes, Properties
- ***Domain-specific*** languages have *predefined* concepts (modeling elements and relationships) that are specific for a domain
  - Examples of domain-specific modelling languages:
    - **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, …

Strengths and Weaknesses of Domain-specfic Modelling Languages

- Strengths
  - Comprehensibility of models
    - Concepts are adequate for stakeholders
  - Guidance for modelers
    - Predefined concepts determine what is relevant for a model
    - Modeling language determines correct usage of elements
  - Standardisation: Reuse of models
    - Common concepts for a domain (e.g. BPMN, ArchiMate)
- Weaknesses
  - Restricted to a specific domain
    - Only what can be expressed with the modelling elements can be modeled

Conceptual Modeling and Metamodelling

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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 predefinded concepts.

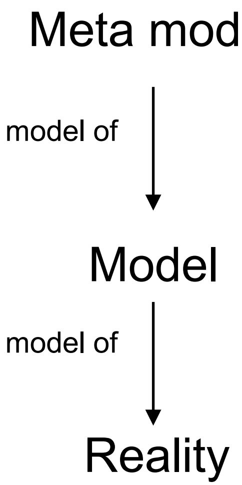
Meta Model

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

Modelling Language

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

Meta-model



Meta model

model of

Model

model of

Reality

- 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

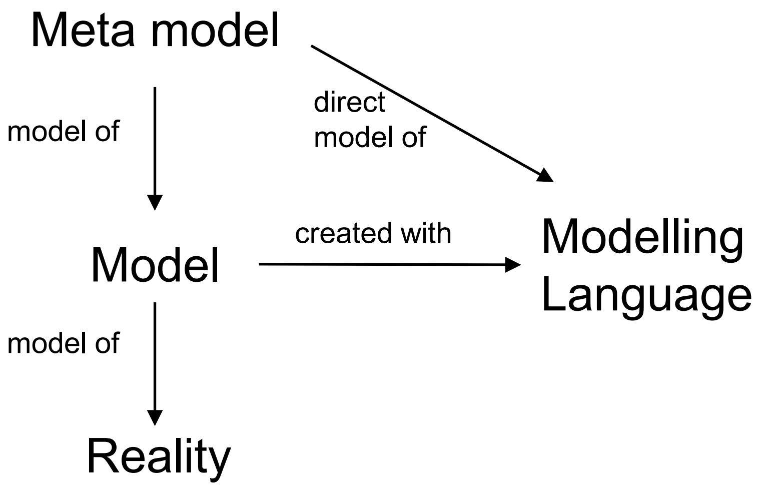
Metamodel:

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:

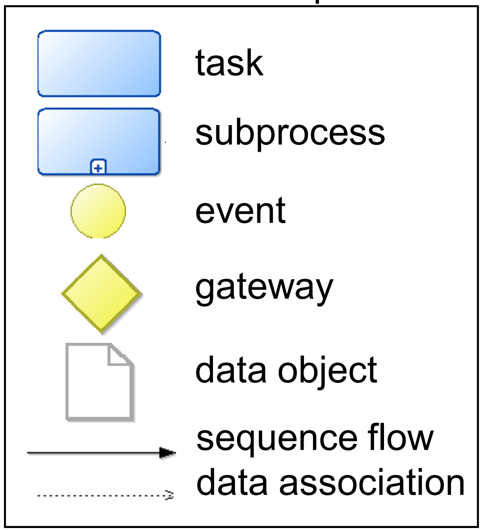
Concepts which can be used to create models.

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



task

subprocess

event

gateway

data object

sequence flow

data association

Illustration: Modeling Language for Business Processes

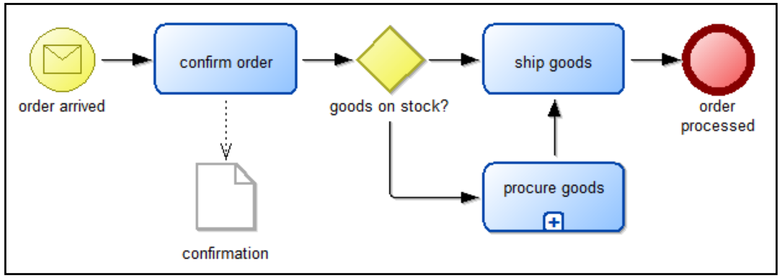
Metamodel:

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.

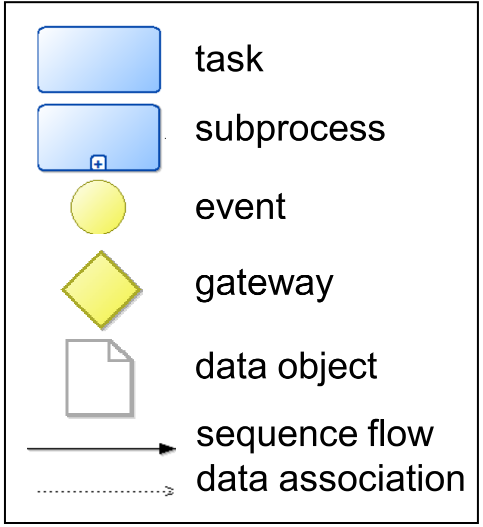
Model:



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"

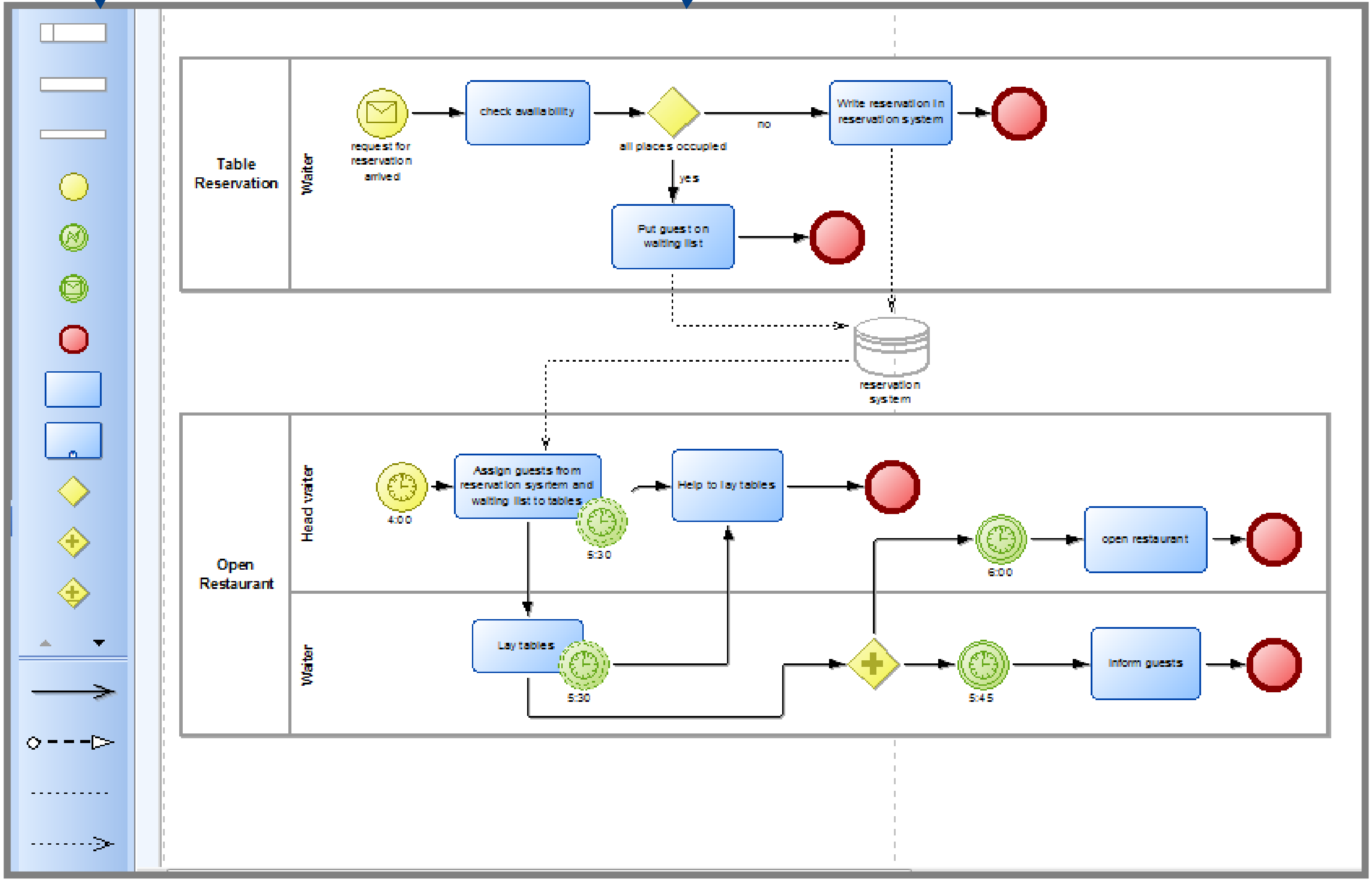
Modelling Language:

Notation/appearance of meta-model concept



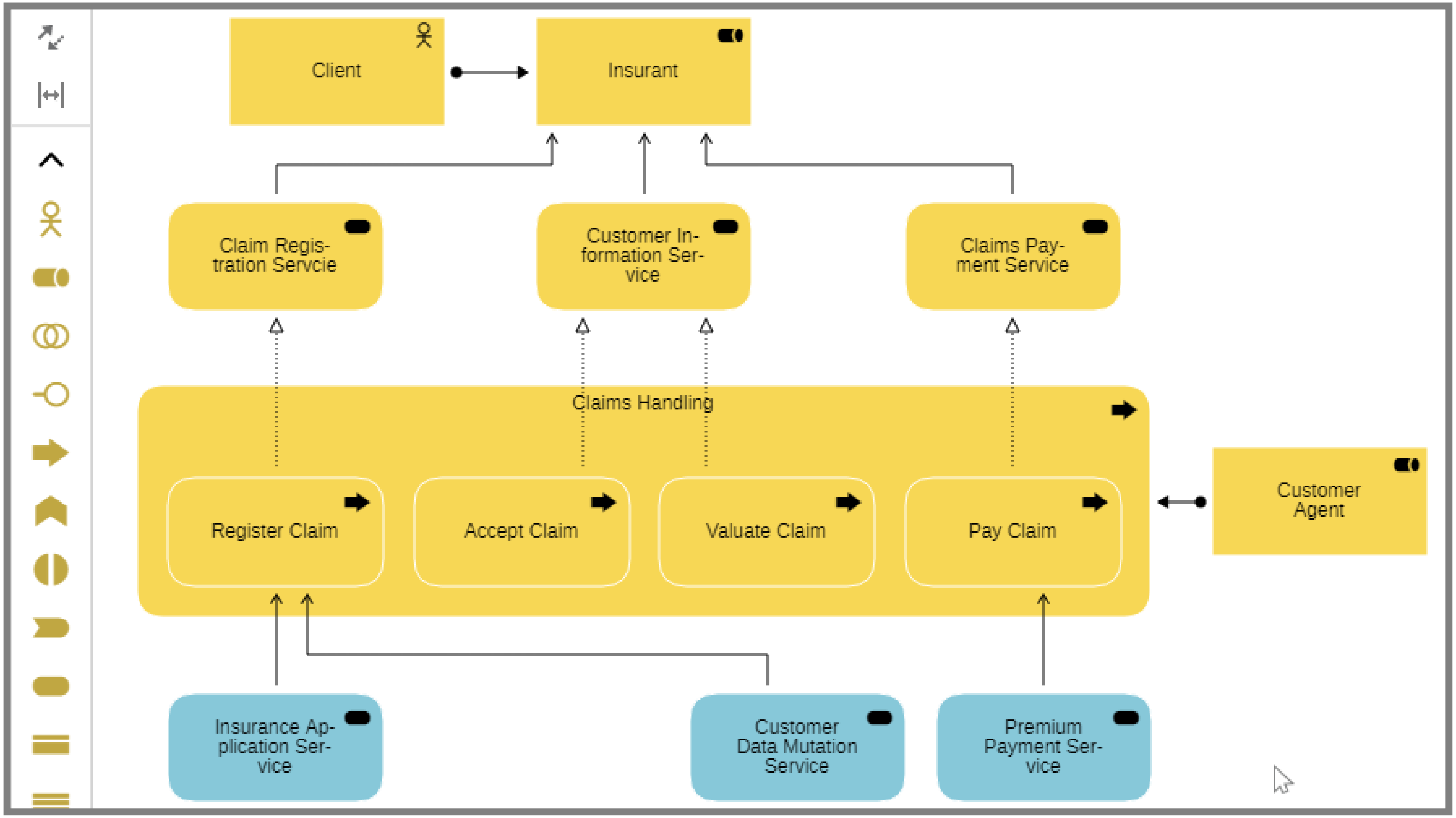
Modelling Language

Model

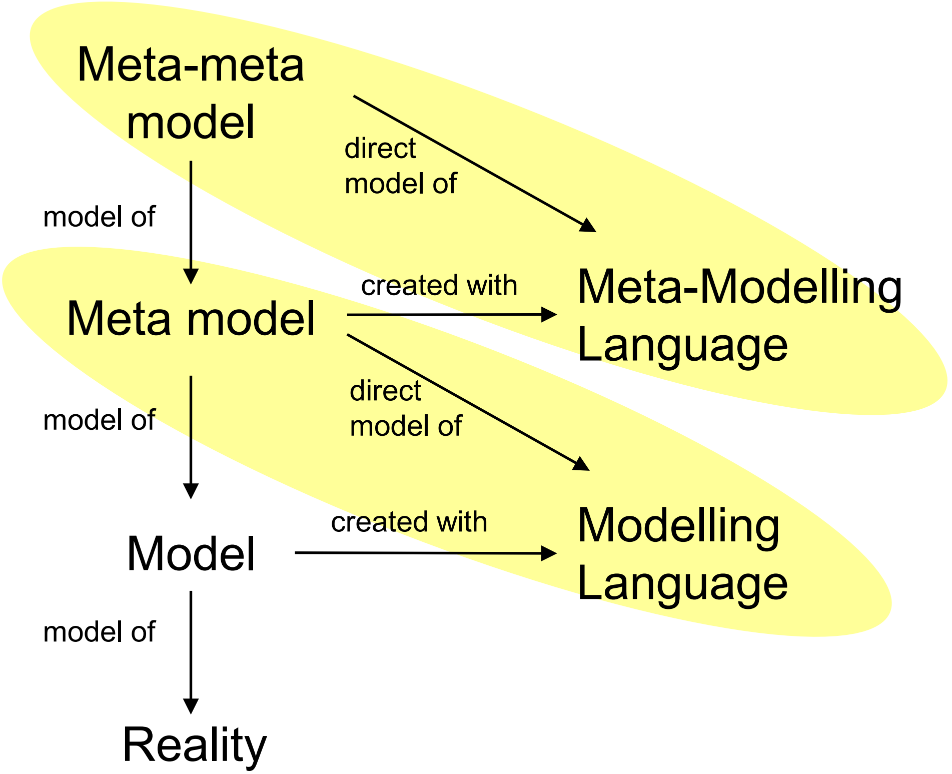


Modelling Language

Model



Meta-meta model

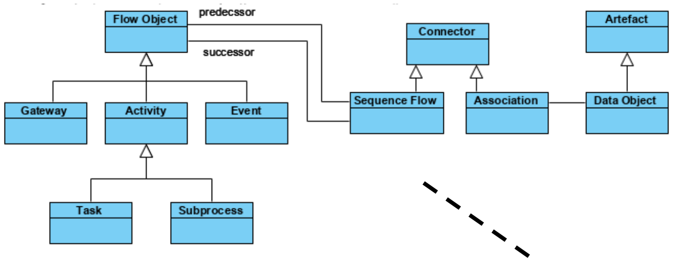


- The meta model must again be described in some language, which is specified in a meta-meta model
- A **meta-meta model** defines the concepts for describing a meta model
- Graphical models usually have to kinds of concepts
  - Modeling elements
  - Relationships
- Examples for meta-modeling languages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

Metamodels can be defined as Class Diagrams

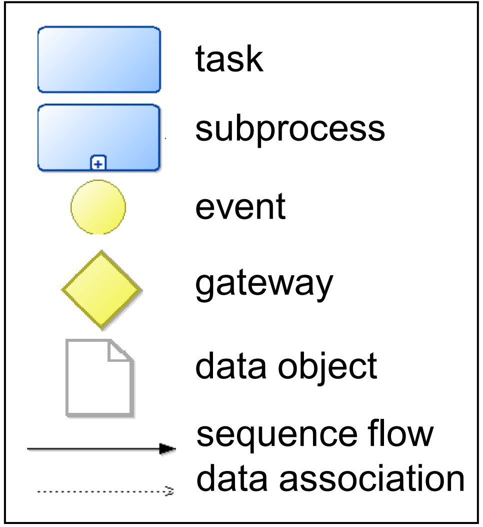
Metamodel:

Concepts which can be used to create models.

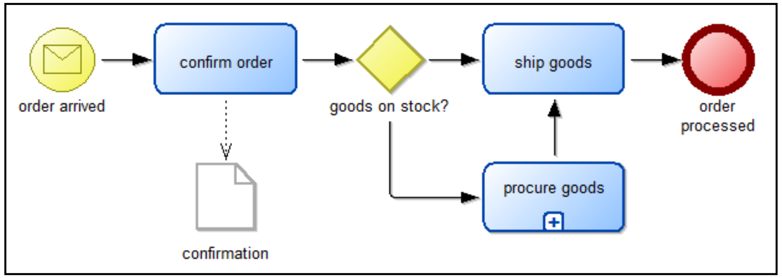


Modelling Language:

Notation/appearance of meta-model concept



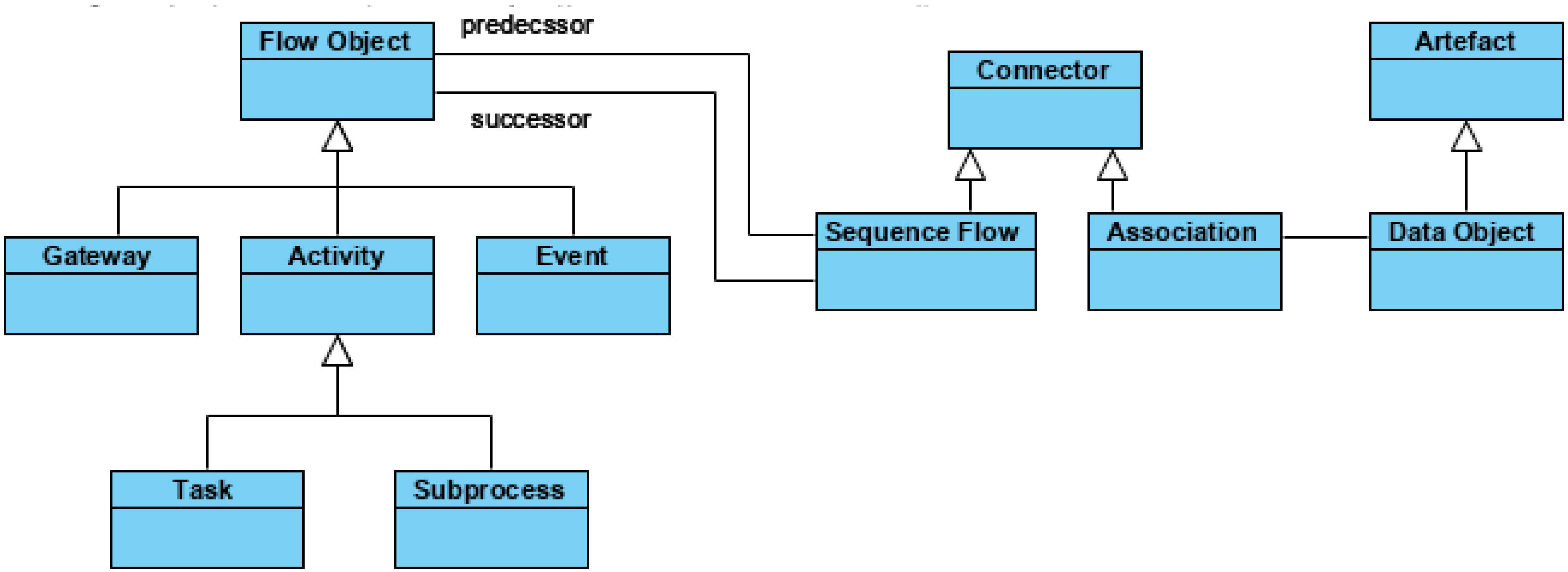
Model:



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"

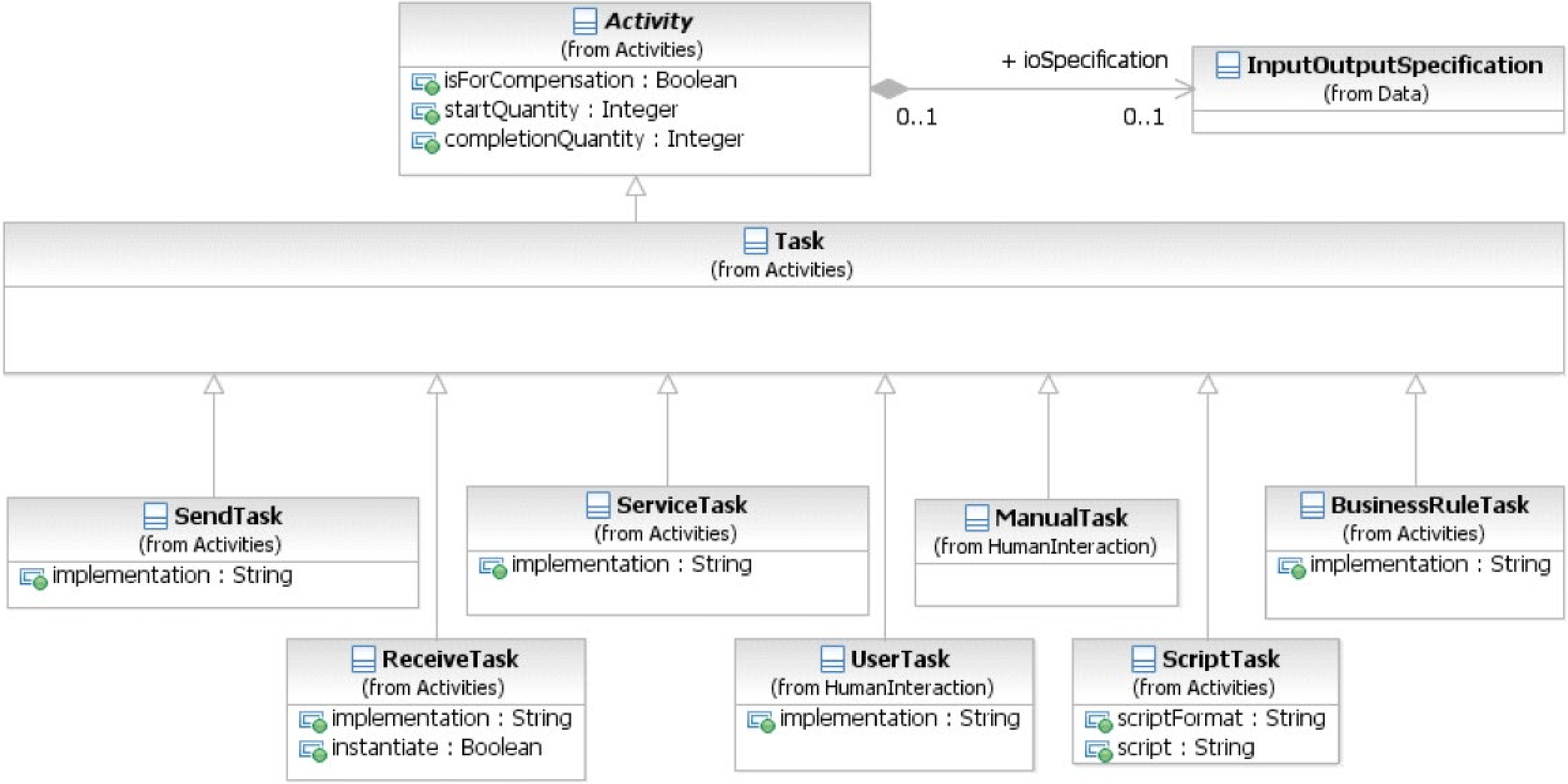
Metamodels can be defined as Class Diagrams

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



(UML Class diagrams where 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



Source: BPMN 2.0 specification

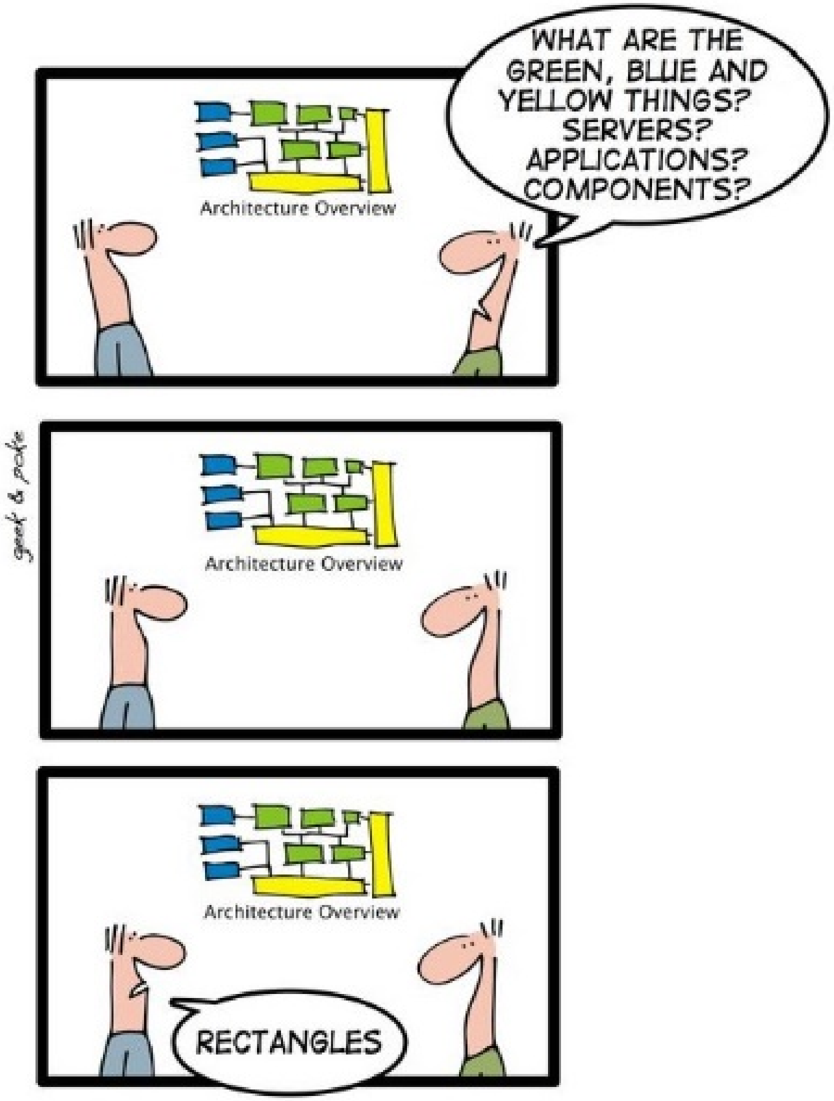
Knowledge in Models

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.

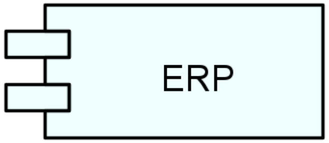
http://blog.bizzdesign.com/how-to-not-fail-when-implementing-strategy

Interpretation of Models

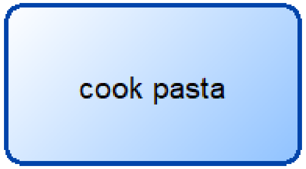


Making the Knowledge in Models explicit

- Humans «know» the meaning of the modeling objects.
  - Meta model: Concepts of the model language
  - Application: Labels/names of the model elements
- Examples:



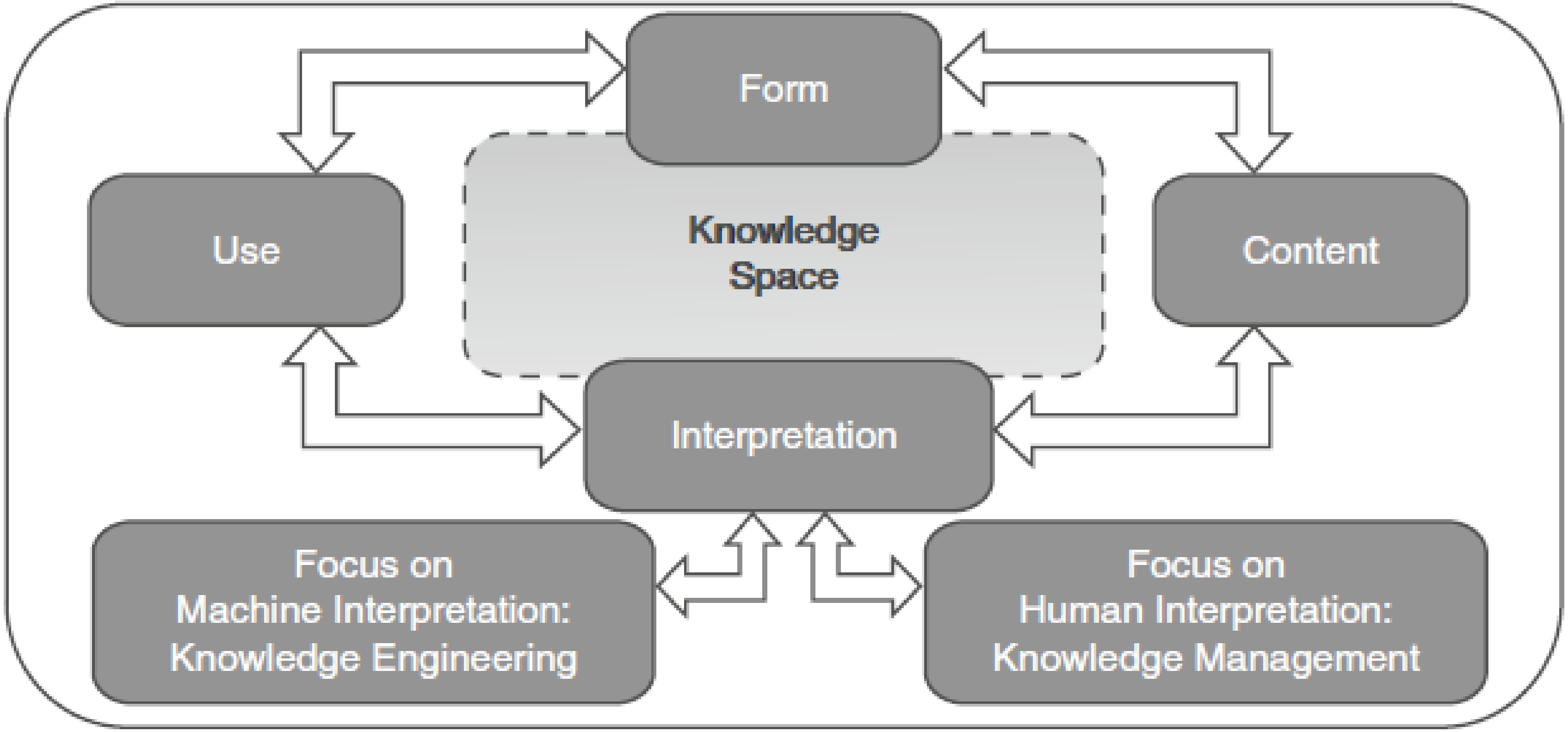
- Meta model: Application Component
- Application: «ERP System» is business software



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

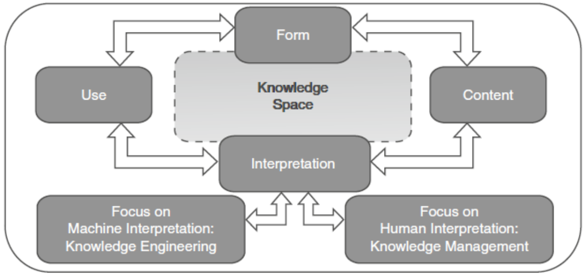
- The objective is to represent the knowledge so that it can be interpreted by a system for decision making and problem solving

Dimensions of a Knowledge Space


Form
Use
Knowledge Space
Content
Interpretation
Focus on Machine Interpretation: Knowledge Engineering
Focus on Human Interpretation: Knowledge Management

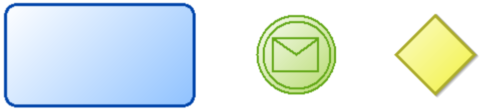
Karagiannis, D., & Woitsch, R. (2010). Knowledge Engineering in Business Process Management.
In *Handbook on Business Process Management 2* (pp. 463–485). Springer.

Dimensions of the Knowledge Space

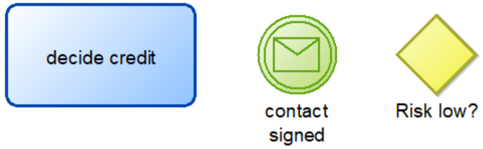

Form
Use
Knowledge Space
Content
Interpretation
Focus on Machine Interpretation: Knowledge Engineering
Focus on Human Interpretation: Knowledge Management

Use:
- process optimization requires knowledge about time and costs
- selection of a cloud service require knowledge about data and functionality

Form: modeling language



Content: Instantiation of concepts



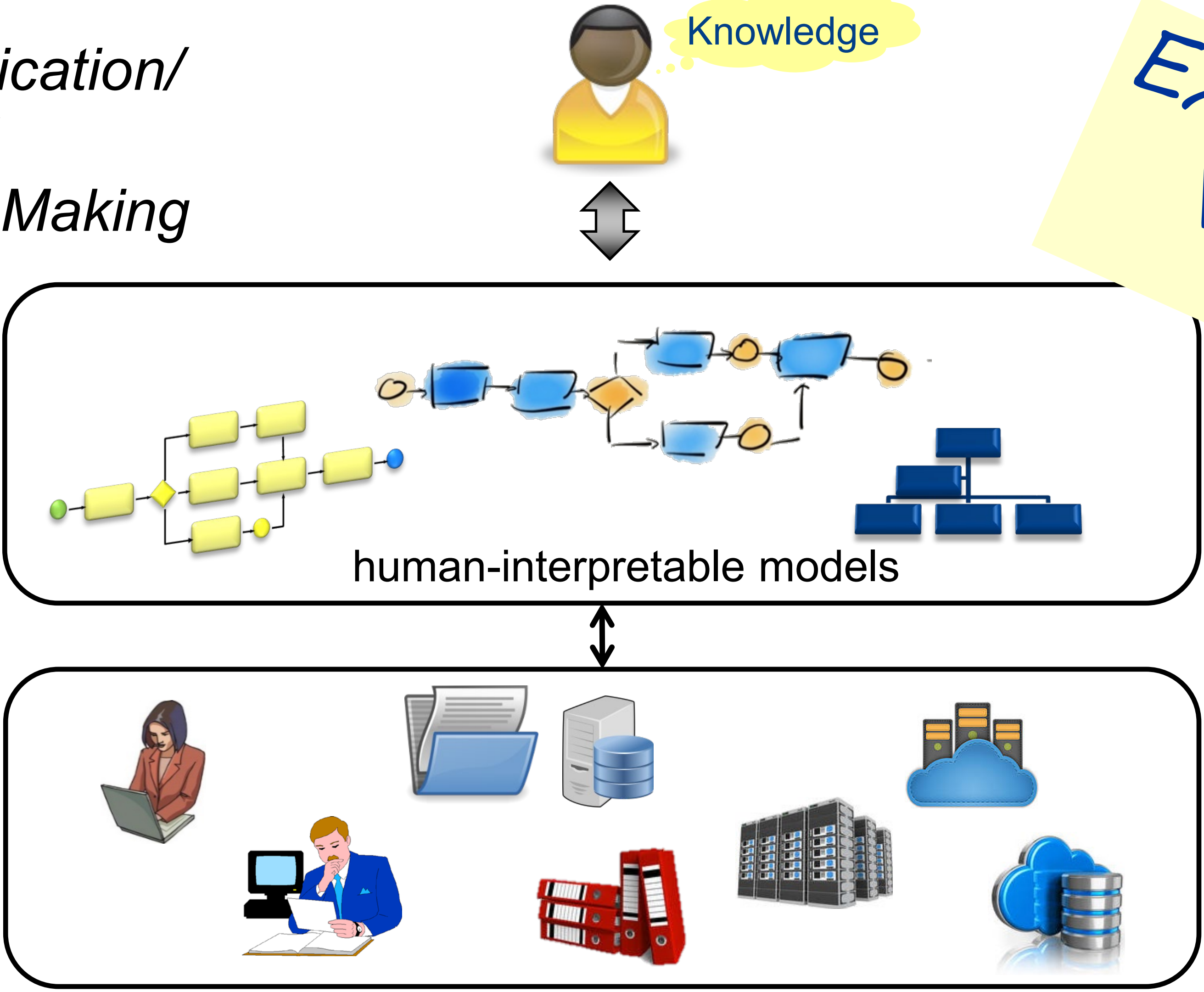
- ***Use***: Stakeholders and their concerns determine the relevant subset of the knowledge
- ***Form***: Syntax and semantic of *meta model concepts*.
- ***Content***: *Instantiation* of meta model concepts for a specific *application* (represented in the labels)
- ***Interpretation***: Giving meaning to a model:
  - Graphical models are cognitively adequate for human
  - Machines need more formal representation

Graphical Models are appropriate for Humans

Communication/
Analysis/
Decision Making

Models

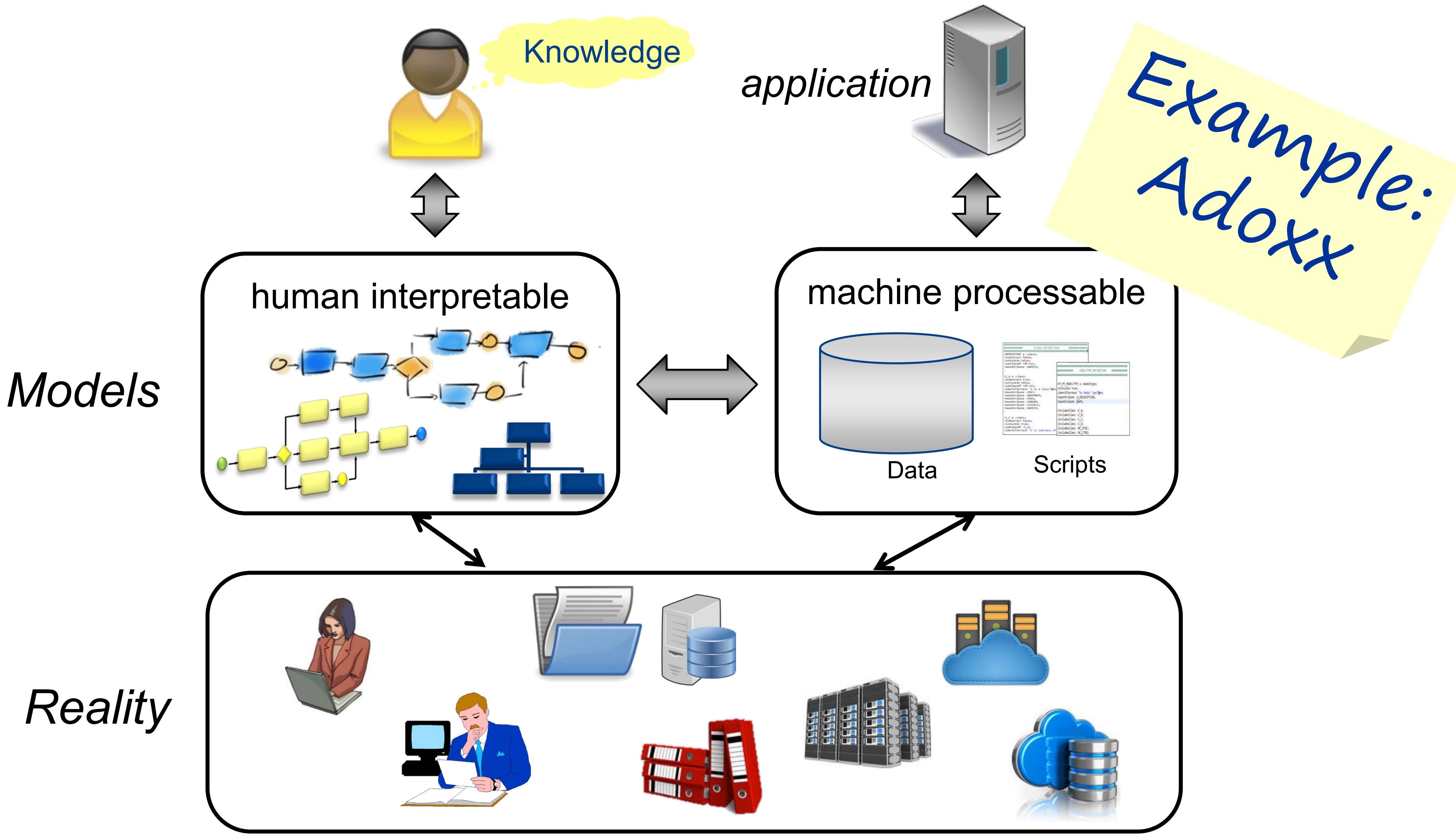
Reality



Example: Visio

Models should allow automated analysis, decision making and digitalization

Graphical Models are Represented in a Database

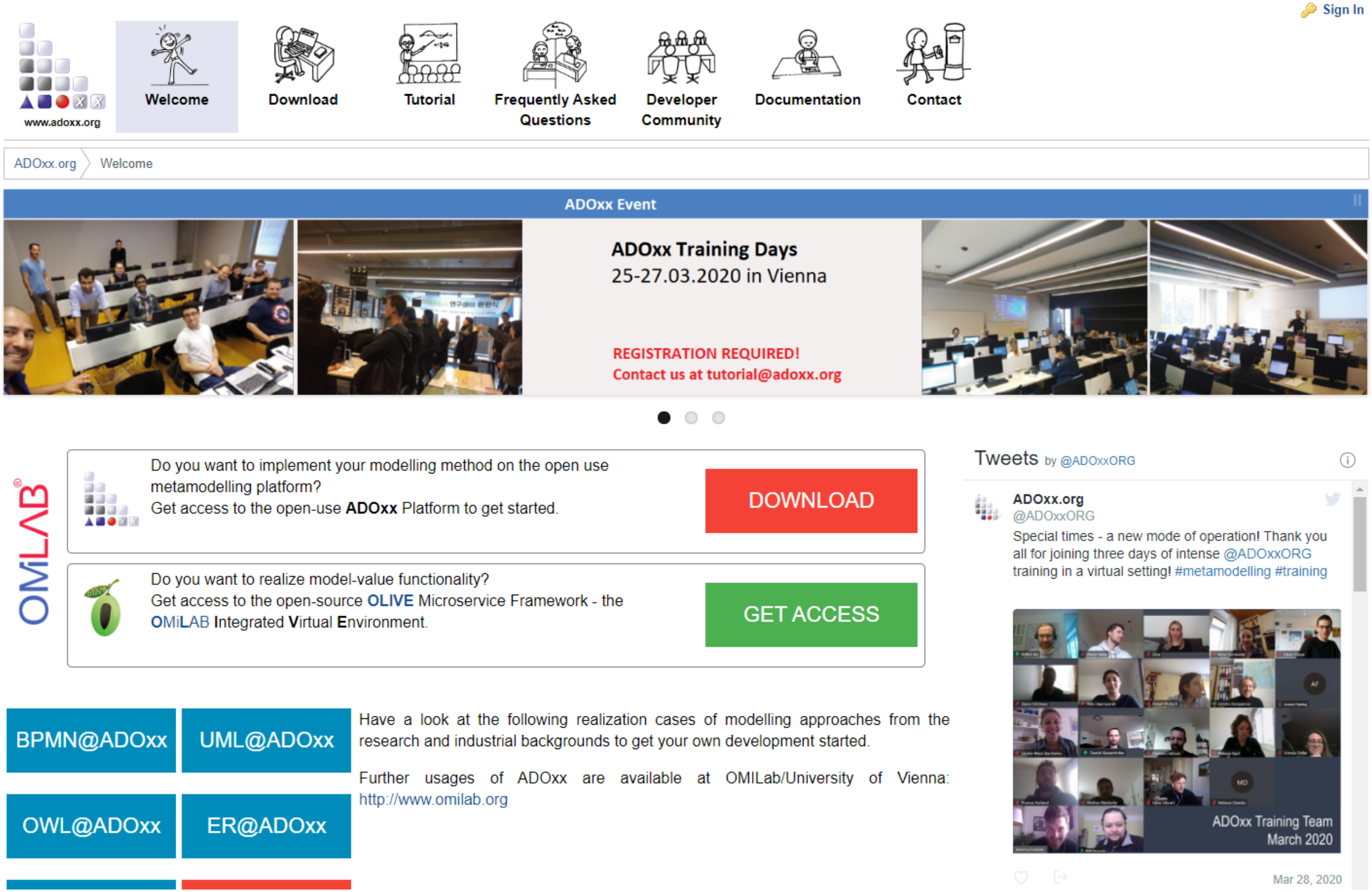


Example: Adoxx

Metamodelling with ADOxx

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

adoxx.org – Download, Tutorials, Community



🔑 Sign In

www.adoxx.org

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

● ○ ○

OMiLAB

Do you want to implement your modelling method on the open use metamodelling 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 **O**MiLAB **I**ntegrated **V**irtual **E**nvironment.

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

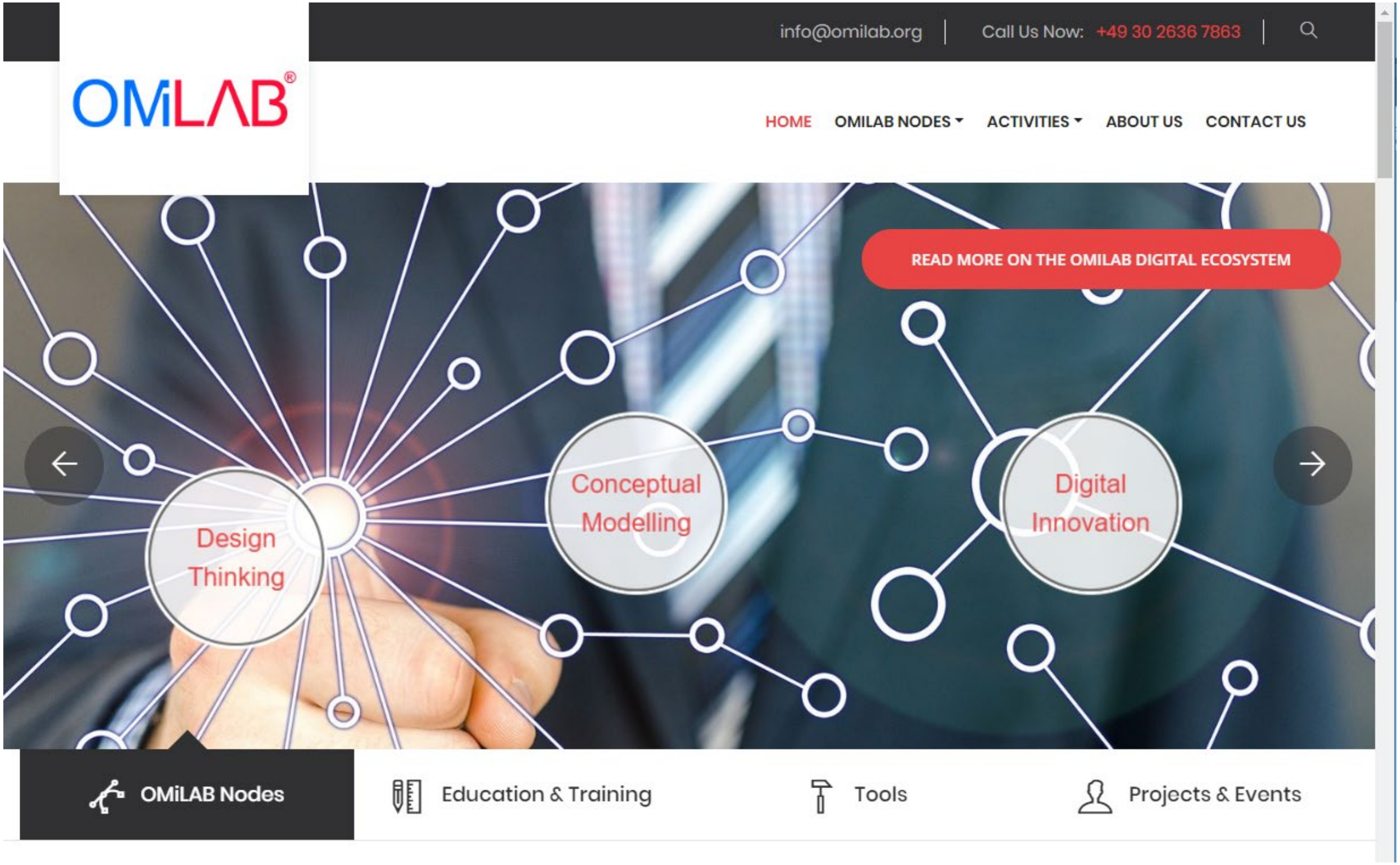
Special times - a new mode of operation! Thank you all for joining three days of intense @ADOxxORG training in a virtual setting! #metamodelling #training

ADOxx Training Team
March 2020

Mar 28, 2020

OMiLAB – A Conceptual Modelling Commnity

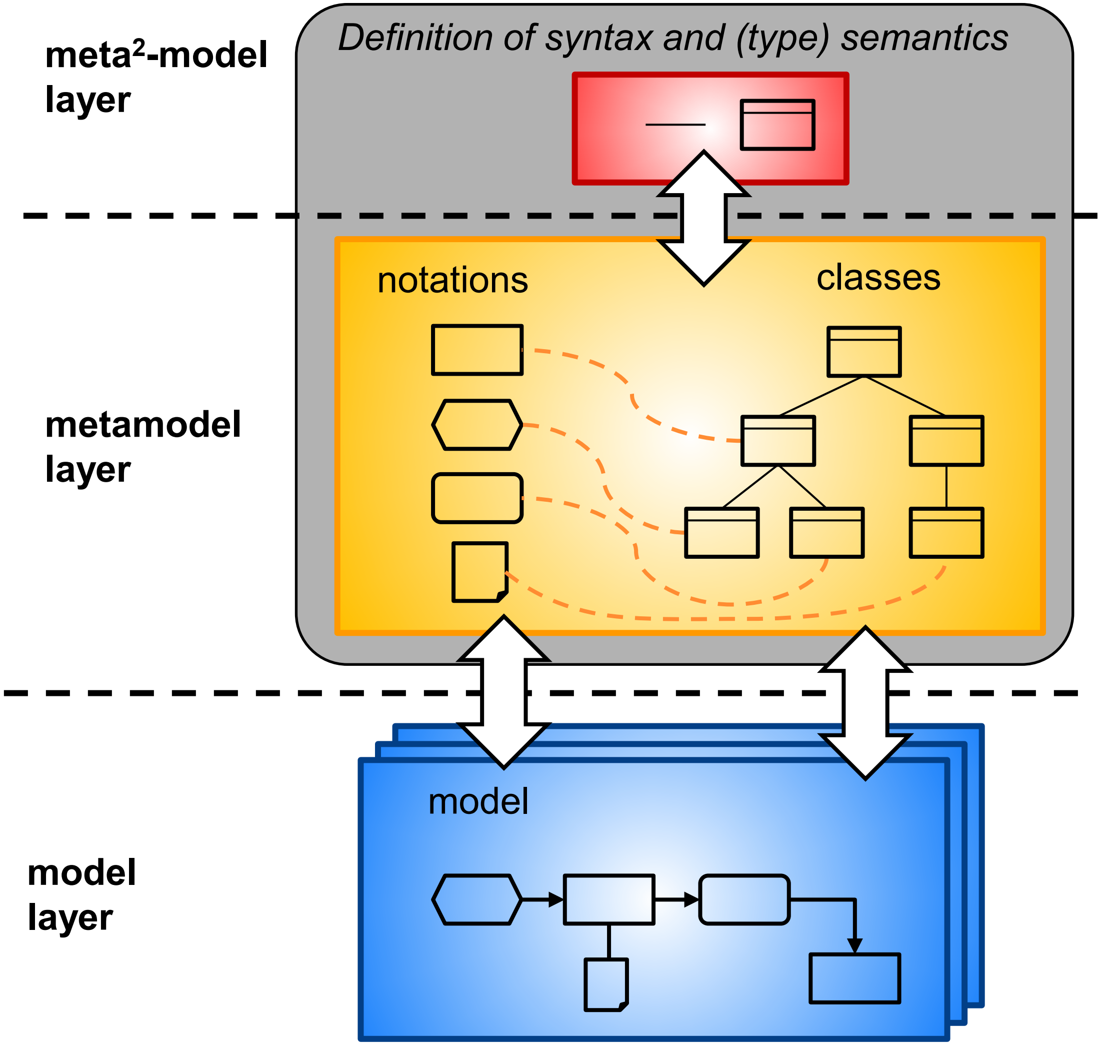
ADOxx is the basis for OMiLAB



The ADOxx Environment

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

Modeling Environment



meta²-model layer

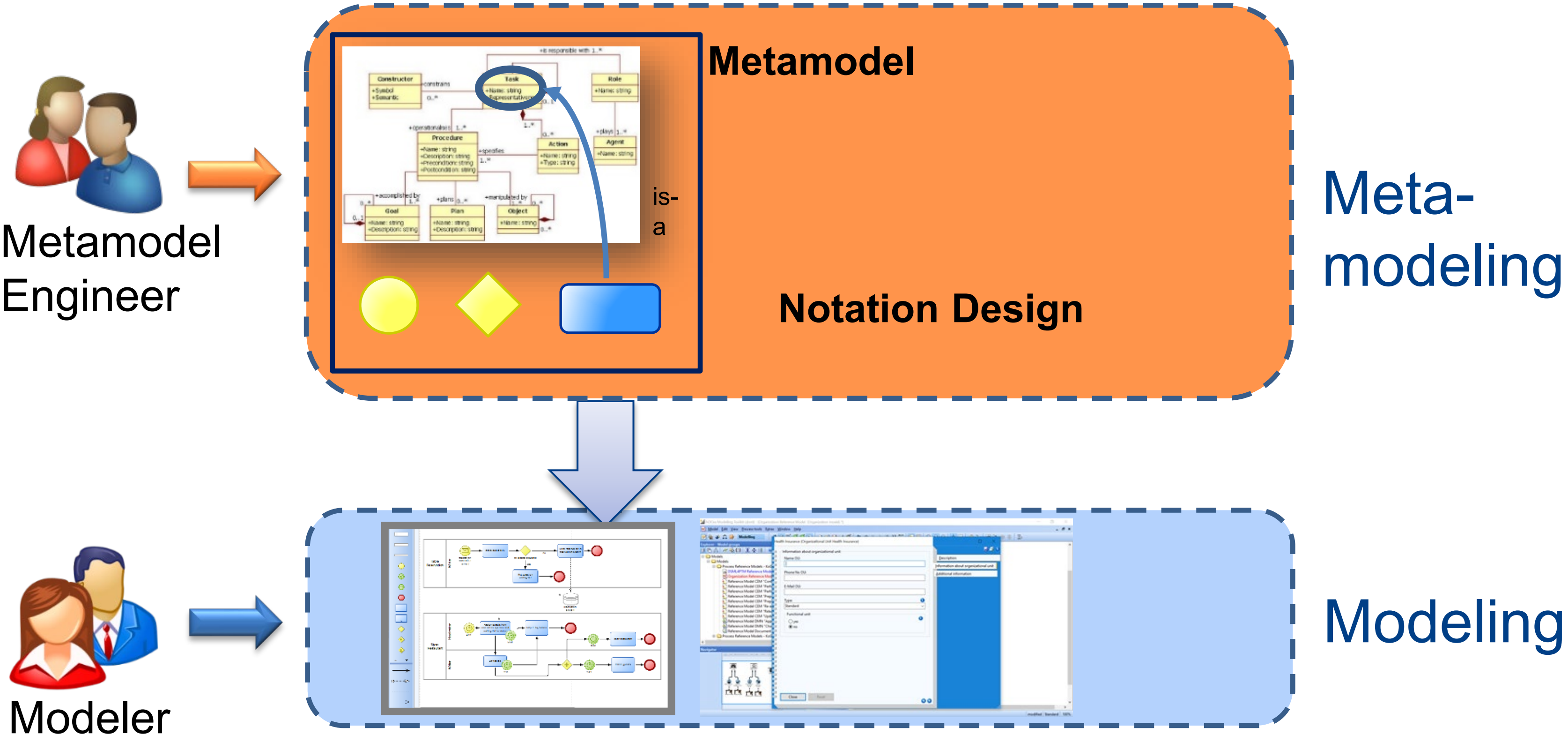
metamodel layer

model layer

ADOxx Development Toolkit

ADOxx Modeling Toolkit

Modeling and Metamodeling



Metamodel Engineer

Metamodel

is-a

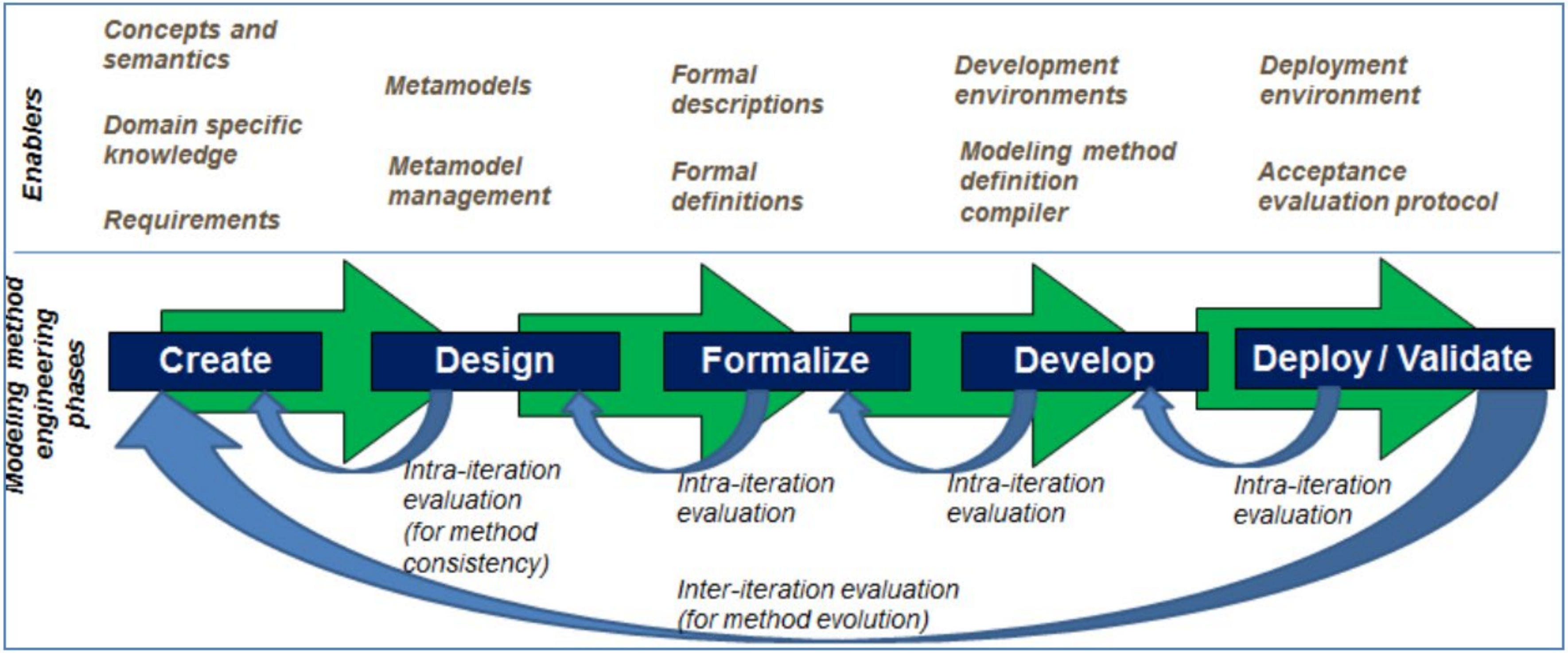
Notation Design

Meta-modeling

Modeler

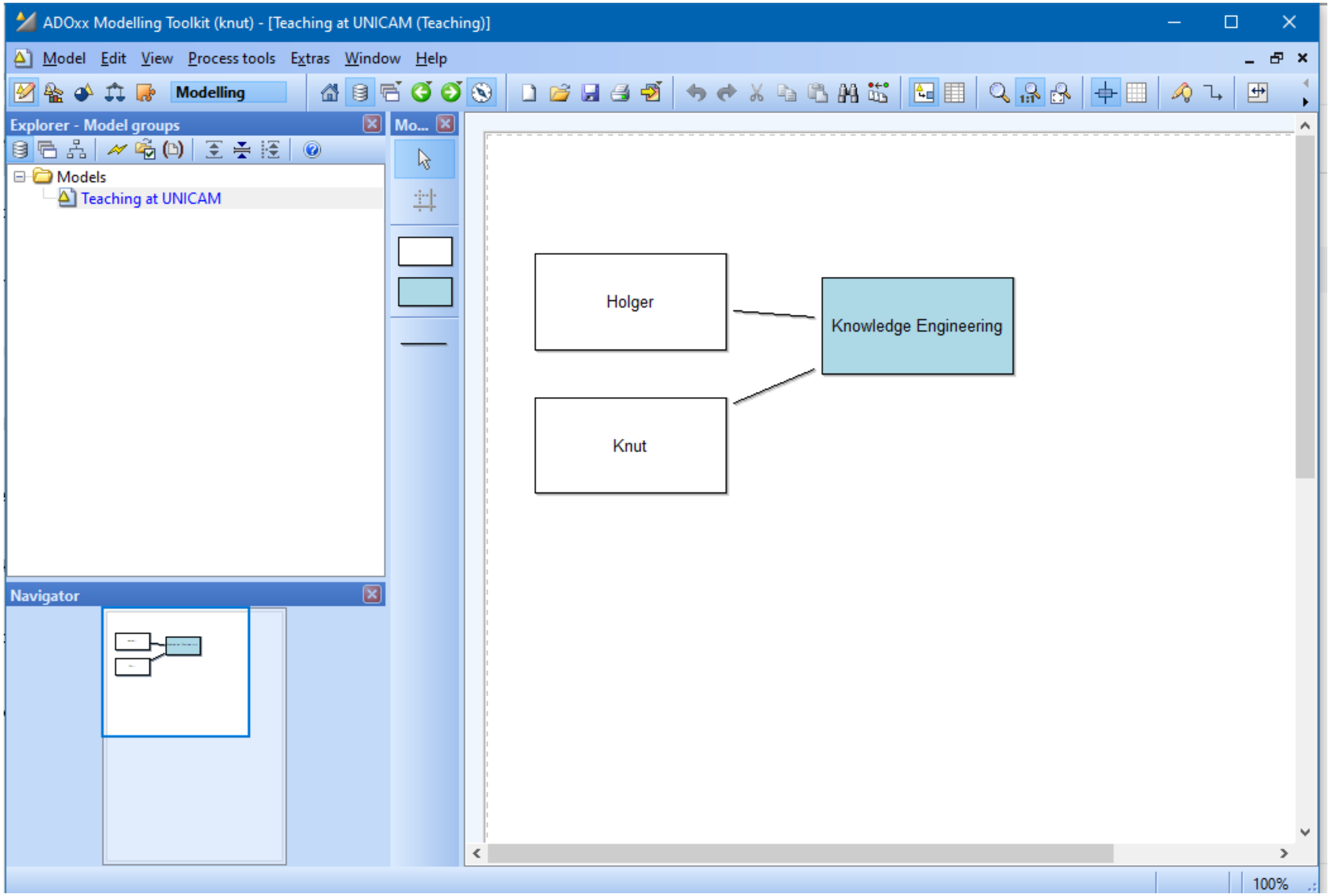
Modeling

The AMME LifeCycle
Agile Modeling Method Engineering


Enablers
Concepts and semantics
Domain specific knowledge
Requirements
Metamodels
Metamodel management
Formal descriptions
Formal definitions
Development environments
Modeling method definition compiler
Deployment environment
Acceptance evaluation protocol
Modeling method engineering phases
Create
Design
Formalize
Develop
Deploy / Validate
Intra-iteration evaluation (for method consistency)
Intra-iteration evaluation
Intra-iteration evaluation
Intra-iteration evaluation
Inter-iteration evaluation (for method evolution)

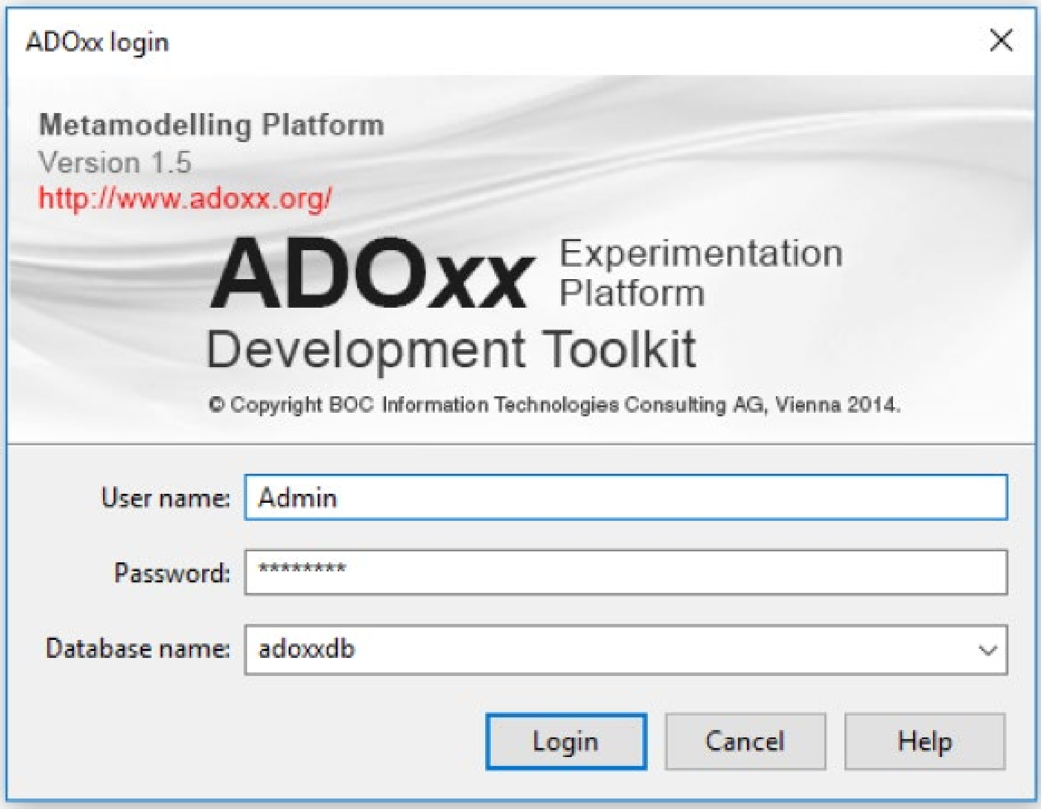
(Karagiannis 2015)

Example: Create a Modeling Language for Teaching


ADOxx Modelling Toolkit (knut) - [Teaching at UNICAM (Teaching)]
Model Edit View Process tools Extras Window Help
Modelling
Explorer - Model groups
Models
Teaching at UNICAM
Mo...
Navigator
Holger
Knut
Knowledge Engineering
100%

Development Toolkit

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



ADOxx login

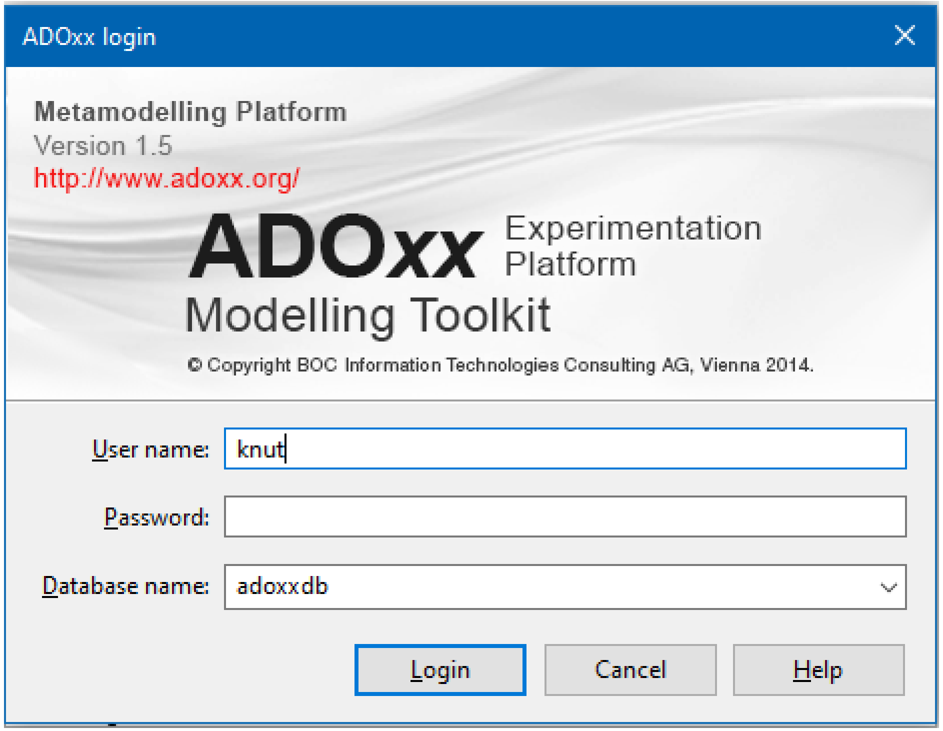
Metamodelling Platform
Version 1.5
http://www.adoxx.org/

© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

User name: Admin

Password: *********

Database name: adoxxdb



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

ADOxx Experimentation Platform
Modelling Toolkit

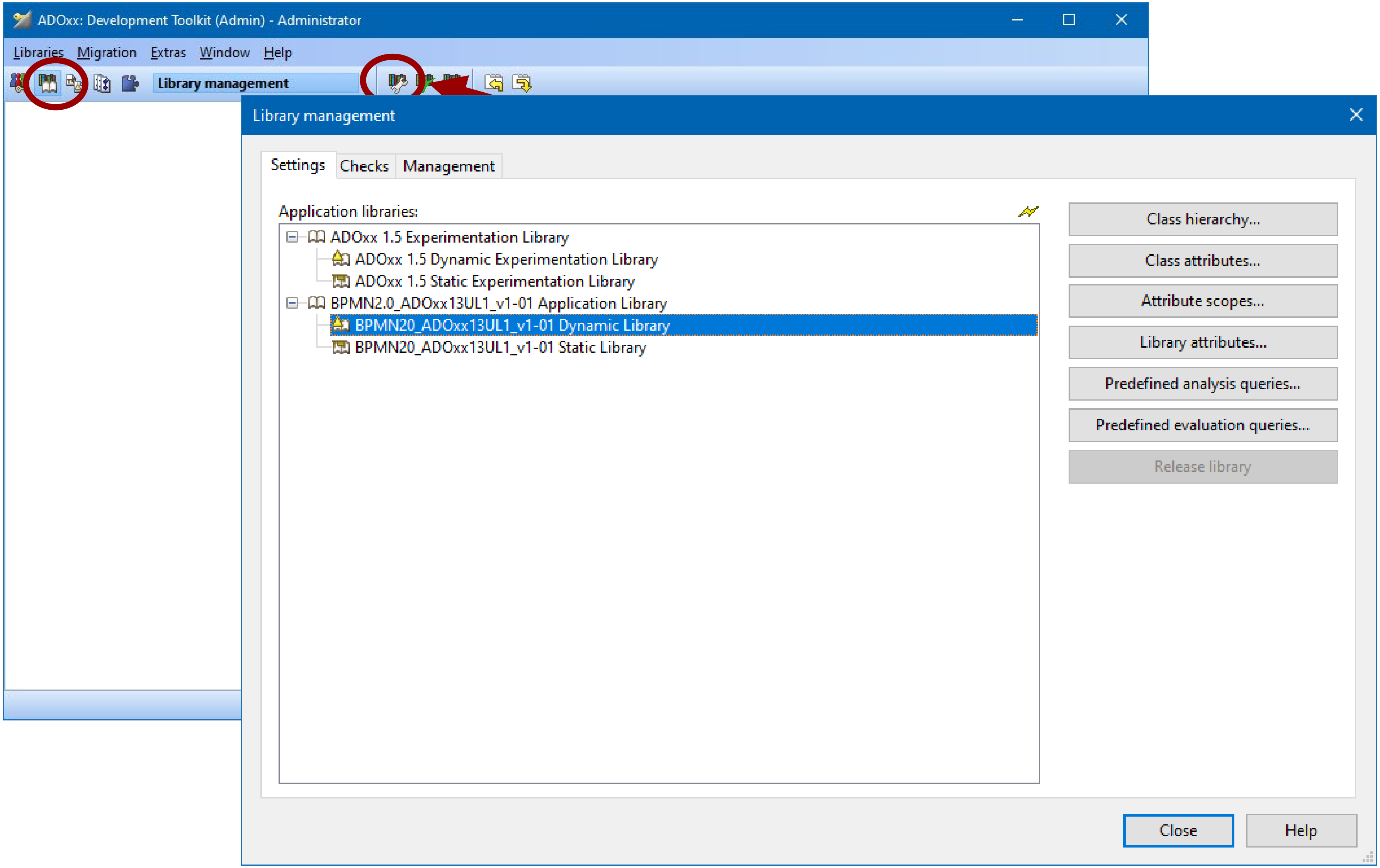
© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

User name: knut

Password:

Database name: adoxxdb

Metamodelling with ADOxx



ADOxx: Development Toolkit (Admin) - Administrator

Libraries Migration Extras Window Help

Library management

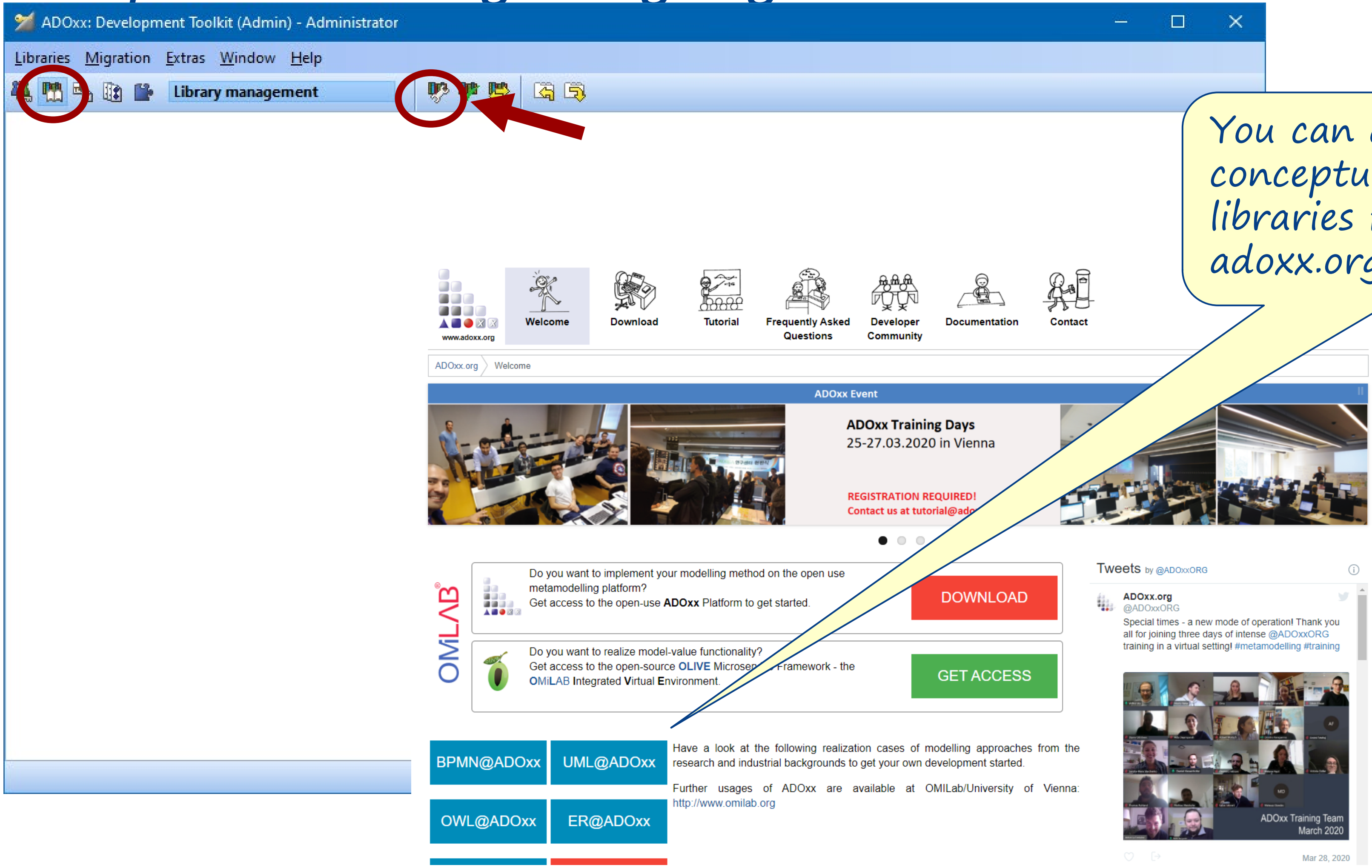
Library management

Settings | Checks | Management

Application libraries:

- 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

Import Modeling Language Libraries



ADOxx: Development Toolkit (Admin) - Administrator

ADOxx Training Days
25-27.03.2020 in Vienna

REGISTRATION REQUIRED!
Contact us at tutorial@ado

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

Do you want to realize model-value functionality?
Get access to the open-source **OLIVE** Microsen Framework - the **OM**i**LAB** **I**ntegrated **V**irtual **E**nvironment.

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

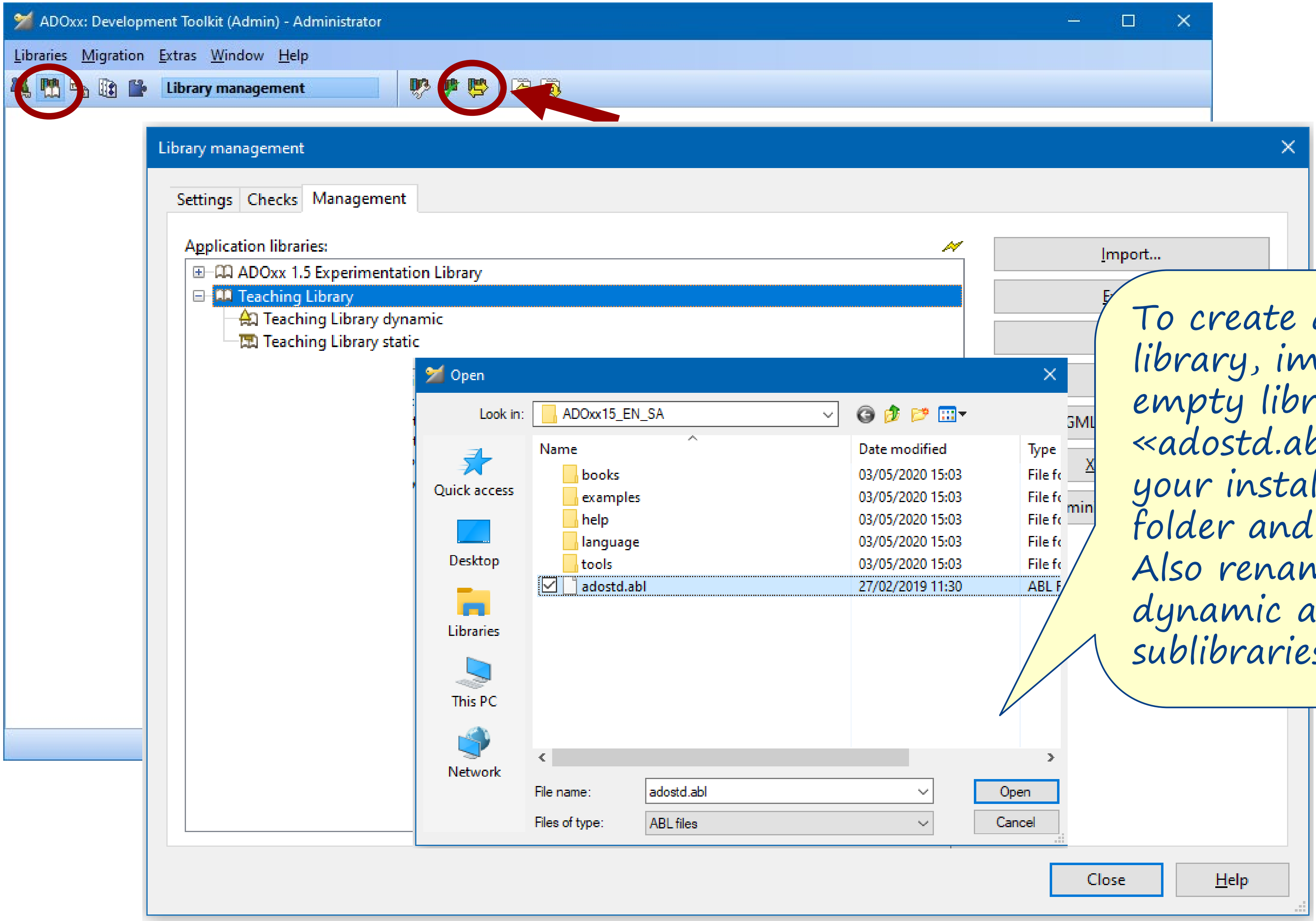
ADOxx.org
@ADOxxORG
Special times - a new mode of operation! Thank you all for joining three days of intense @ADOxxORG training in a virtual setting! #metamodelling #training

ADOxx Training Team
March 2020

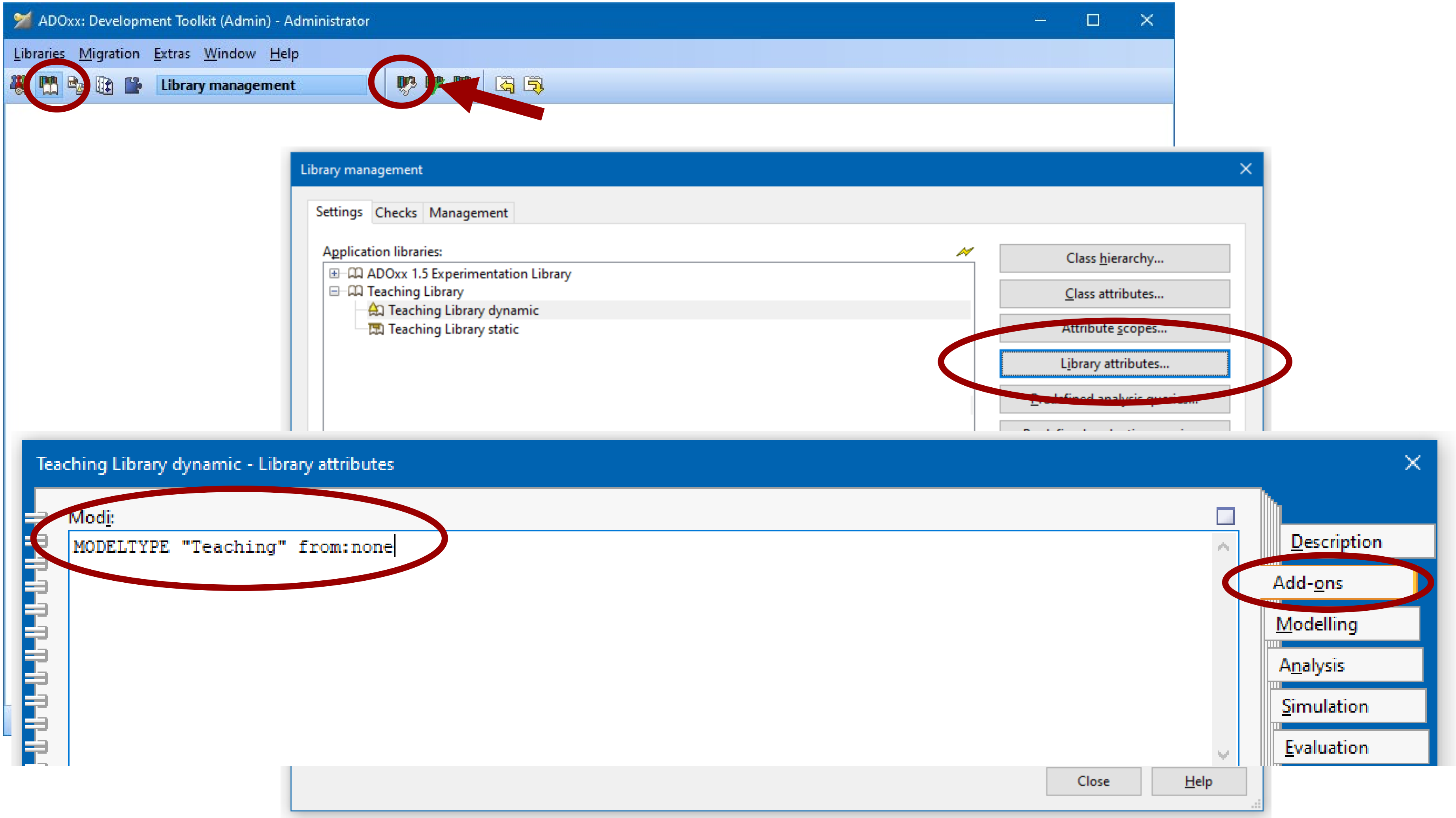
Mar 28, 2020

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

Create a new Modeling Language Library


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
Import...
Open
Look in: ADOxx15_EN_SA
Quick access
Desktop
Libraries
This PC
Network
Name
Date modified
Type
books 03/05/2020 15:03 File fo
examples 03/05/2020 15:03 File fo
help 03/05/2020 15:03 File fo
language 03/05/2020 15:03 File fo
tools 03/05/2020 15:03 File fo
adostd.abl 27/02/2019 11:30 ABL F
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

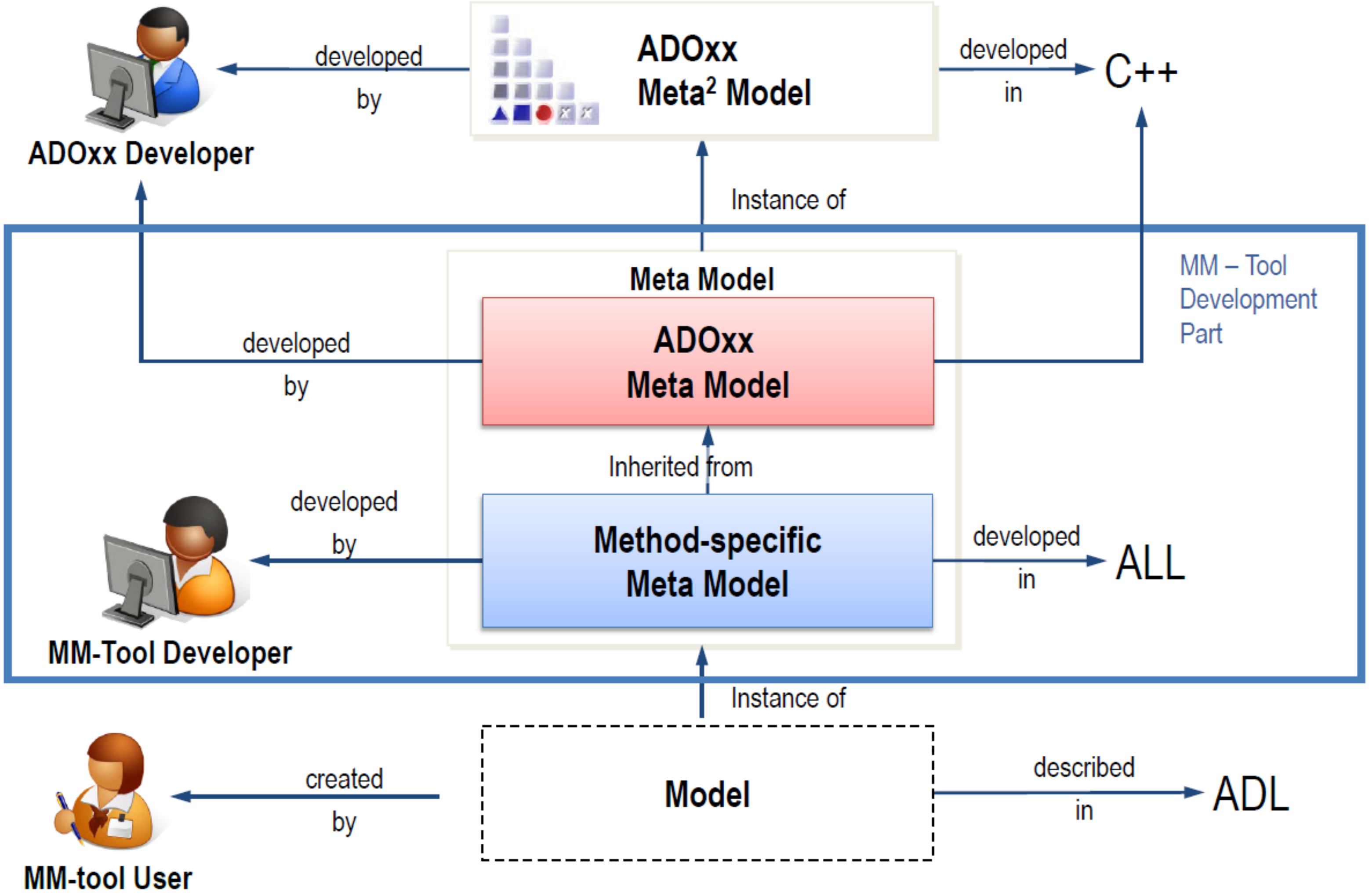


Modi:

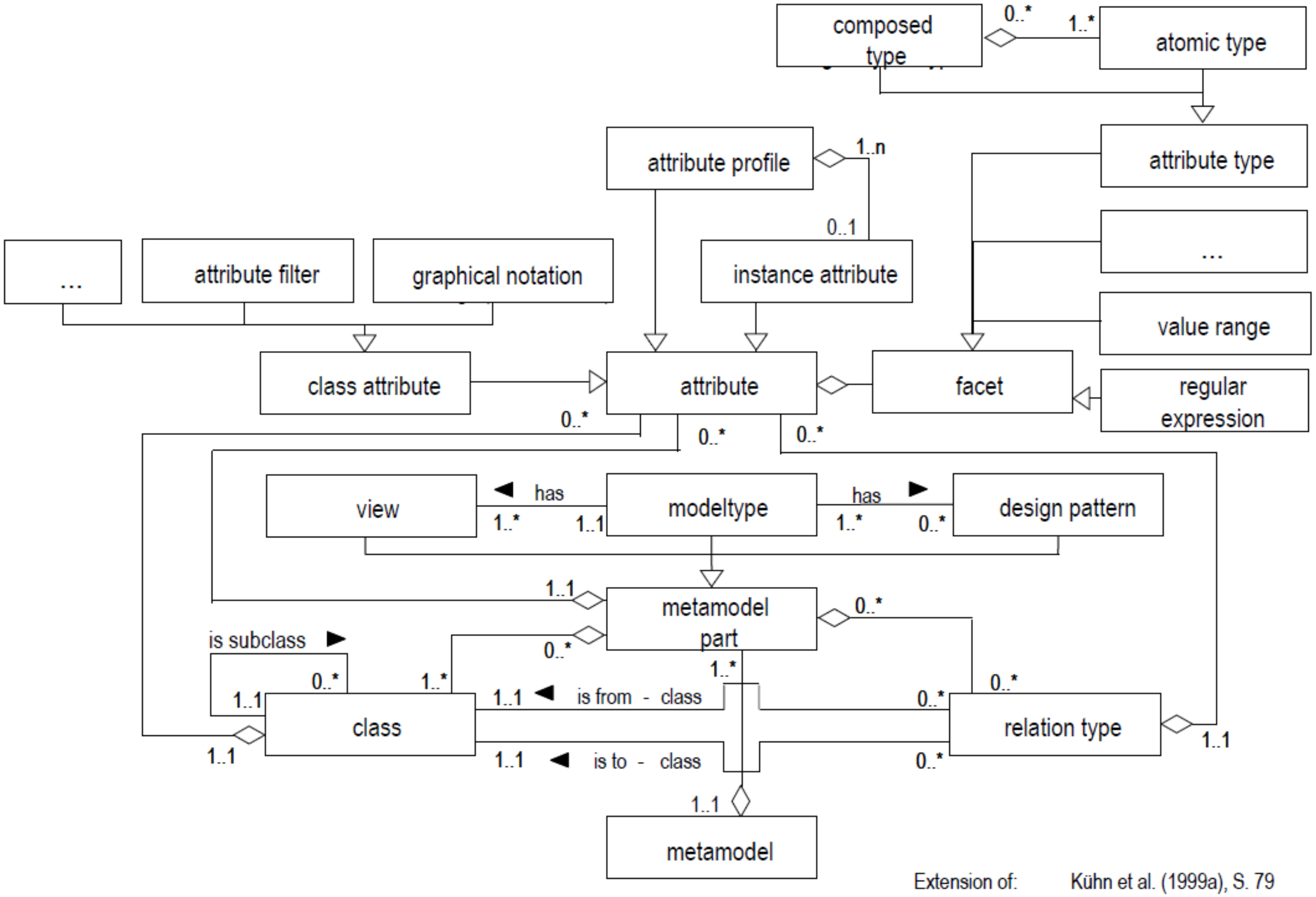
```
MODELTYPE "Teaching" from:none
```

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

Meta Modelling Platforms Hierarchyin ADO*xx*

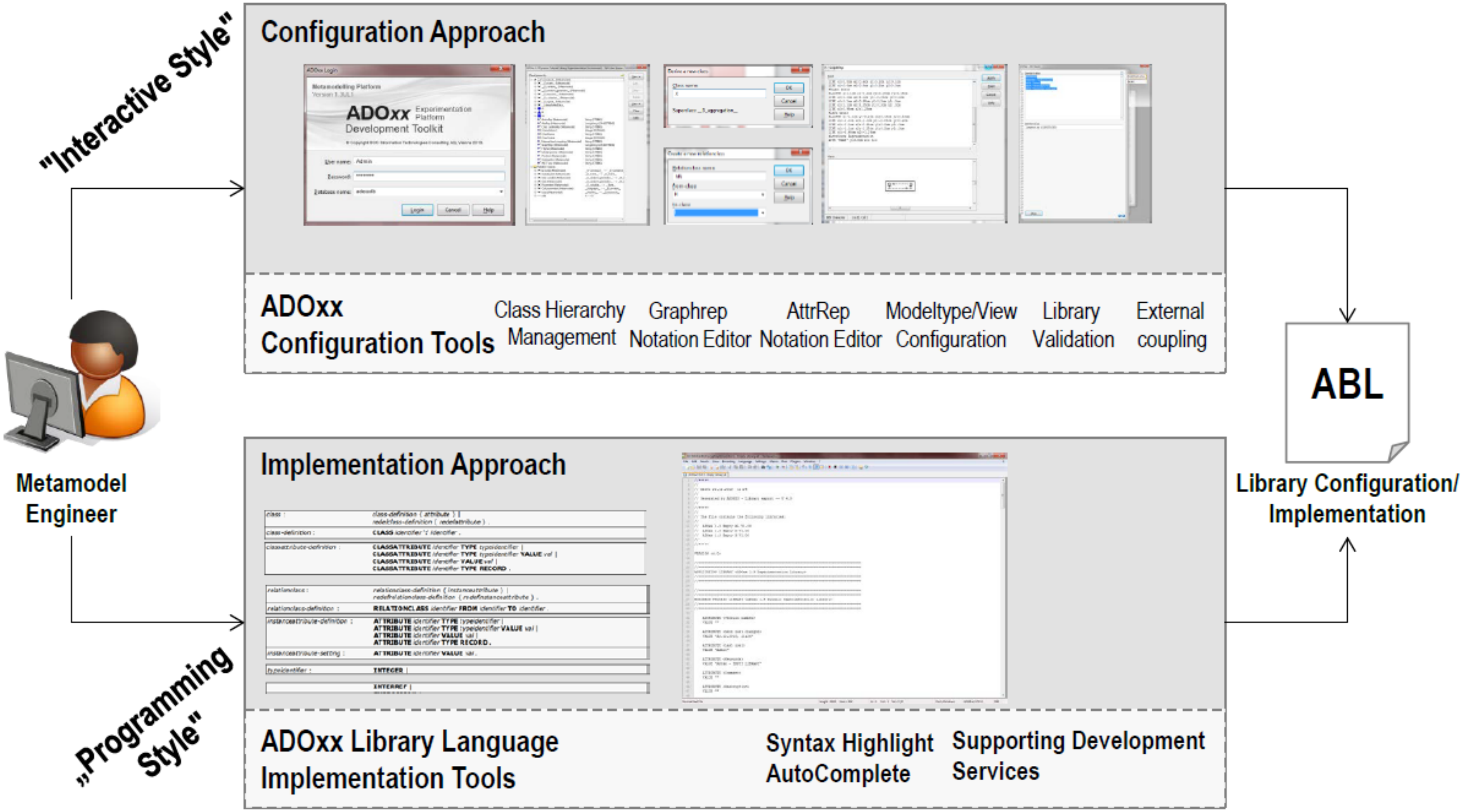


Meta^2 Model: Meta Model of Meta Modelling Language



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

Development Approaches in ADOxx – Configuration and Implementation



"Interactive Style"

Configuration Approach

ADOxx Configuration Tools

Class Hierarchy Management

Graphrep Notation Editor

AttrRep Notation Editor

Modeltype/View Configuration

Library Validation

External coupling

Metamodel Engineer

ABL

Library Configuration/ Implementation

„Programming Style"

Implementation Approach

class :	*class-definition { attribute } |* *redefclass-definition { redefattribute } .*
class-definition :	**CLASS** *identifier ':' identifier .*
classattribute-definition :	**CLASSATTRIBUTE** *identifier* **TYPE** *typeidentifier |* **CLASSATTRIBUTE** *identifier* **TYPE** *typeidentifier* **VALUE** *val |* **CLASSATTRIBUTE** *identifier* **VALUE** *val |* **CLASSATTRIBUTE** *identifier* **TYPE RECORD** .
relationclass :	*relationclass-definition { instanceattribute } |* *redefrelationclass-definition { redefinstanceattribute } .*
relationclass-definition :	**RELATIONCLASS** *identifier* **FROM** *identifier* **TO** *identifier .*
instanceattribute-definition :	**ATTRIBUTE** *identifier* **TYPE** *typeidentifier |* **ATTRIBUTE** *identifier* **TYPE** *typeidentifier* **VALUE** *val |* **ATTRIBUTE** *identifier* **VALUE** *val |* **ATTRIBUTE** *identifier* **TYPE RECORD.**
instanceattribute-setting :	**ATTRIBUTE** *identifier* **VALUE** *val .*
typeidentifier :	**INTEGER** |
	INTERREF |

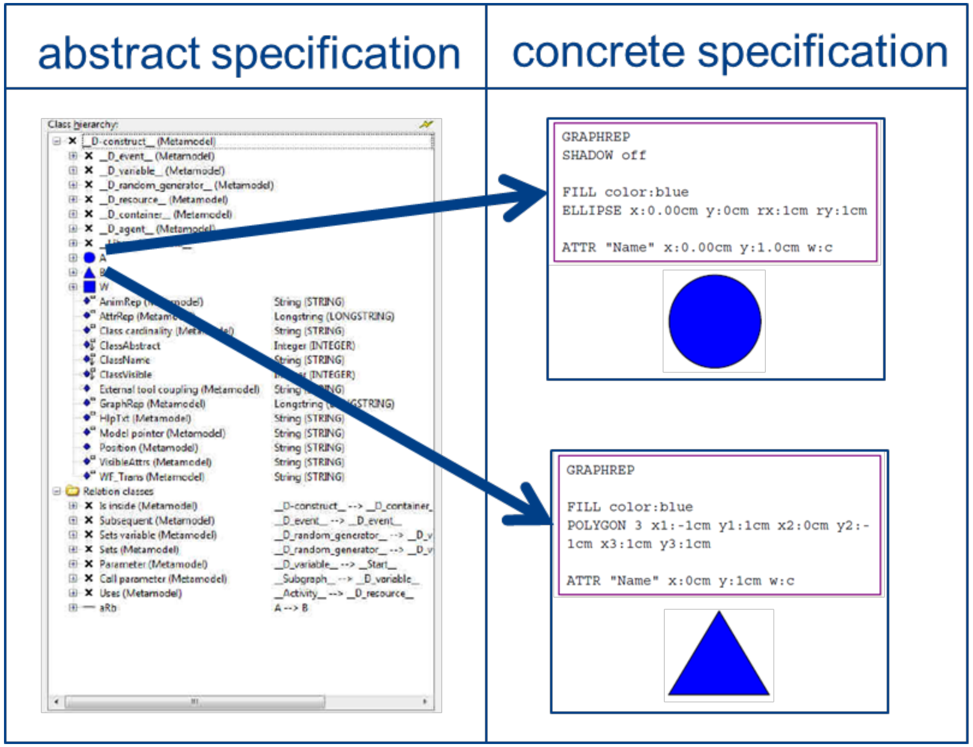
ADOxx Library Language Implementation Tools

Syntax Highlight AutoComplete

Supporting Development Services

Abstract and Concrete Specification

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



abstract specification	concrete specification

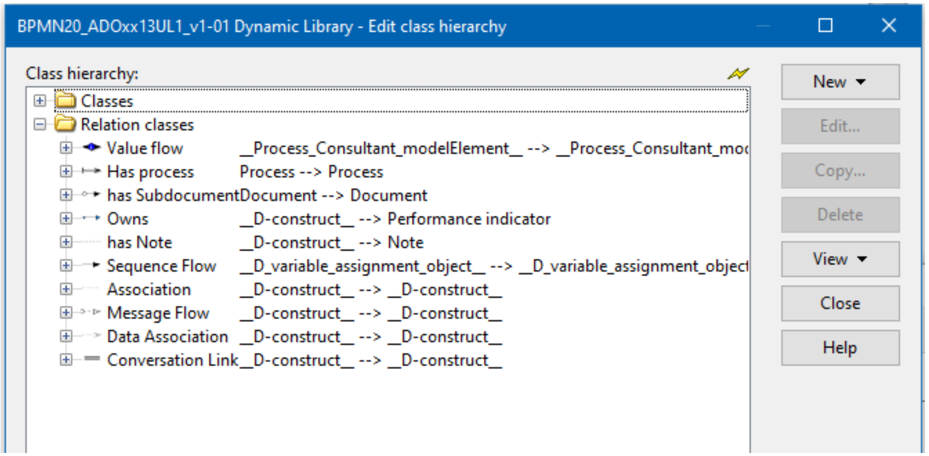
```
Class hierarchy:
⊟ ✕ __D-construct__ (Metamodel)
   ⊞ ✕ __D_event__ (Metamodel)
   ⊞ ✕ __D_variable__ (Metamodel)
   ⊞ ✕ __D_random_generator__ (Metamodel)
   ⊞ ✕ __D_resource__ (Metamodel)
   ⊞ ✕ __D_container__ (Metamodel)
   ⊞ ✕ __D_agent__ (Metamodel)
   ⊞ ✕ [illegible]
   ⊞ ● A
   ⊞ ▲ B
   ⊞ ■ W
     ◆ AnimRep (Metamodel)             String (STRING)
     ◆ AttrRep (Metamodel)             Longstring (LONGSTRING)
     ◆ Class cardinality (Metamodel)   String (STRING)
     ◆ ClassAbstract                   Integer (INTEGER)
     ◆ ClassName                       String (STRING)
     ◆ ClassVisible                    Integer (INTEGER)
     ◆ External tool coupling (Metamodel) String (STRING)
     ◆ GraphRep (Metamodel)            Longstring (LONGSTRING)
     ◆ HlpTxt (Metamodel)              String (STRING)
     ◆ Model pointer (Metamodel)       String (STRING)
     ◆ Position (Metamodel)            String (STRING)
     ◆ VisibleAttrs (Metamodel)        String (STRING)
     ◆ WF_Trans (Metamodel)            String (STRING)
⊟ Relation classes
   ⊞ ✕ Is inside (Metamodel)           __D-construct__ --> __D_container_
   ⊞ ✕ Subsequent (Metamodel)          __D_event__ --> __D_event__
   ⊞ ✕ Sets variable (Metamodel)       __D_random_generator__ --> __D_v
   ⊞ ✕ Sets (Metamodel)                __D_random_generator__ --> __D_v
   ⊞ ✕ Parameter (Metamodel)           __D_variable__ --> __Start__
   ⊞ ✕ Call parameter (Metamodel)      __Subgraph__ --> __D_variable__
   ⊞ ✕ Uses (Metamodel)                __Activity__ --> __D_resource__
   ⊞ — aRb                             A --> B
```

```
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:blue
POLYGON 3 x1:-1cm y1:1cm x2:0cm y2:-
1cm x3:1cm y3:1cm

ATTR "Name" x:0cm y:1cm w:c
```

Class Hierarchies

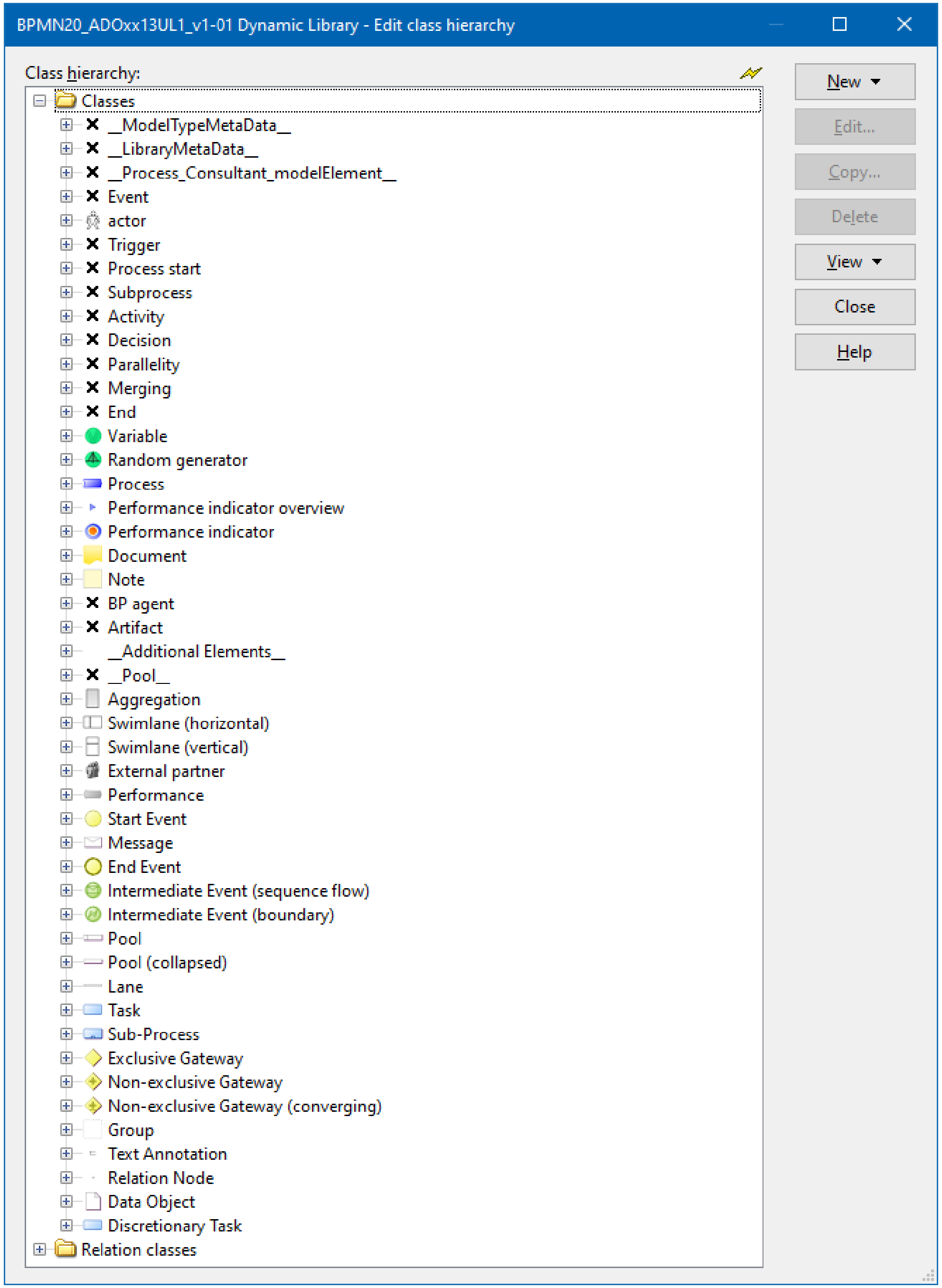
- ADOxx distinguishes
  - Classes
  - Relation classes



BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

Class hierarchy:

- Classes
- Relation classes
  - Value flow _Process_Consultant_modelElement_ --> _Process_Consultant_mod
  - Has process Process --> Process
  - has SubdocumentDocument --> Document
  - Owns _D-construct_ --> Performance indicator
  - has Note _D-construct_ --> Note
  - Sequence Flow _D_variable_assignment_object_ --> _D_variable_assignment_object
  - Association _D-construct_ --> _D-construct_
  - Message Flow _D-construct_ --> _D-construct_
  - Data Association _D-construct_ --> _D-construct_
  - Conversation Link _D-construct_ --> _D-construct_



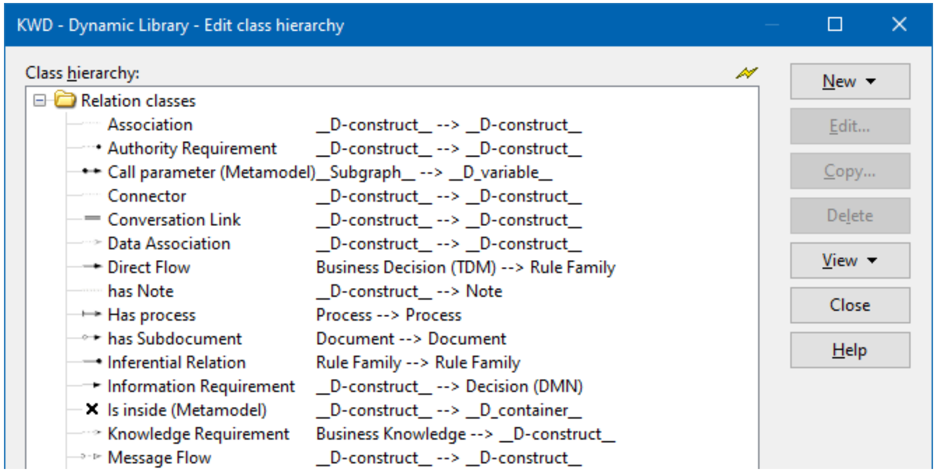
BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

Class hierarchy:

- Classes
  - _ModelTypeMetaData_
  - _LibraryMetaData_
  - _Process_Consultant_modelElement_
  - Event
  - actor
  - Trigger
  - Process start
  - Subprocess
  - Activity
  - Decision
  - Parallelity
  - Merging
  - End
  - Variable
  - Random generator
  - Process
  - Performance indicator overview
  - Performance indicator
  - Document
  - Note
  - BP agent
  - Artifact
  - _Additional Elements_
  - _Pool_
  - Aggregation
  - Swimlane (horizontal)
  - Swimlane (vertical)
  - External partner
  - Performance
  - Start Event
  - Message
  - End Event
  - Intermediate Event (sequence flow)
  - Intermediate Event (boundary)
  - Pool
  - Pool (collapsed)
  - Lane
  - Task
  - Sub-Process
  - Exclusive Gateway
  - Non-exclusive Gateway
  - Non-exclusive Gateway (converging)
  - Group
  - Text Annotation
  - Relation Node
  - Data Object
  - Discretionary Task
- Relation classes

Class Hierarchies

- ADOxx distinguishes
  - Classes
  - Relation classes

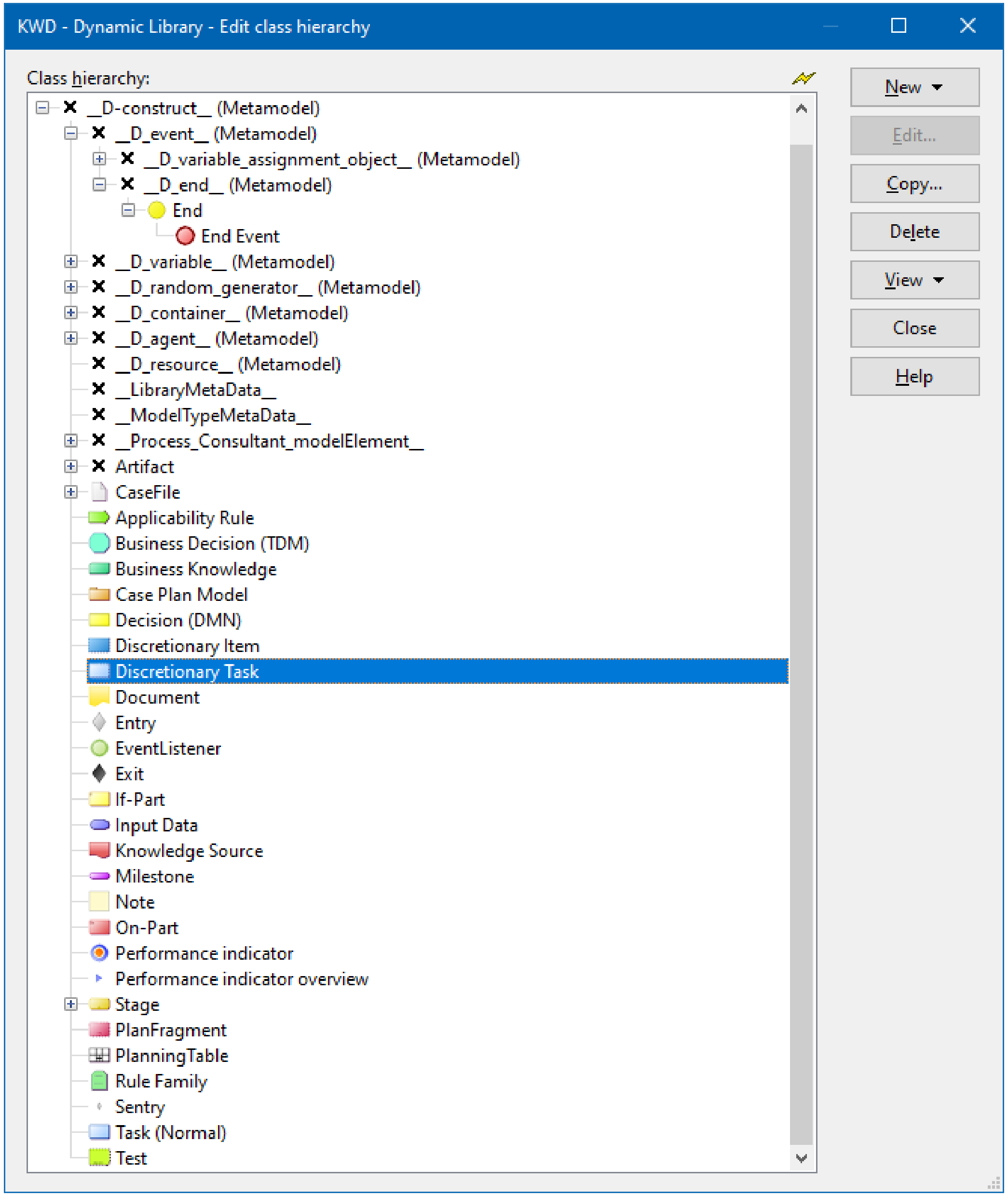


KWD - Dynamic Library - Edit class hierarchy

Class hierarchy:

Relation classes

Association	_D-construct_ --> _D-construct_
Authority Requirement	_D-construct_ --> _D-construct_
Call parameter (Metamodel)	_Subgraph_ --> _D_variable_
Connector	_D-construct_ --> _D-construct_
Conversation Link	_D-construct_ --> _D-construct_
Data Association	_D-construct_ --> _D-construct_
Direct Flow	Business Decision (TDM) --> Rule Family
has Note	_D-construct_ --> Note
Has process	Process --> Process
has Subdocument	Document --> Document
Inferential Relation	Rule Family --> Rule Family
Information Requirement	_D-construct_ --> Decision (DMN)
Is inside (Metamodel)	_D-construct_ --> _D_container_
Knowledge Requirement	Business Knowledge --> _D-construct_
Message Flow	_D-construct_ --> _D-construct_

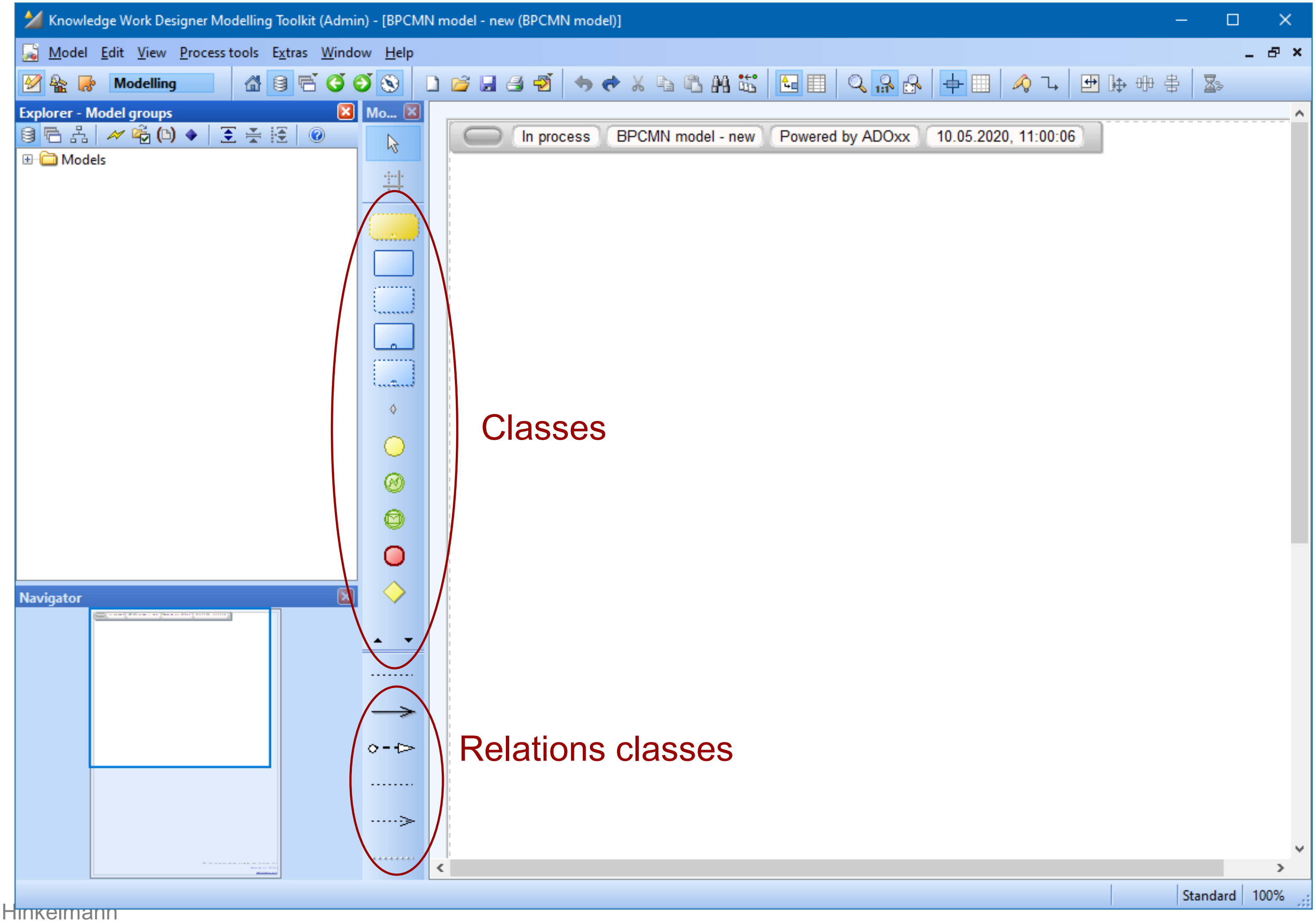


KWD - Dynamic Library - Edit class hierarchy

Class hierarchy:

- _D-construct_ (Metamodel)
  - _D_event_ (Metamodel)
    - _D_variable_assignment_object_ (Metamodel)
    - _D_end_ (Metamodel)
      - End
        - End Event
  - _D_variable_ (Metamodel)
  - _D_random_generator_ (Metamodel)
  - _D_container_ (Metamodel)
  - _D_agent_ (Metamodel)
  - _D_resource_ (Metamodel)
  - _LibraryMetaData_
  - _ModelTypeMetaData_
  - _Process_Consultant_modelElement_
  - Artifact
  - CaseFile
  - Applicability Rule
  - Business Decision (TDM)
  - Business Knowledge
  - Case Plan Model
  - Decision (DMN)
  - Discretionary Item
  - Discretionary Task
  - Document
  - Entry
  - EventListener
  - Exit
  - If-Part
  - Input Data
  - Knowledge Source
  - Milestone
  - Note
  - On-Part
  - Performance indicator
  - Performance indicator overview
  - Stage
  - PlanFragment
  - PlanningTable
  - Rule Family
  - Sentry
  - Task (Normal)
  - Test

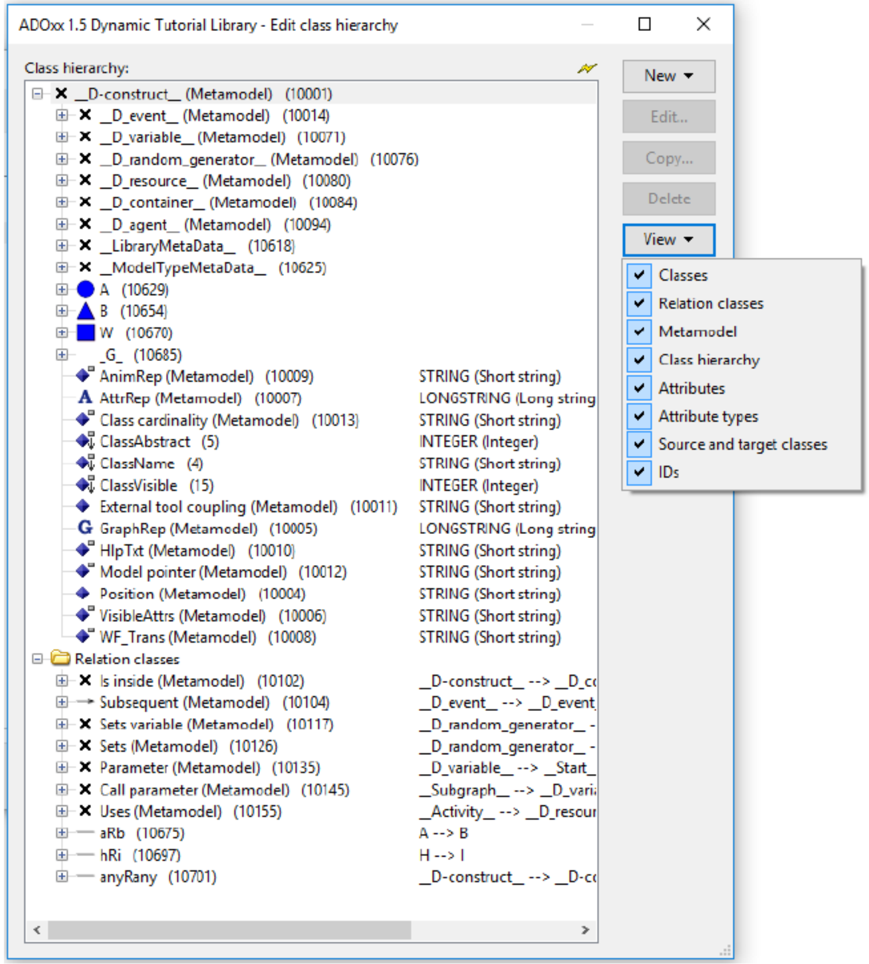
Appearance of Classes in the Modelling Toolkit



Classes

Relations classes

Views of the Class Hierarchy



ADOxx 1.5 Dynamic Tutorial Library - Edit class hierarchy

Class hierarchy:

```
⊟ ✕ __D-construct__ (Metamodel)  (10001)
   ⊞ ✕ __D_event__ (Metamodel)  (10014)
   ⊞ ✕ __D_variable__ (Metamodel)  (10071)
   ⊞ ✕ __D_random_generator__ (Metamodel)  (10076)
   ⊞ ✕ __D_resource__ (Metamodel)  (10080)
   ⊞ ✕ __D_container__ (Metamodel)  (10084)
   ⊞ ✕ __D_agent__ (Metamodel)  (10094)
   ⊞ ✕ __LibraryMetaData__  (10618)
   ⊞ ✕ __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
   ⊞ ✕ Is inside (Metamodel)  (10102)                 __D-construct__ --> __D_c
   ⊞ → Subsequent (Metamodel)  (10104)                __D_event__ --> __D_event
   ⊞ ✕ Sets variable (Metamodel)  (10117)             __D_random_generator__ -
   ⊞ ✕ Sets (Metamodel)  (10126)                      __D_random_generator__ -
   ⊞ ✕ Parameter (Metamodel)  (10135)                 __D_variable__ --> __Start_
   ⊞ ✕ Call parameter (Metamodel)  (10145)            __Subgraph__ --> __D_vari
   ⊞ ✕ Uses (Metamodel)  (10155)                      __Activity__ --> __D_resour
   ⊞ — aRb  (10675)                                   A --> B
   ⊞ — hRi  (10697)                                   H --> I
   ⊞ — anyRany  (10701)                               __D-construct__ --> __D-c
```

☑ 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.

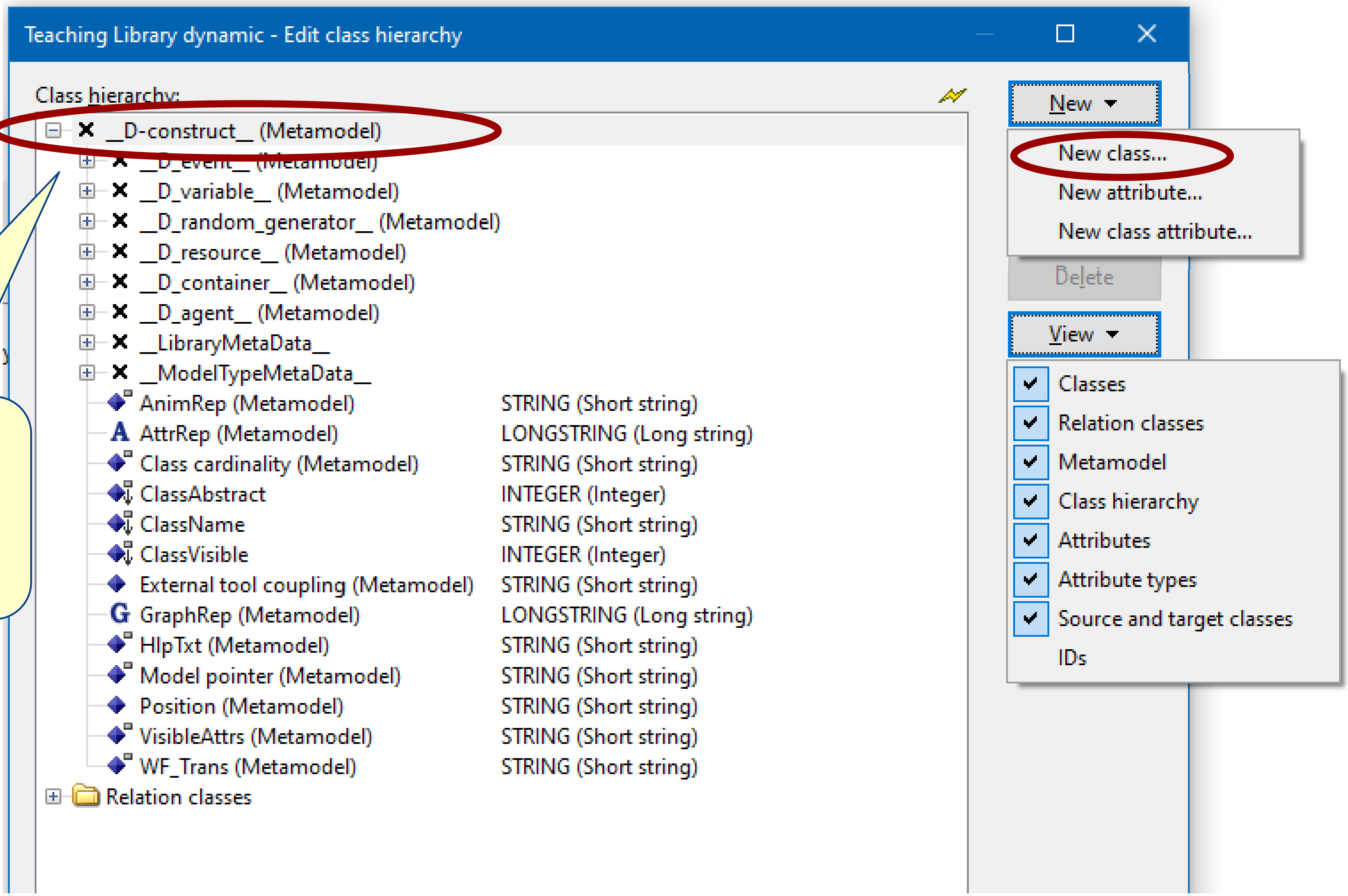
IDs

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



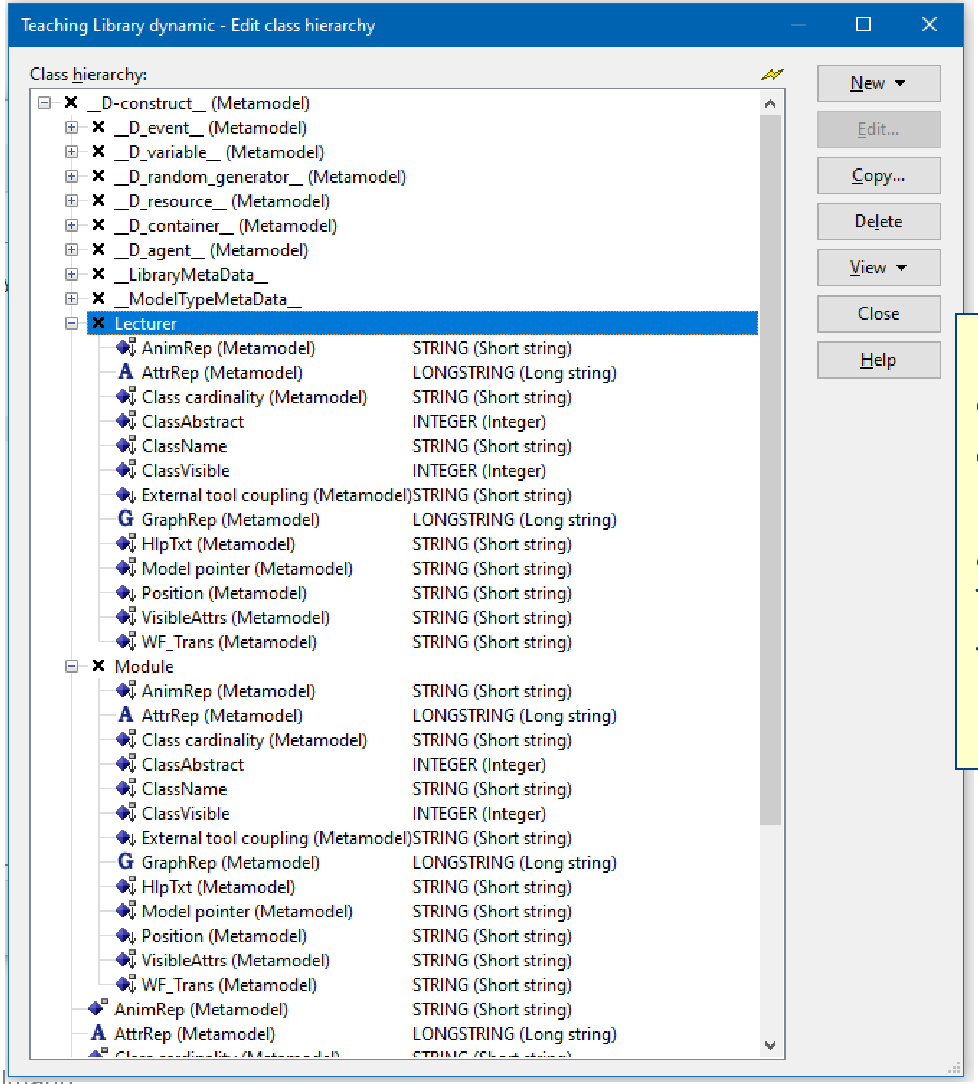
Teaching Library dynamic - Edit class hierarchy

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)
     A 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)
     G 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
```

There are predefined abstract classes which have specific functionaliy

New Classes for Lecturer and Module



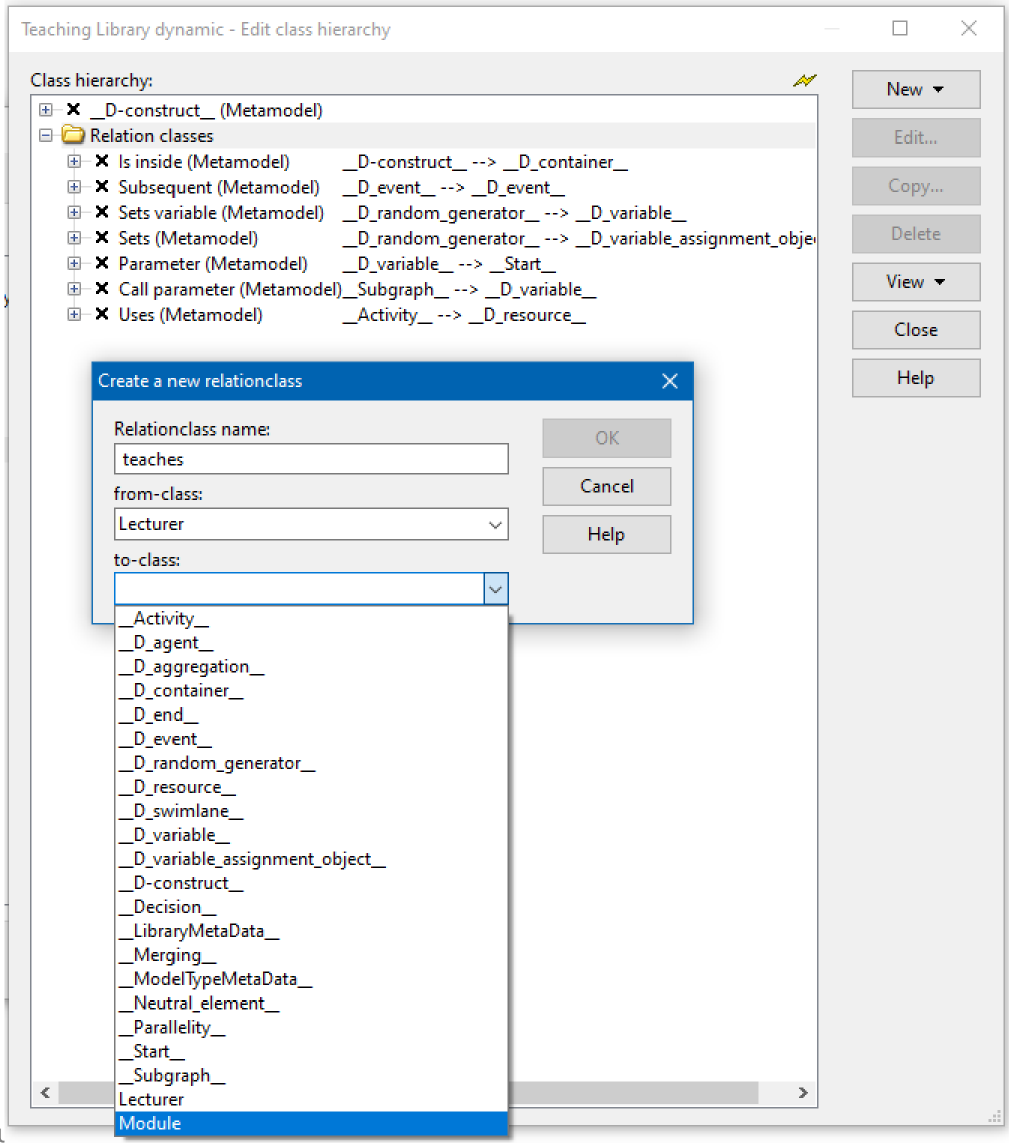
Teaching Library dynamic - Edit class hierarchy

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_
    ⊟ ✕ Lecturer
          AnimRep (Metamodel)                 STRING (Short string)
        A 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)
        G 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)
    ⊟ ✕ Module
          AnimRep (Metamodel)                 STRING (Short string)
        A 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)
        G 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)
    A AttrRep (Metamodel)                     LONGSTRING (Long string)
      Class cardinality (Metamodel)           STRING (Short string)
```

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:

- ☒ __D-construct__ (Metamodel)
- Relation classes
  - ☒ Is inside (Metamodel) __D-construct__ --> __D_container__
  - ☒ Subsequent (Metamodel) __D_event__ --> __D_event__
  - ☒ Sets variable (Metamodel) __D_random_generator__ --> __D_variable__
  - ☒ Sets (Metamodel) __D_random_generator__ --> __D_variable_assignment_obje
  - ☒ Parameter (Metamodel) __D_variable__ --> __Start__
  - ☒ Call parameter (Metamodel)__Subgraph__ --> __D_variable__
  - ☒ Uses (Metamodel) __Activity__ --> __D_resource__

Create a new relationclass

Relationclass name: teaches

from-class: Lecturer

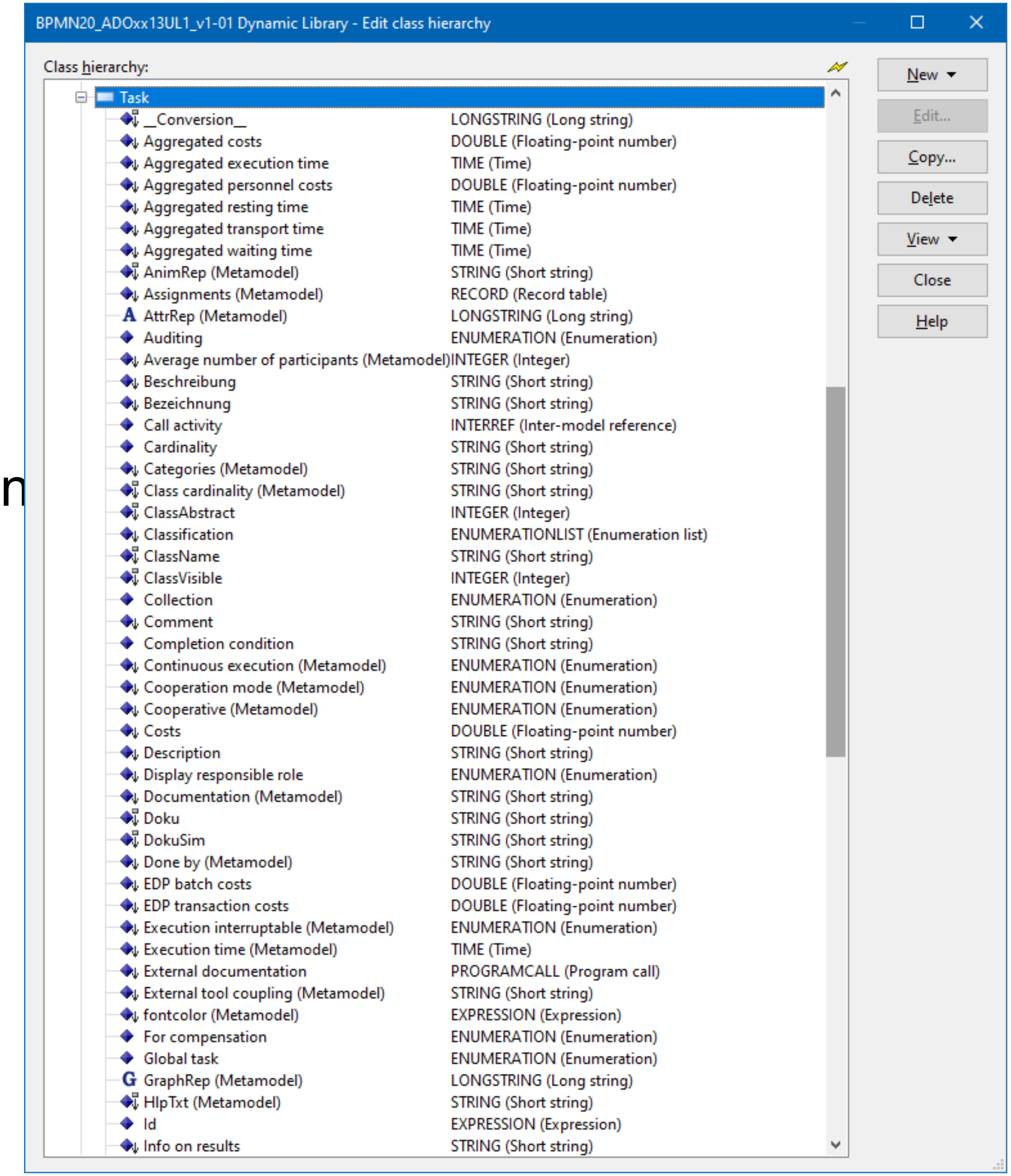
to-class:

```
__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__
__Parallelity__
__Start__
__Subgraph__
Lecturer
Module
```

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

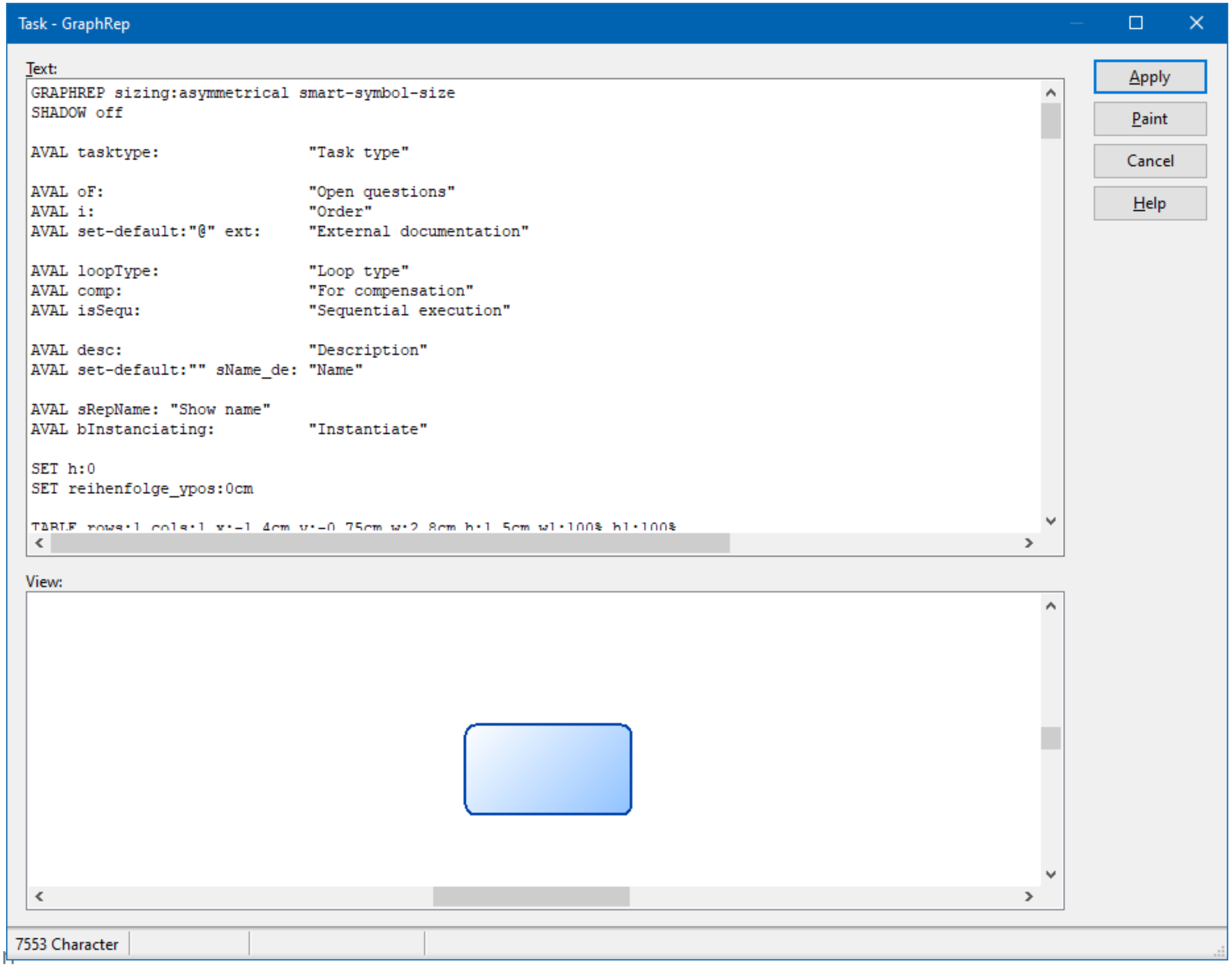
Class hierarchy:

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)

Special Attribute GraphRep

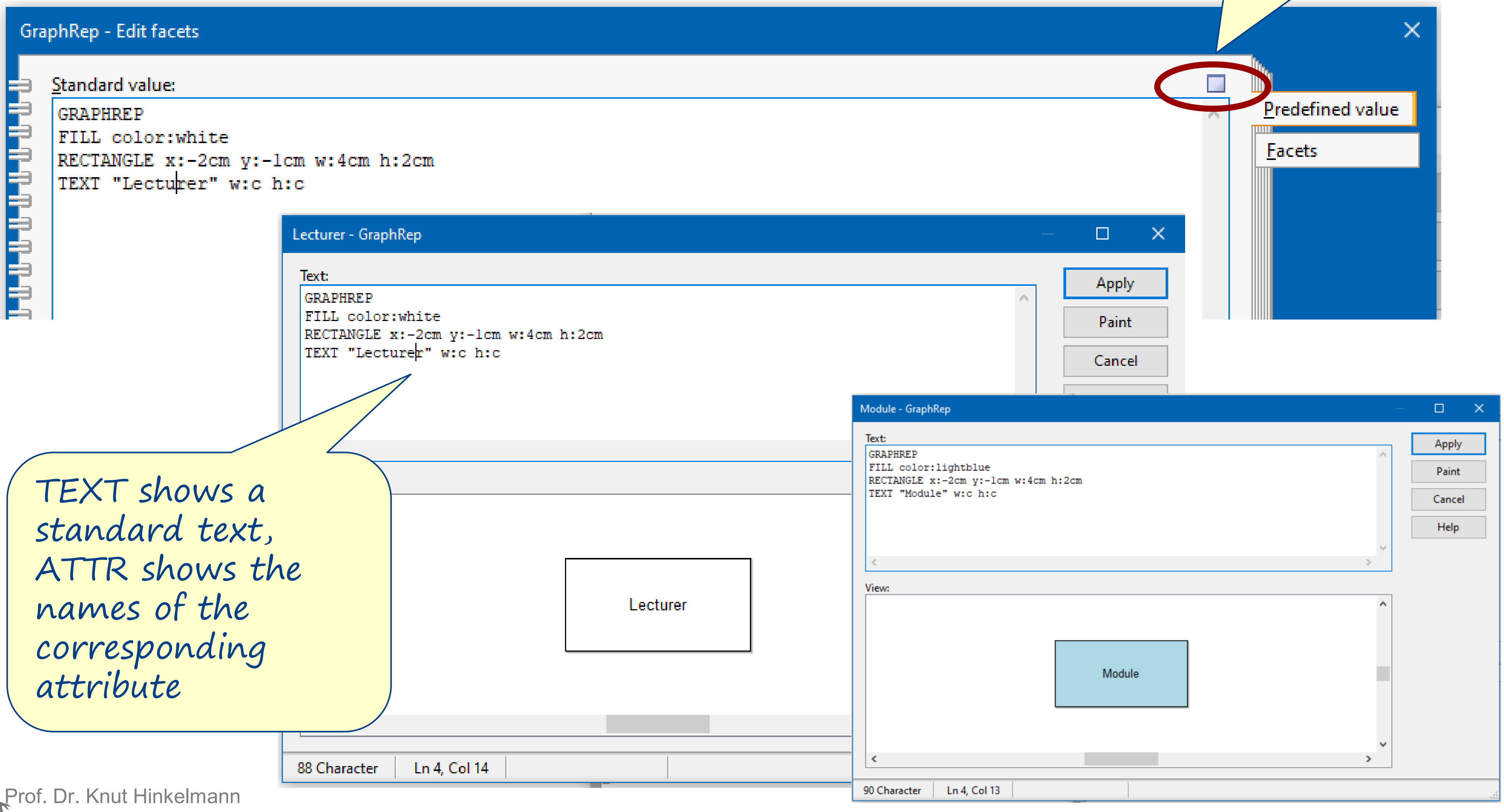
GraphRep: A script language for the graphical representation



```
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%
```

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

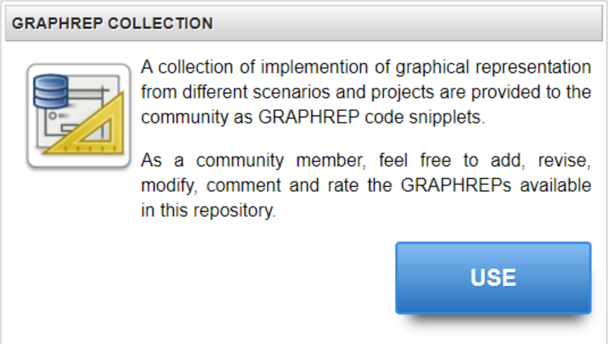
Text:

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

View:

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

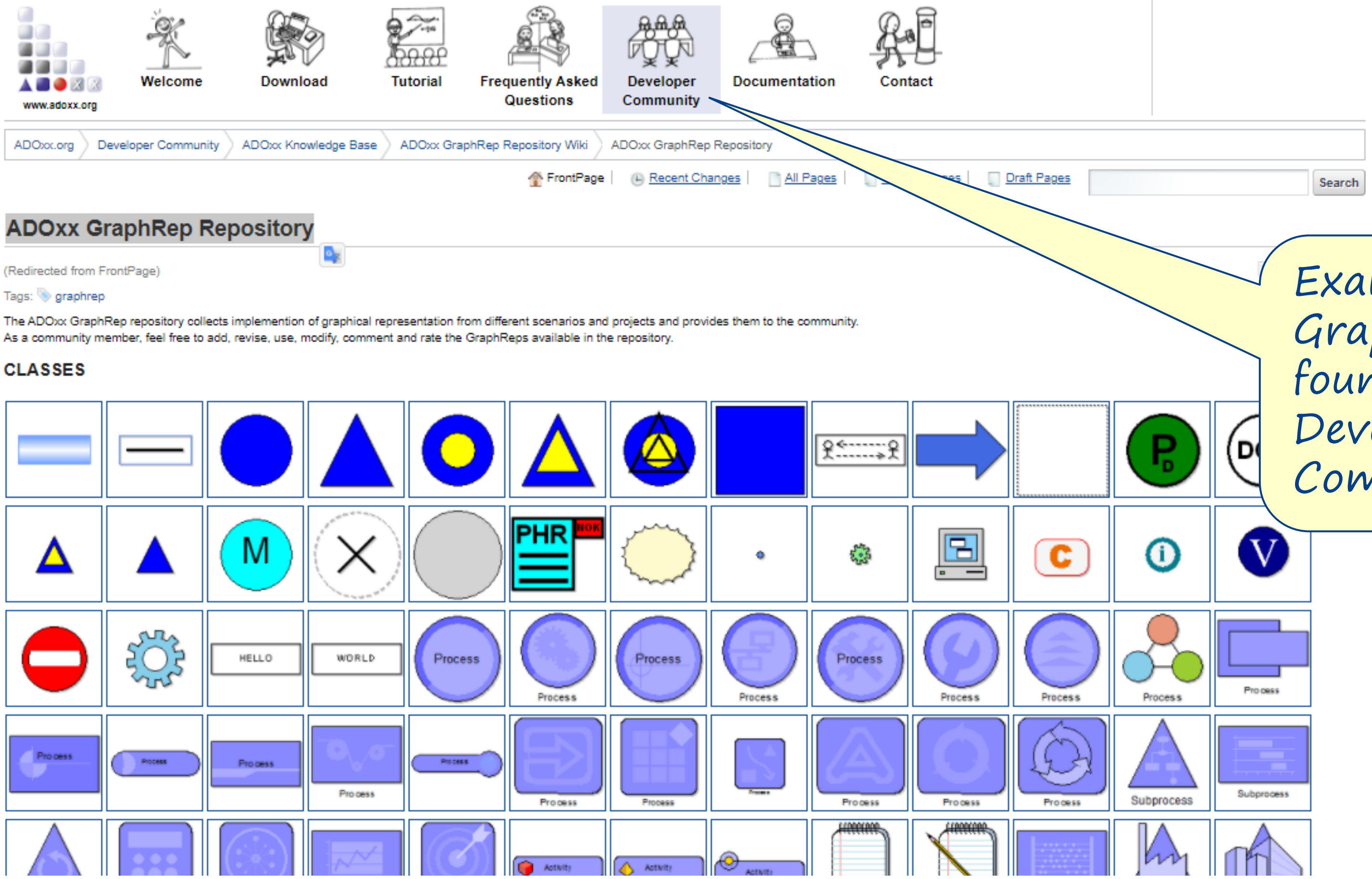
ADOxx GraphRep Repository



GRAPHREP COLLECTION

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

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



ADOxx GraphRep Repository

(Redirected from FrontPage)

Tags: graphrep

The ADOxx GraphRep repository collects implemention 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.

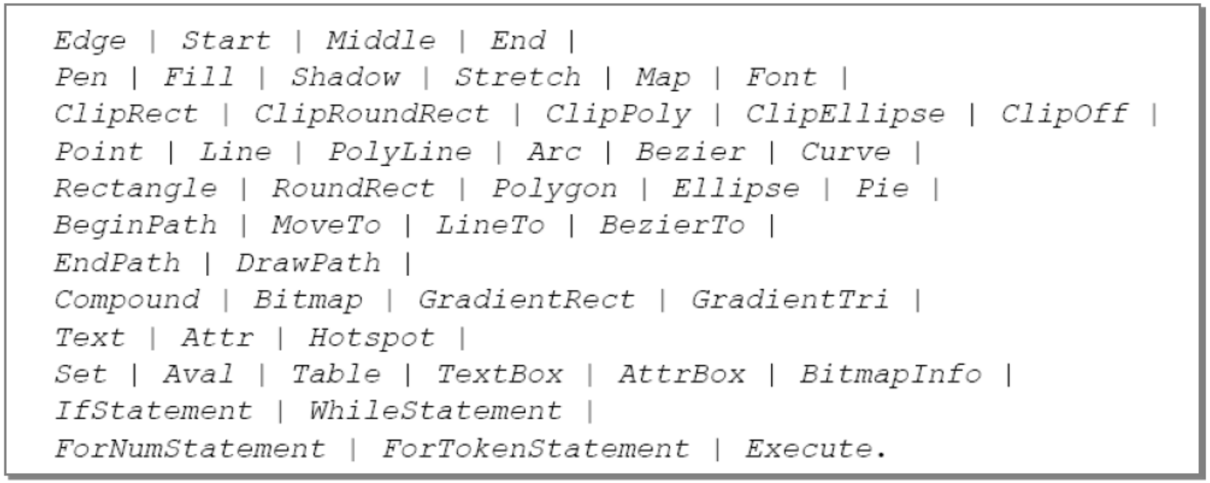
CLASSES

Examples of GraphReps can be found in the ADOxx Developer Community

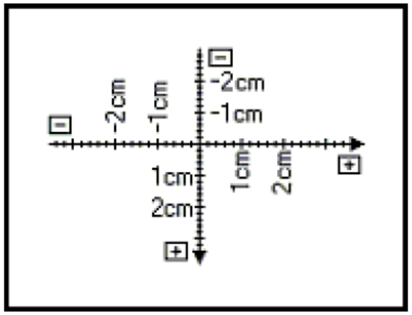
GraphRep Elements

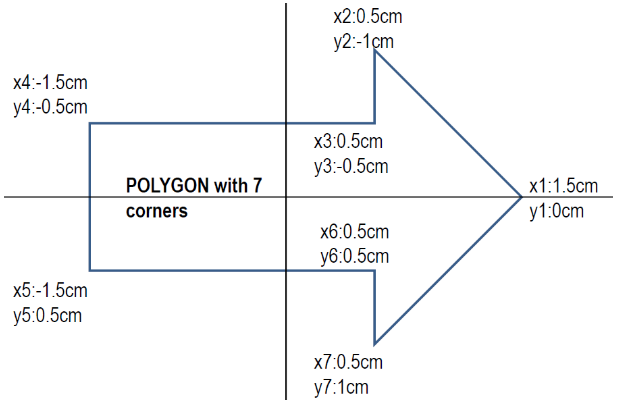
- Types of elements
  - Style elements
  - Shape elements
  - Variable assigning elements
  - Context elements
  - Control elements
- Elements are placed on x-y-axes

GraphRep Elements



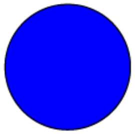
```
Edge | Start | Middle | End |
Pen | Fill | Shadow | Stretch | Map | Font |
ClipRect | ClipRoundRect | ClipPoly | ClipEllipse | ClipOff |
Point | Line | PolyLine | Arc | Bezier | Curve |
Rectangle | RoundRect | Polygon | Ellipse | Pie |
BeginPath | MoveTo | LineTo | BezierTo |
EndPath | DrawPath |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





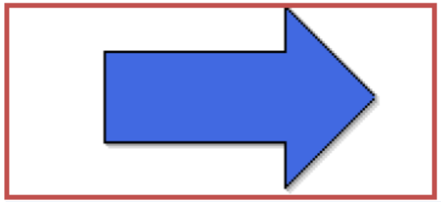
GraphRep Examples

```
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
```

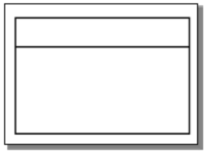
```
ATTR "Name" y:1.4cm w:c h:c
```



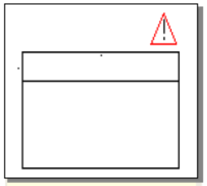
In case attribute name is available, it is shown here

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

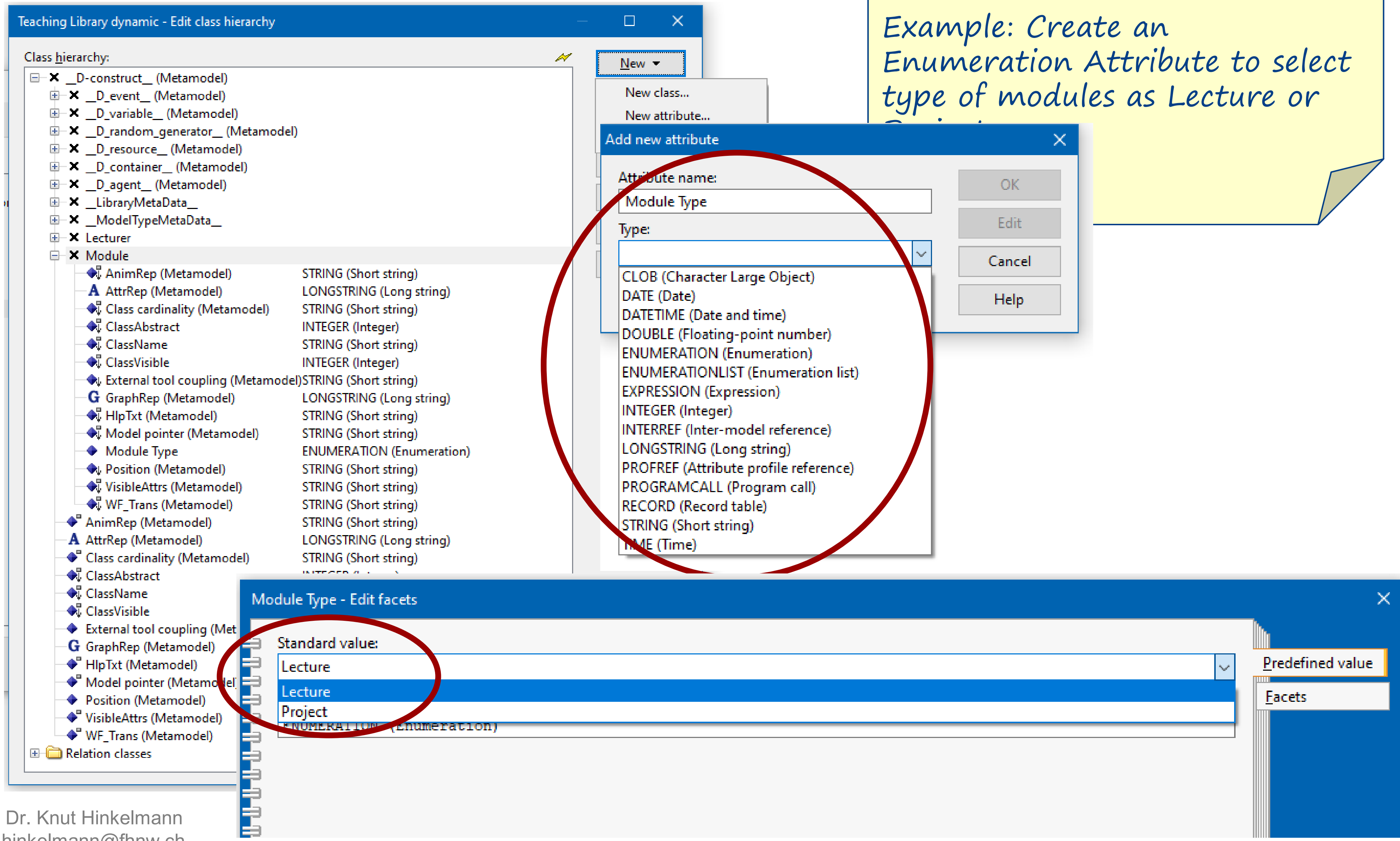


Condition not fulfilled

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:

```
⊟ ✕ _D-construct_ (Metamodel)
   ⊞ ✕ _D_event_ (Metamodel)
   ⊞ ✕ _D_variable_ (Metamodel)
   ⊞ ✕ _D_random_generator_ (Metamodel)
   ⊞ ✕ _D_resource_ (Metamodel)
   ⊞ ✕ _D_container_ (Metamodel)
   ⊞ ✕ _D_agent_ (Metamodel)
   ⊞ ✕ _LibraryMetaData_
   ⊞ ✕ _ModelTypeMetaData_
   ⊞ ✕ Lecturer
   ⊟ ✕ 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 (Met
     GraphRep (Metamodel)
     HlpTxt (Metamodel)
     Model pointer (Metamodel)
     Position (Metamodel)
     VisibleAttrs (Metamodel)
     WF_Trans (Metamodel)
⊞ Relation classes
```

Add new attribute

Attribute name: Module Type

Type:

- CLOB (Character Large Object)
- DATE (Date)
- DATETIME (Date and time)
- DOUBLE (Floating-point number)
- ENUMERATION (Enumeration)
- ENUMERATIONLIST (Enumeration list)
- EXPRESSION (Expression)
- INTEGER (Integer)
- INTERREF (Inter-model reference)
- LONGSTRING (Long string)
- PROFREF (Attribute profile reference)
- PROGRAMCALL (Program call)
- RECORD (Record table)
- STRING (Short string)
- TIME (Time)

Module Type - Edit facets

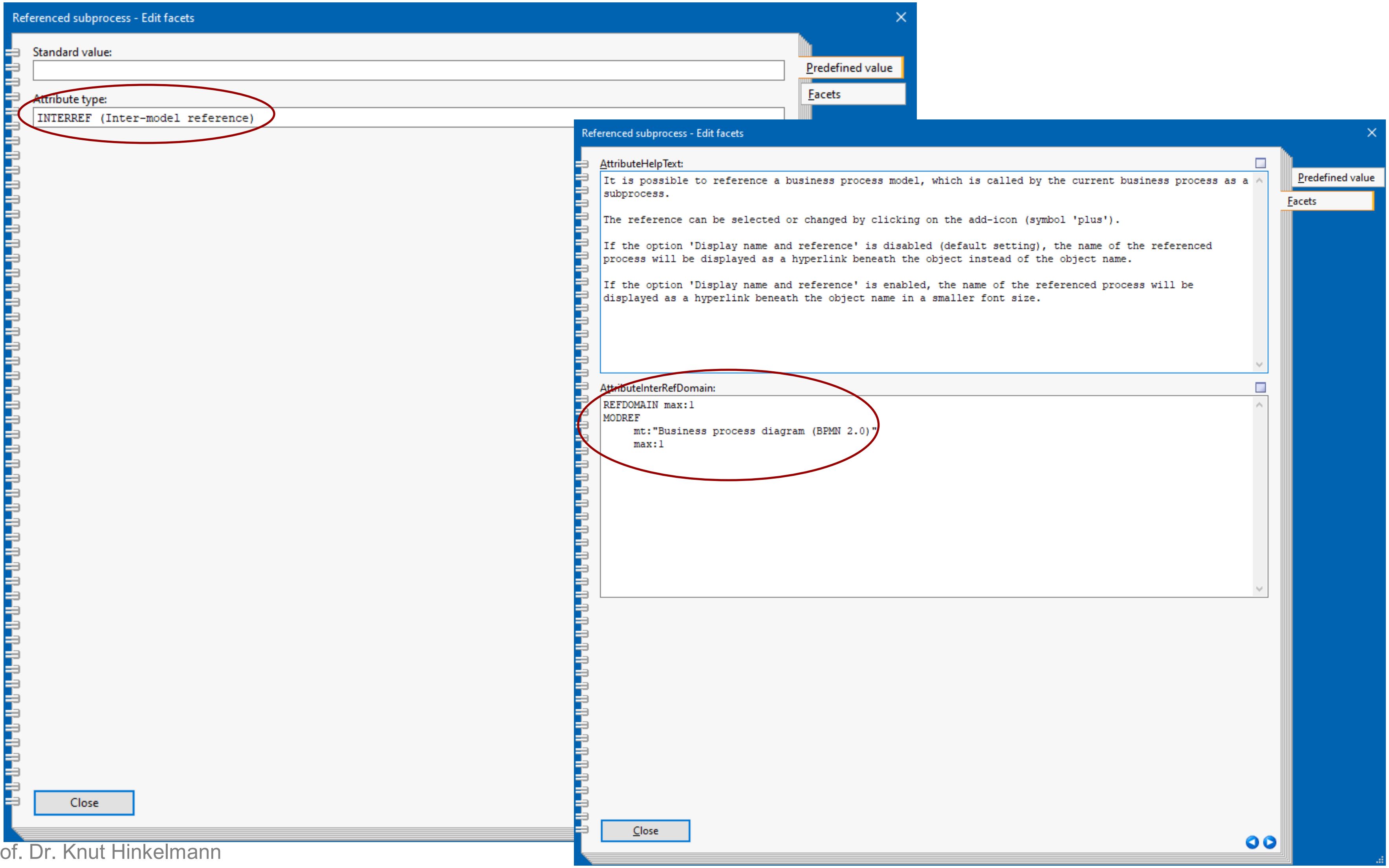
Standard value: Lecture

- Lecture
- Project

ENUMERATION (Enumeration)

References

Referencing a Subprocess



Referenced subprocess - Edit facets
Predefined value
Facets
Close

Standard value:

Attribute type:

```
INTERREF (Inter-model reference)
```

Referenced subprocess - Edit facets
Predefined value
Facets
Close

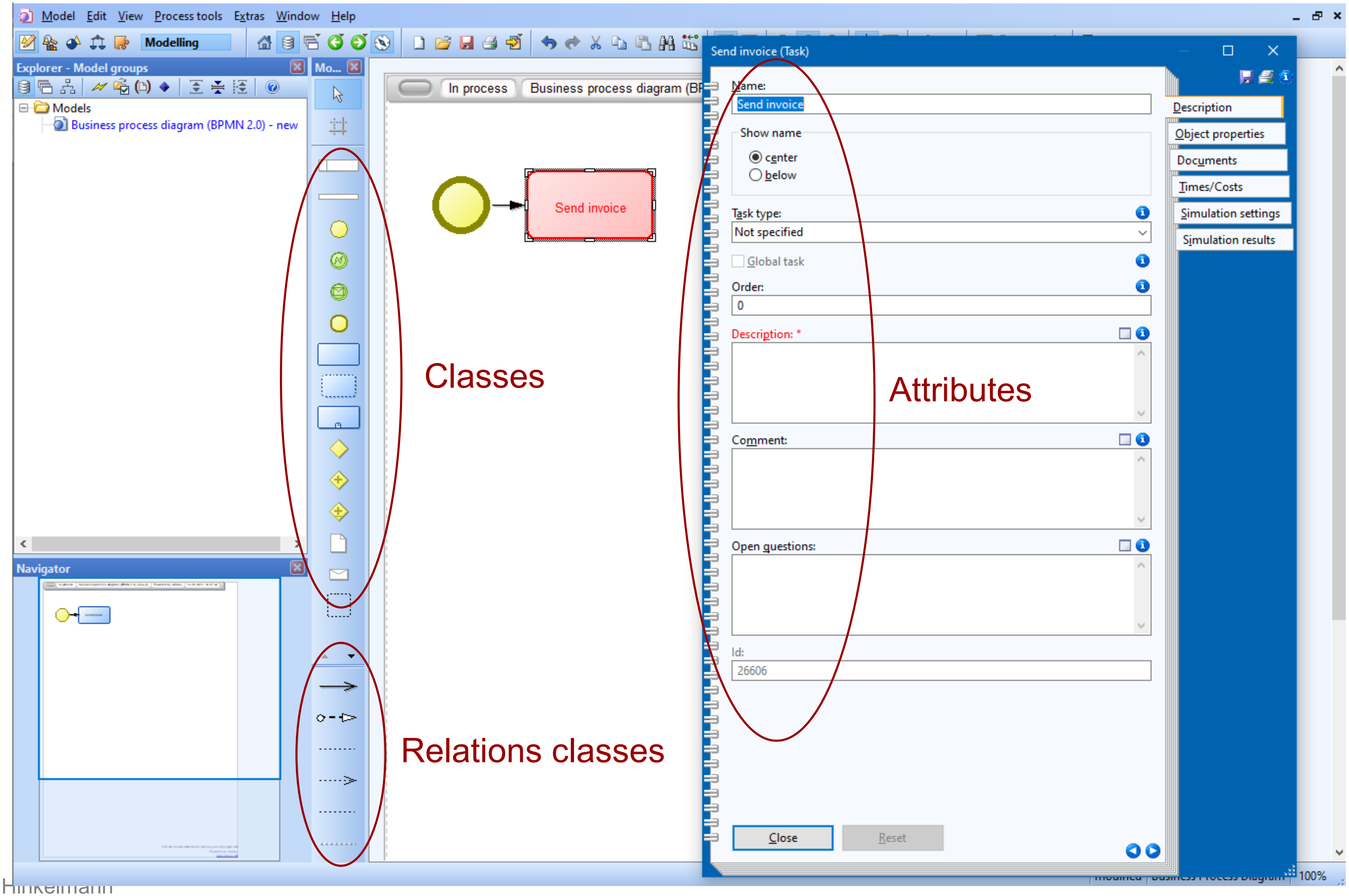
AttributeHelpText:

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

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

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

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

AttributeInterRefDomain:

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

Appearance of Classes in the Modelling Toolkit



Name: Send invoice

Show name
● center
○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

Open questions:

Id: 26606

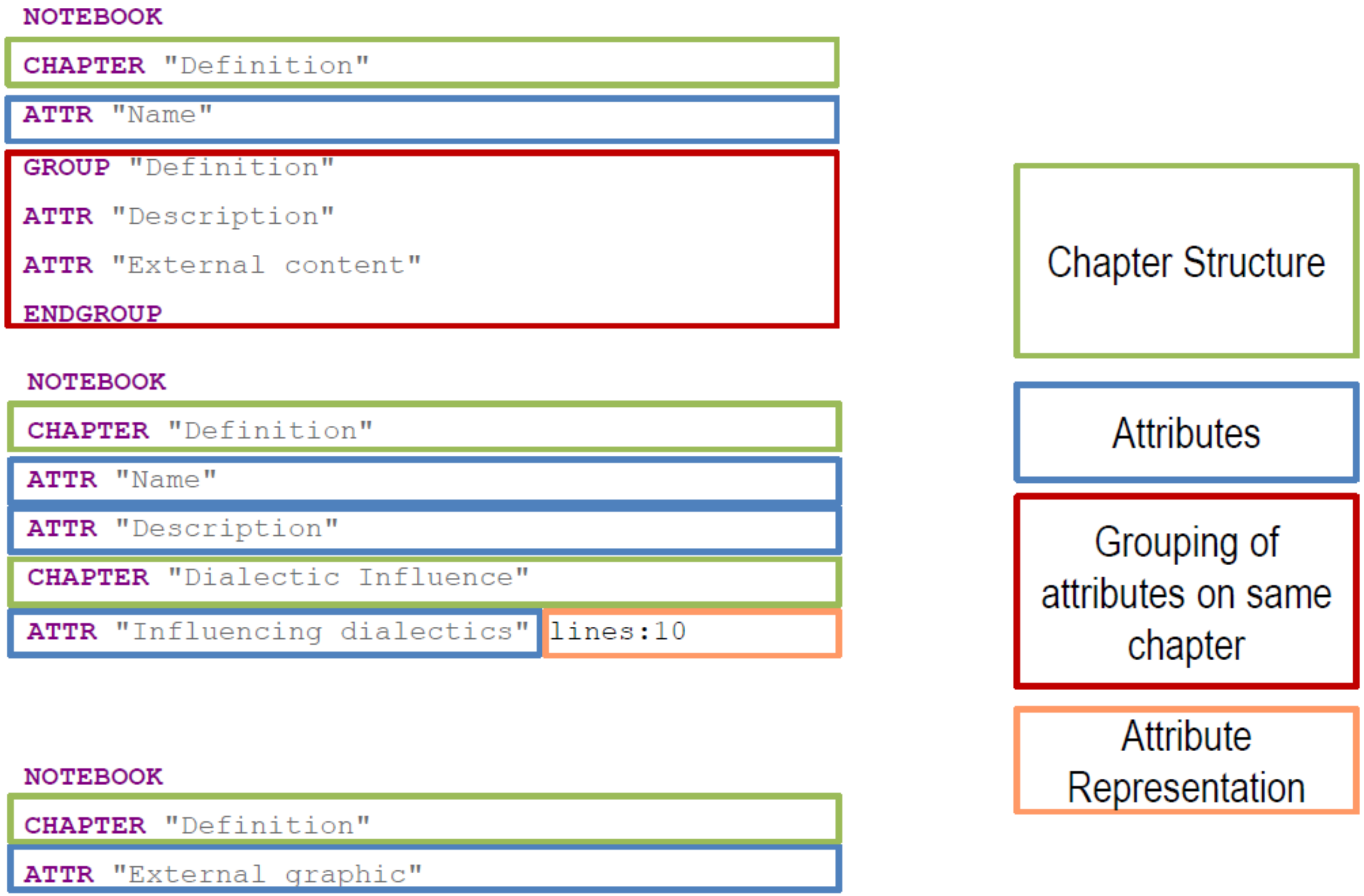
Classes

Attributes

Relations classes

AttrRep

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



```
NOTEBOOK
CHAPTER "Definition"
ATTR "Name"
GROUP "Definition"
ATTR "Description"
ATTR "External content"
ENDGROUP
```

```
NOTEBOOK
CHAPTER "Definition"
ATTR "Name"
ATTR "Description"
CHAPTER "Dialectic Influence"
ATTR "Influencing dialectics" lines:10
```

```
NOTEBOOK
CHAPTER "Definition"
ATTR "External graphic"
```

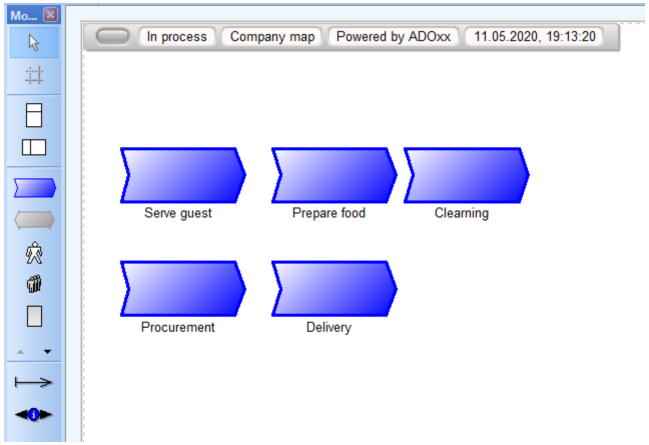
Chapter Structure

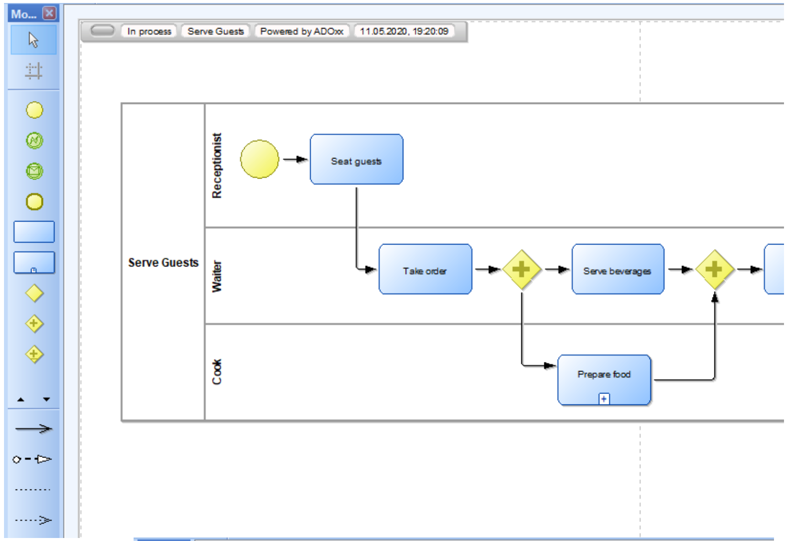
Attributes

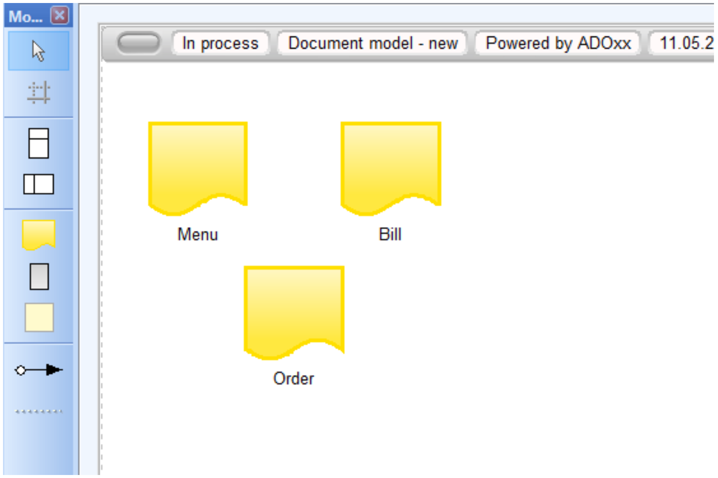
Grouping of attributes on same chapter

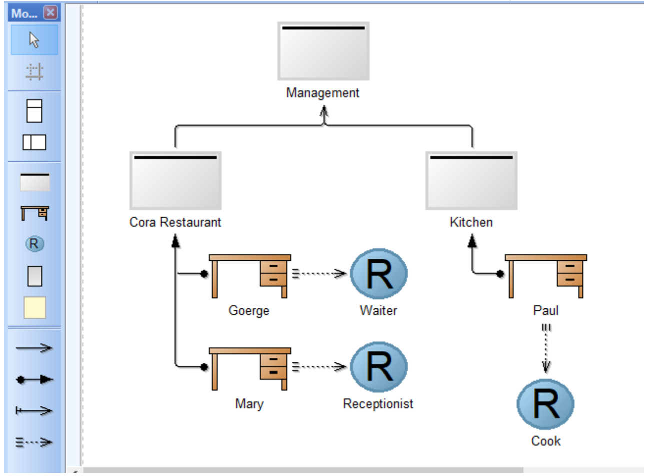
Attribute Representation

Model Types: Represention Views on the Knowledge

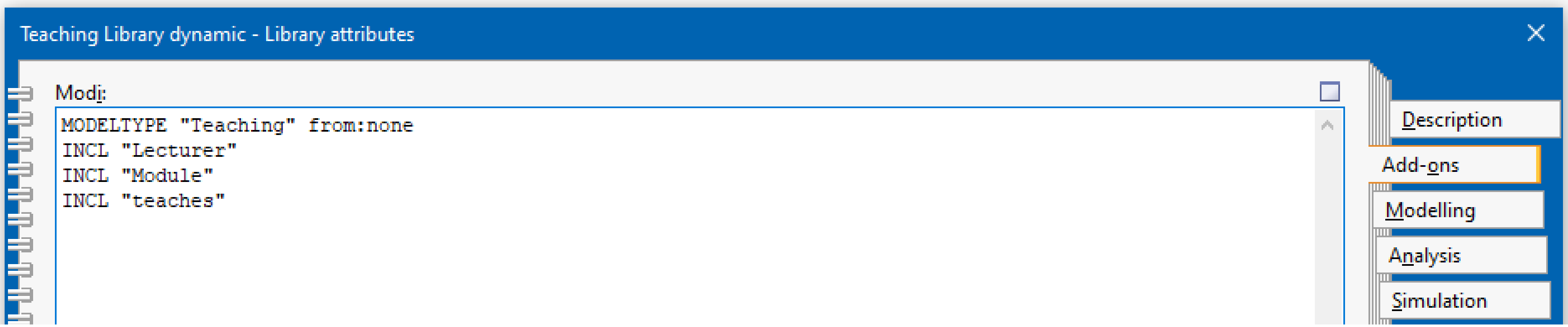






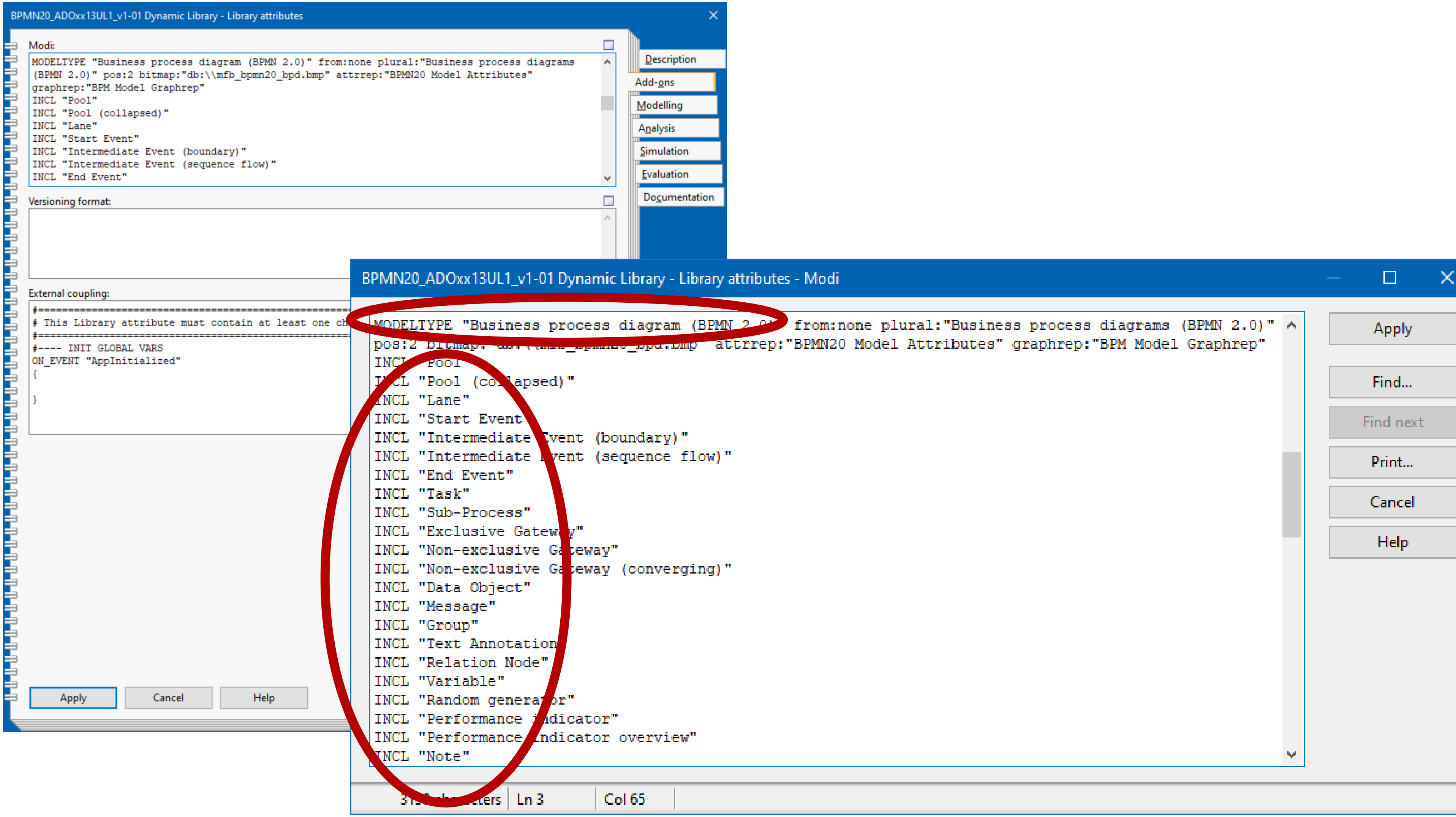


Example



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

Classes are assigned to Model Types



BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes

Modi:

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

Versioning format:

External coupling:

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

}
```

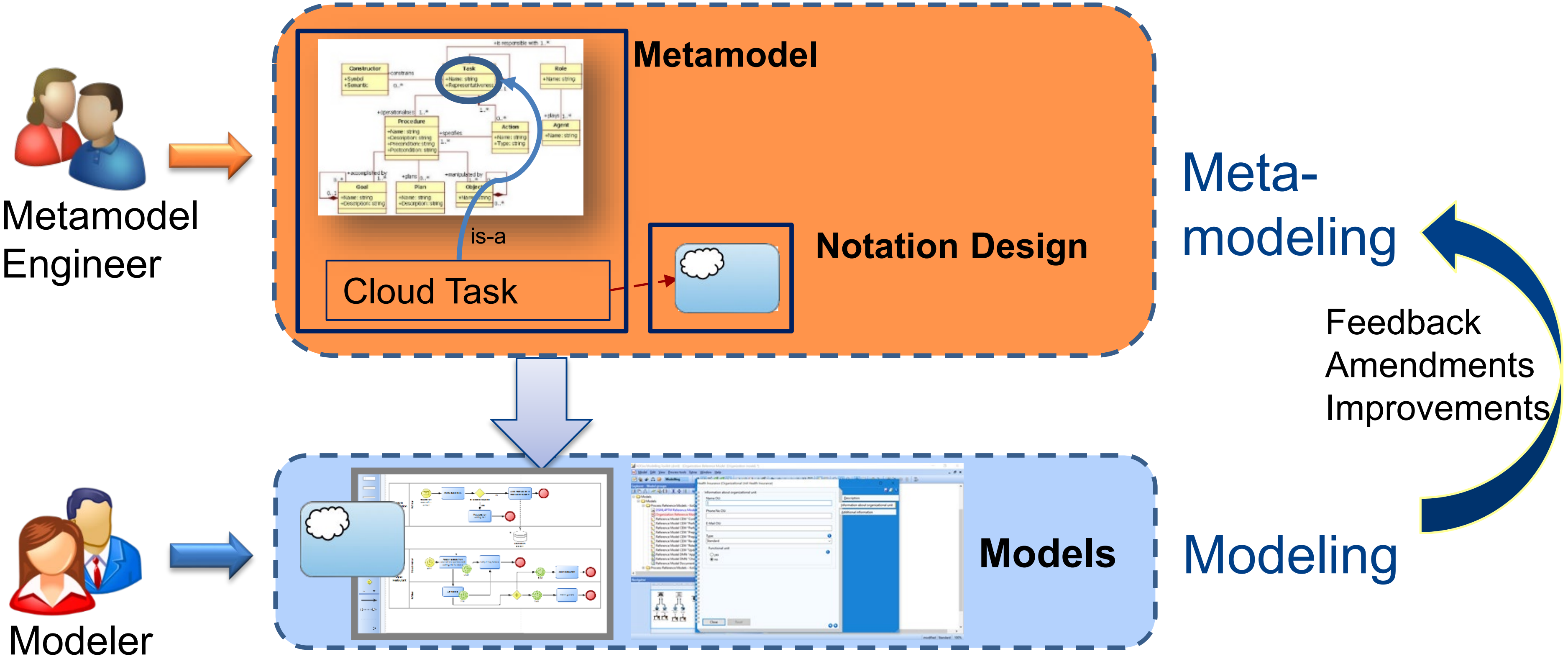
BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes - Modi

```
MODELTYPE "Business process diagram (BPMN 2.0)" from:none plural:"Business process diagrams (BPMN 2.0)"
pos:2 bitmap:"db:\\mfb_bpmn20_bpd.bmp" attrrep:"BPMN20 Model Attributes" graphrep:"BPM Model Graphrep"
INCL "Pool"
INCL "Pool (collapsed)"
INCL "Lane"
INCL "Start Event"
INCL "Intermediate Event (boundary)"
INCL "Intermediate Event (sequence flow)"
INCL "End Event"
INCL "Task"
INCL "Sub-Process"
INCL "Exclusive Gateway"
INCL "Non-exclusive Gateway"
INCL "Non-exclusive Gateway (converging)"
INCL "Data Object"
INCL "Message"
INCL "Group"
INCL "Text Annotation"
INCL "Relation Node"
INCL "Variable"
INCL "Random generator"
INCL "Performance indicator"
INCL "Performance indicator overview"
INCL "Note"
```

31[illegible] characters | Ln 3 | Col 65

Change of Metamodel

- Example: new task type Cloud Task


Metamodel Engineer
Metamodel
is-a
Cloud Task
Notation Design
Meta-modeling
Feedback
Amendments
Improvements
Models
Modeling
Modeler