Conceptual Modelling

Knut Hinkelmann

Models

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

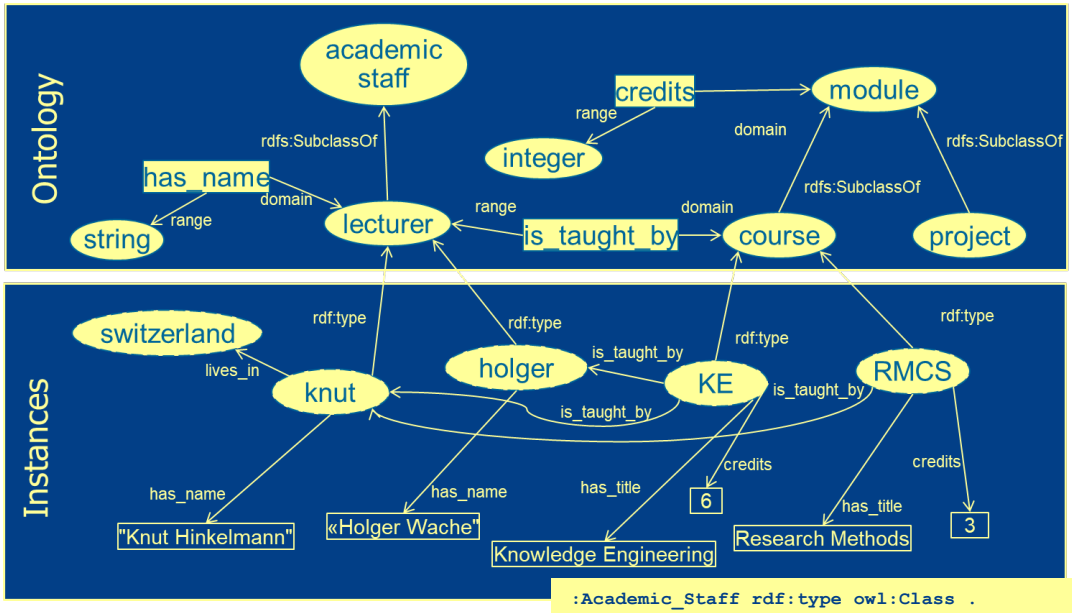
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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" .
```

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

General-purpose Modelling vs. Conceptual Modelling

Modelling

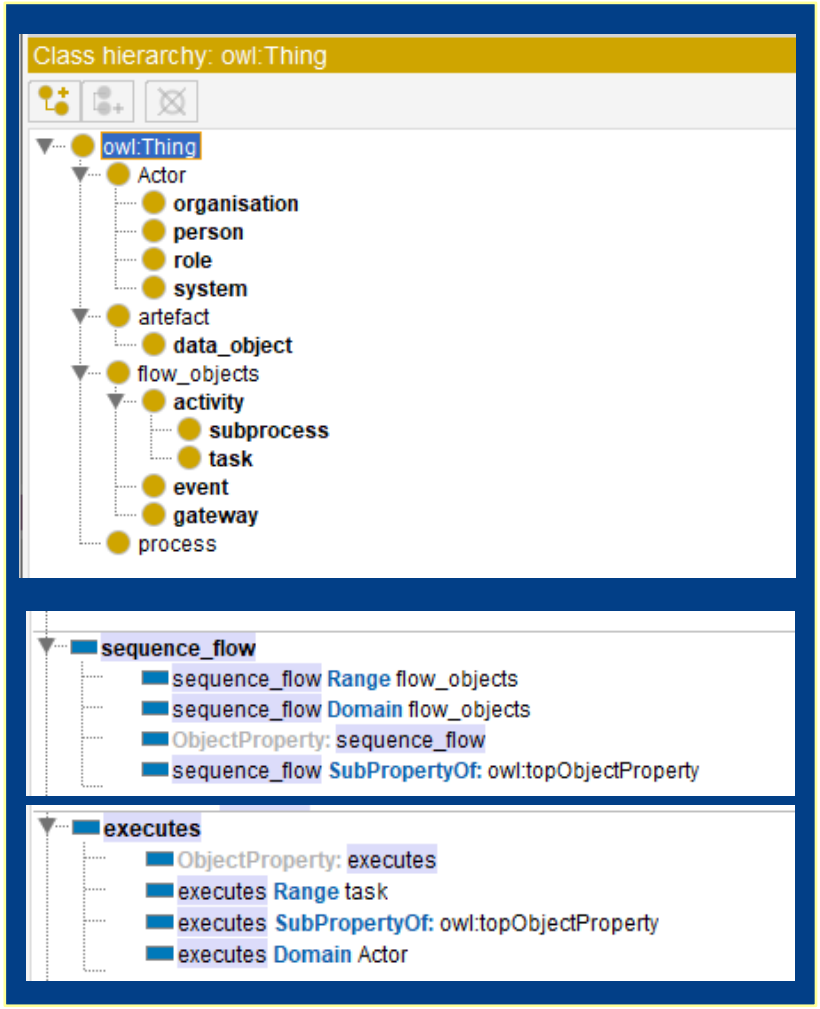
Representing all relevant aspects of a domain in a defined language.

- ***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

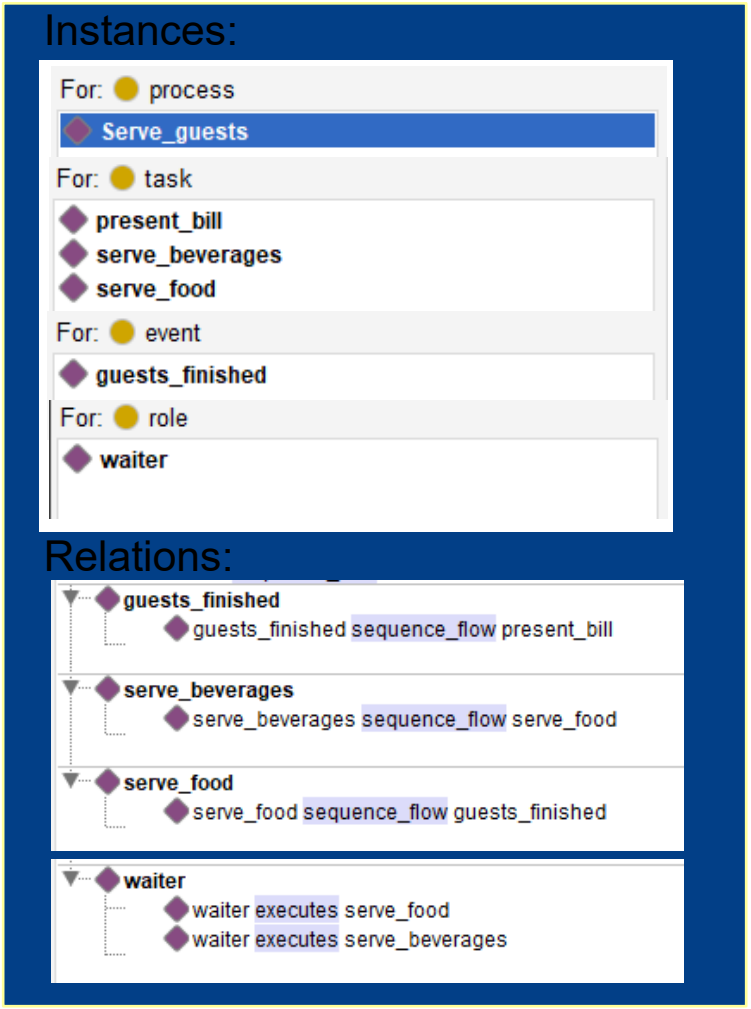
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Example: Concepts and Instances for Process Modelling

Business Process Ontology (Metamodel):



Process Model for Serve Guests



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

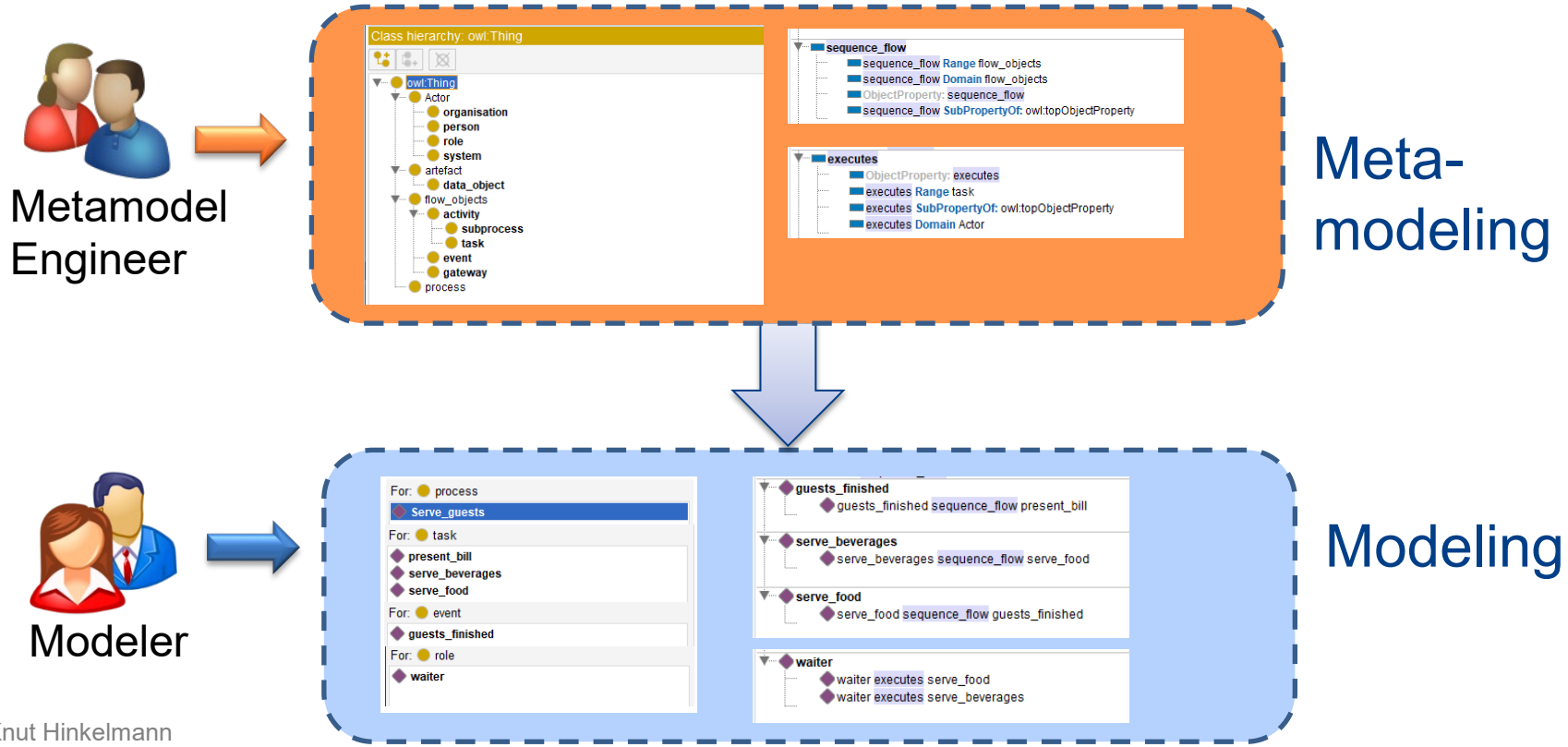
The specification of the domain-specific concepts that can be used for modeling

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

Modeling and Metamodeling

Conceptual Modeling consists of two phases

- Metamodeling: Defining (domain-specific) concepts
- Modeling: Creating models using these concepts



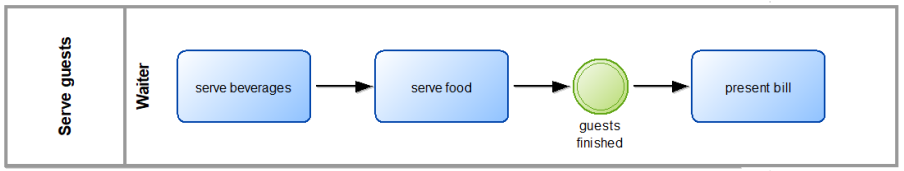
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

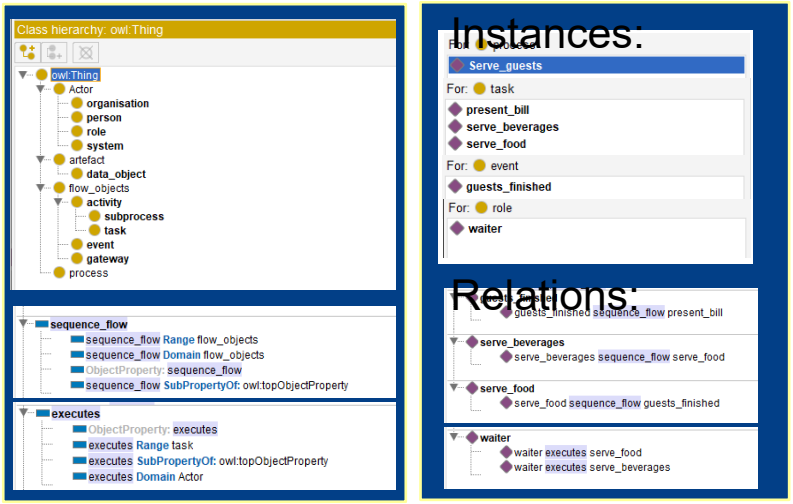
Strengths and Weaknesses of Conceptual Modelling

- 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

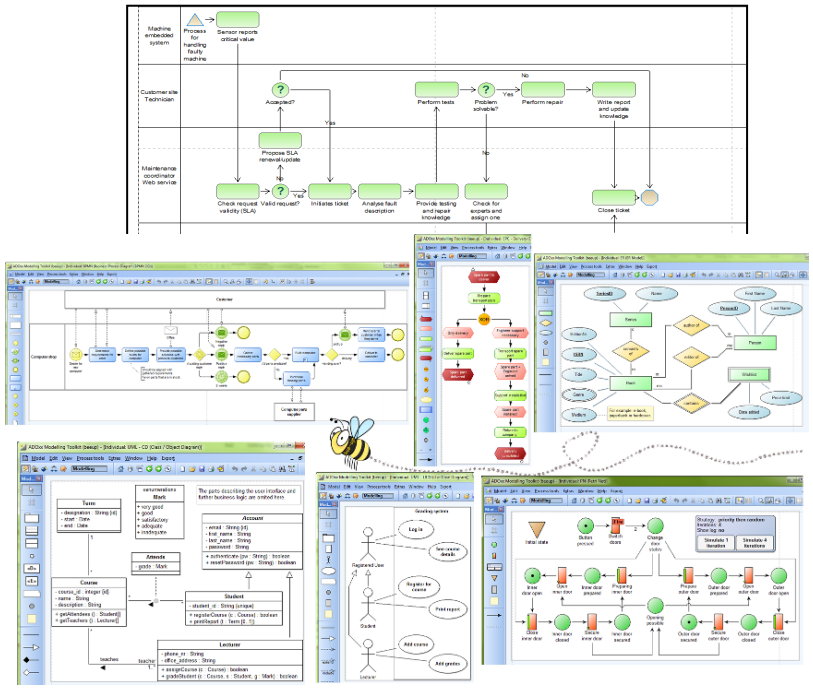
- Creating knowledge graphs is difficult for domain experts – it requires skills in modeling language
- Creating graphical models is more adequate for non-technical
- Objective: Create ontologies (knowledge graphs) from graphical models





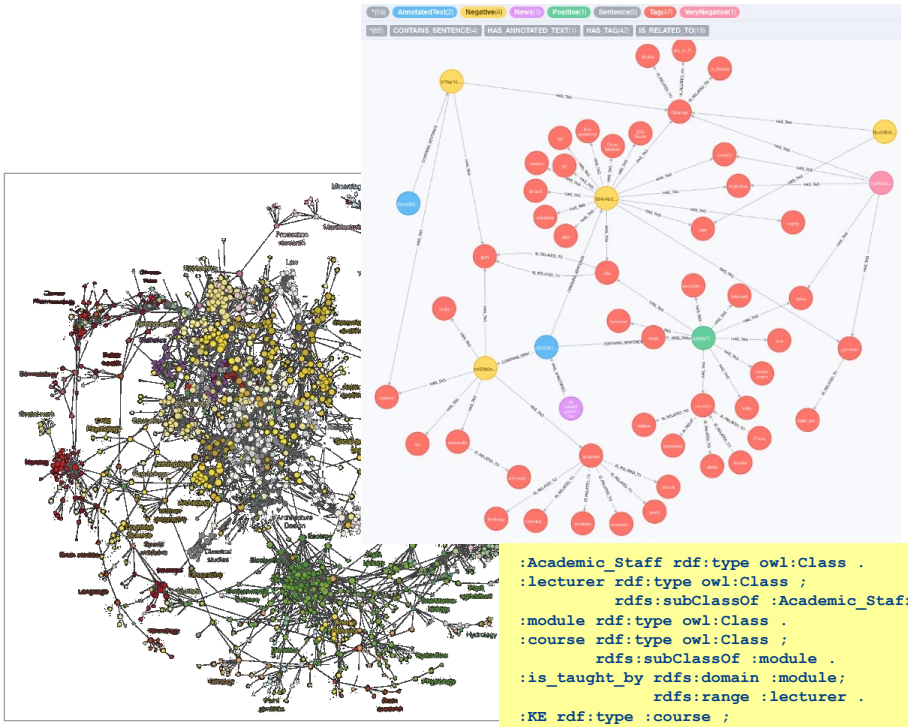
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Conceptual (graphical) Models



Modeling using predefined *concepts*

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" .
```

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

Agenda

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

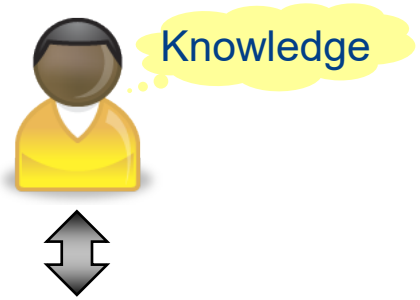
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Conceptual Modeling and Metamodelling with Graphical Models

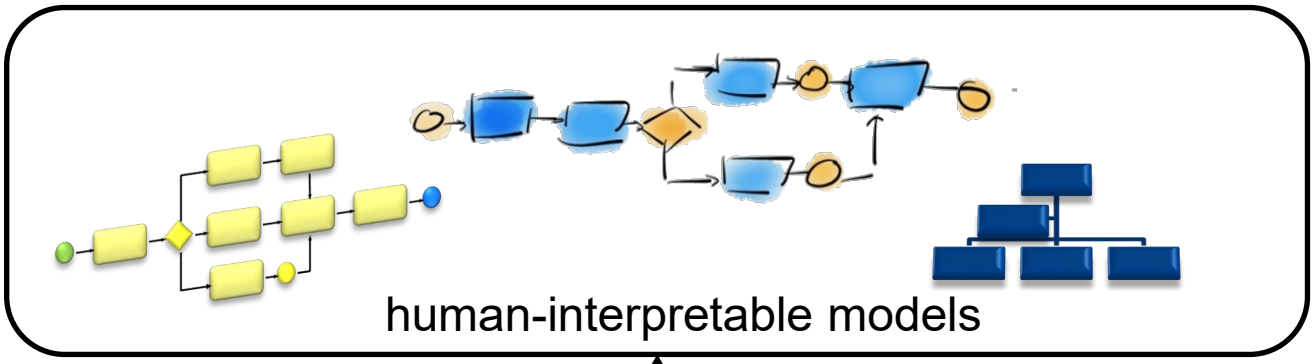
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Graphical Models are appropriate for Humans

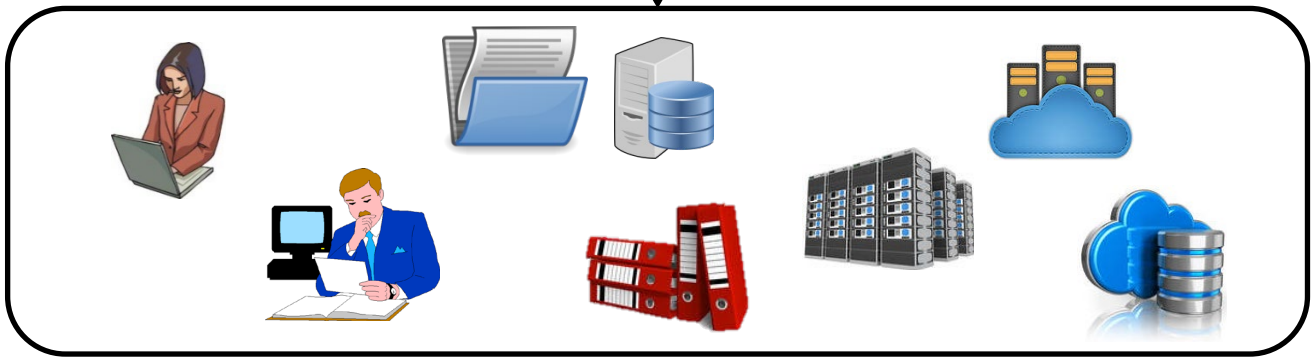
Communication/
Analysis/
Decision Making



Models

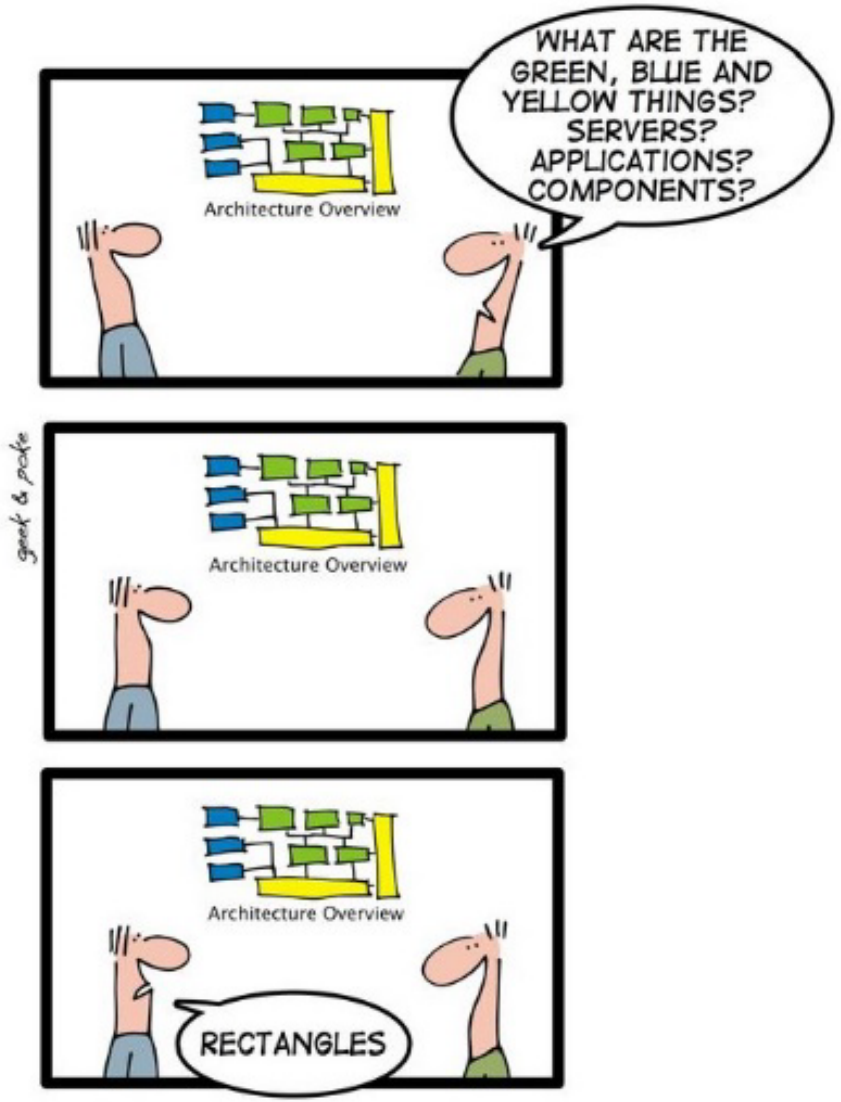


Reality



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

Interpretation of Models



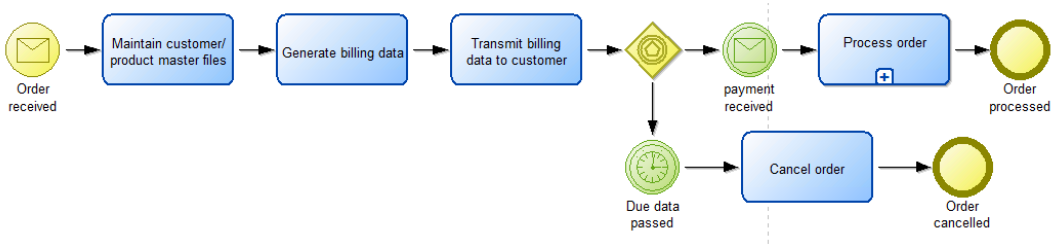
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

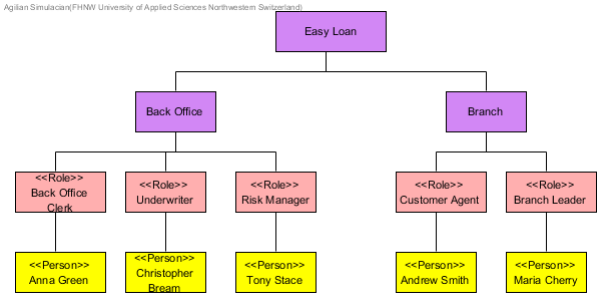
Models

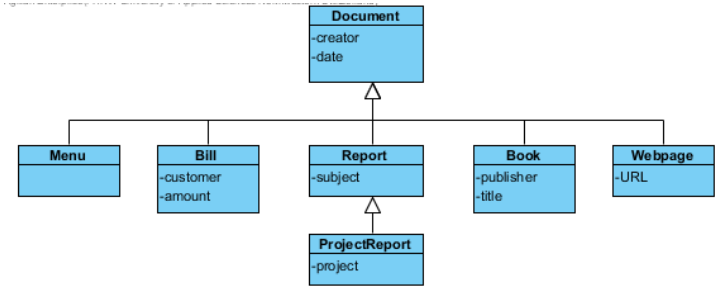
- Models are not mere pictures; rather, they
  - provide a precise, meaningful description that can be visualized in different ways for different stakeholders;
  - can also be used to analyze the impact of changes, cost, risk, security, compliance and other relevant KPIs.

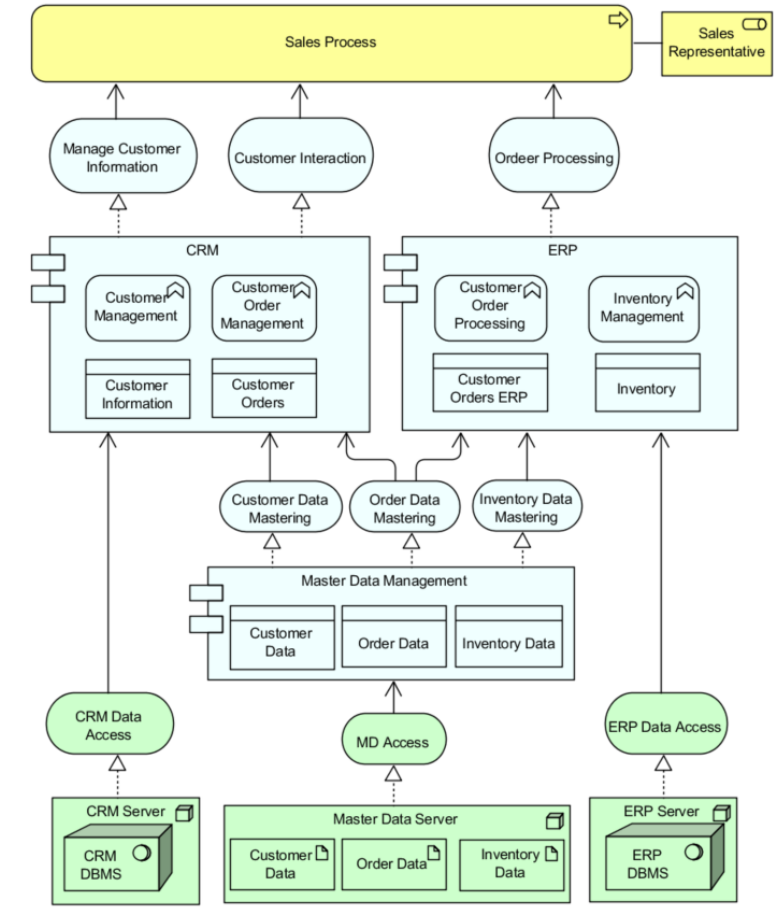
http://blog.bizzdesign.com/how-to-not-fail-when-implementing-strategy

Enterprise Models using Domain-Specific Graphical Modeling









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

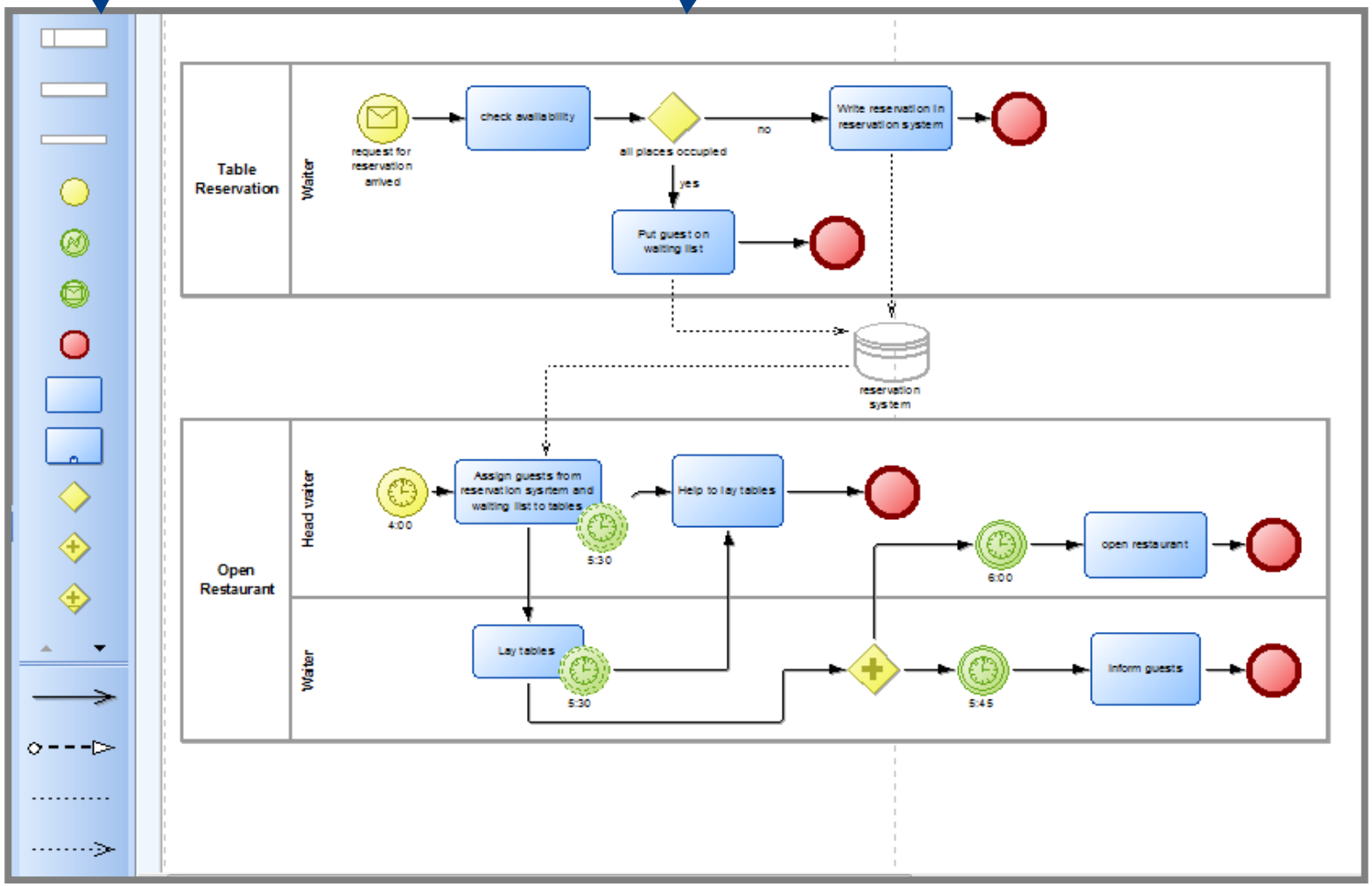
Domain-specific Graphical Modelling Languages

- 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, …

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

Modelling Language

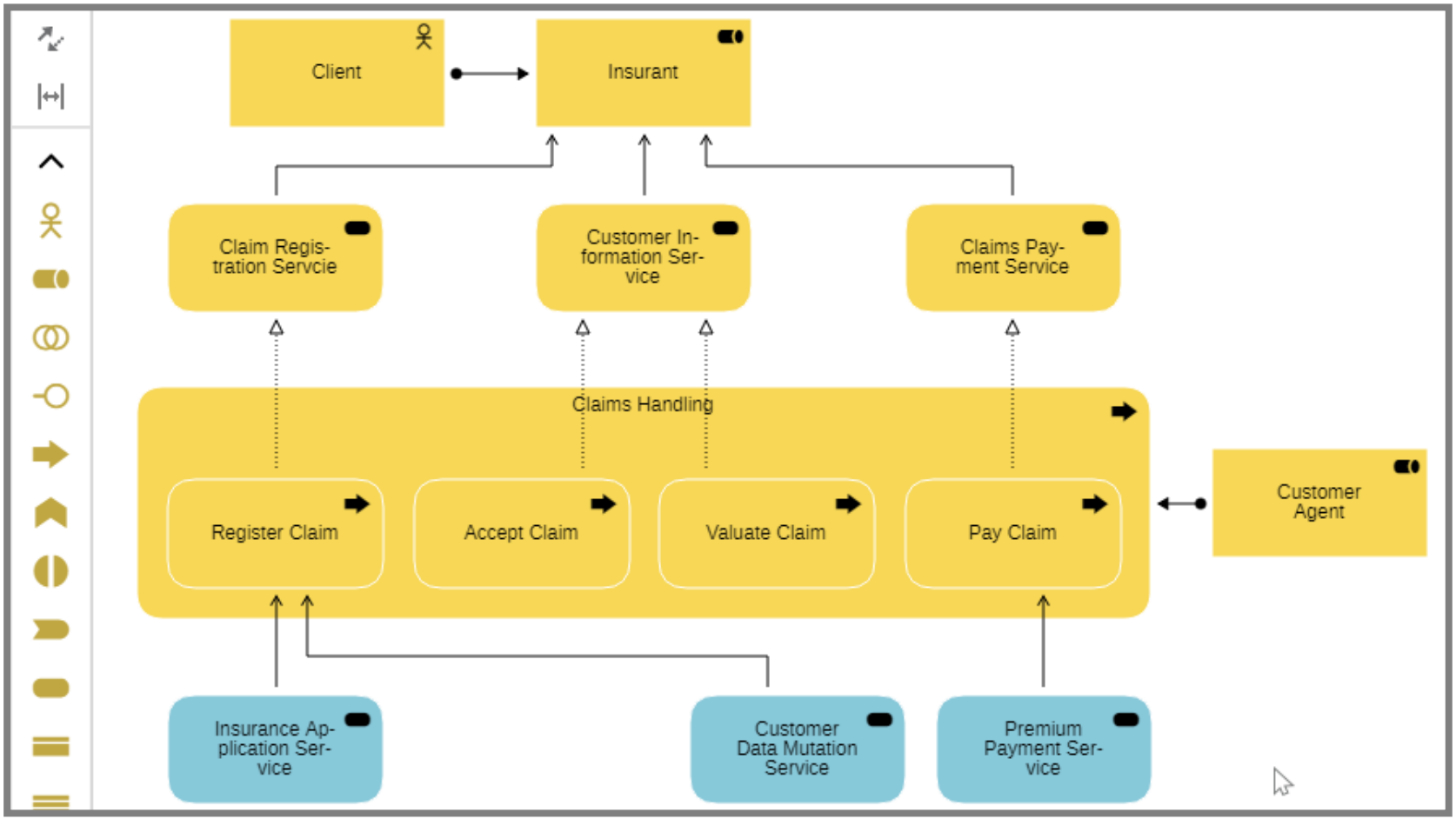
Model



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

Modelling Language

Model



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

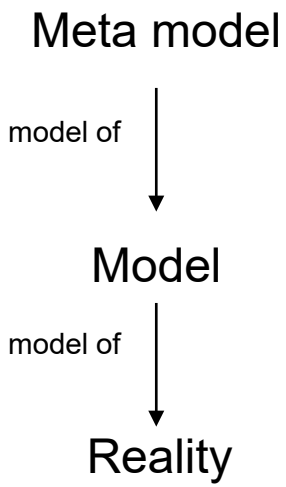
The specification of the domain-specific concepts that can be used for modeling

Modelling Language

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

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

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

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

Concepts for Business Process Models

Metamodel:

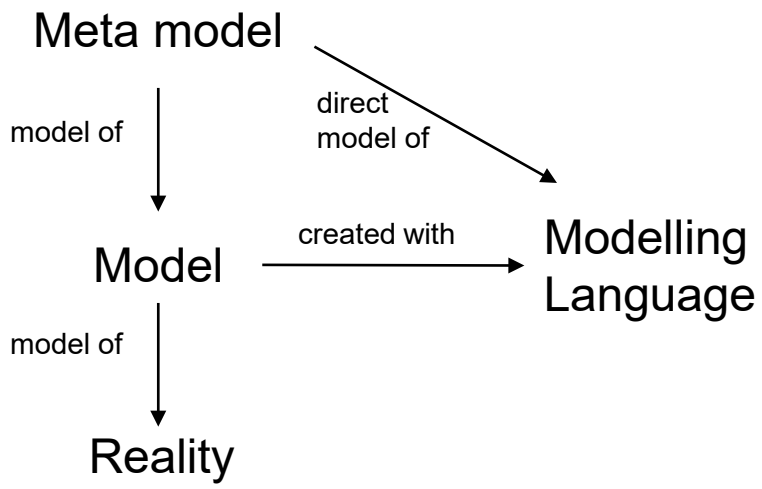
Concepts which can be used to create models.

Example: A process model consists of concepts for

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

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

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

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

Illustration: Modeling Language for Business Processes

Meta model:

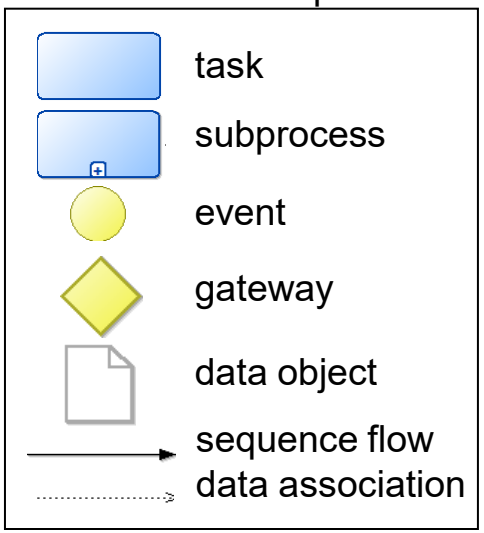
Concepts which can be used to create models.

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



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

Illustration: Modeling Language for Business Processes

Metamodel:

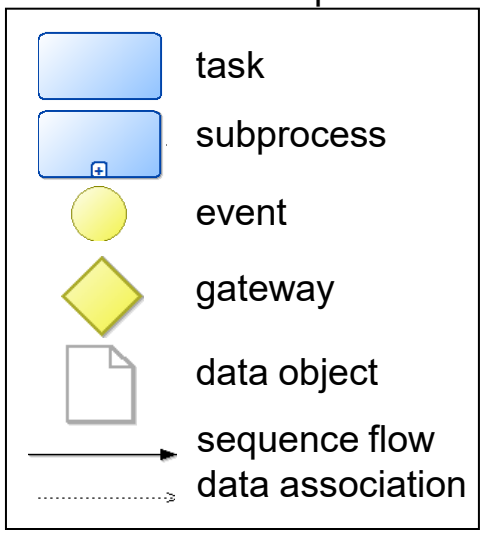
Concepts which can be used to create models.

Example: A process model consists of concepts for

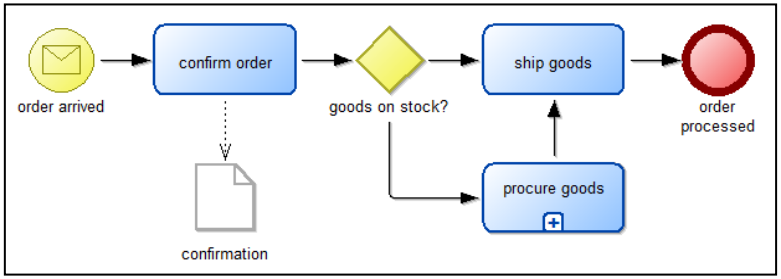
- Model elements: event, task, subprocess, gateway, data object
- Relationships: sequence flow, data association.

Modelling Language:

Notation/appearance of meta-model concept



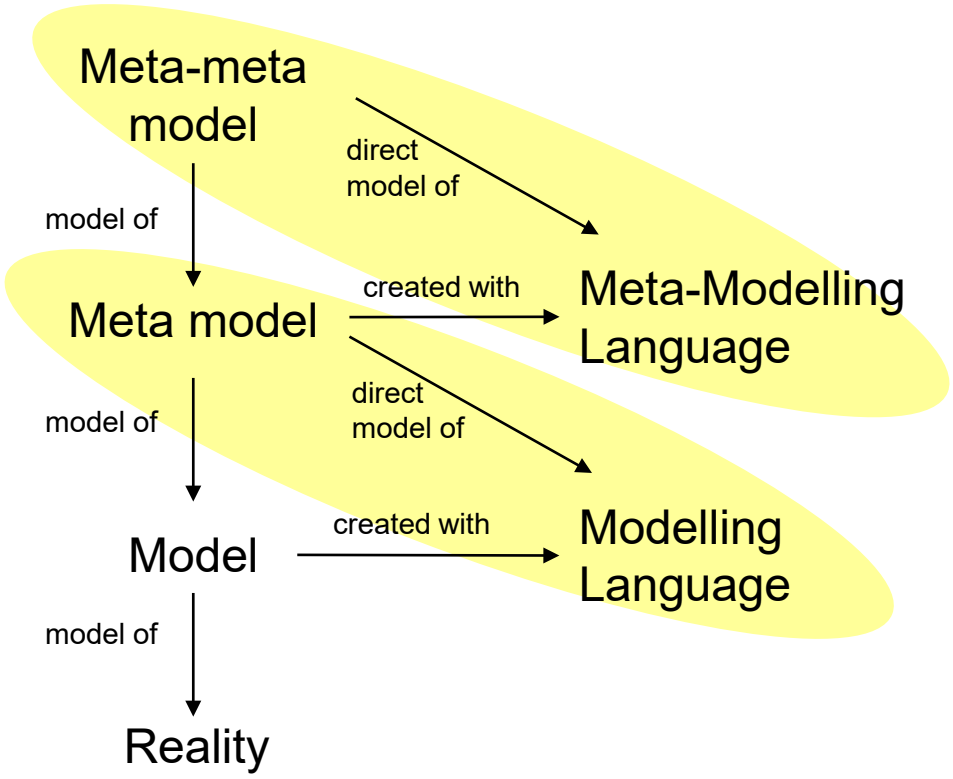
Model:



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"

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

Meta-meta model



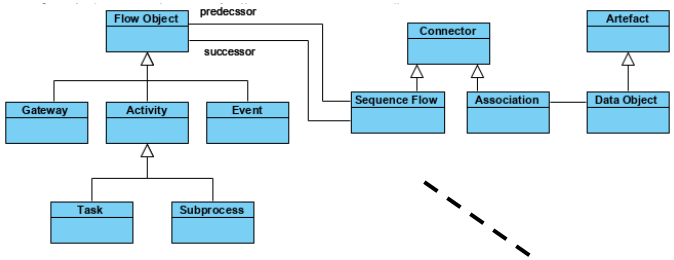
- The meta model must again be described in some language, which is specified in a meta-meta model
- A **meta-meta model** defines the concepts for describing a meta model
- Graphical models usually have to kinds of concepts
  - Modeling elements
  - Relationships
- Examples for meta-modeling languages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

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

Metamodels can be defined as Class Diagrams

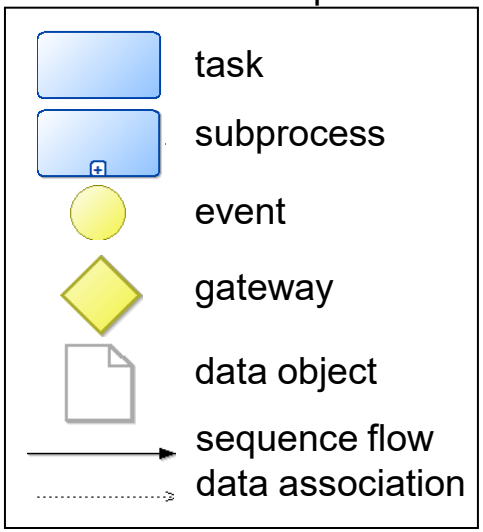
Metamodel:

Concepts which can be used to create models.

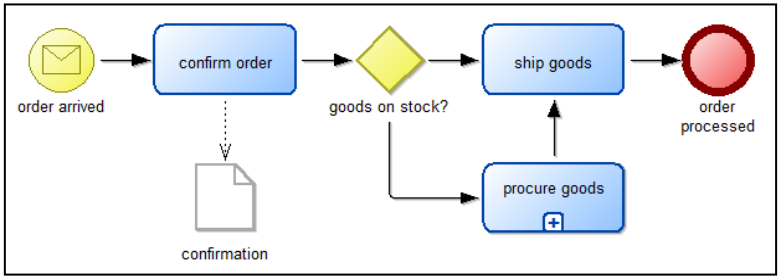


Modelling Language:

Notation/appearance of meta-model concept



Model:

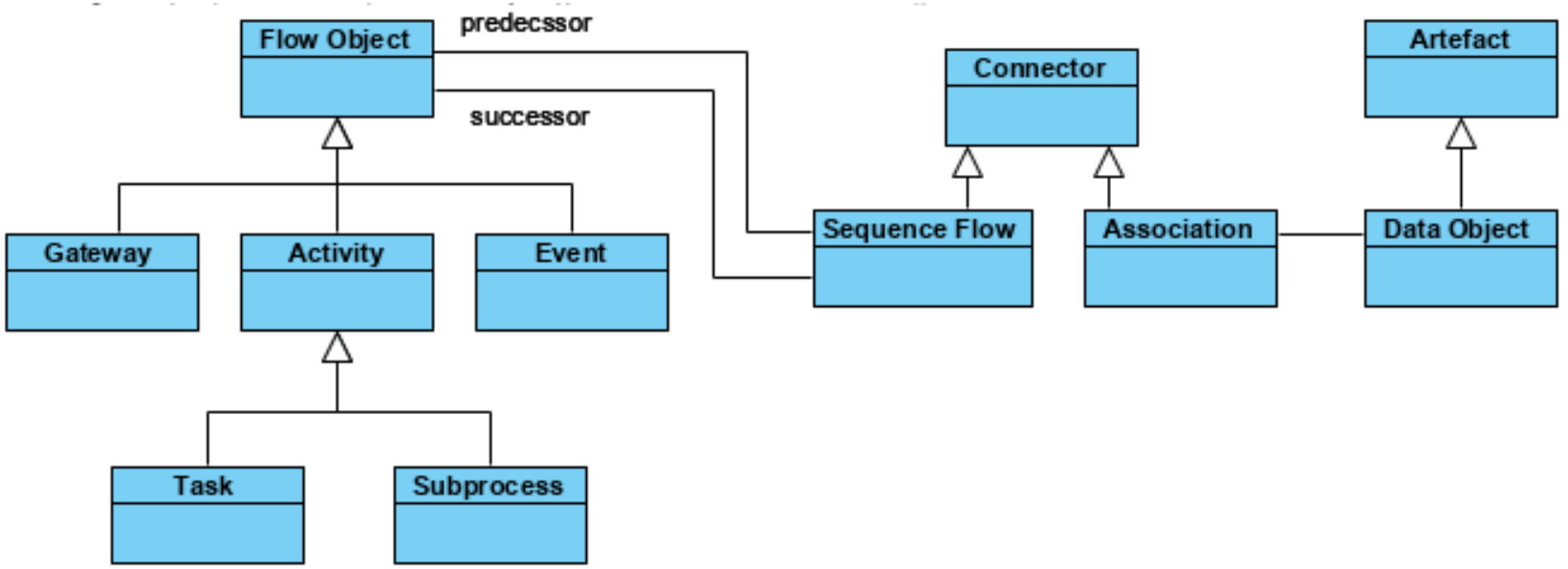


A model contains instances of the concepts defined in the meta-model. The object „confirm order“ represents a real entity; it is an instance of the concept «task"

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

Metamodels can be defined as Class Diagrams

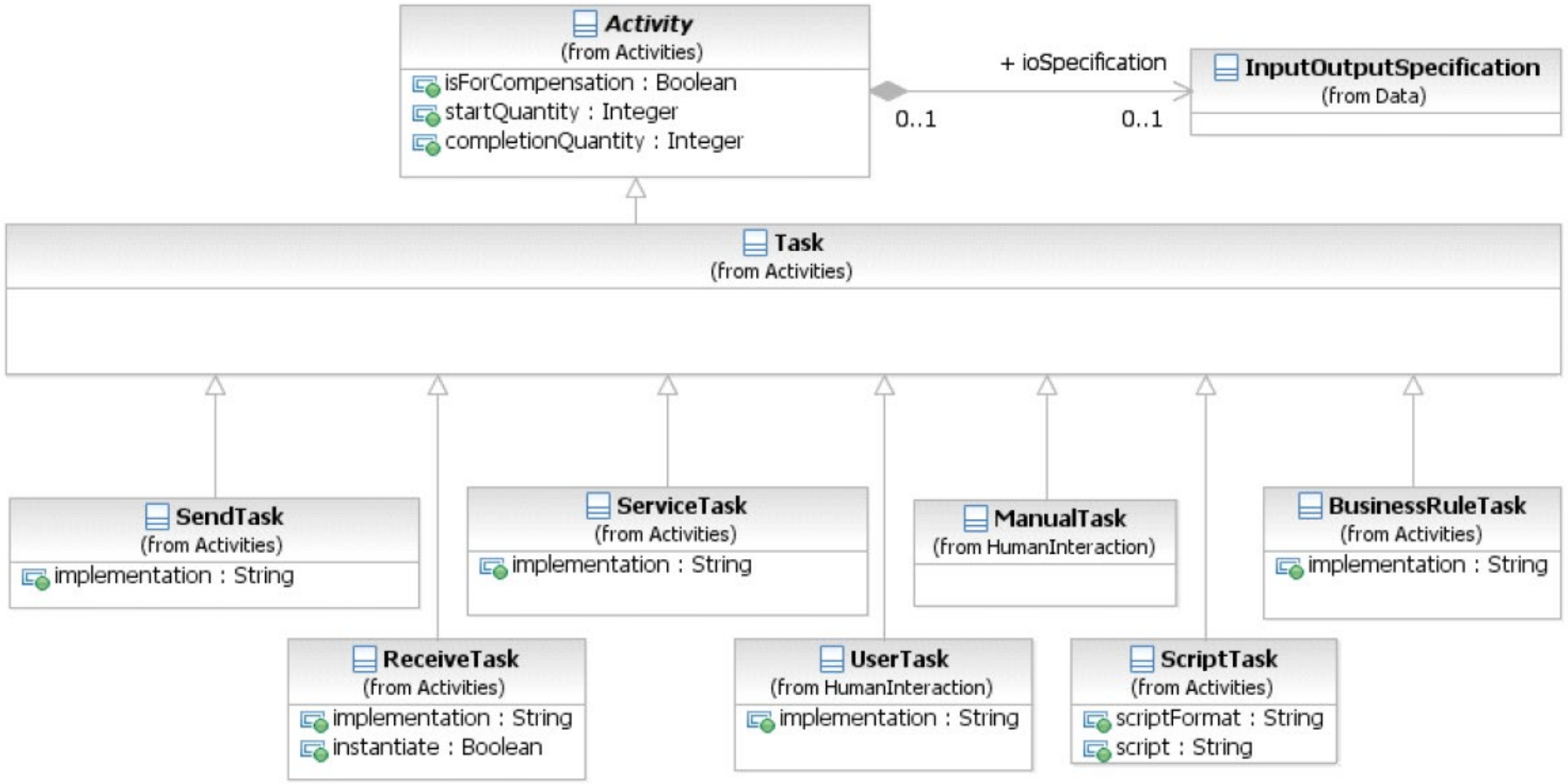
A Metamodeling language one can describe meta models
Metamodel corresponds to a knowledge base
Metamodels can be represented as class diagrams



(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)

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

Subset of the BPMN Metamodel as UML Class Diagram



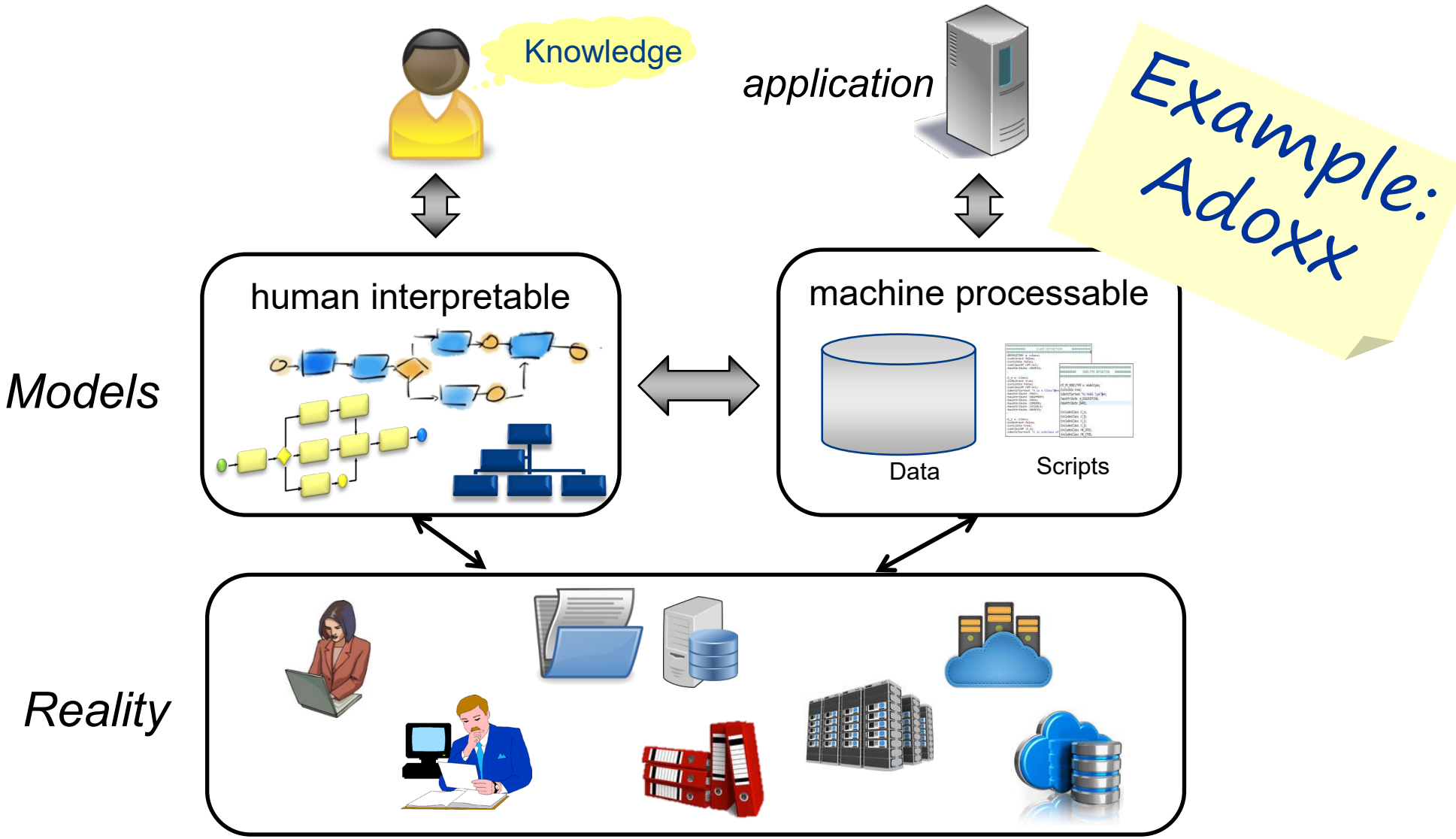
Source: BPMN 2.0 specification

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

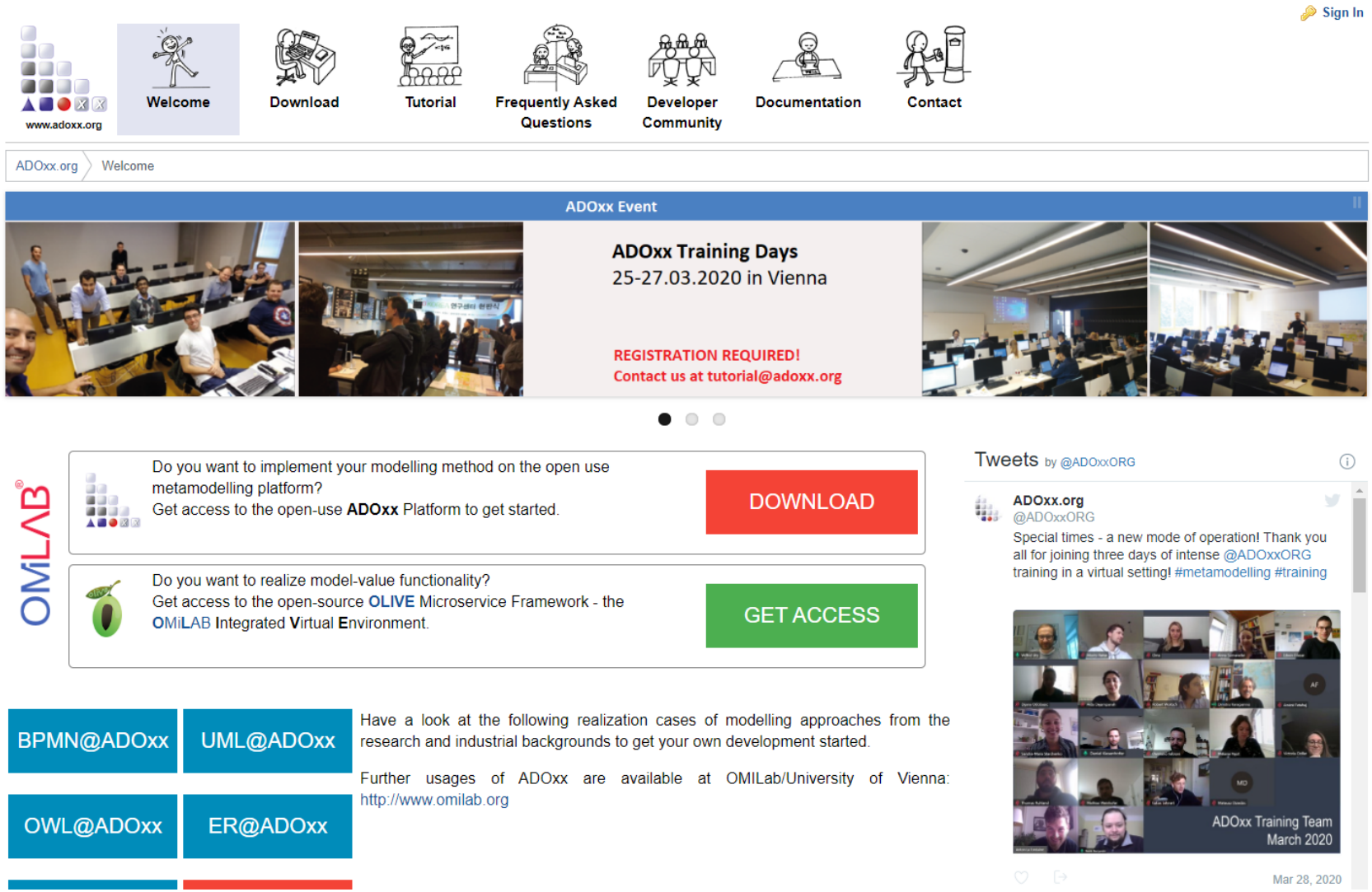
Metamodelling with ADOxx

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

Graphical Models represented in a Database



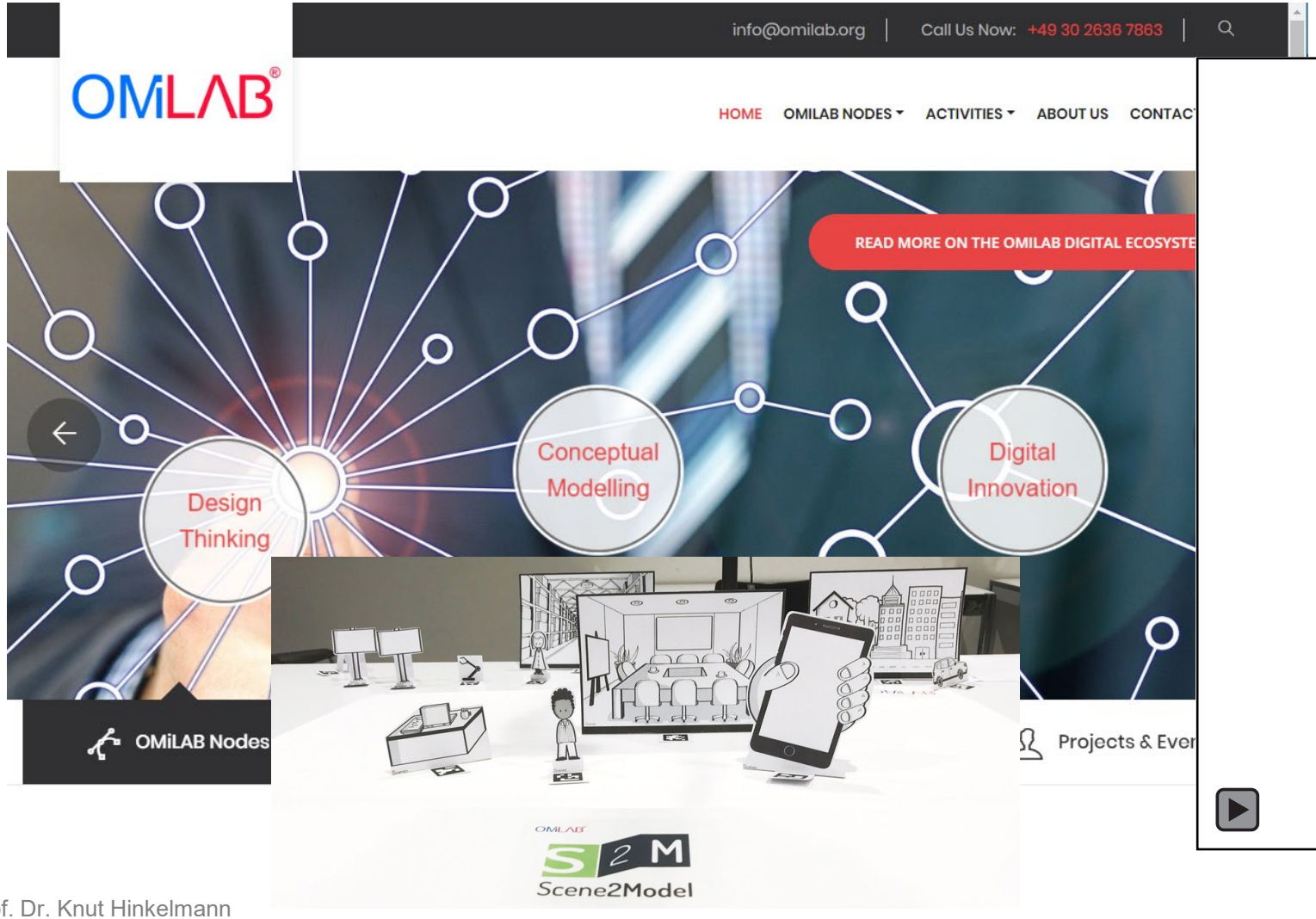
adoxx.org – Download, Tutorials, Community



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

OMiLAB – A Conceptual Modelling Community

ADOxx is the basis for OMiLAB

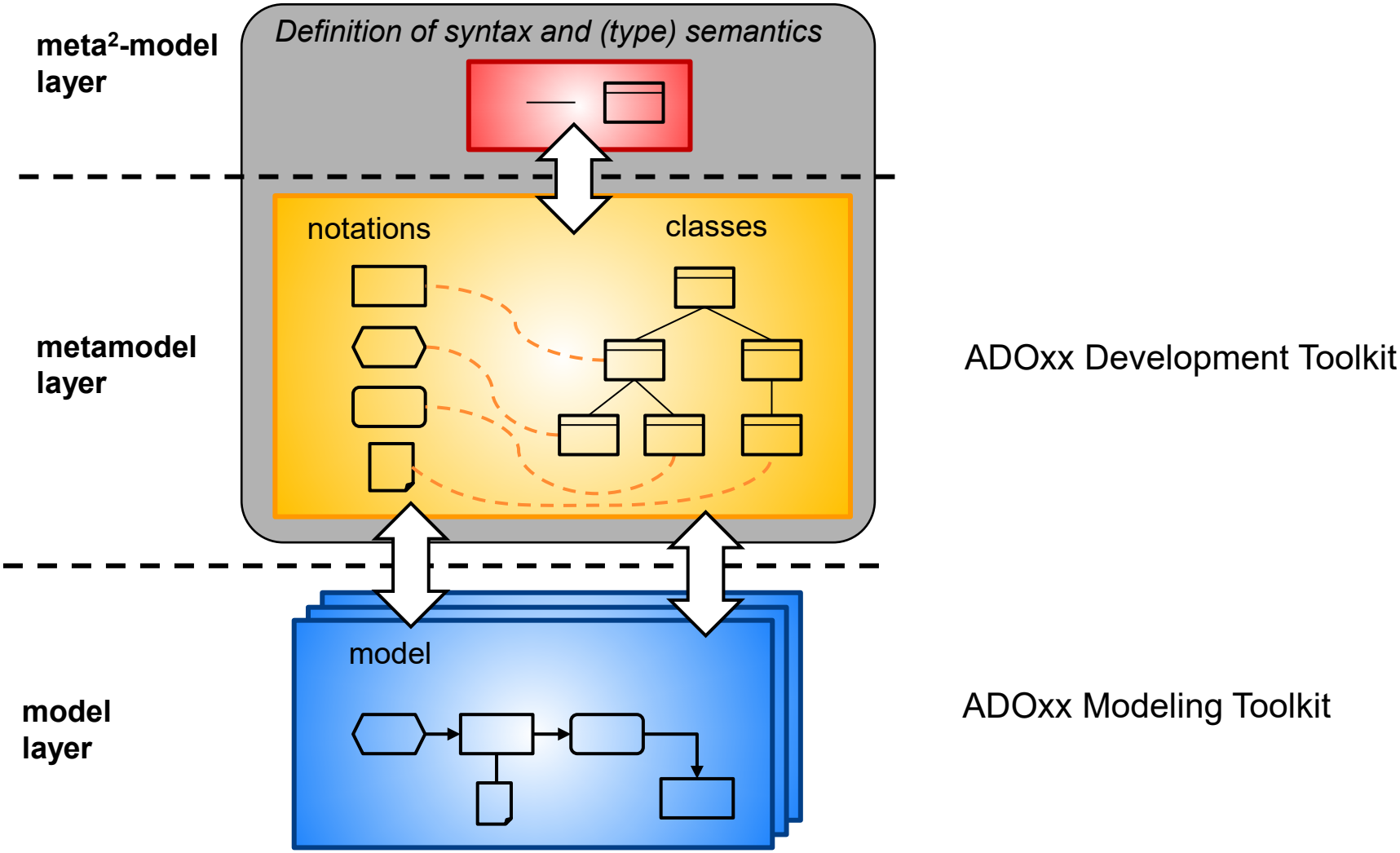


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

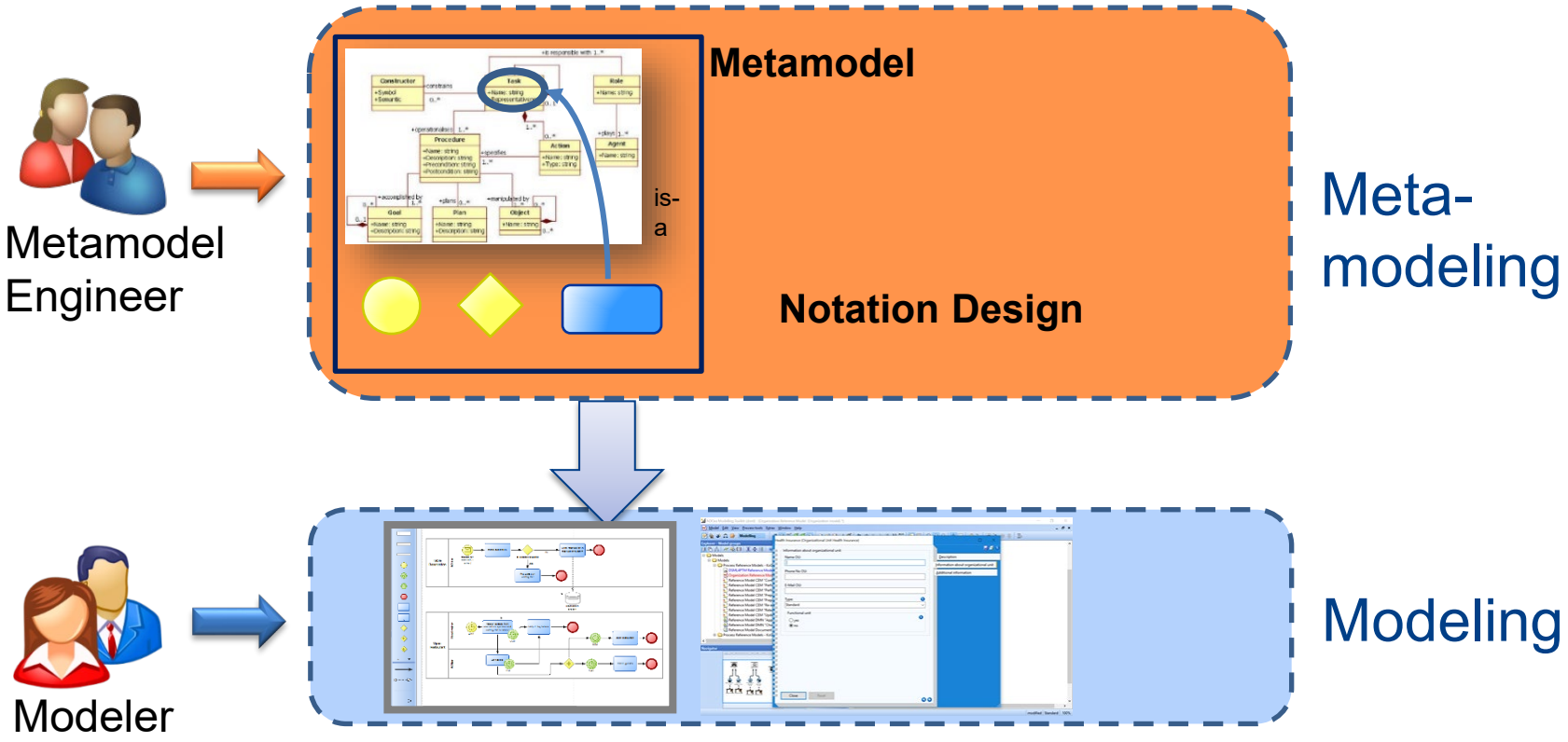
The ADOxx Environment

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

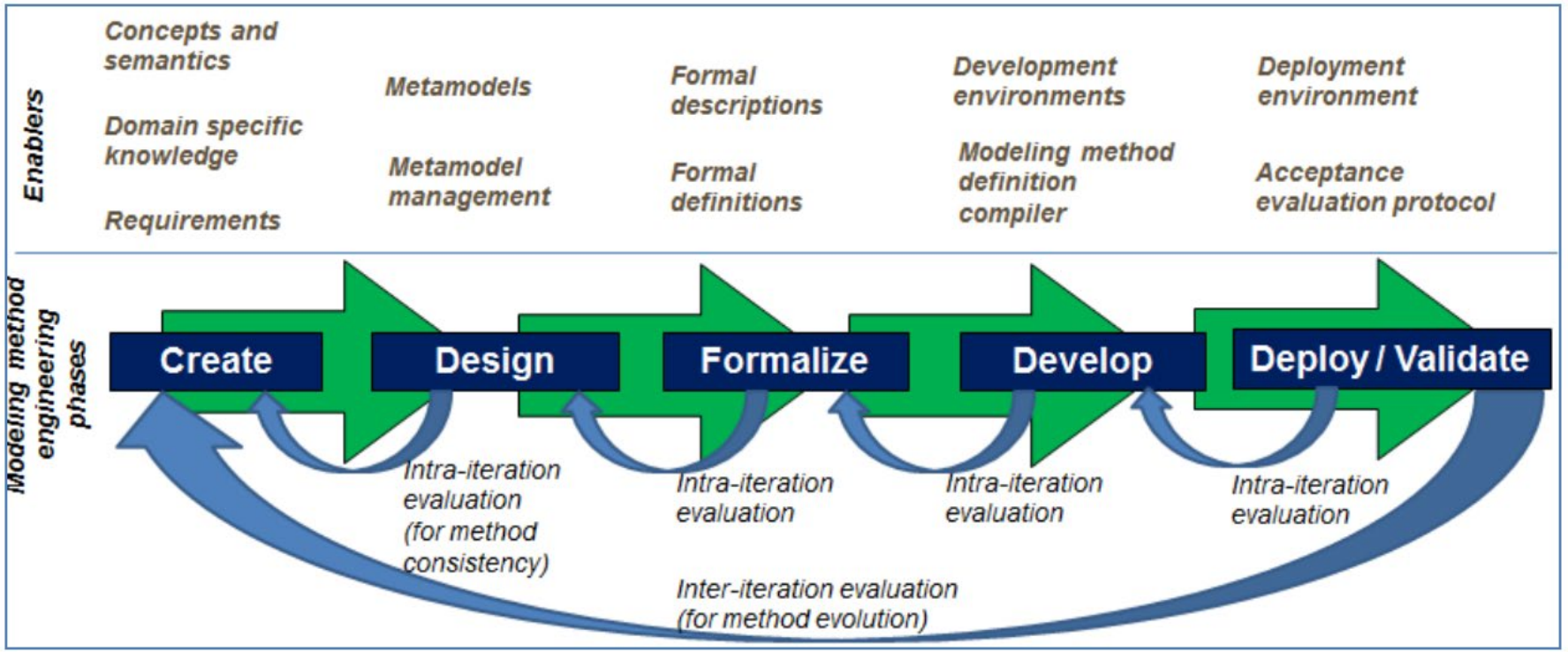
Modeling Environment



Modeling and Metamodeling



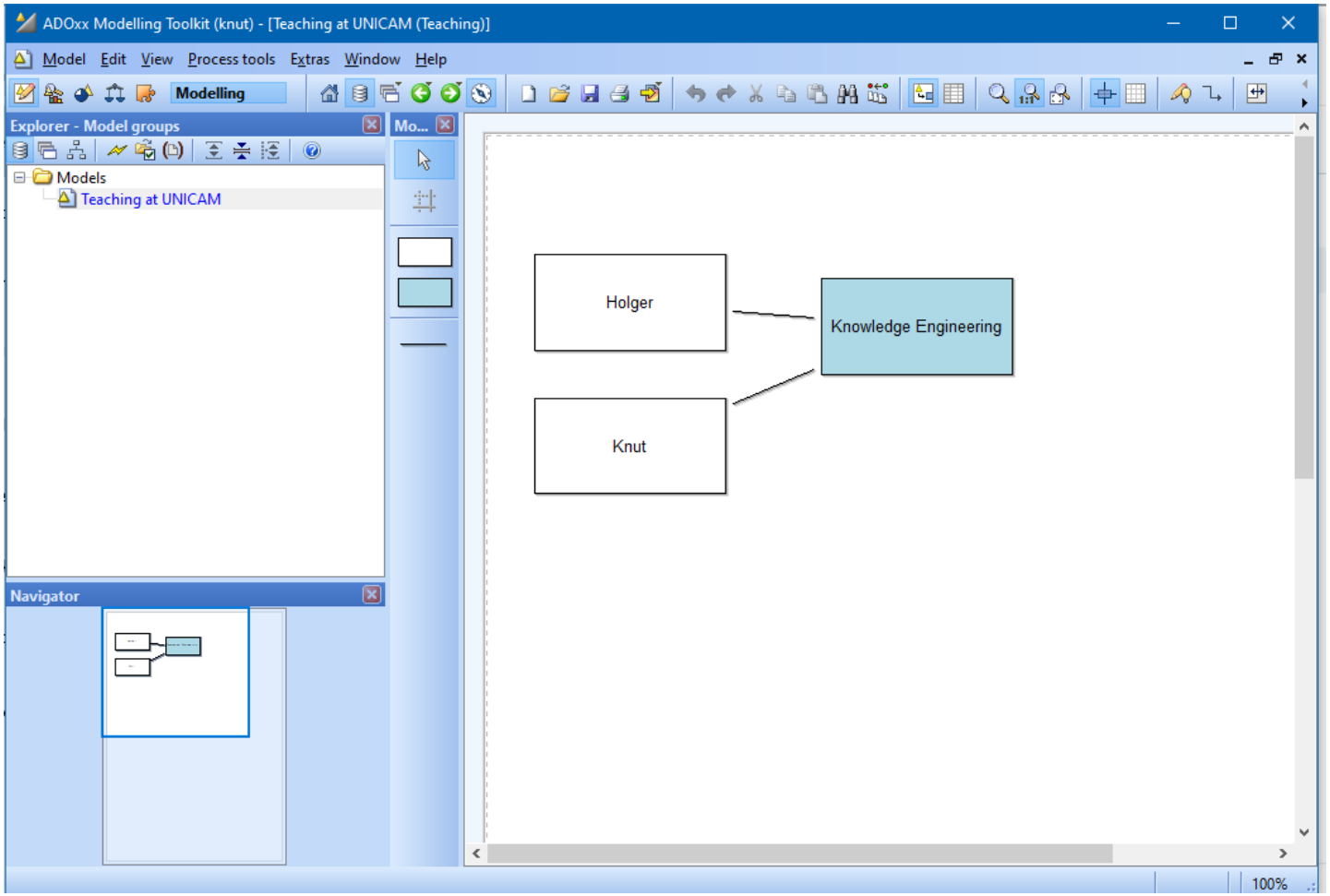
The AMME LifeCycle
Agile Modeling Method Engineering



(Karagiannis 2015)

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

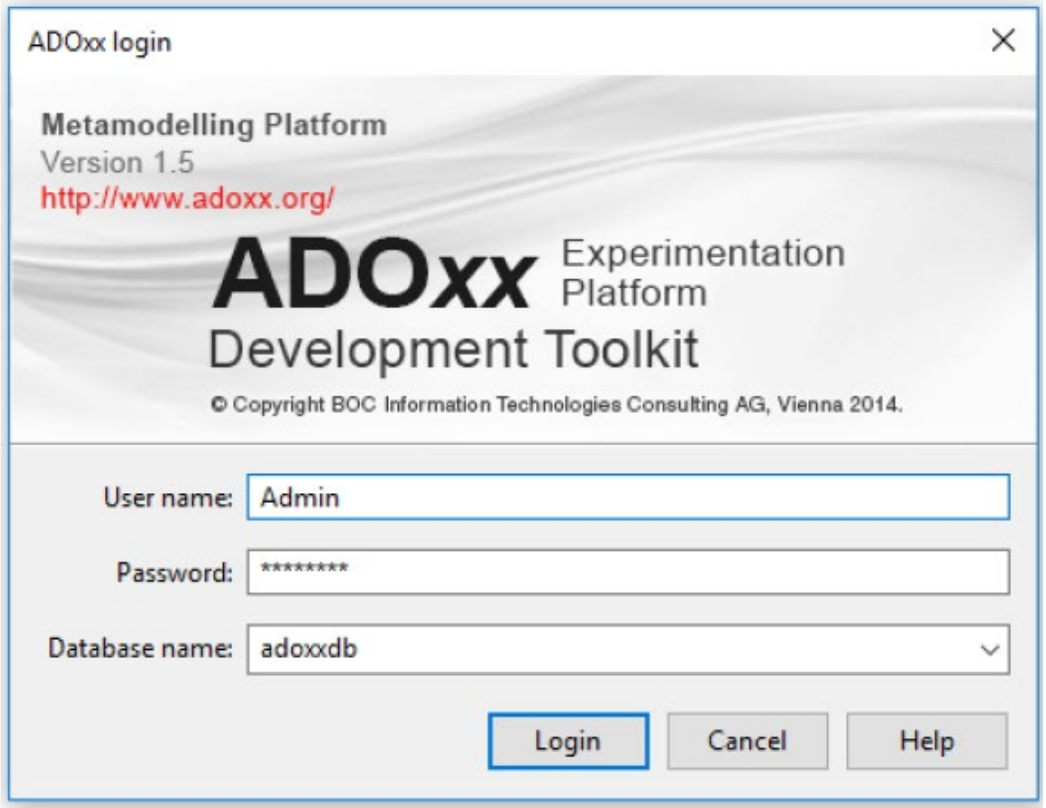
Example: Create a Modeling Language for Teaching



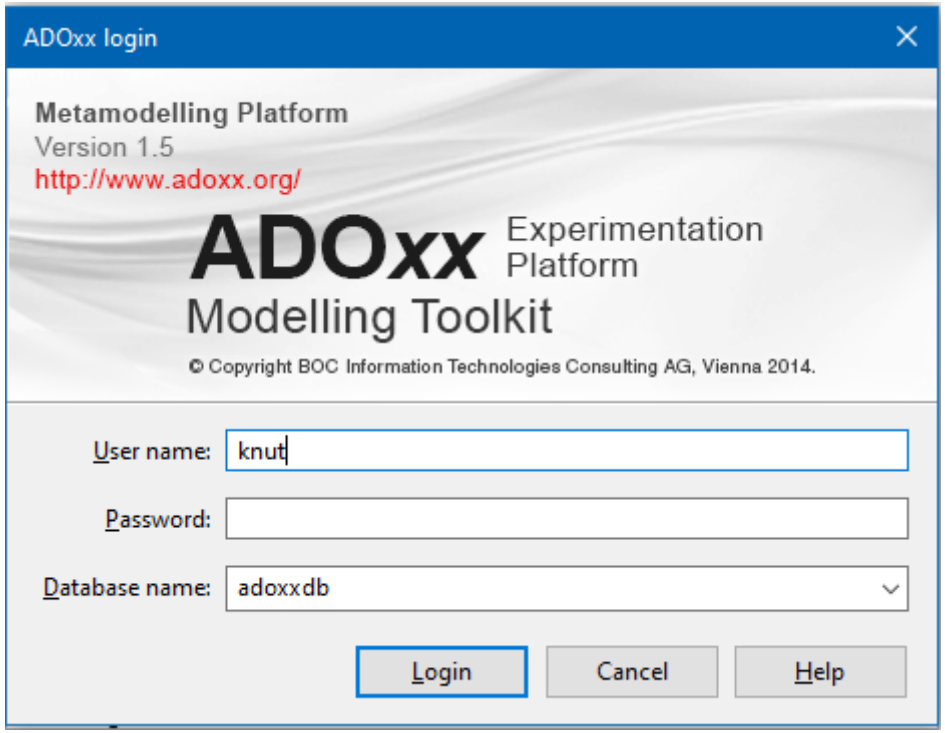
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Development Toolkit

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

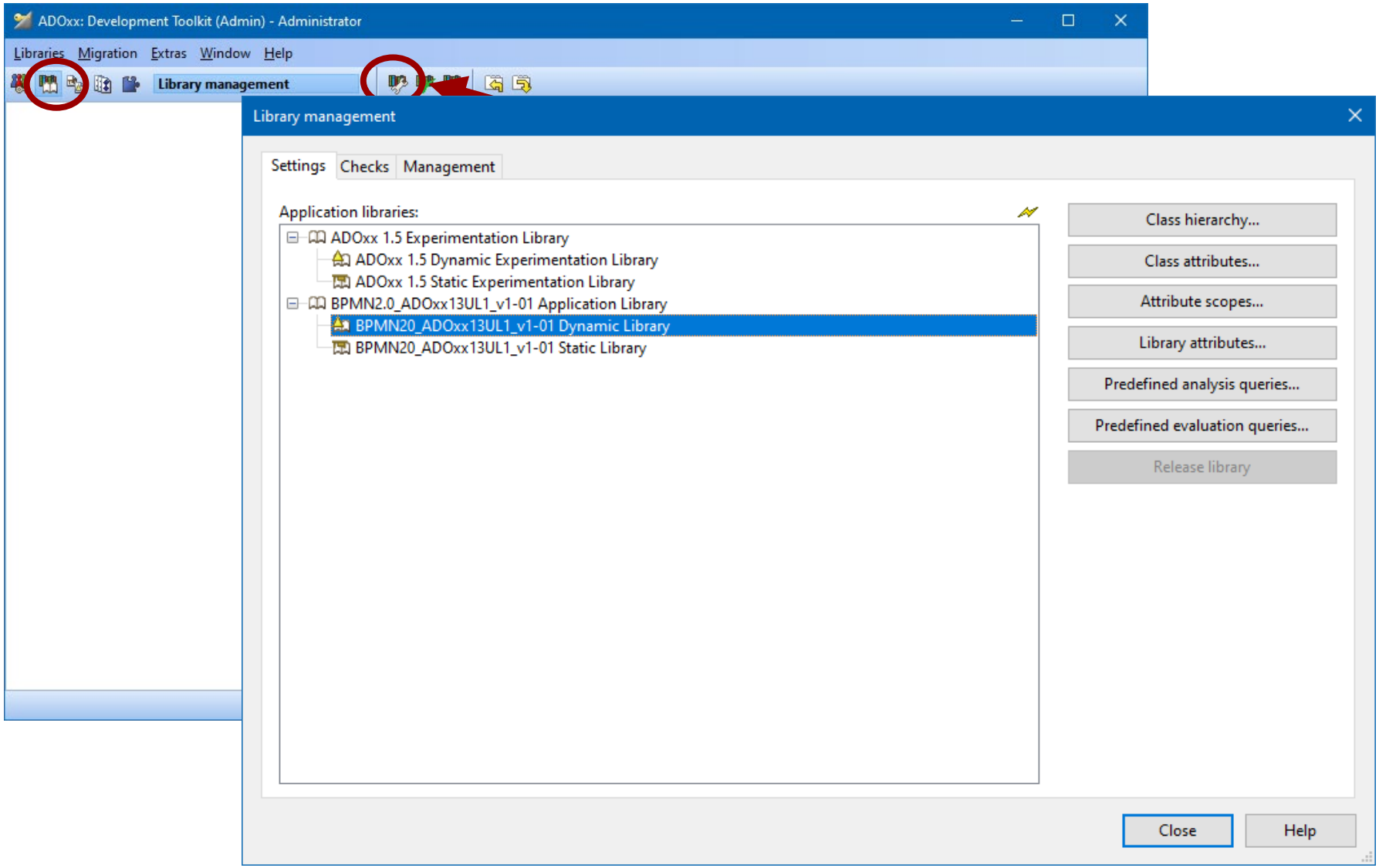


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch



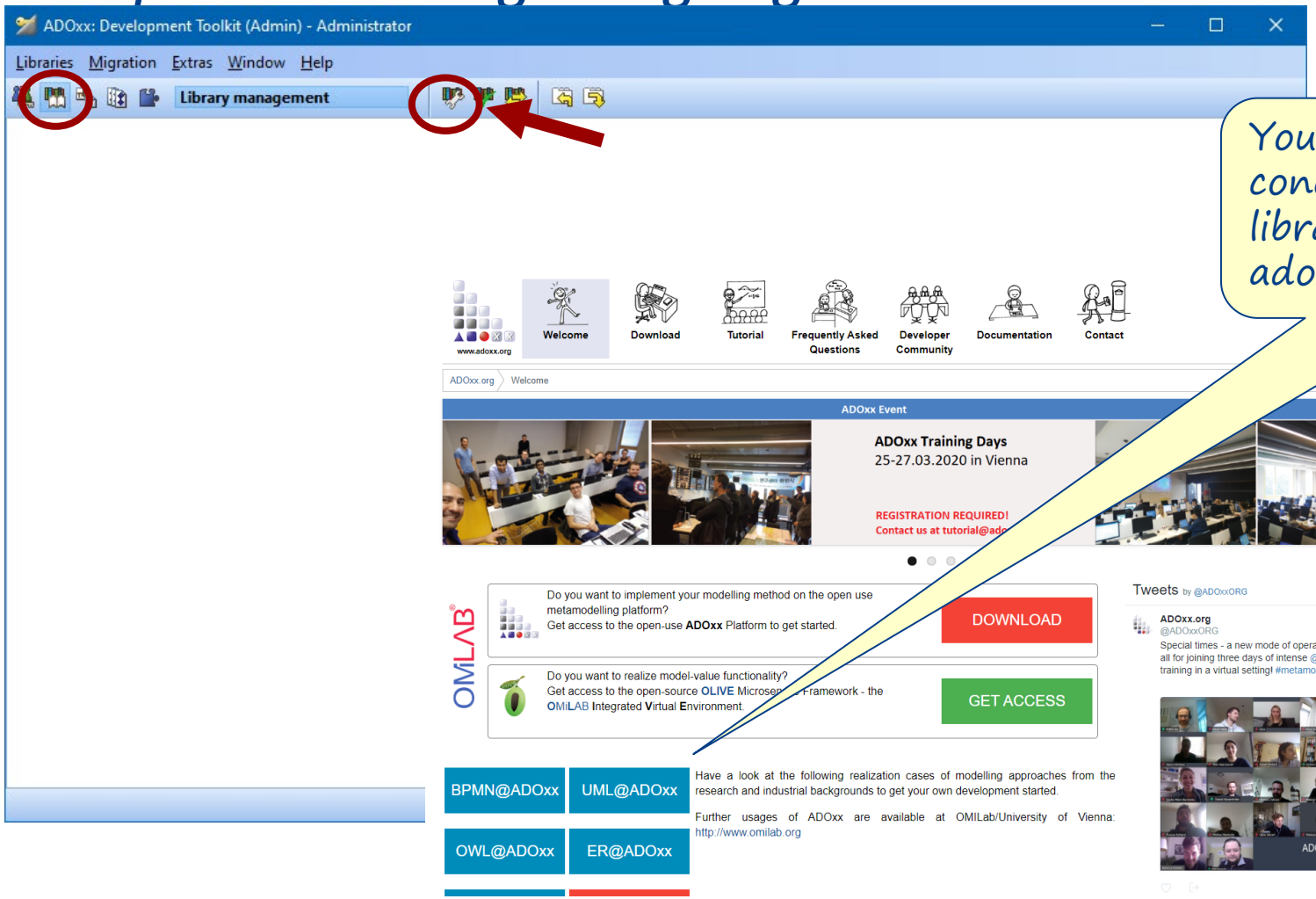
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Metamodelling with ADOxx



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

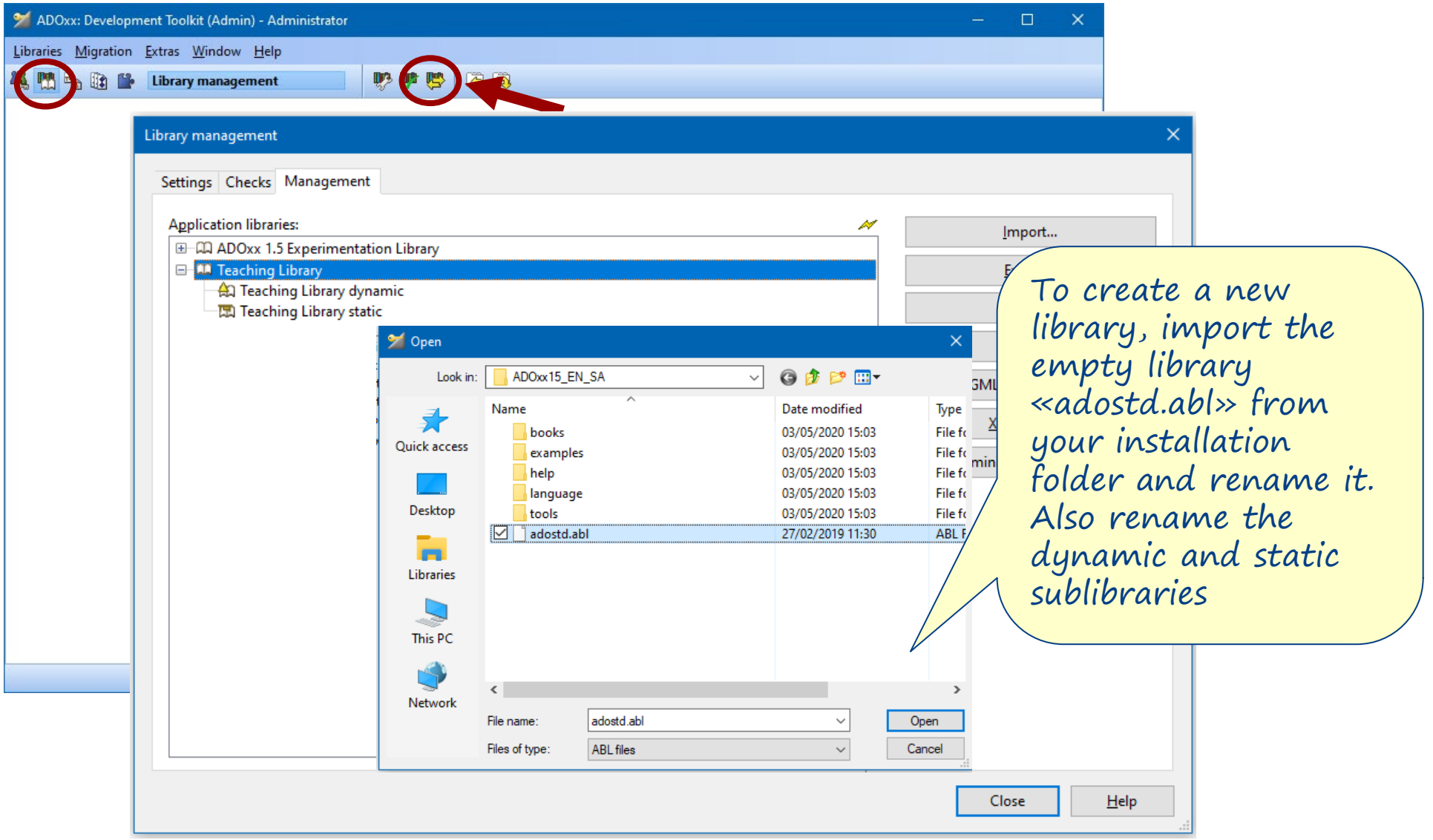
Import Modeling Language Libraries



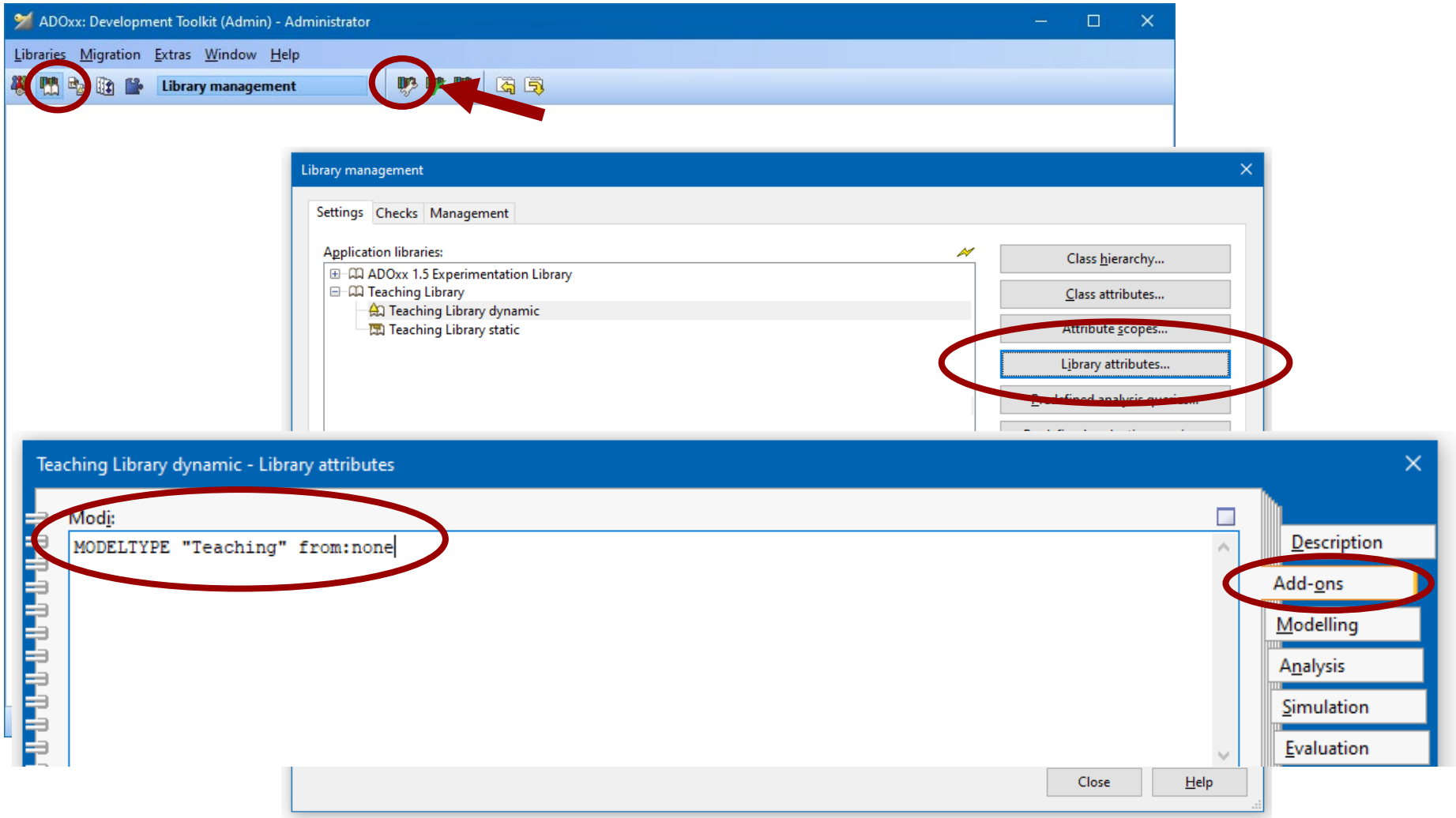
You can download conceptual modeling libraries from adoxx.org, e.g. BPMN,

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

Create a new Modeling Language Library



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

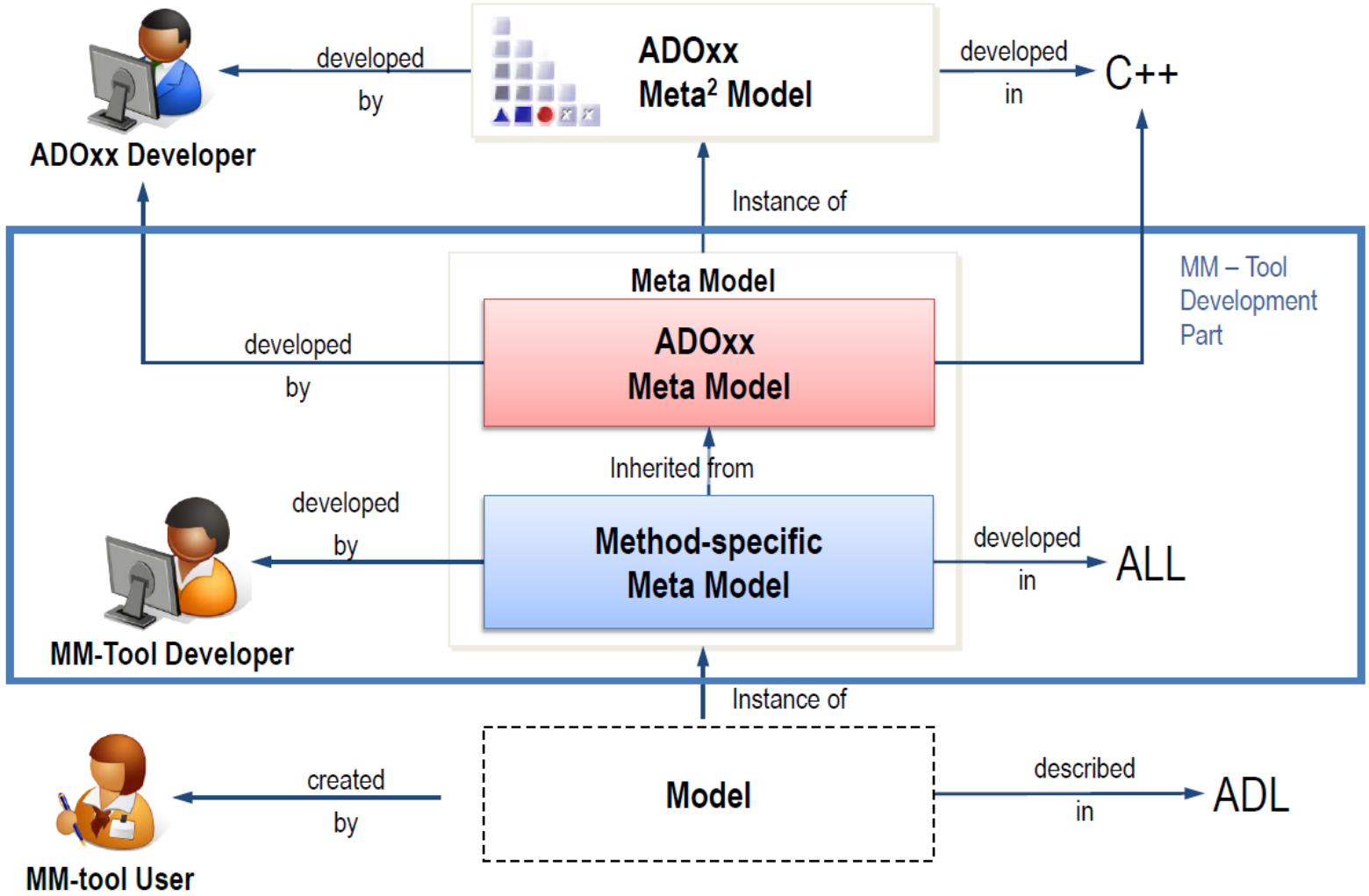


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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 . . . Agile development of ADOxx platform in parallel to modelling method development
 MM-Tool Developer	Developing an Meta Modelling Tool	Domain Knowledge Method Knowledge Platform Knowledge	
 ADOxx Developer	Implementation of tool specific and ADOxx functionality	Platform Knowledge ADOxx Technology Skills	

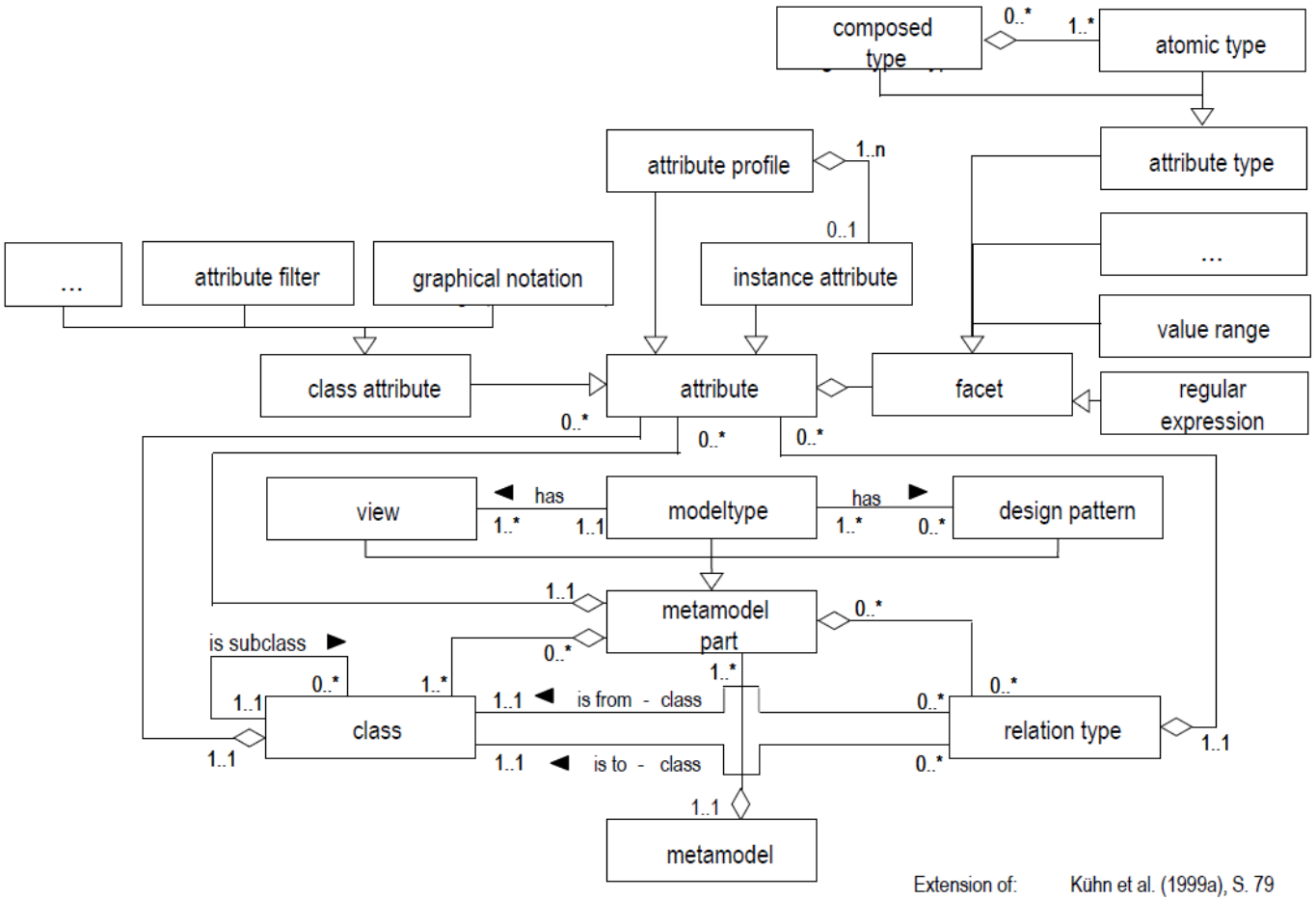
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Meta Modelling Platforms Hierarchyin ADOxx



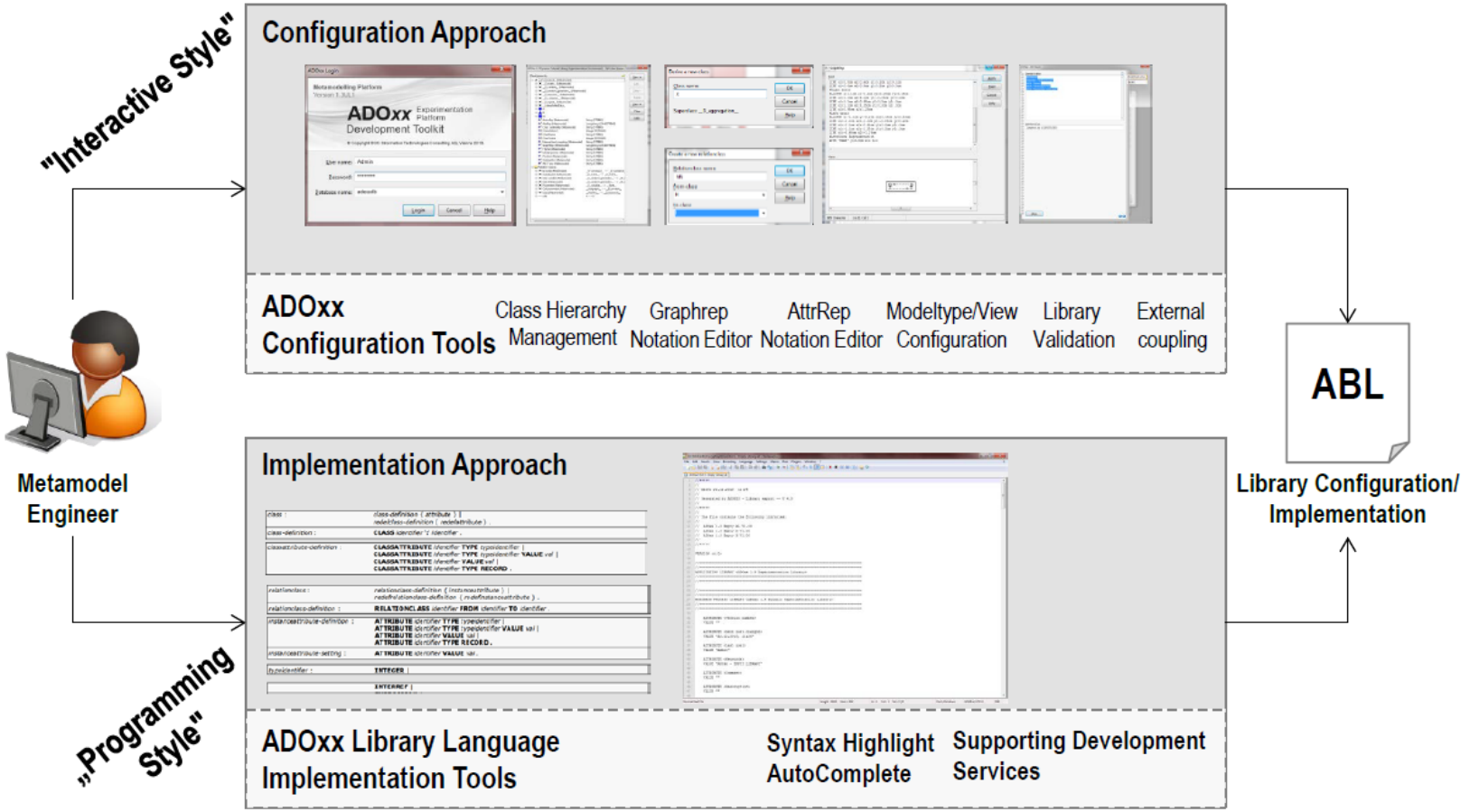
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Meta² Model: Meta Model of Meta Modelling Language



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

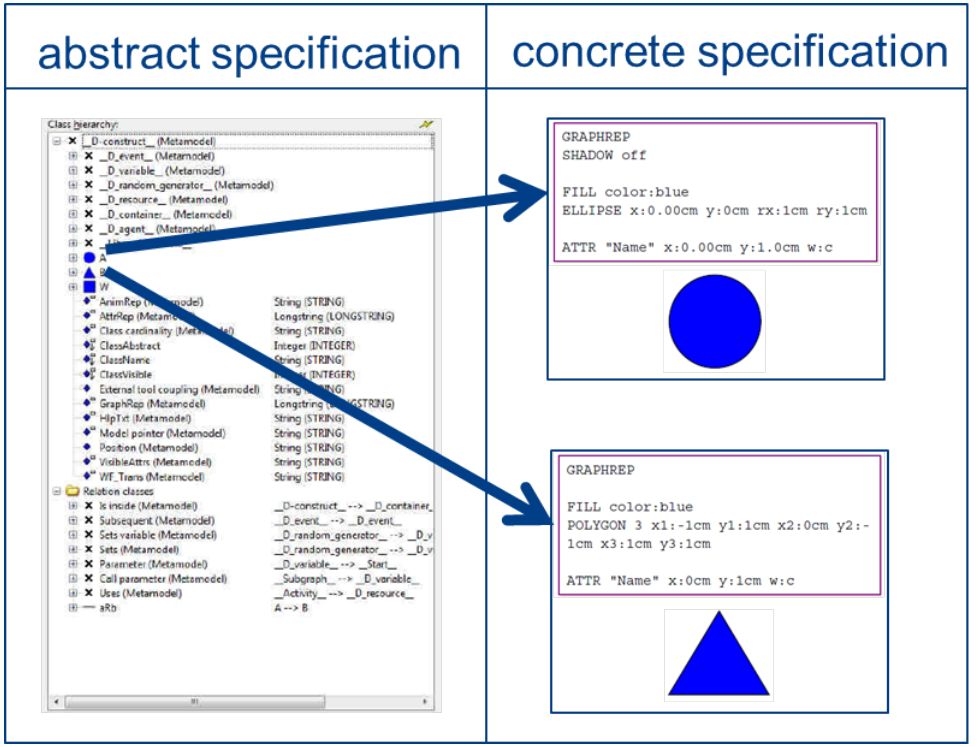
Development Approaches in ADOxx – Configuration and Implementation



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

Metamodel and Modeling Language in ADOxx

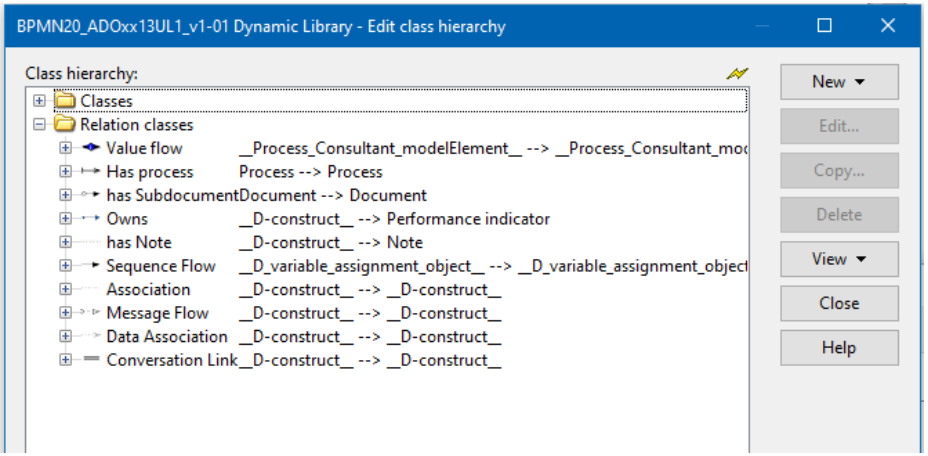
- The meta model of a model language is defined by
  - Classes of elements and relations
  - Class hierarchy
  - Attributes of the elements
- The notation is defined by
  - special attribute GraphRep

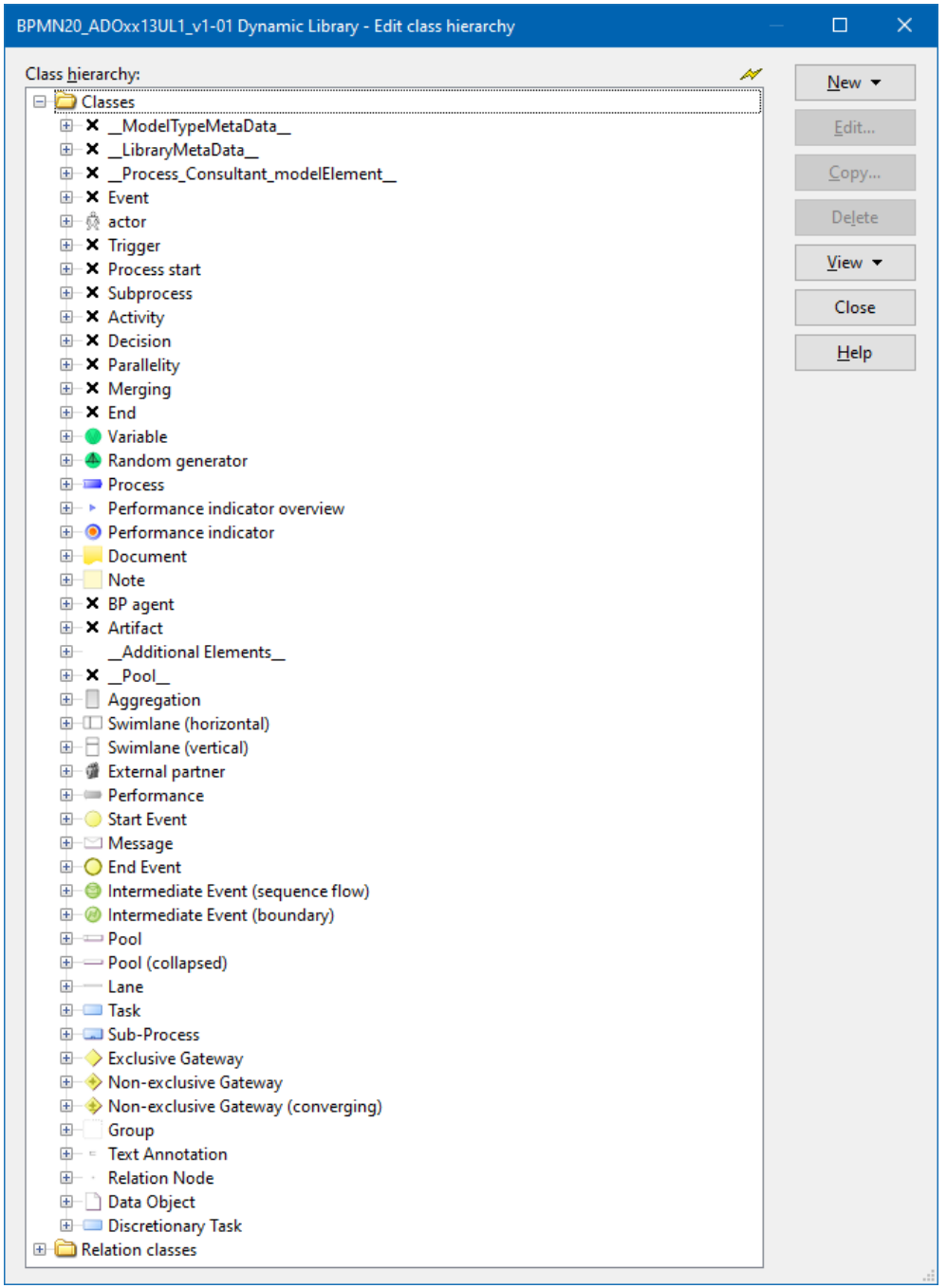


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Class Hierarchies

- ADOxx distinguishes
  - Classes
  - Relation classes

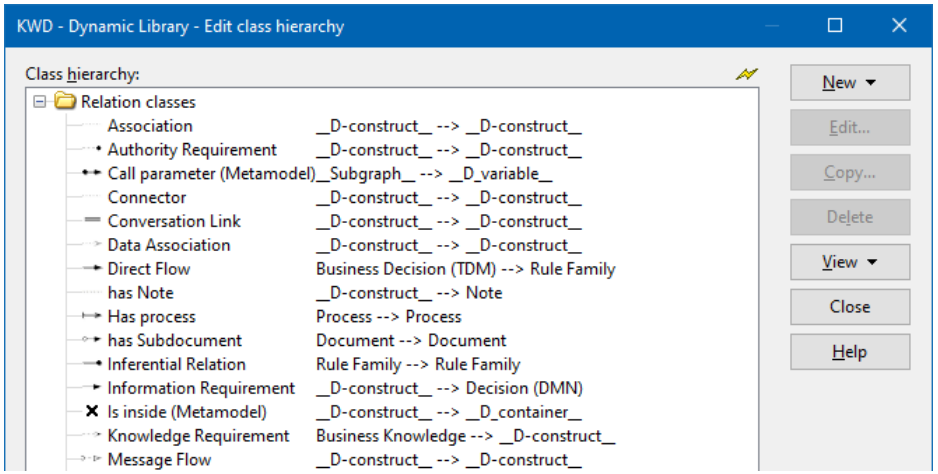


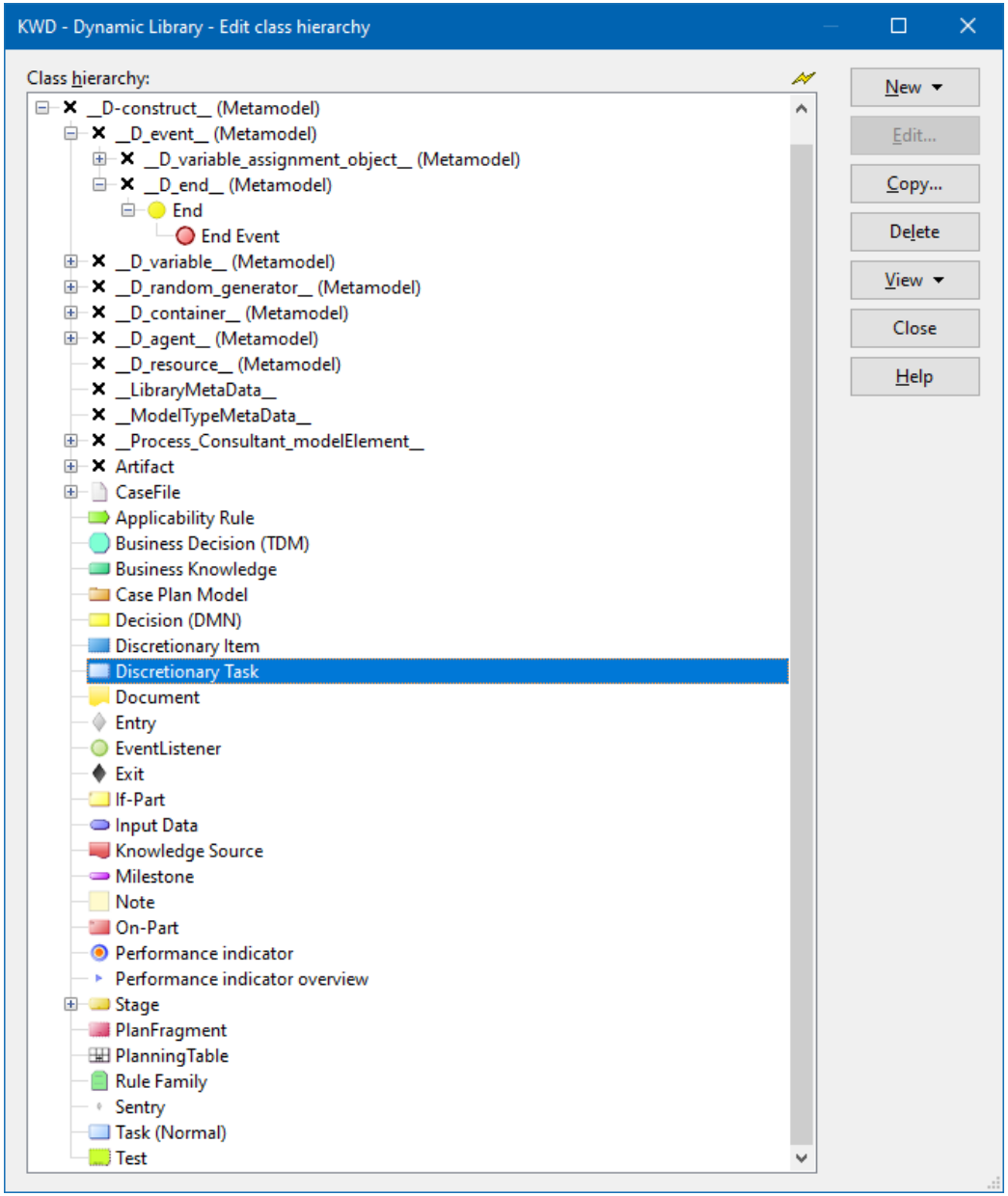


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Class Hierarchies

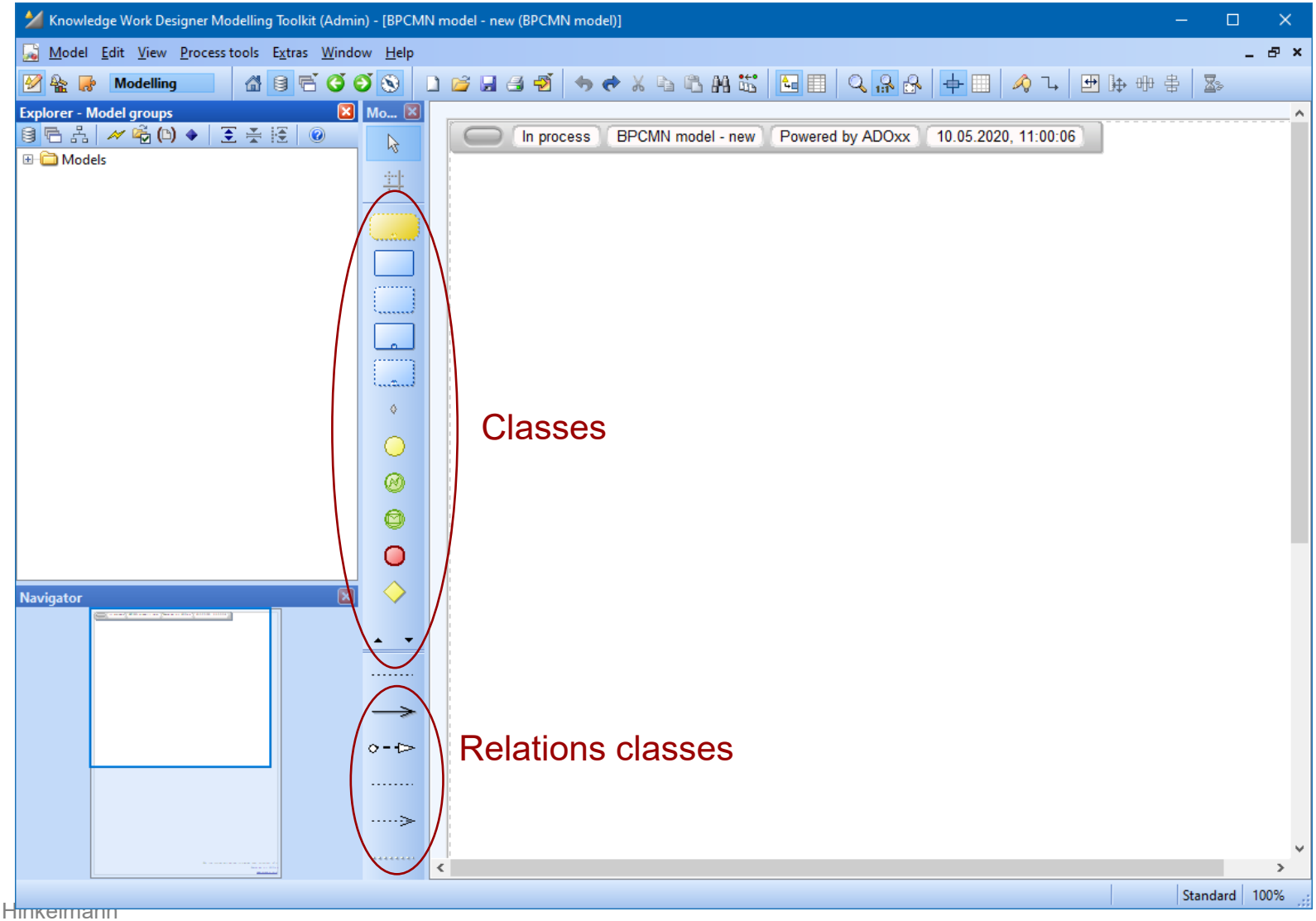
- ADOxx distinguishes
  - Classes
  - Relation classes





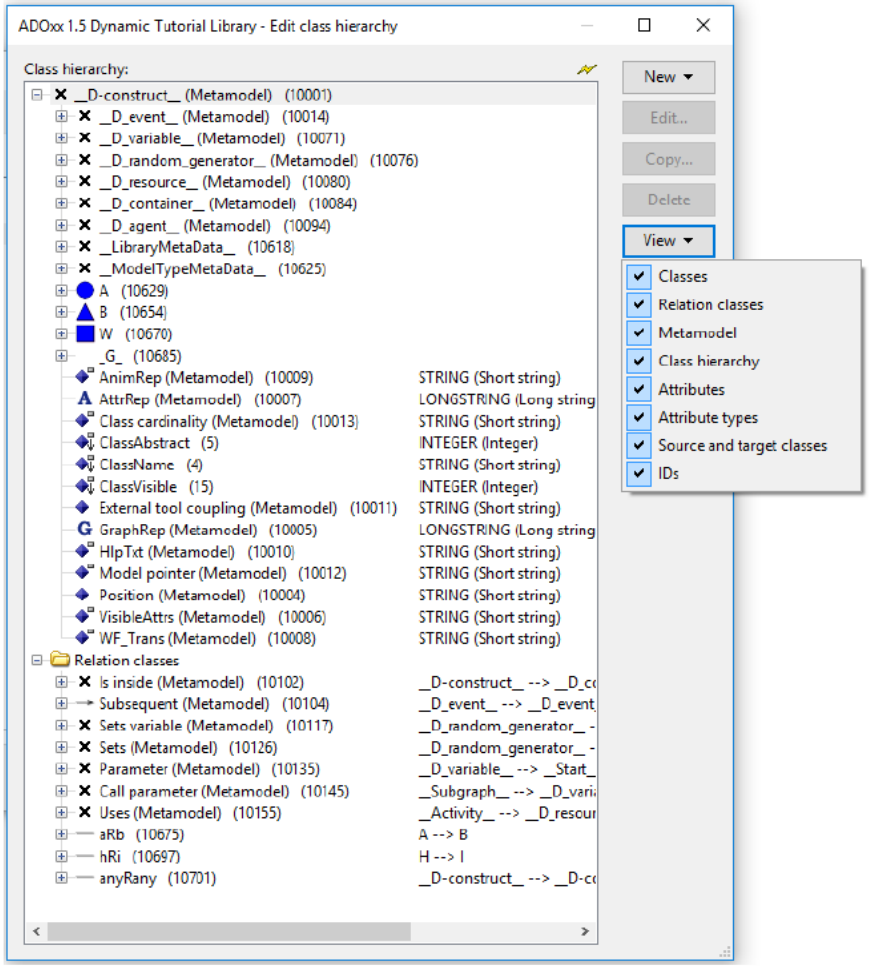
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Appearance of Classes in the Modelling Toolkit



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

Views of the Class Hierarchy



Classes

All visible classes will be shown

Relation classes

All available relation classes will be shown

Metamodel

All classes will be shown

Class hierarchy

All classes will be shown with their inheritance in a hierarchy

Attributes

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

Attribute types

The type of each attribute will be shown

Source- and Target-classes

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

IDs

Shows ID numbers of classes and attributes

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

Icons in Class Hierarchy

 **Class** (the icon shows the graphical definition of the object and can therefore vary)

 **Class** (without a graphical definition)

 **Attribute**

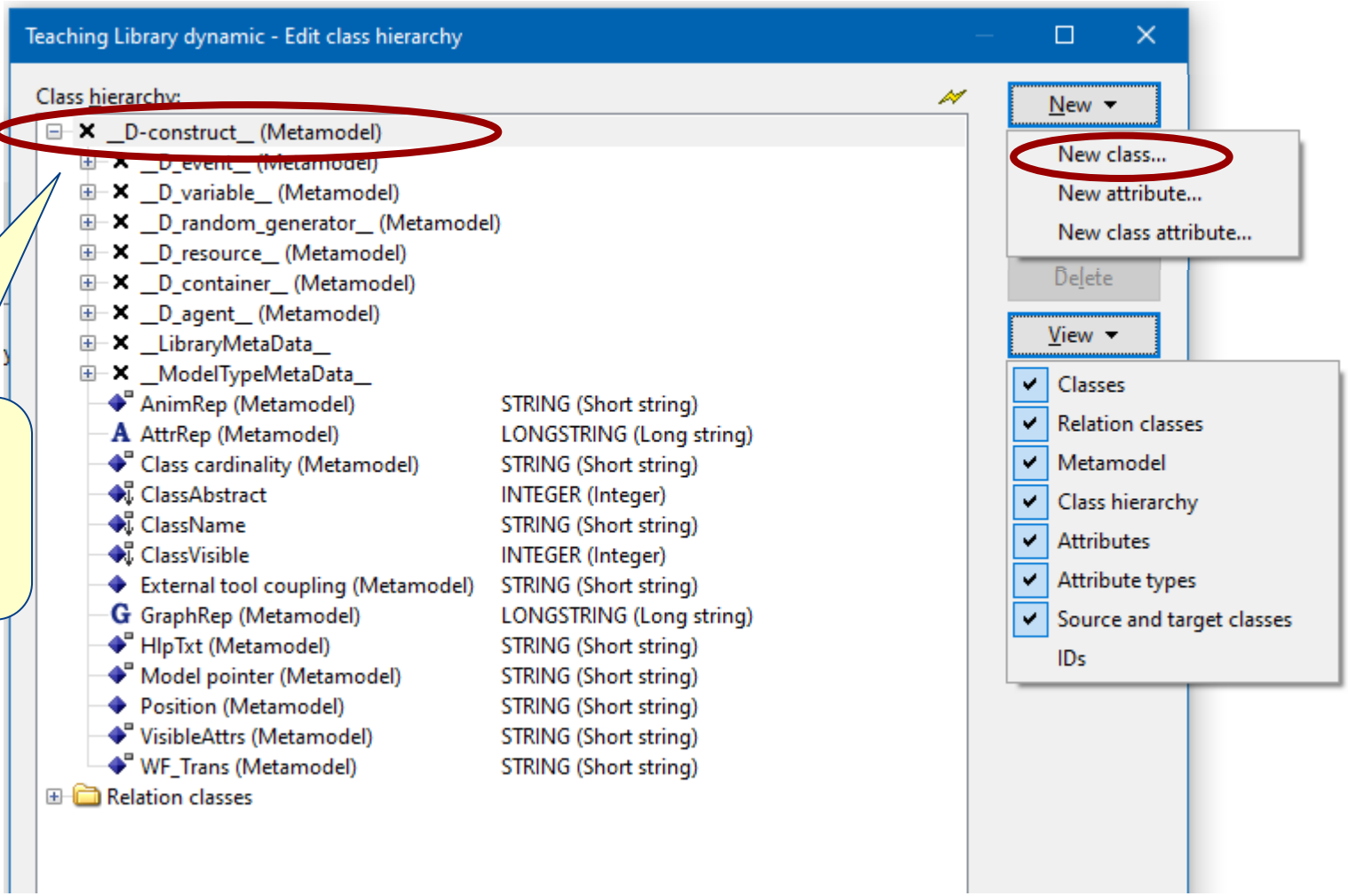
 **Attribute** (inherited from another class)

 **Class attribute**

 **Class attribute** (inherited from another class)

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

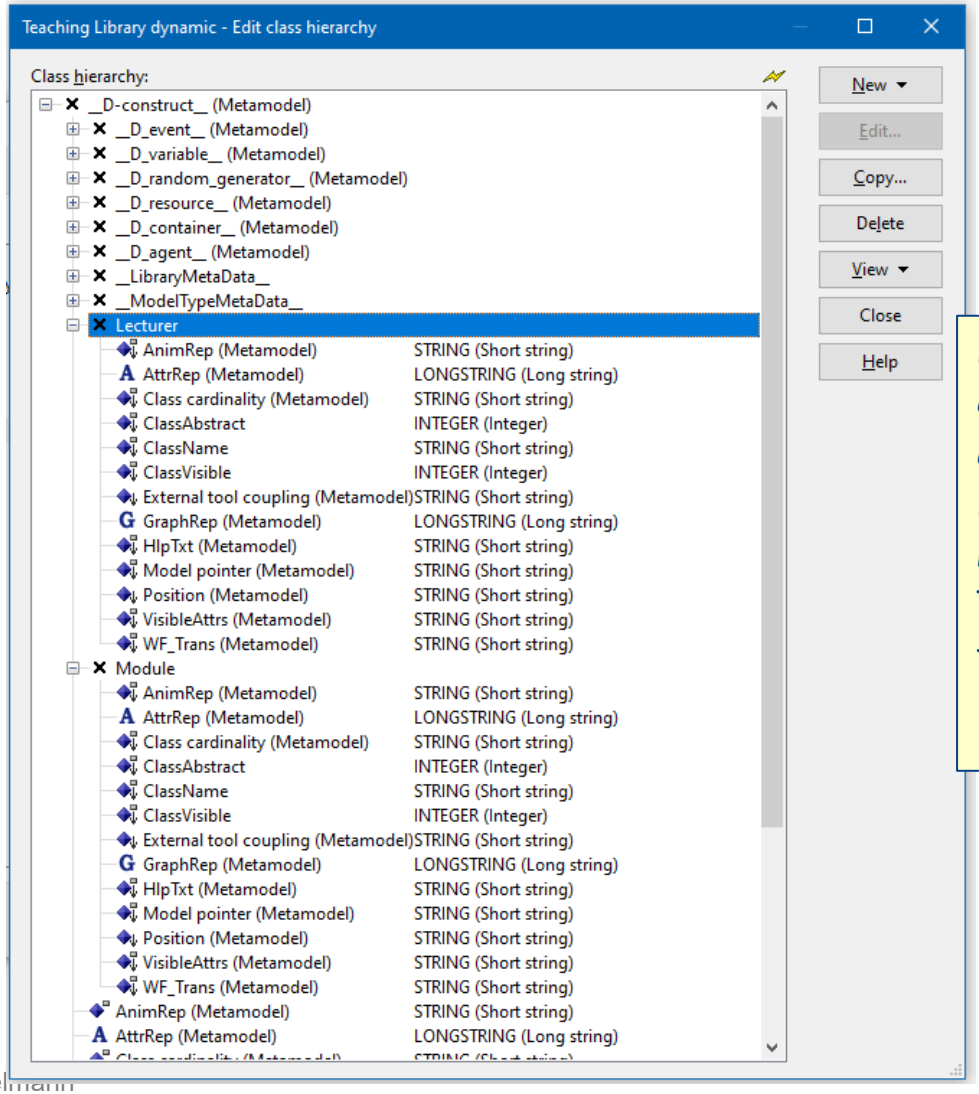
Creating new Classes



There are predefined abstract classes which have specific functionaliy

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

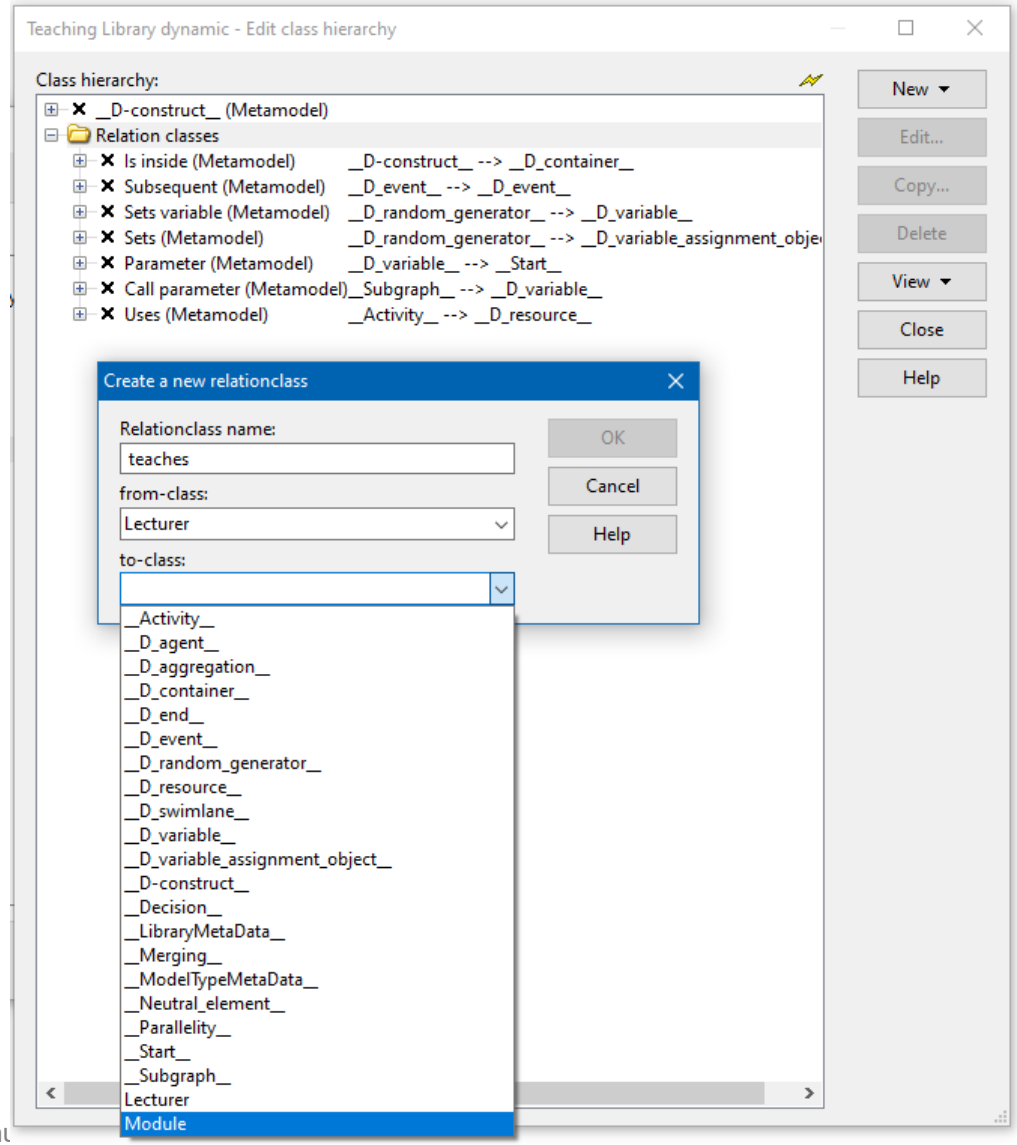
New Classes for Lecturer and Module



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

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

Defininig a new Relation

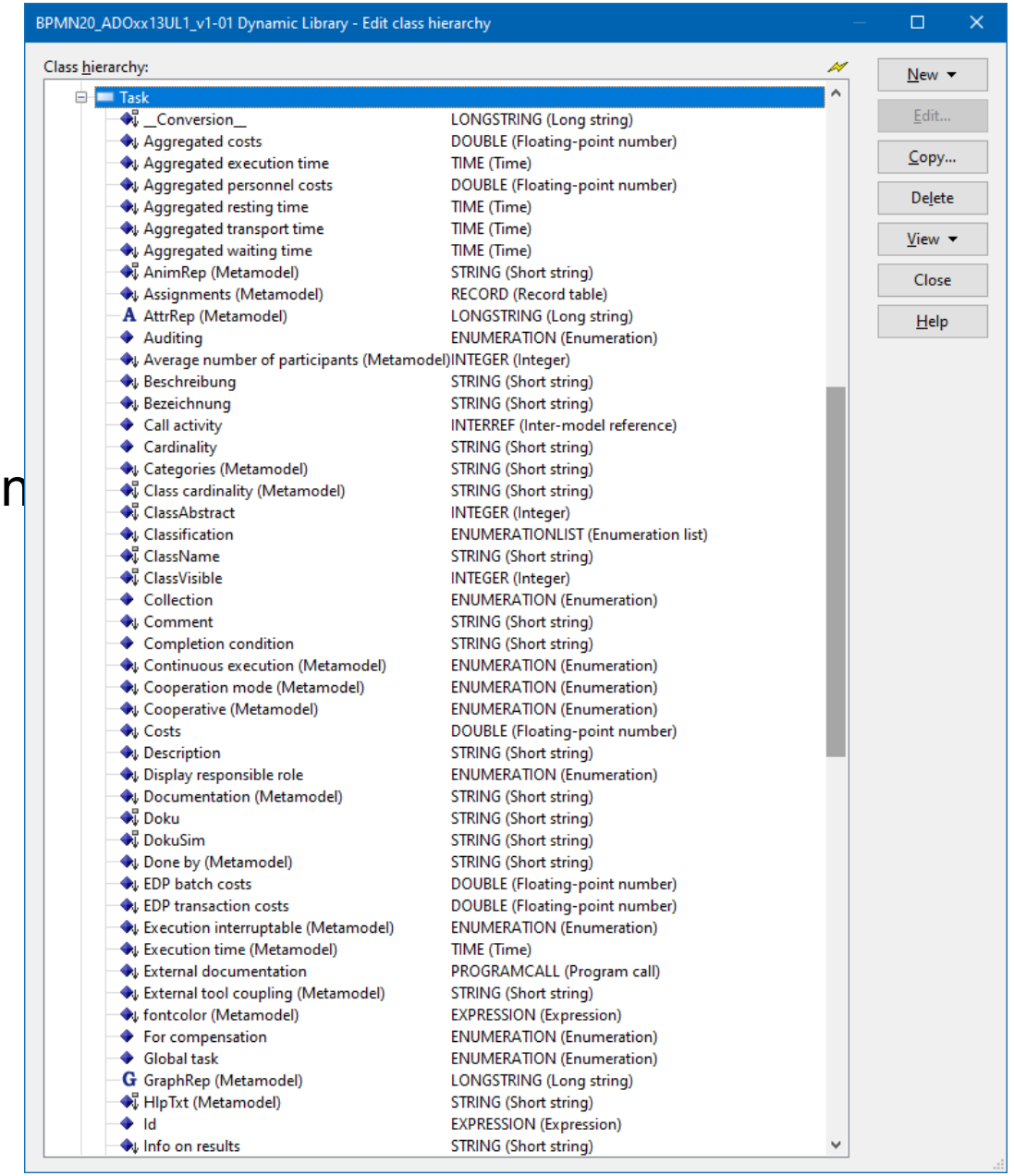


Example: A new relation «teaches» for elements from class «Lecturer» to class «Module»

Prof. Dr. Knu
knut.hinkelmann@fhnw.ch

Attributes

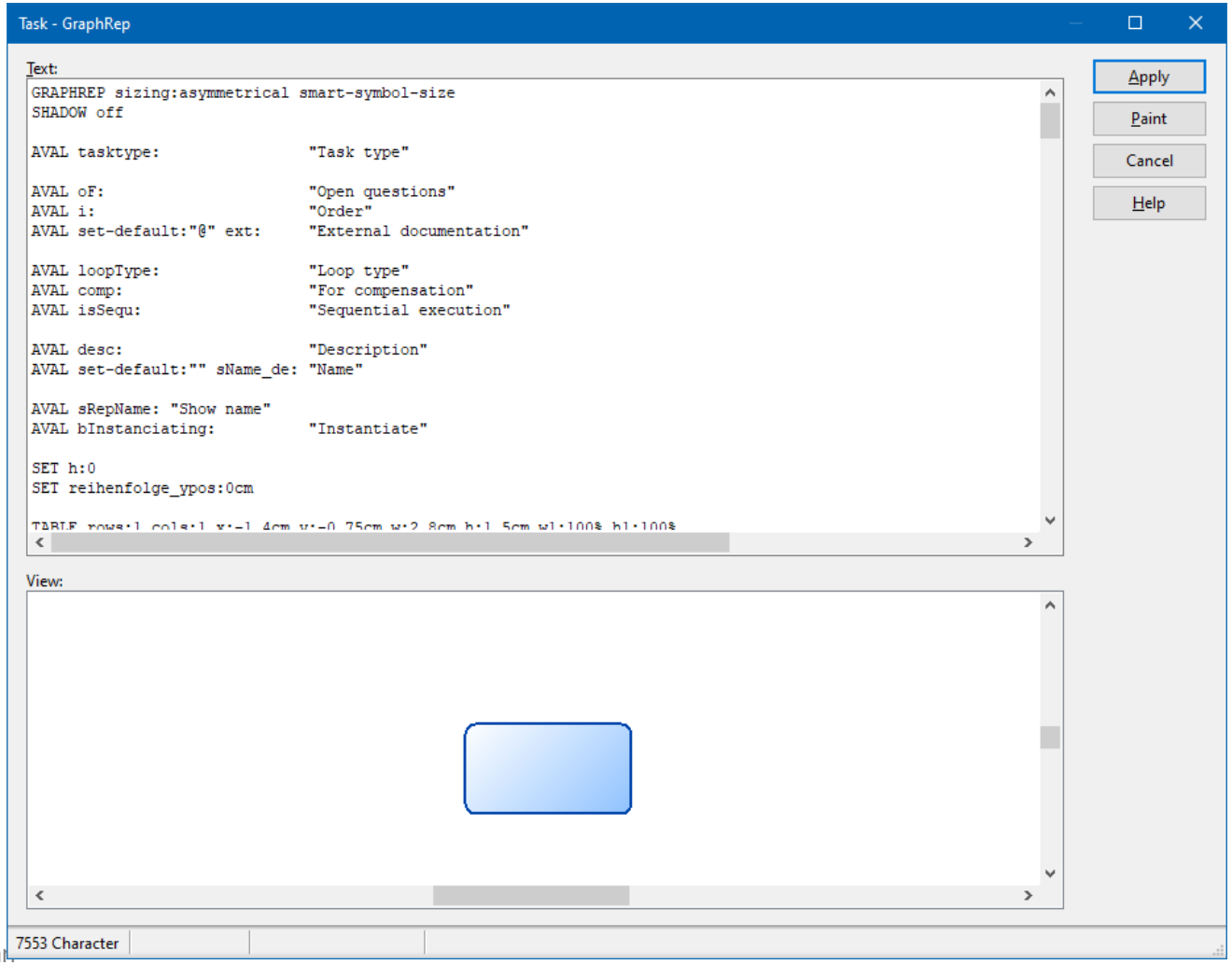
- Kinds of Attributes
  - Properties of Models
  - Graphical Representation
  - References



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

Special Attribute GraphRep

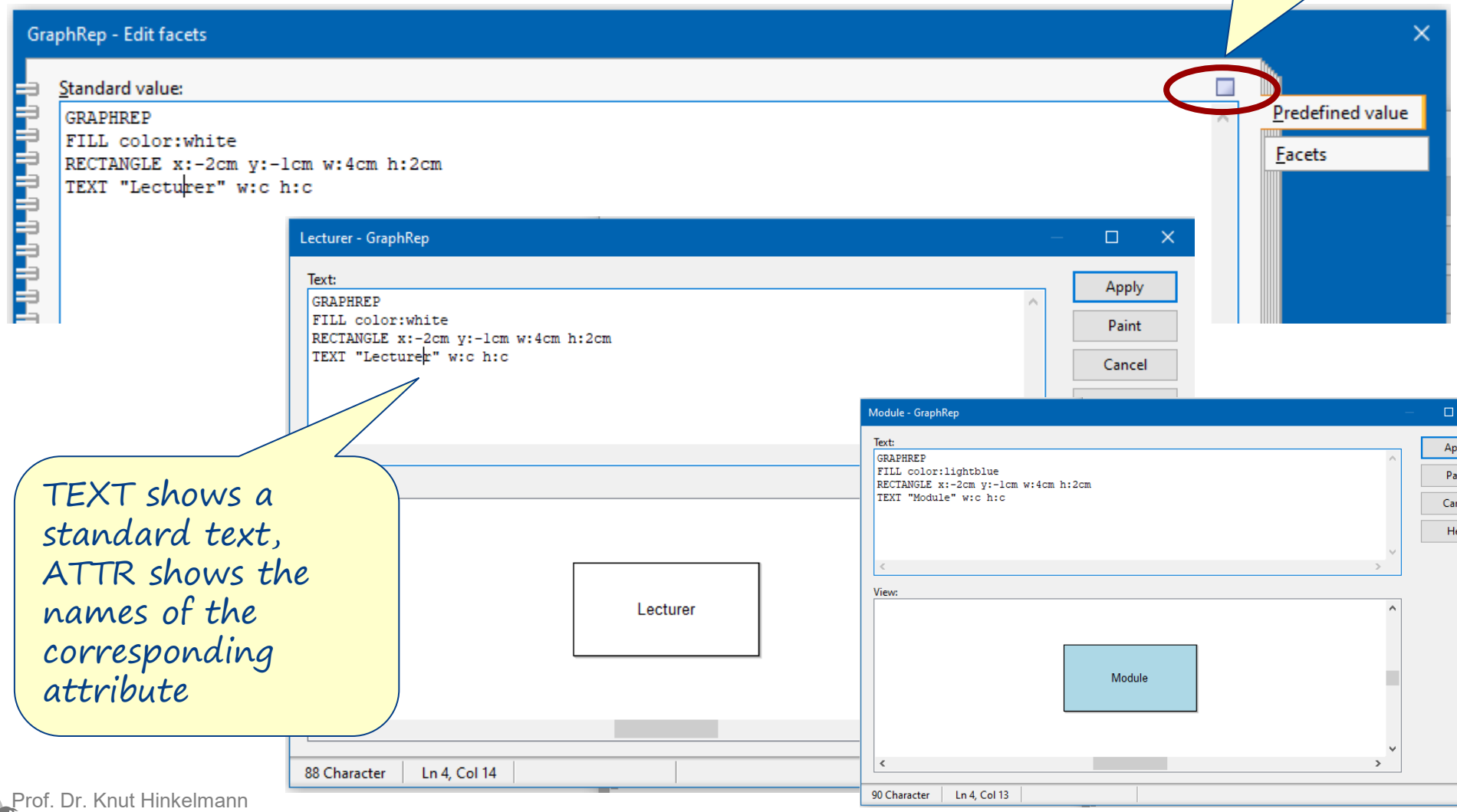
GraphRep: A script language for the graphical representation



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

Defining a GraphRep

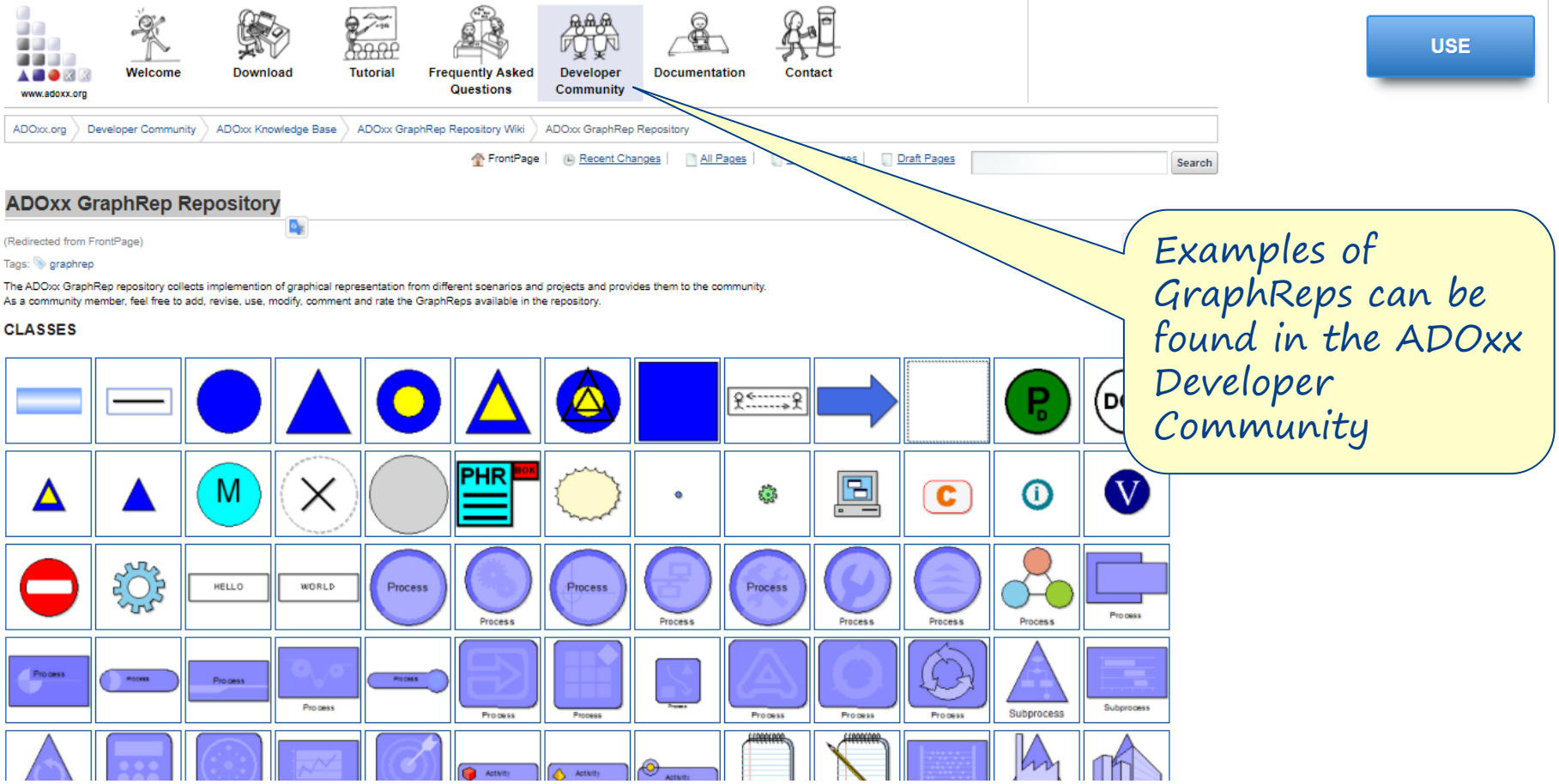
With the help button you can define and test the graphics



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

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

ADOxx GraphRep Repository



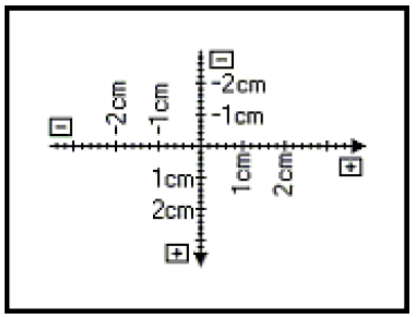
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

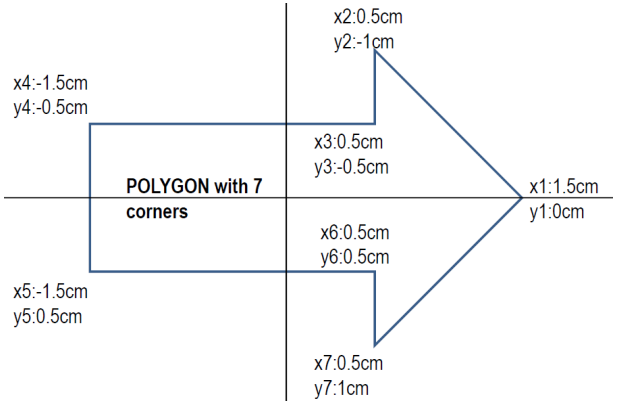
GraphRep Elements

- 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





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

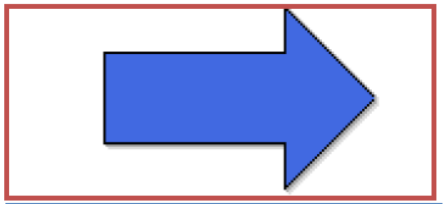
GraphRep Examples

```
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
```

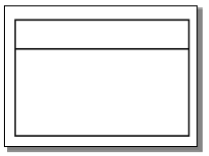
```
GRAPHREP
FILL color:royalblue
POLYGON 7 x1:1.5cm y1:0cm x2:0.5cm
y2:-1cm x3:0.5cm y3:-0.5cm x4:-1.5cm
y4:-0.5cm x5:-1.5cm y5:0.5cm
x6:0.5cm y6:0.5cm x7:0.5cm y7:1cm
ATTR "Name" y:1.4cm w:c h:c
```



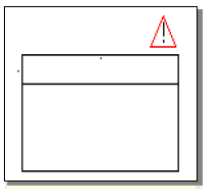
In case attribute name is available, it is shown here

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



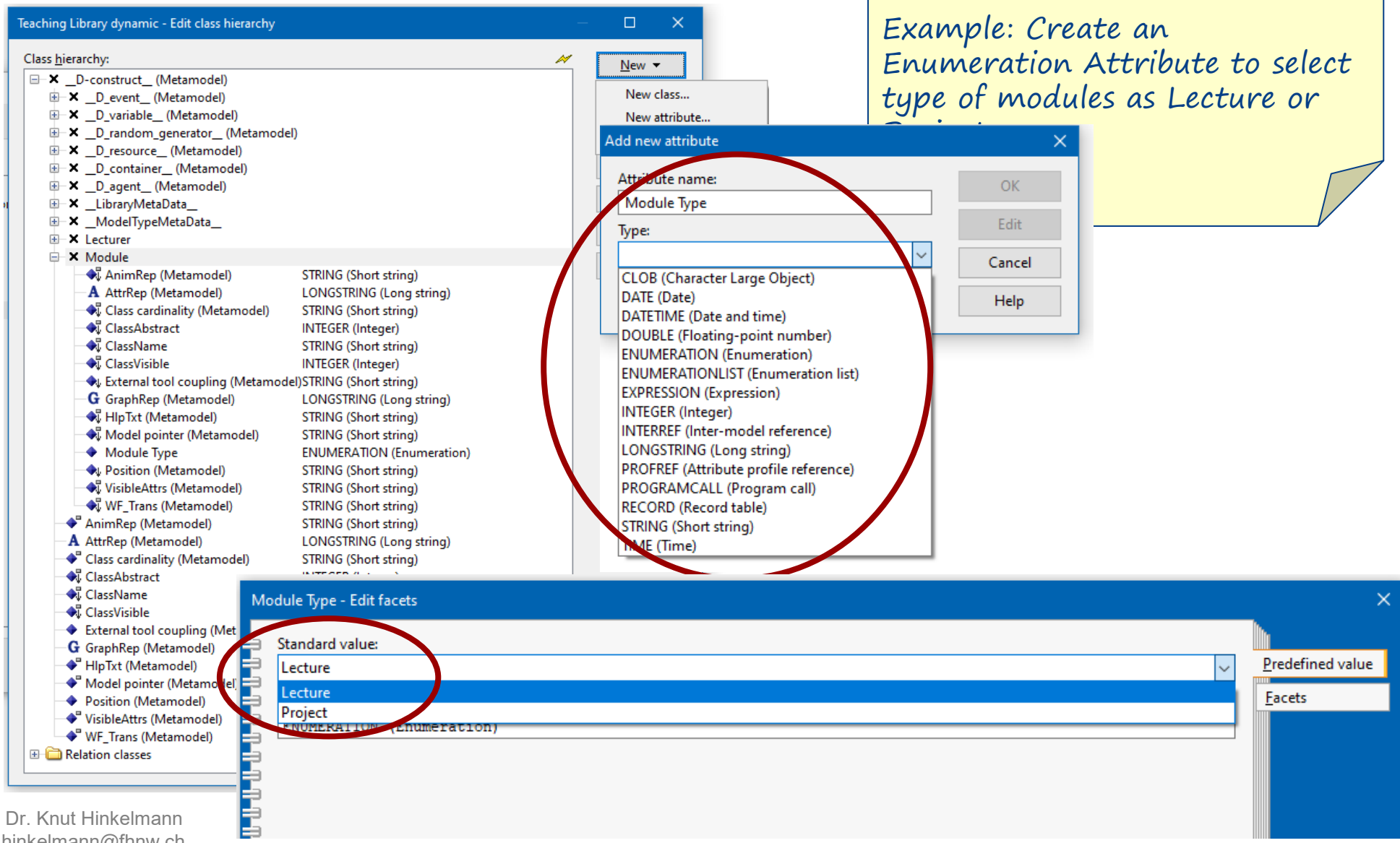
Condition not fulfilled

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

Defining a new Attribute

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

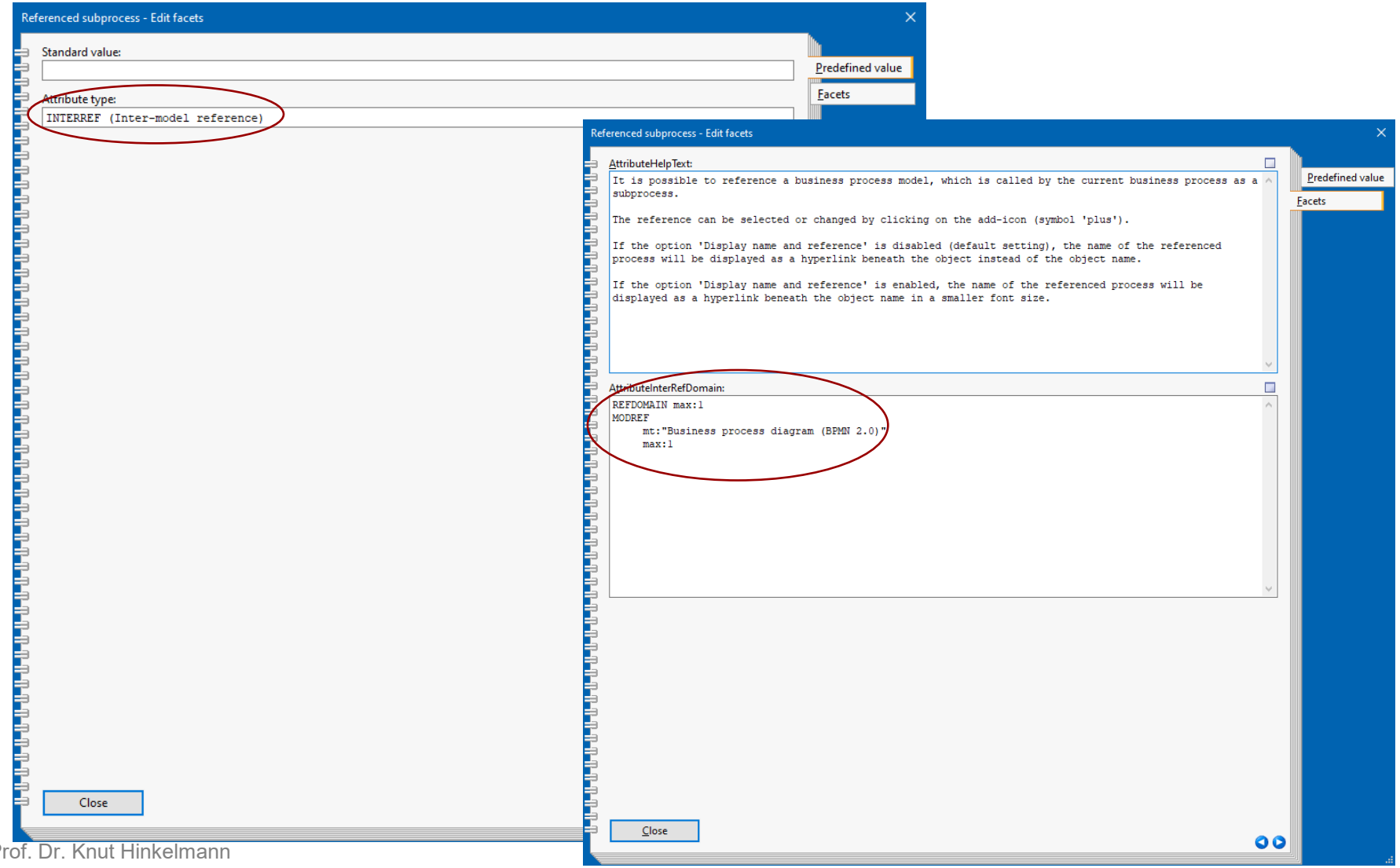
Example: Create an Enumeration Attribute to select type of modules as Lecture or



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

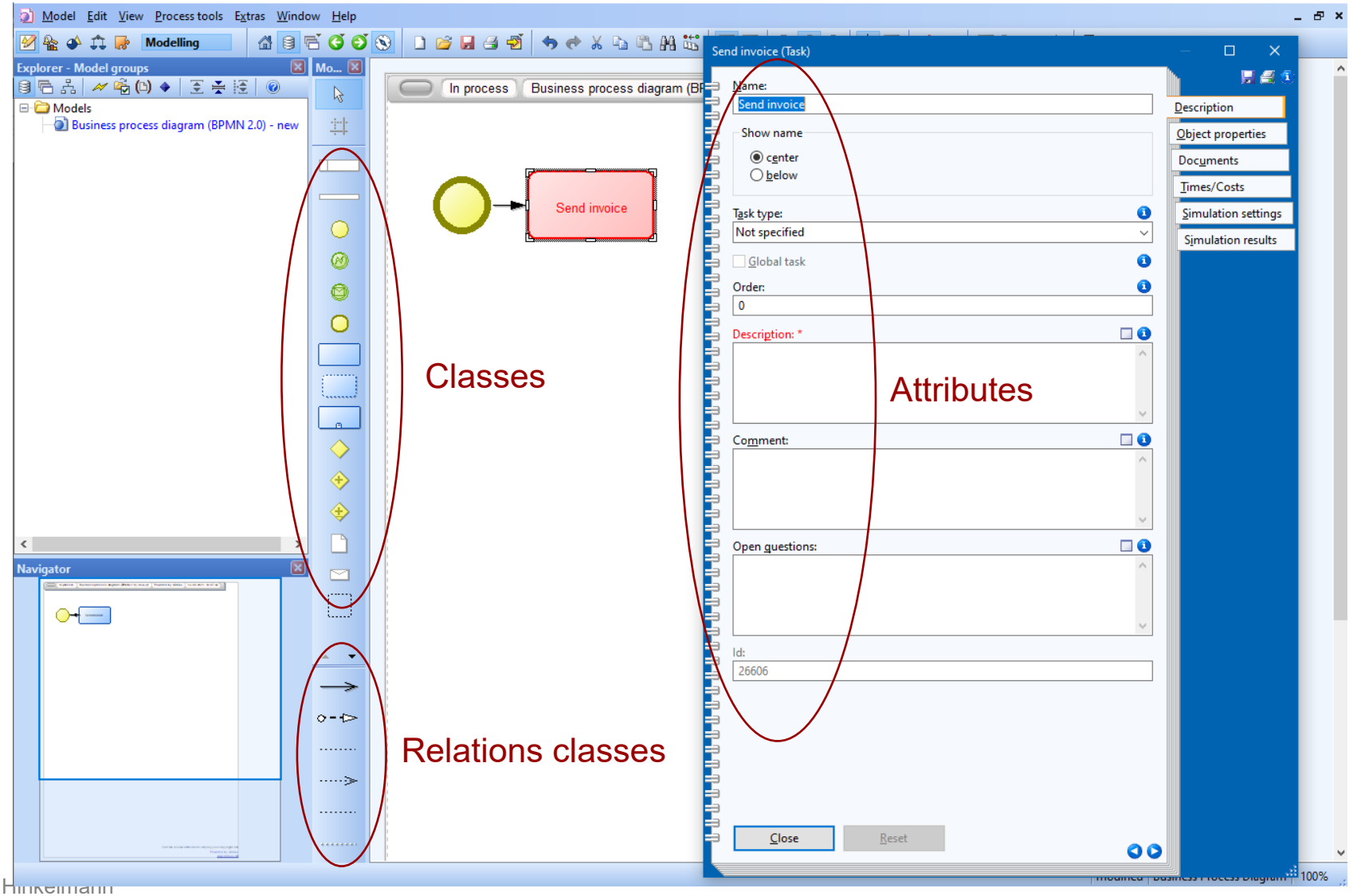
References

Referencing a Subprocess



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

Appearance of Classes in the Modelling Toolkit



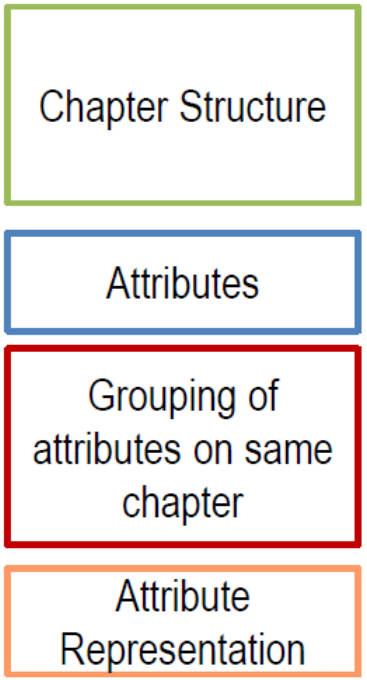
AttrRep

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

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

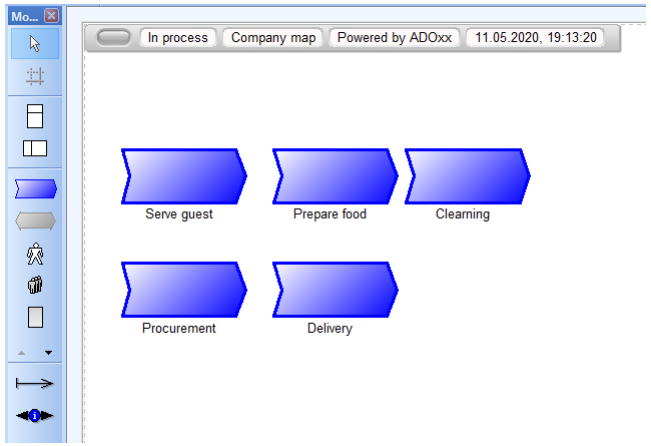
```
NOTEBOOK
CHAPTER "Definition"
ATTR "Name"
ATTR "Description"
CHAPTER "Dialectic Influence"
ATTR "Influencing dialectics" lines:10
```

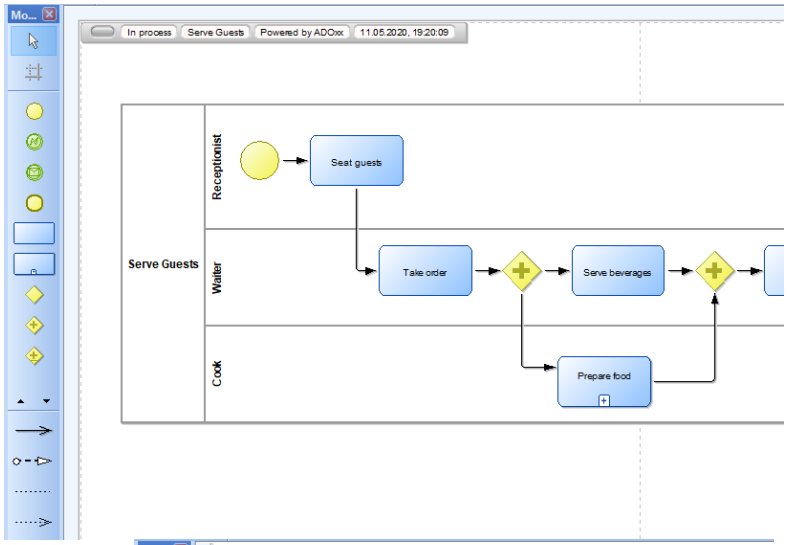
```
NOTEBOOK
CHAPTER "Definition"
ATTR "External graphic"
```

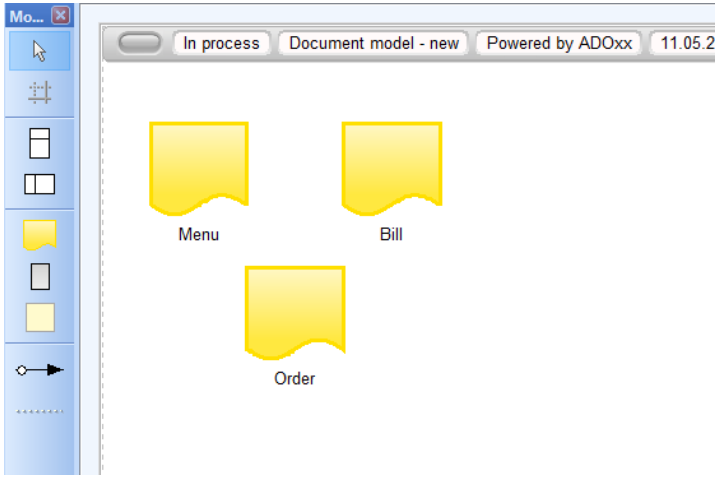


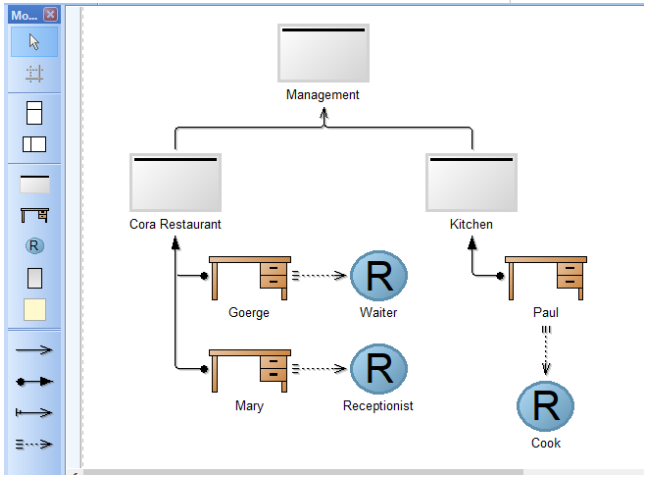
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Model Types: Represention Views on the Knowledge



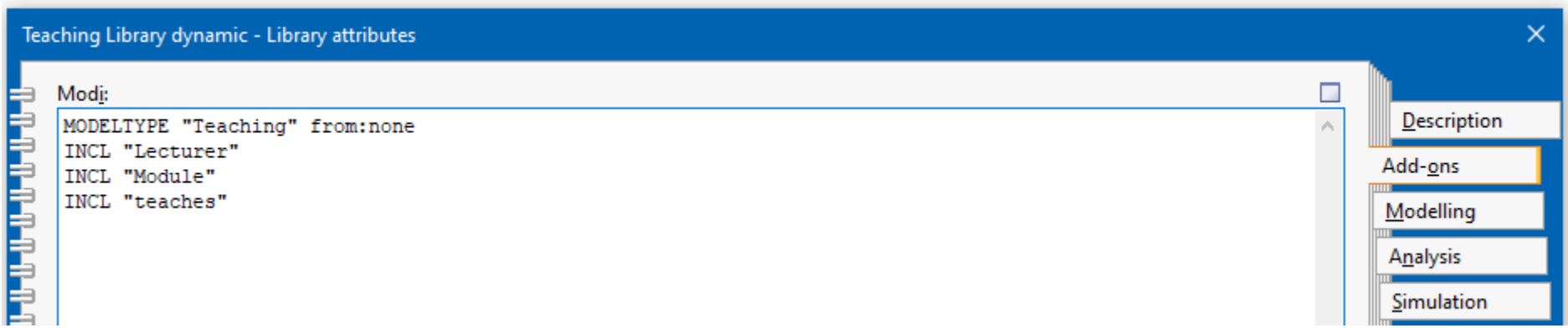






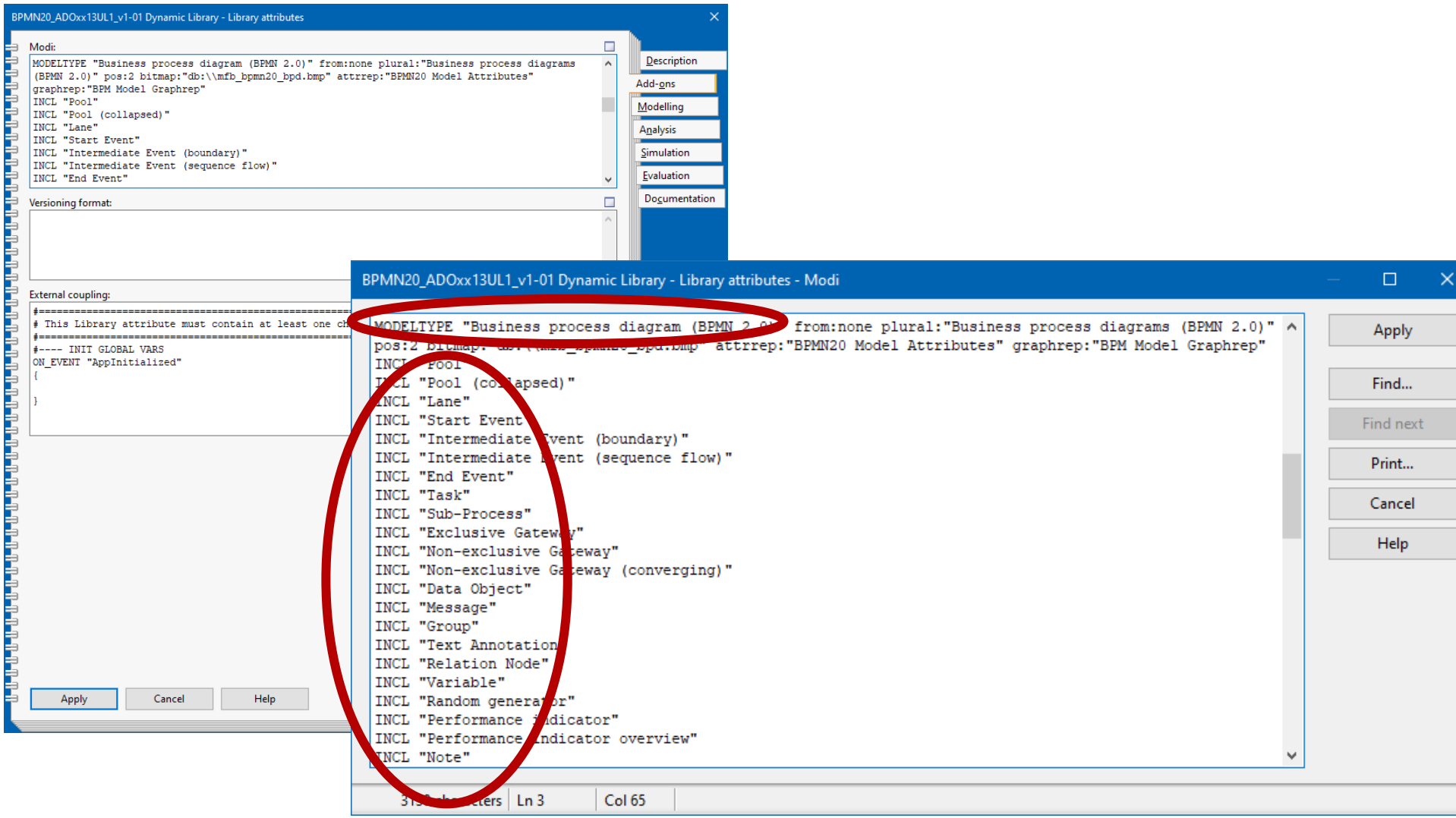
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Example



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

Classes are assigned to Model Types

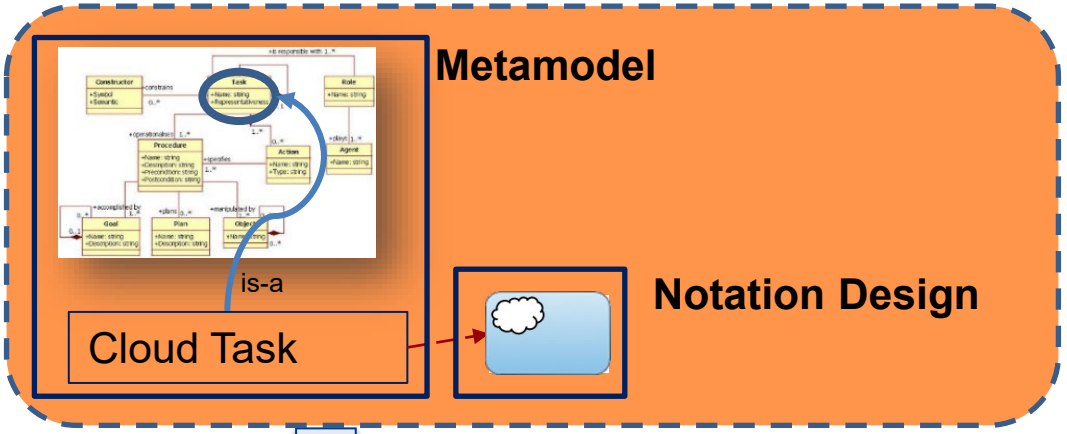


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Change of Metamodel

- Example: new task type Cloud Task

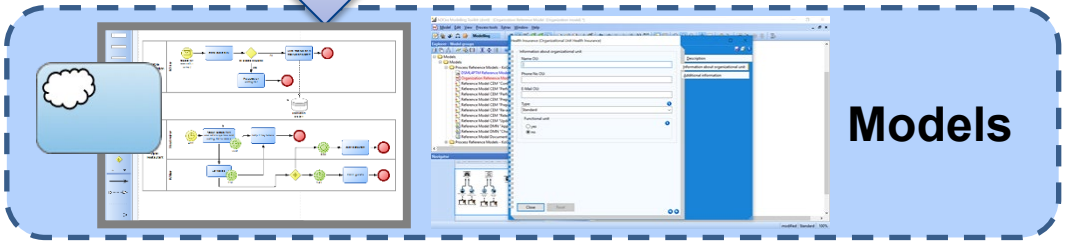
Metamodel Engineer



Meta-modeling

Feedback
Amendments
Improvements

Modeler



Modeling