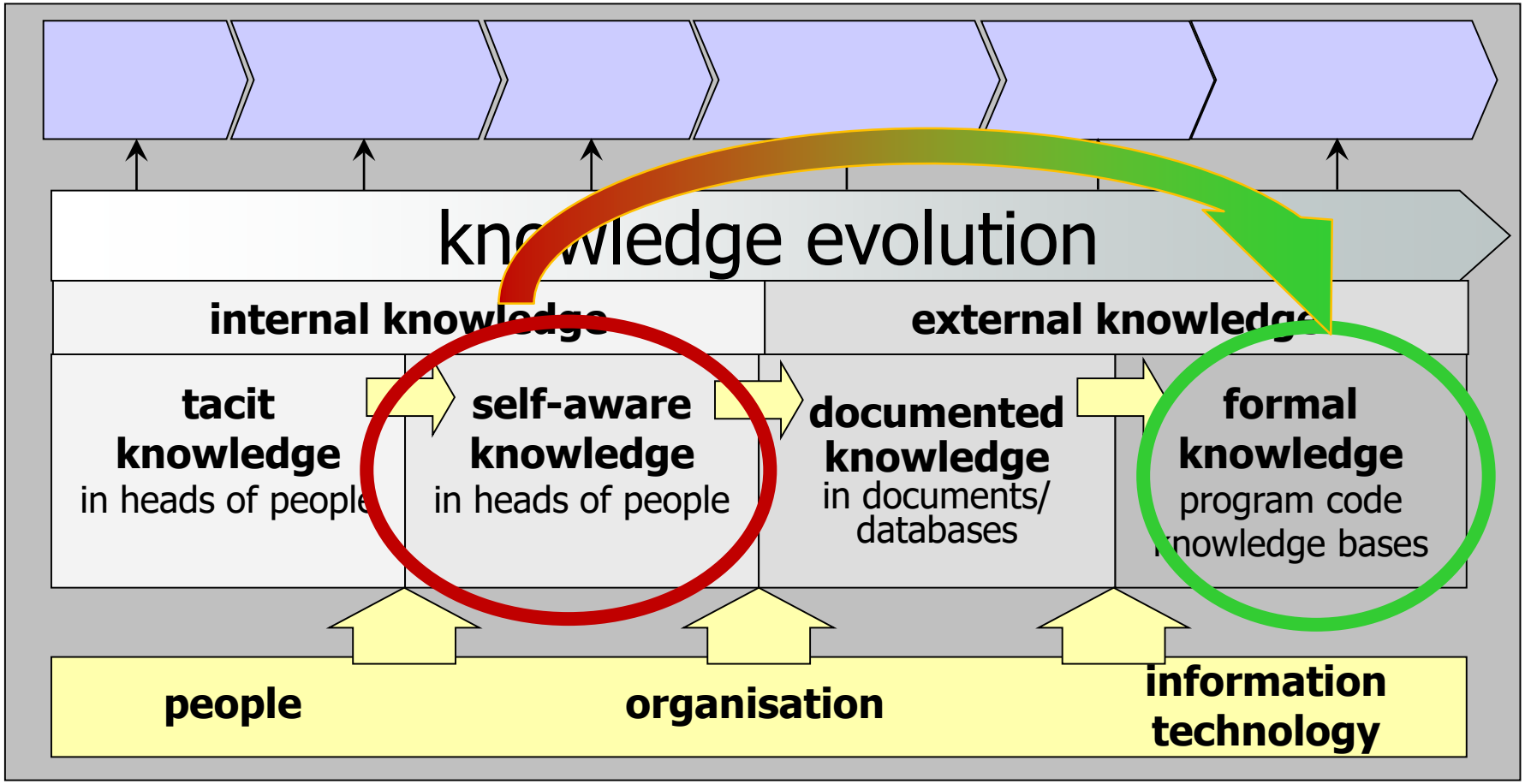
Modelling and Metamodelling

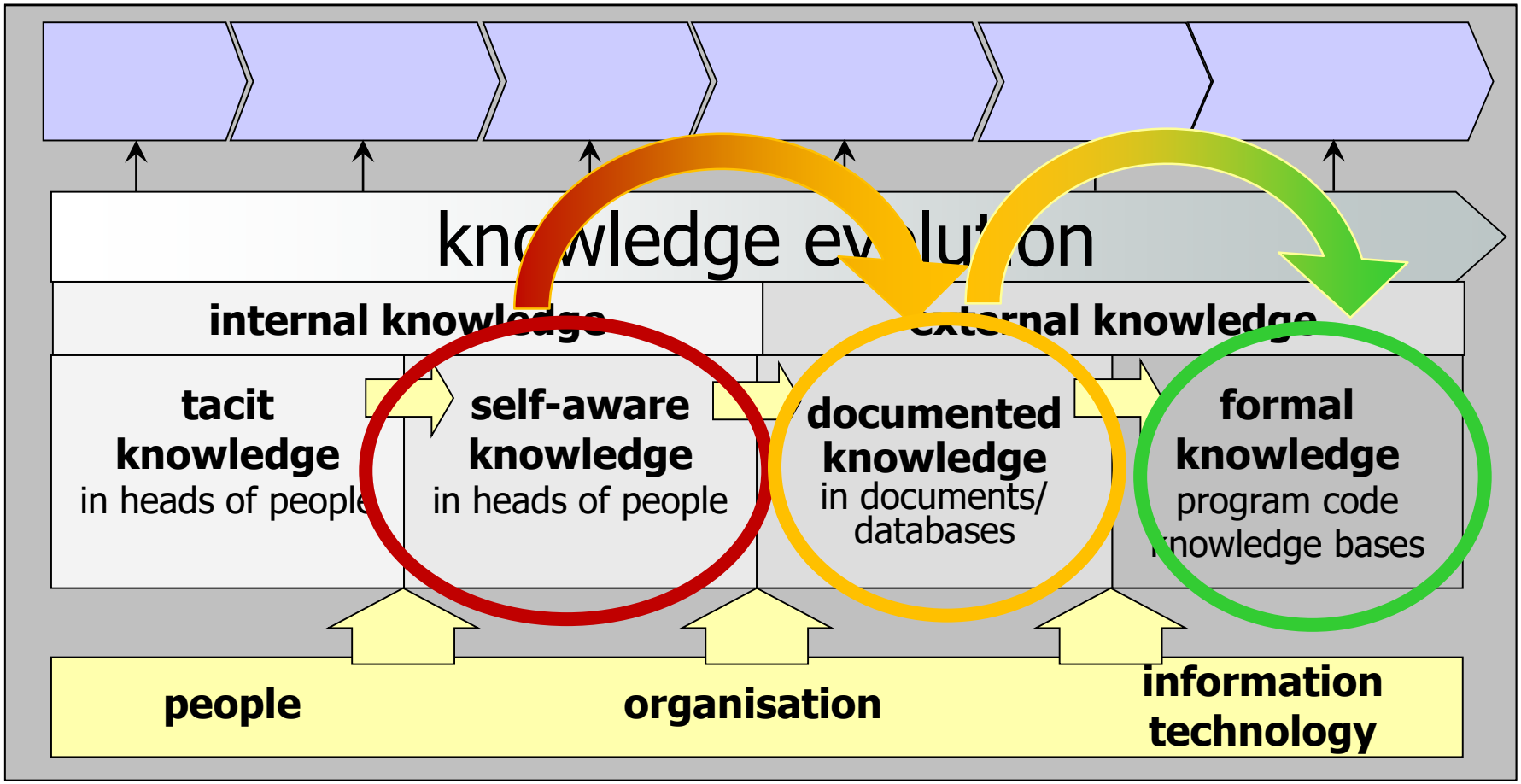
Knut Hinkelmann

Ontology Engineering



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

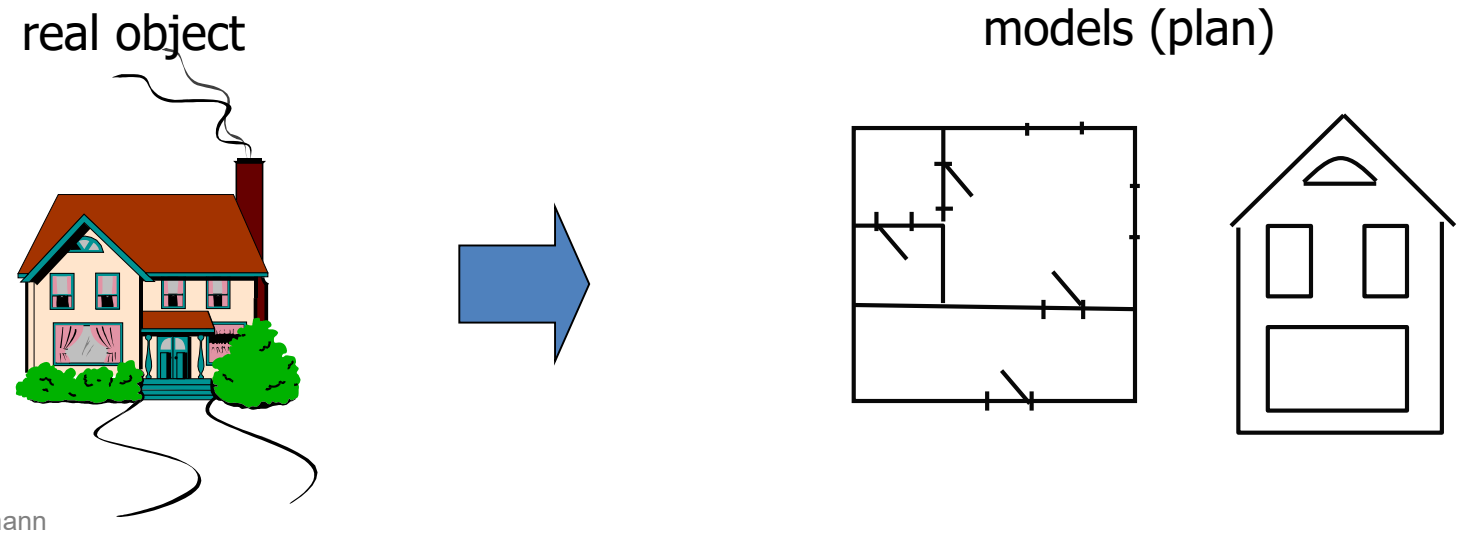
A Two-step Approach for Building a Knowledge Base



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

Models

- A Model is a reproduction of a *relevant* part of reality which contains the essential aspects to be investigated.
- Relevance depends on the
  - purpose (also called concern or goal)
  - stakeholders

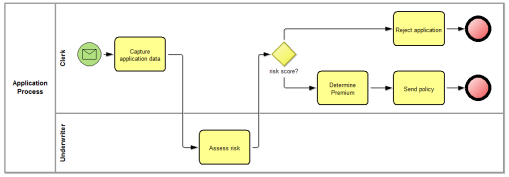


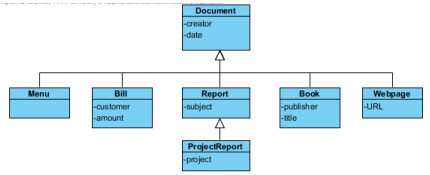
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Models

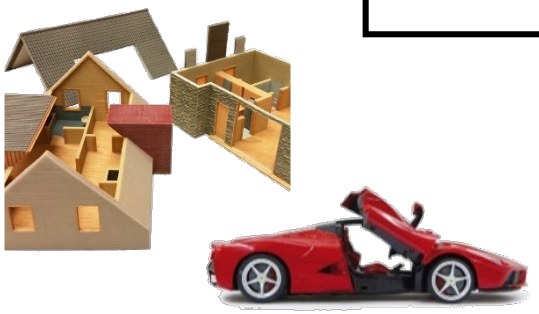
The Application Process
In the business process for health insurance, first the application data are captured by the clerk. Then the risk assessment is made by the underwriter. Depending on the risk score, the clerk determines the premiums and sends the policy or the application is rejected.

- There can be different kinds of models
  - textual model
  - graphical model
  - conceptual models
  - mathematical model
  - physical model





$$E = m\,c^2$$



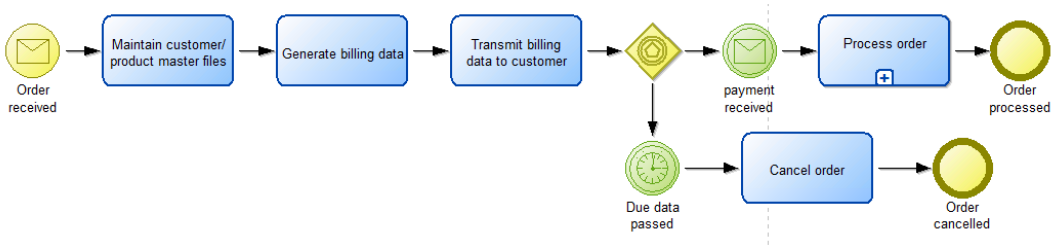
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

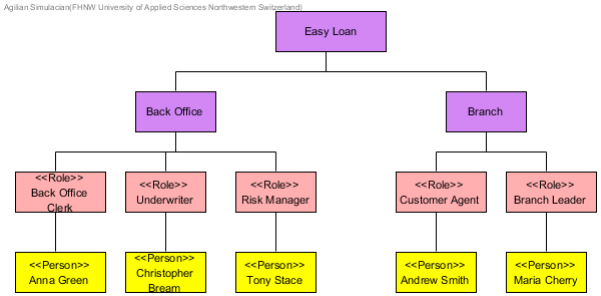
Visual Communication

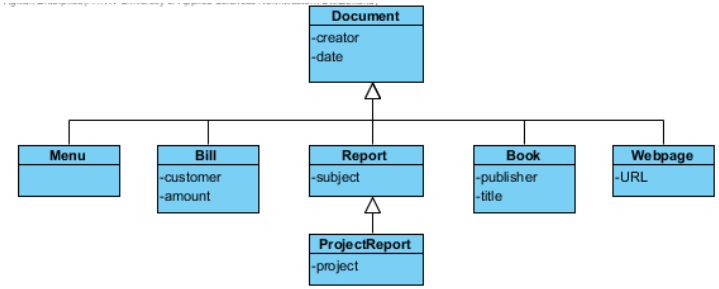
- A picture is worth a thousand words
- Graphical Models are easier to understand than text

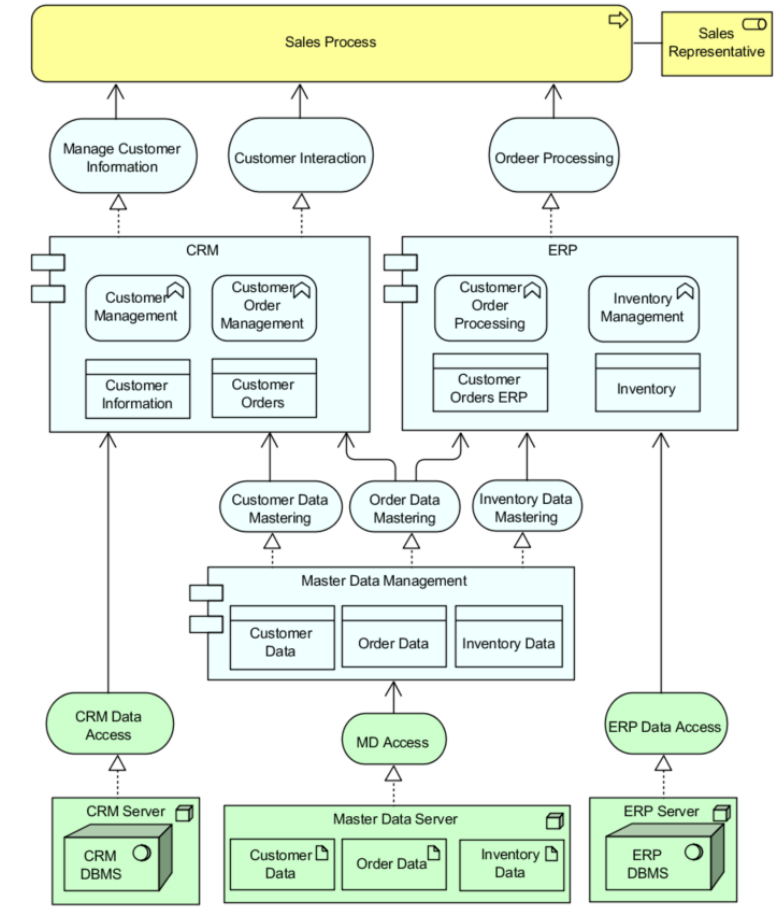
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Enterprise Models





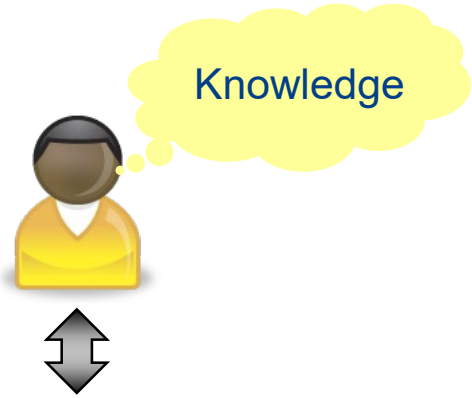




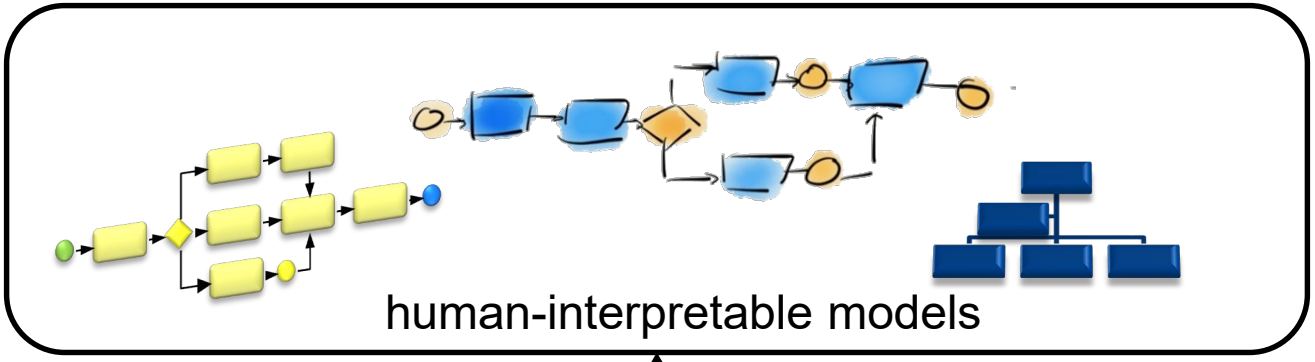
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Human Problem Solving

Communication/
Analysis/
Decision Making



Models



Reality

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

Models and Modelling

Model

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

Modelling

Describing and representing all relevant aspects of a domain in a defined ***modelling language***.

Result of modelling is a model.

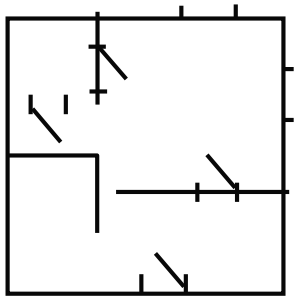
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Model in Architecture

real object

house

model



architect's drawing
(plan)

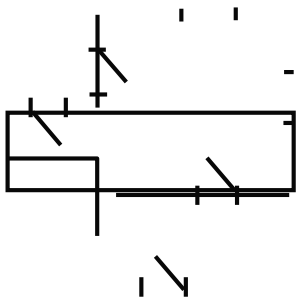
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Model and Modelling Language in Architecture

real object

house

model

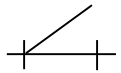


architect's drawing
(plan)

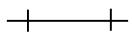
modelling language
(concrete syntax)

object types:

wall



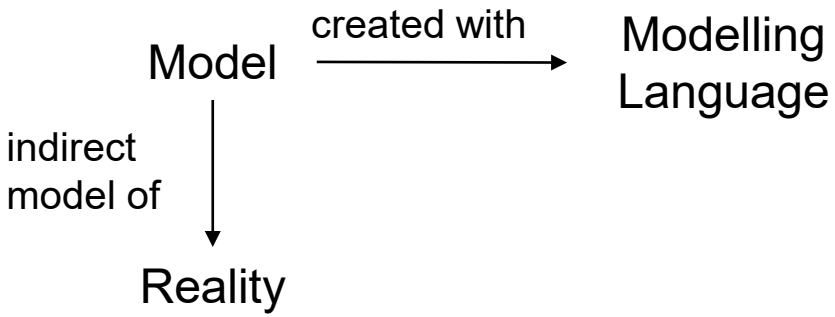
door



window

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

Modelling Language

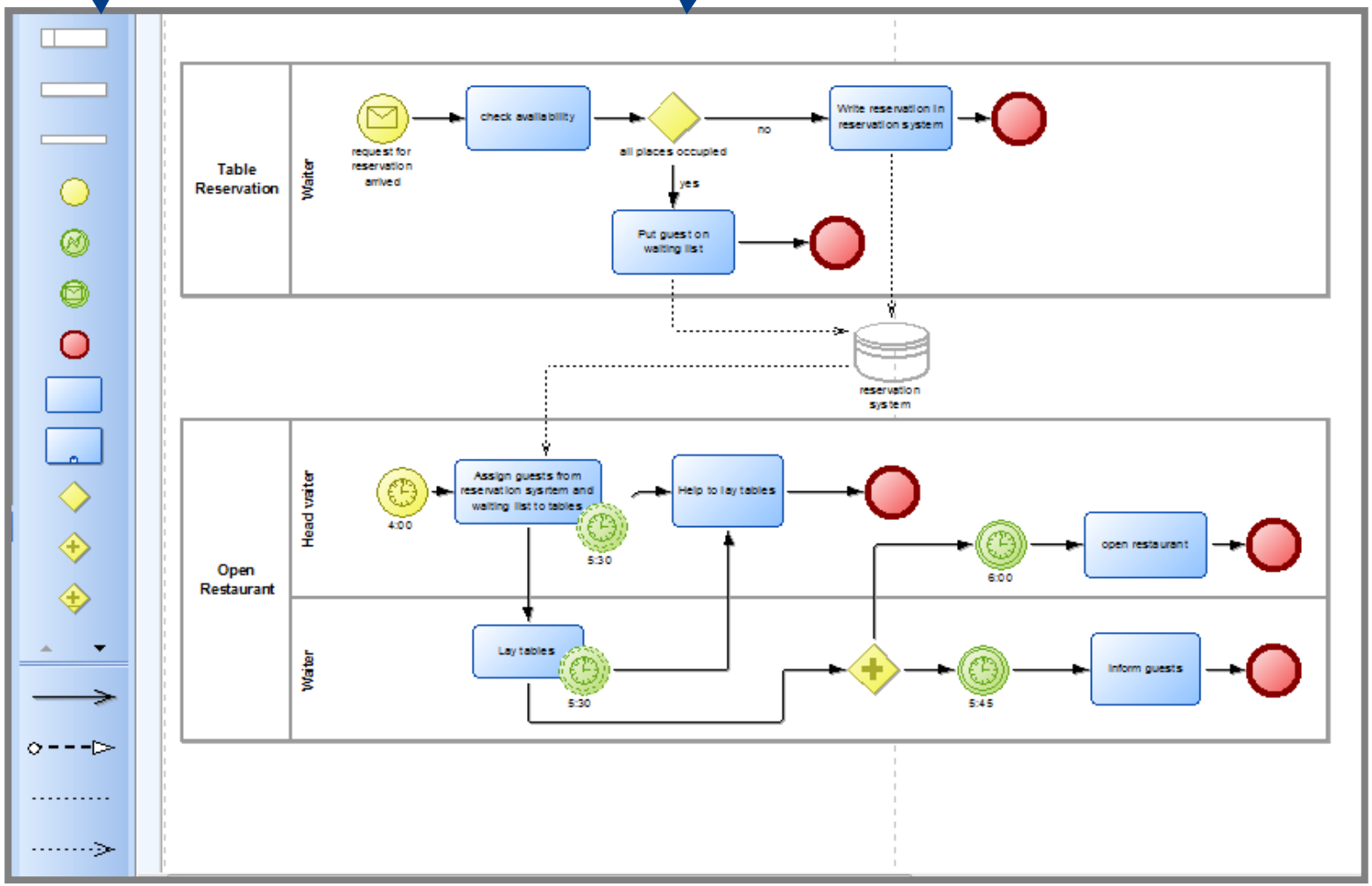


- A modelling "language" specifies the building blocks (elements) from which a model can be made.
- There can be different types of modelling languages, depending on the kind of model
  - graphical model
  - textual description
  - mathematical model
  - conceptual model

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

Modelling Language

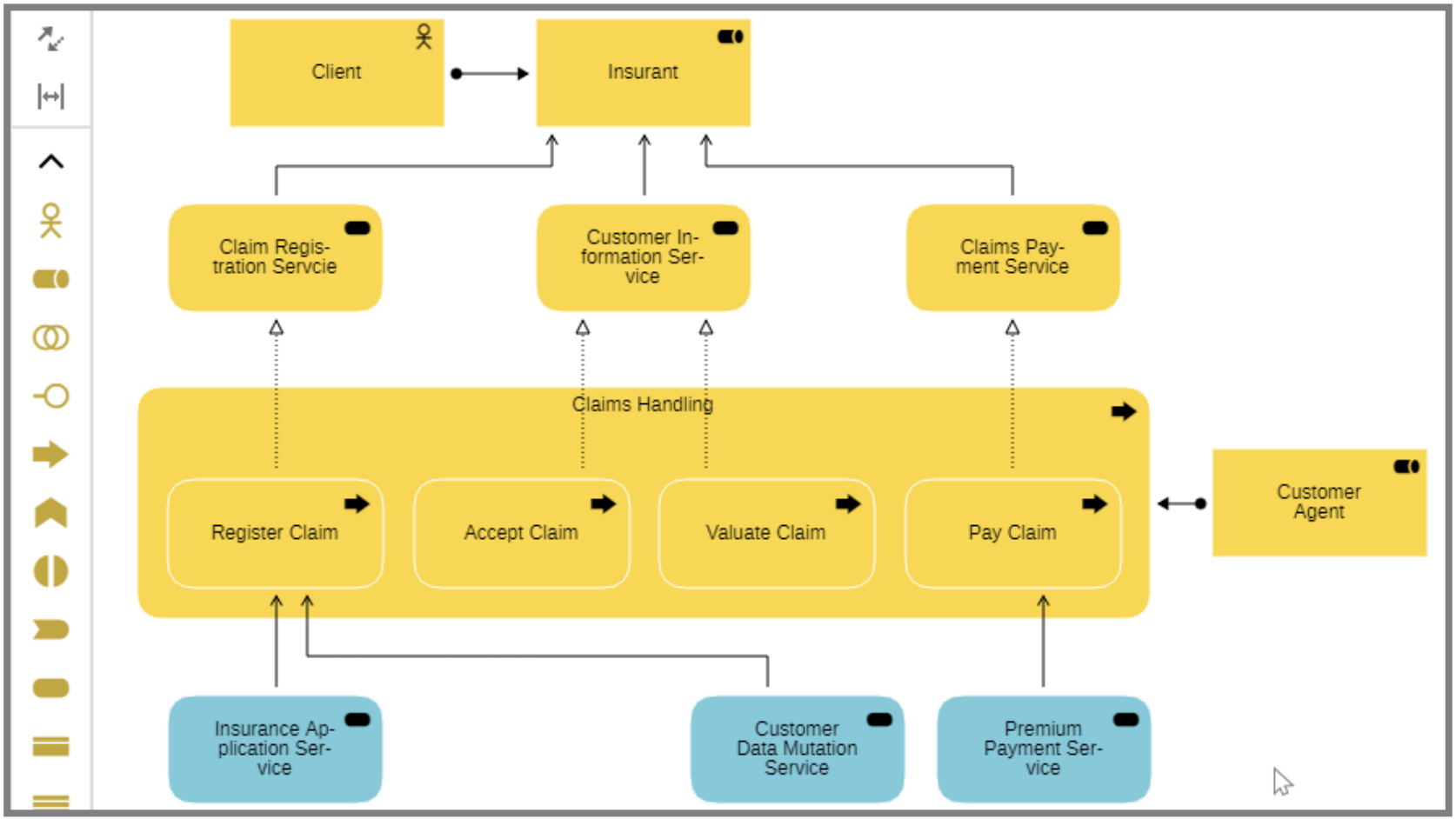
Model



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

Modelling Language

Model



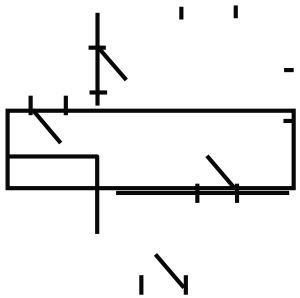
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Model and Meta-Model in Architecture

real object

house

model

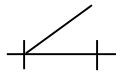


architect's drawing (plan)

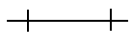
modelling language
(concrete syntax)

object types:

wall



door



window

meta-model
(abstract syntax)

object types:

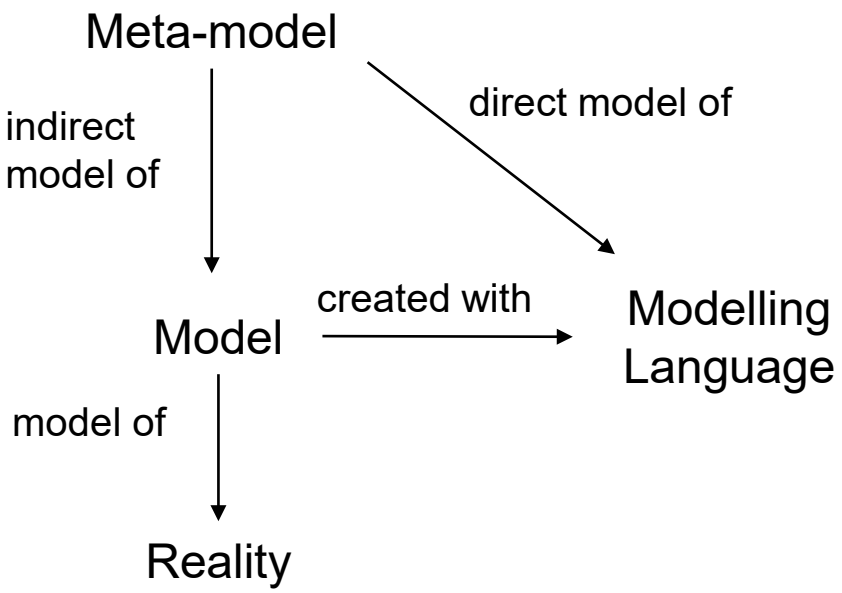
- wall
- door
- window

rules:

- a door is adjacent to a wall on both sides
- Windows are on outer walls.

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

Meta-model



A meta-model defines the semantics of the modelling language, i.e. the building blocks that can be used to make a model. It defines the

- object types that can be used to represent a model
- relations between object types
- attributes of the object types
- rules to combine object types and relations

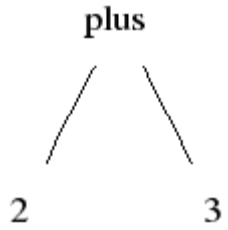
- The meta-model is the abstract syntax, the modelling language is the concrete syntax.

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

Meta Model vs Model Language = Abstract vs. Concrete Syntax

Abstract Syntax

- Deep structure of a language.
- What are the significant parts of the expression?
- Example: a sum expression has two operand expressions as its significant parts



Concrete Syntax

- Surface level of a language.
- What does the expression look like?

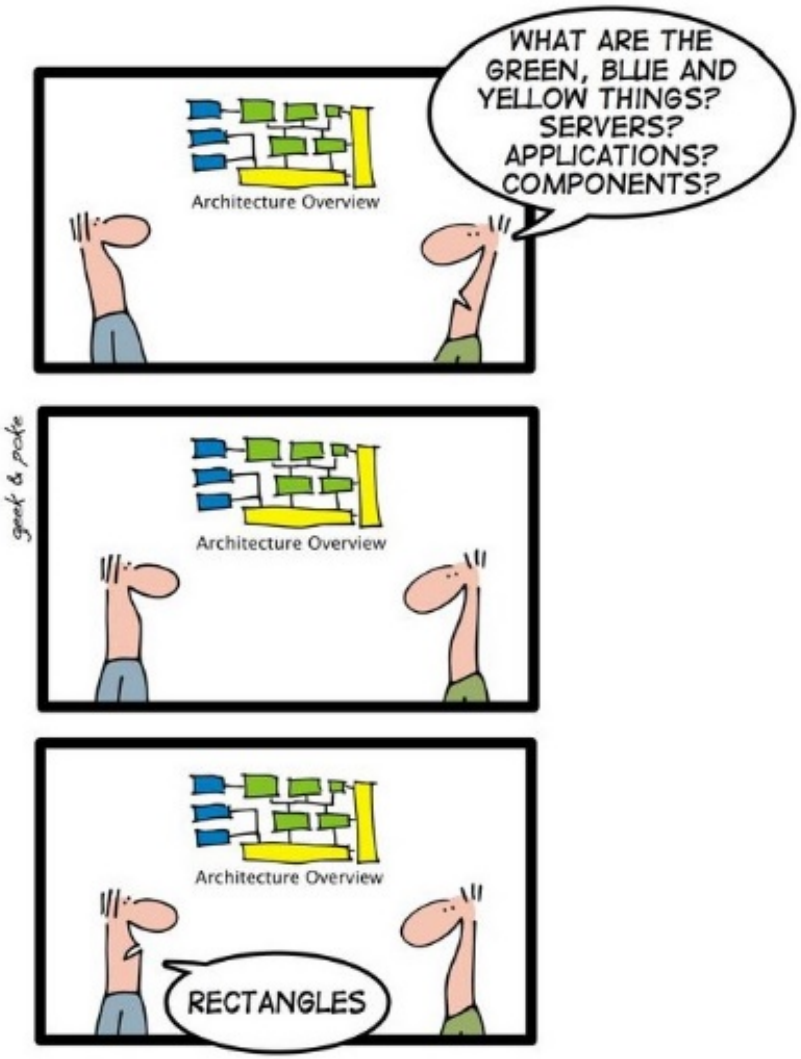
Example: *the same* sum expression can look in different ways:

```
2 + 3                          -- infix

(+ 2 3)                        -- prefix

(2 3 +)                        -- postfix

bipush 2                       -- JVM
bipush 3
iadd

the sum of 2 and 3             -- English
```

http://www.cse.chalmers.se/edu/year/2011/course/TIN321/lectures/proglang-02.html

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

What is the Meaning (Semantics) of a Modelling Language?



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

Metamodel and Modelling Language

Metamodel

- The *metamodel* defines the modelling elements (concepts, relations) and their semantics (= meaning)
  - ♦ WHAT can be modeled
- The *metamodel* corresponds to the *abstract syntax*

Modelling language

- The *modelling language* defines the notation/appearance of the modelling elements
  - ♦ HOW can it be modeled
- The *modelling language* corresponds to the *concrete syntax*

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

Illustration: Meta-model and Model for Processes

Metamodel:

Abstract syntax:
Concepts and relations which can be used to create models.

Example: A class and object diagram consists of concepts for

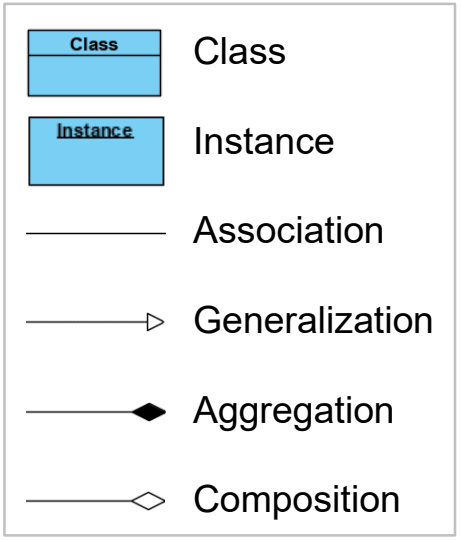
- «classes», «instances»,

and relations for

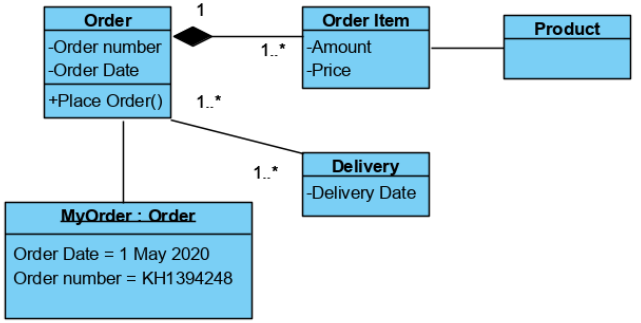
- «association», «generalization», «aggregation» and «composition»

Modelling Language:

Concrete Syntax (notation, appearance) of meta-model elements



Model:



A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

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

Illustration: Meta-model and Model for Processes

Metamodel:

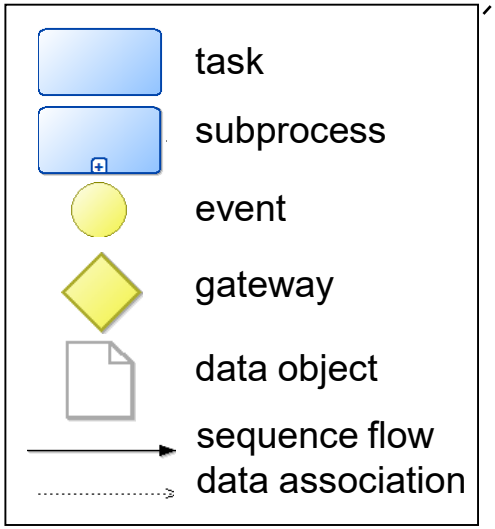
Abstract syntax:
Concepts and relations which can be used to create models.

Example: A process model consists of concepts for
- «task», «subprocess», «event», «gateway», «data object»

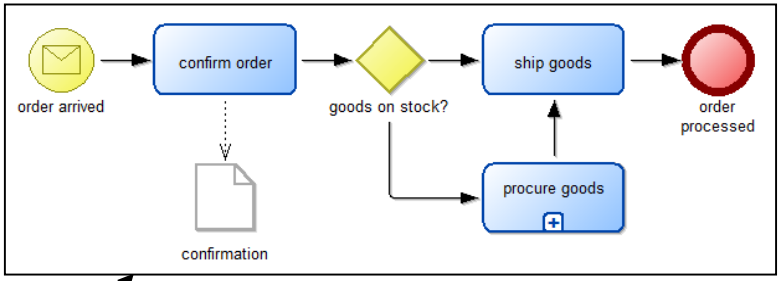
and relations for
- «sequence flow», «data association».

Modelling Language:

Concrete syntax:
Notation/appearance of meta-model elements



Model:

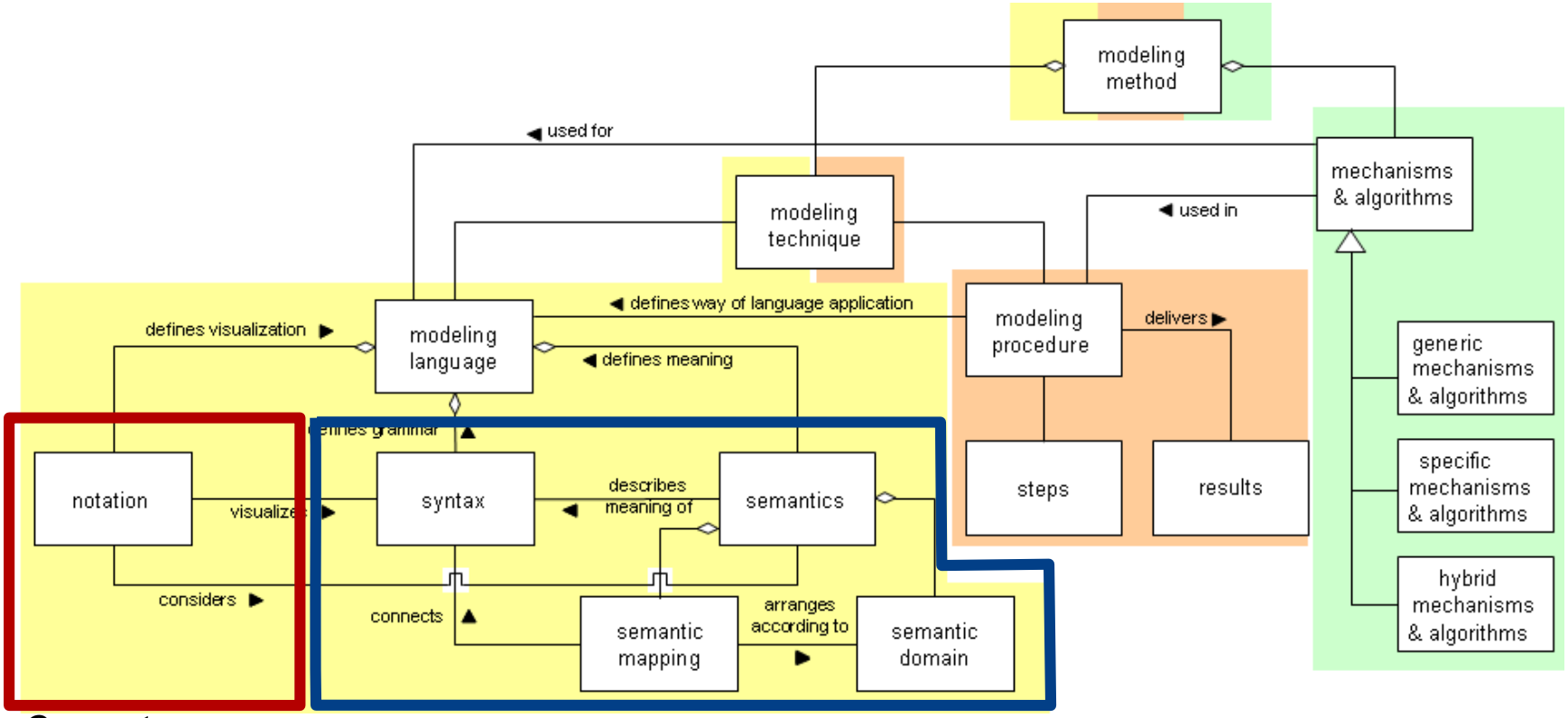


A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

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

Components of Modelling Methods

A Modelling Language is Part of a Modelling Modelling
A Modelling Language consists of the Metamodel (Abstract Syntax and Semantics) and the Notation

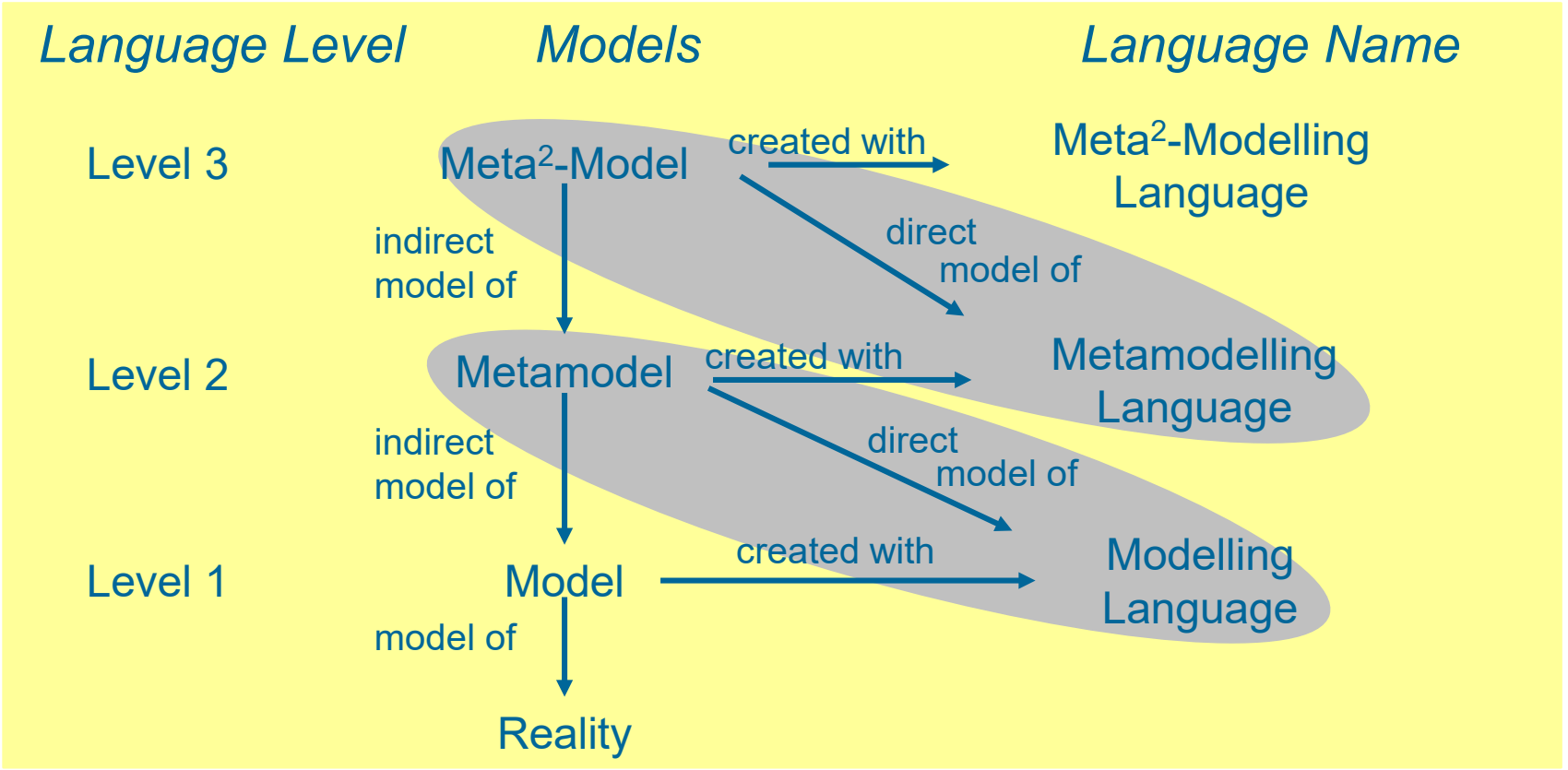


(Karagiannis & Kühn 2002)

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

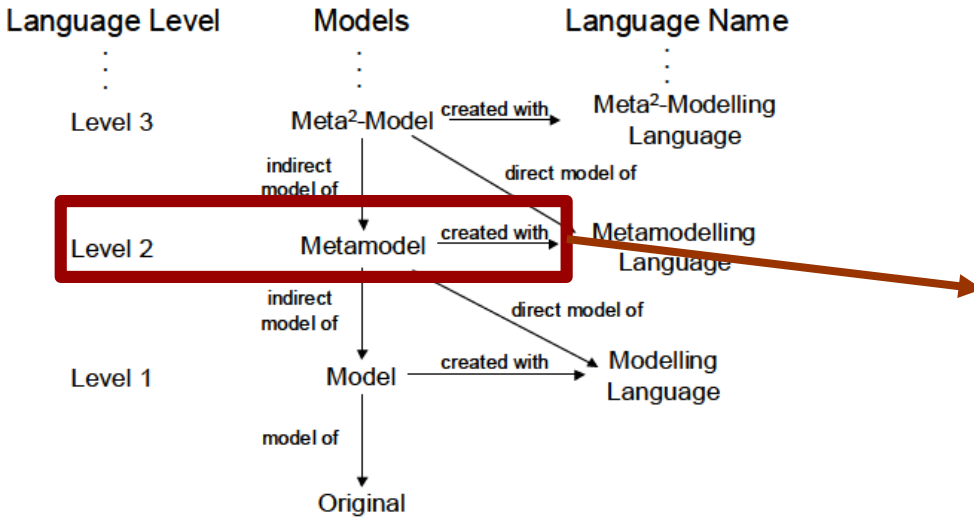
Meta Model Hierarchy

The meta-model must again be described in some language, which has to be specified in a meta-meta-model

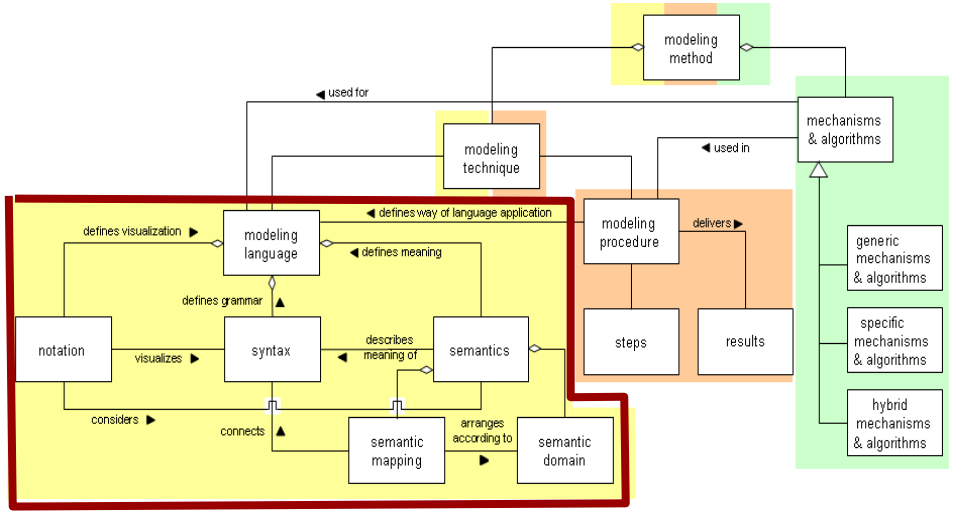


Karagiannis, D. & Kühn, H., 2002. Metamodelling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer, eds. *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.

Metamodelling
Modeling Language Definition

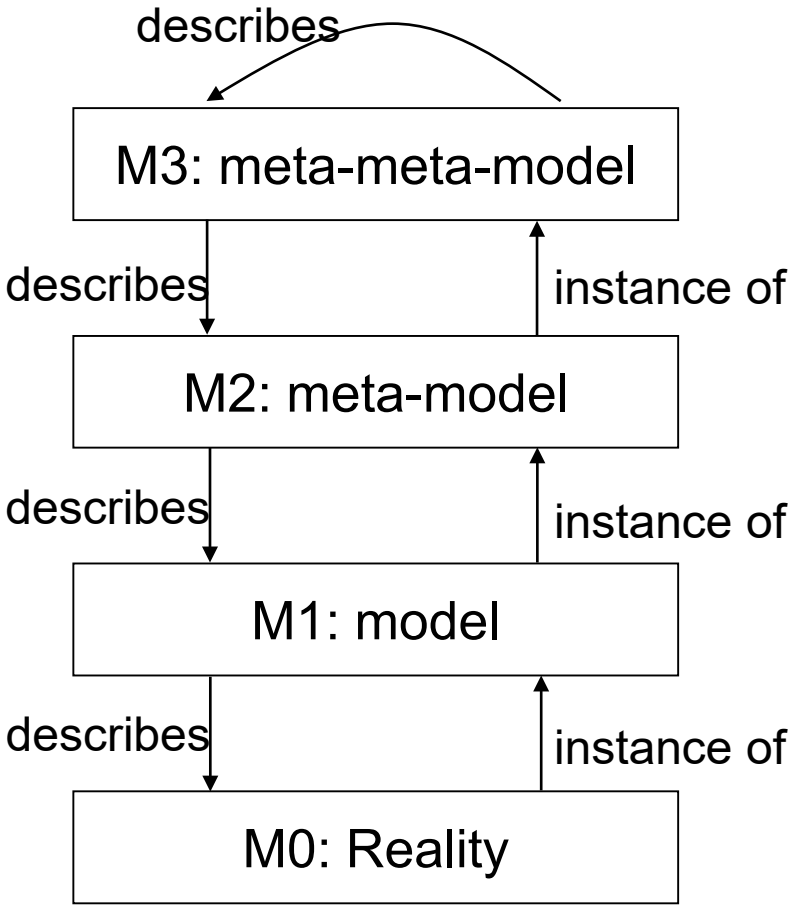


(Strahringer, 1996)



(Karagiannis & Kühn, 2002)

The Model Stack

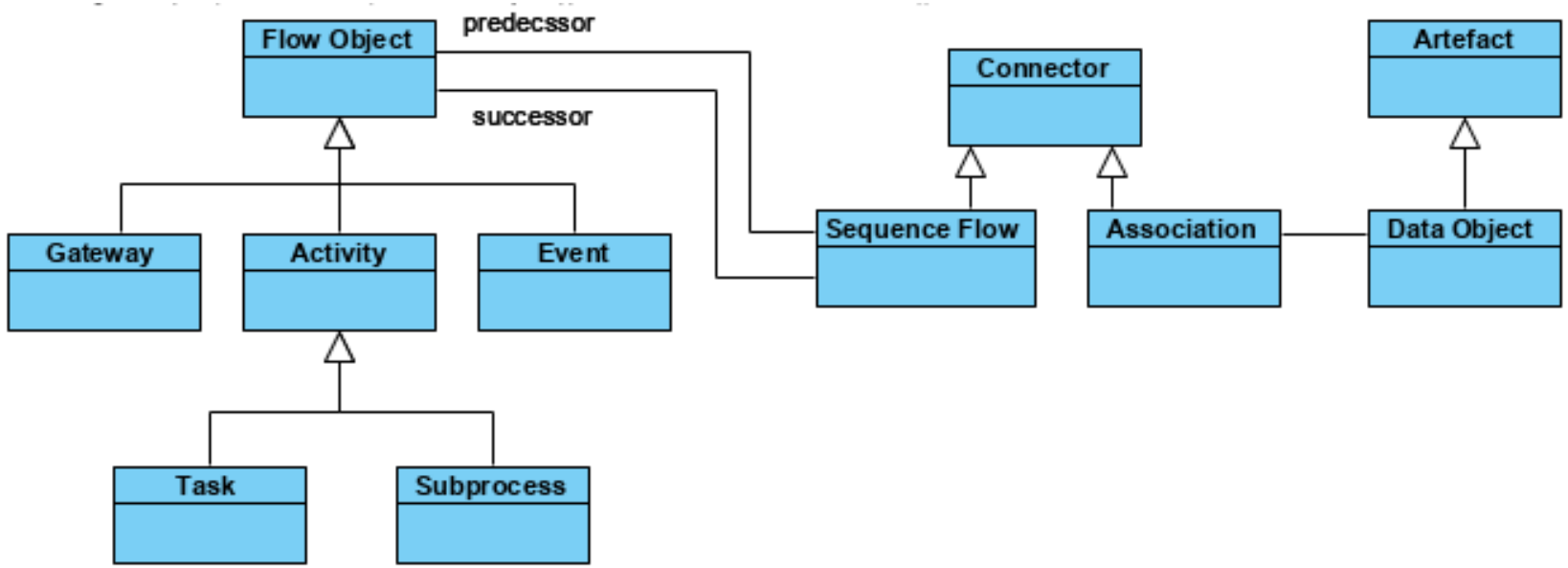


- A model is a ***simplified representation of a reality***
- A meta-model defines a **modelling language** in which a model can be expressed.
- A meta-meta model defines the **language in which a meta-model** can be expressed.

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

Metamodels can be defined as Class Diagrams

Metamodel correspond to a knowledge base
Metamodels can be represented graphically as (a subset of) UML class diagrams



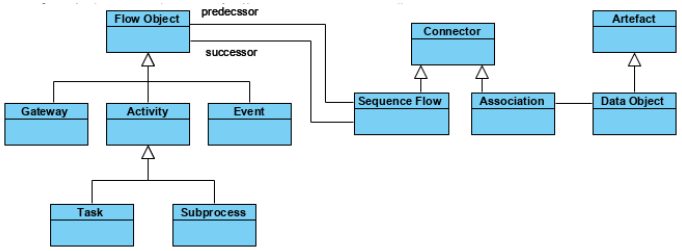
(UML Class diagrams where originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

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

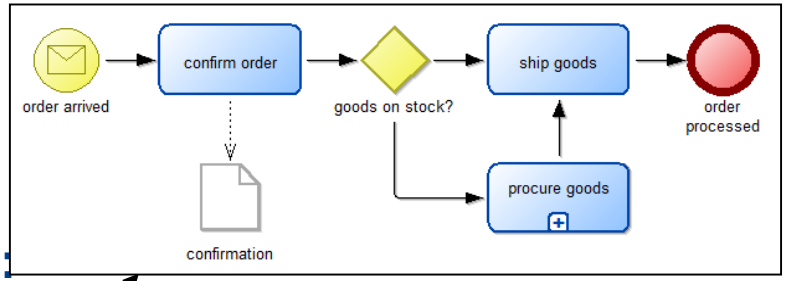
A Metamodel for Processes

Meta-model:

- Classes and relations that can be used for modelling

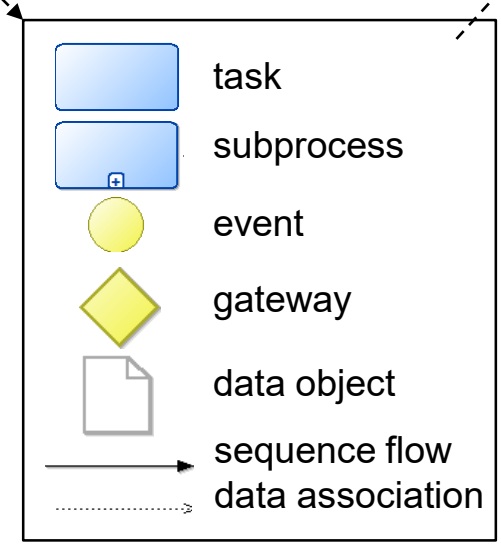


Model:



Modelling Language:

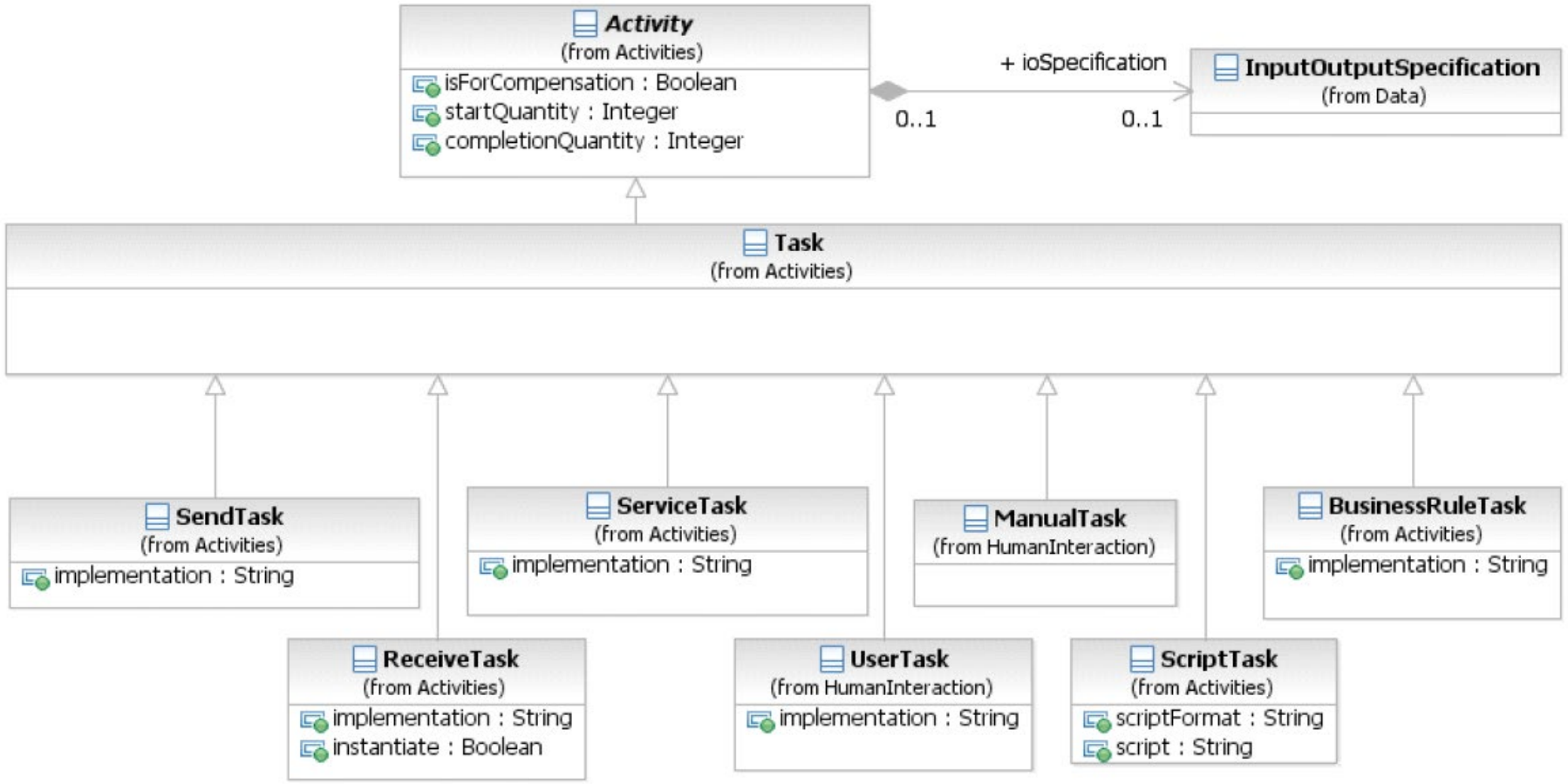
Concrete Syntax (notation, appearance) of meta-model elements



A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

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

Subset of the BPMN Metamodel in UML

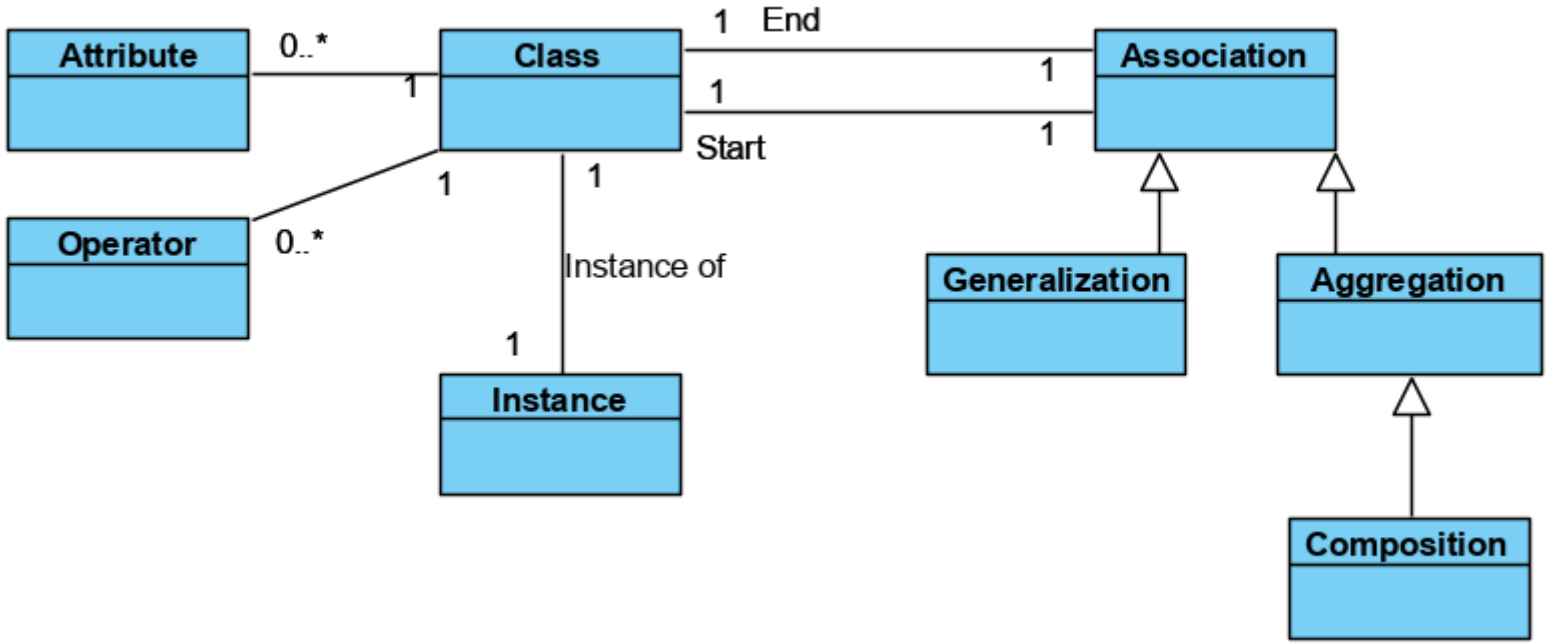


Source: BPMN 2.0 specification

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

A Metamodel for UML Class Diagrams

UML Class Diagrams can be used to model the metamodel for UML class diagrams themselves

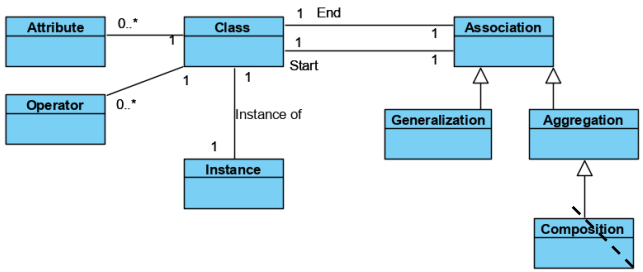


(UML Class diagrams where originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

A Metamodel for Class Diagrams

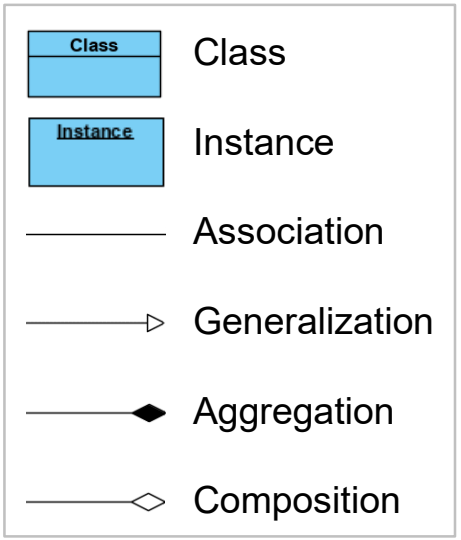
Meta-model:

- Classes and relations that can be used for modelling

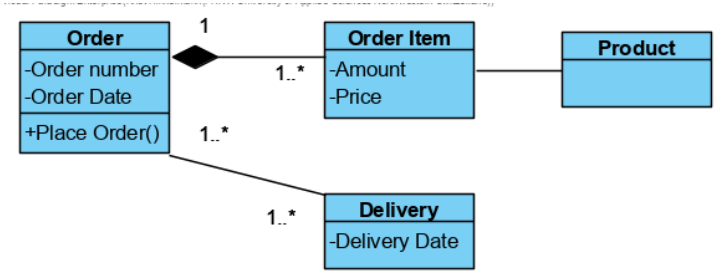


Modelling Language:

Concrete Syntax (notation, appearance) of meta-model elements



Model:



A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

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

Domain-specific vs. General-purpose Modelling Languages

- General-purpose modelling languages can be used to represent any kind of knowledge
- Domain-specific languages are notations which are defined to model knowledge about a specific domain

General-purpose Modelling Languages

- General-purpose modelling languages can be used to represent any kind of knowledge
- They can be used, if no domain-specific modelling language is available (for a view)
- There are a wide range of generalo-purpose modelling languages
  - Natural language allows to express any knowledge
  - Formal languages: Typically a subset of Logic
  - Graphical Diagrams
- General-purpose graphical modelling languages have been developed in a many difference fields:
  - Artificial Intelligence: Semantic networks, Ontologies
  - Data Modelling: Entity Relationship Diagrams
  - Object-Oriented Programming: UML Class Diagrams

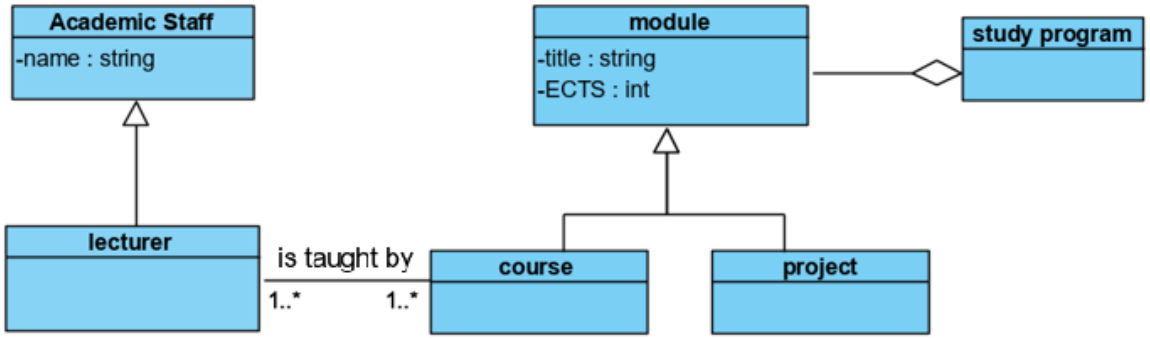
The Metamodel for a General-purpose Modelling Language

- The metamodel for a general-purpose modelling language has only few modelling elements
  - Class
  - Attribute
  - Association
  - Instance
- This can be modelled with Class Diagrams, e.g.
  - (a subset of) UML Class Diagrams
  - Ontology Languages
- Modelling means to
  - define classes
  - create instances of these classes

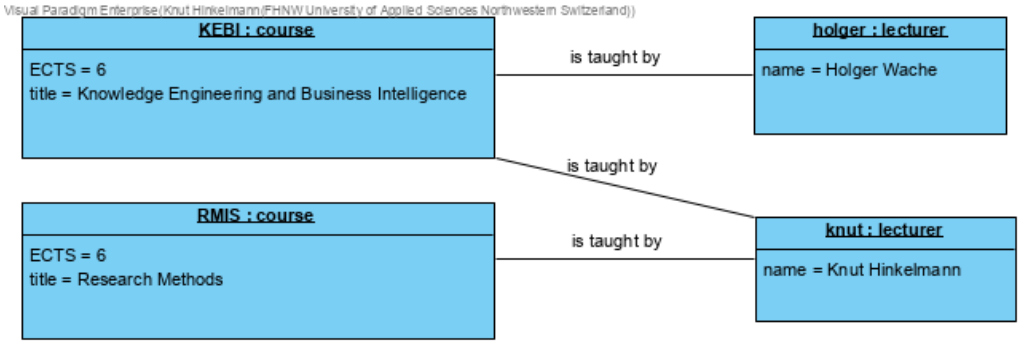
Modelling with a General-purpose Modelling Language

- Class Diagrams are general-purpose modelling languages; one can define classes and relations for any domain
- A model consists of objects which are instances of these classes

Classes
(=metamodel)



Instances
(= model)



Strengths and Weaknesses of General-Purpose Modelling Languages

- Strengths
  - Applicability
    - Can be used to represent everything
    - Every model in the same language
    - Low learning curve for the language
- Weakness
  - No guidance: Users have to …
    - determine how to structure a domain
    - to identify relevant concepts
  - Restrictred reusability
    - Different applications use different concepts

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

Domain-specific Modelling Languages

- Modelling languages have modelling elements for typical concepts and relations of a domain of discourse
  - Predefined classes, relations and constraints
  - Specific shapes for modelling elements and relations
- Modelling means to create instances of theses classes and relations
- Examples of domain-specific modelling languages:
  - **BPMN** is a domain-specific language for business processes
    - Concepts: task, event, gateway, ….
    - Relations: sequence flow, message flow, data association, …
  - **ArchiMate** is a domain-specific language for enterprise architectures
    - Concepts: process, actor, role, business object, …
    - Relations: uses, realizes, …

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

Strengths and Weaknesses of Domain-specfic Modelling Languages

- Strengths
  - Comprehensiblity of models
    - concepts and relations are adequate for stakeholders
    - domain-specific shapes
  - 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

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

- If there is no domain-specific modelling language for a domain of interest, we can
  1. Use a general-purpose modelling language
  2. Define a new domain-specific modelling language
     - From scratch
     - By adapting an existing one

→ meta modelling

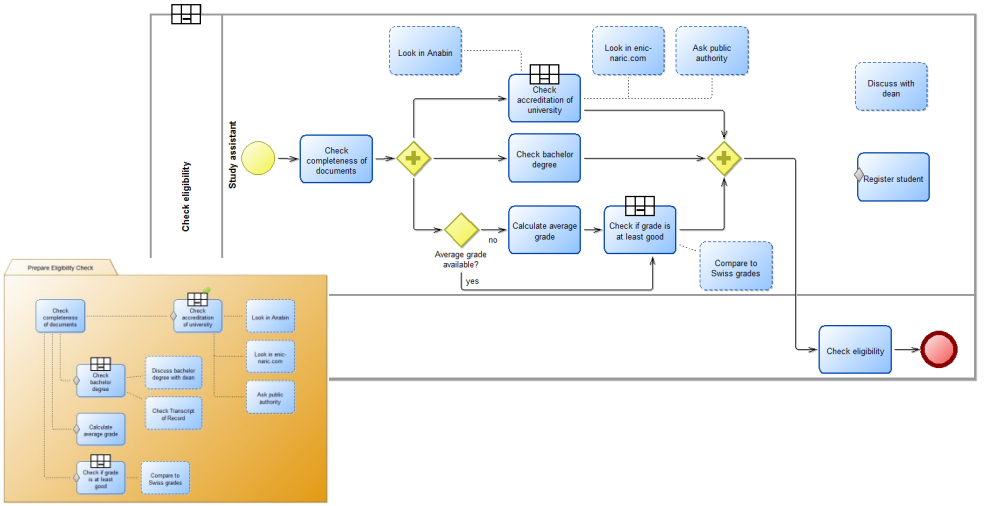
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

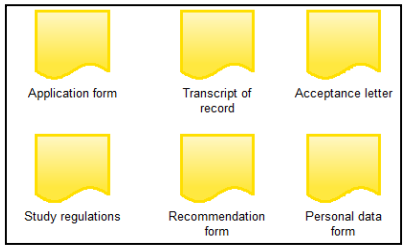
Knowledge Work Designer

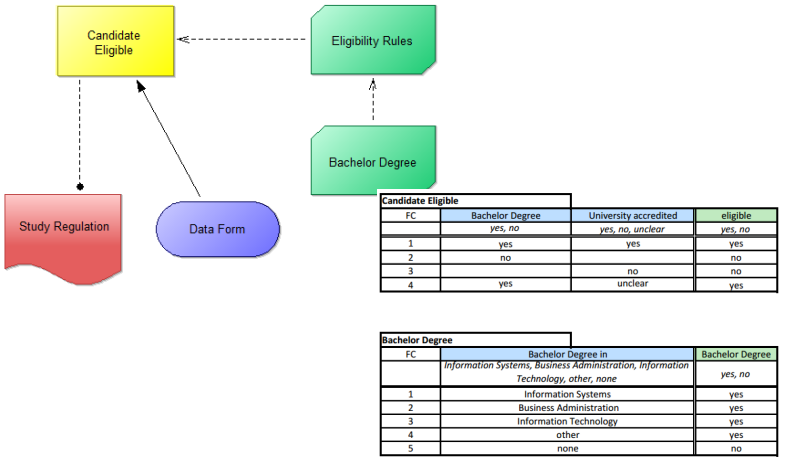
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Modeling of Knowledge Work

- Process Logic
  - Structured Processes (BPM)
  - Case Models (CMMN)
  - Combination (BPCMN)
- Decision Logic
  - Decision Models (DMN)
  - Document Model







Candidate Eligible

FC	Bachelor Degree	University accredited	eligible
	yes, no	*yes, no, unclear*	*yes, no*
1	yes	yes	yes
2	no		no
3		no	no
4	yes	unclear	yes

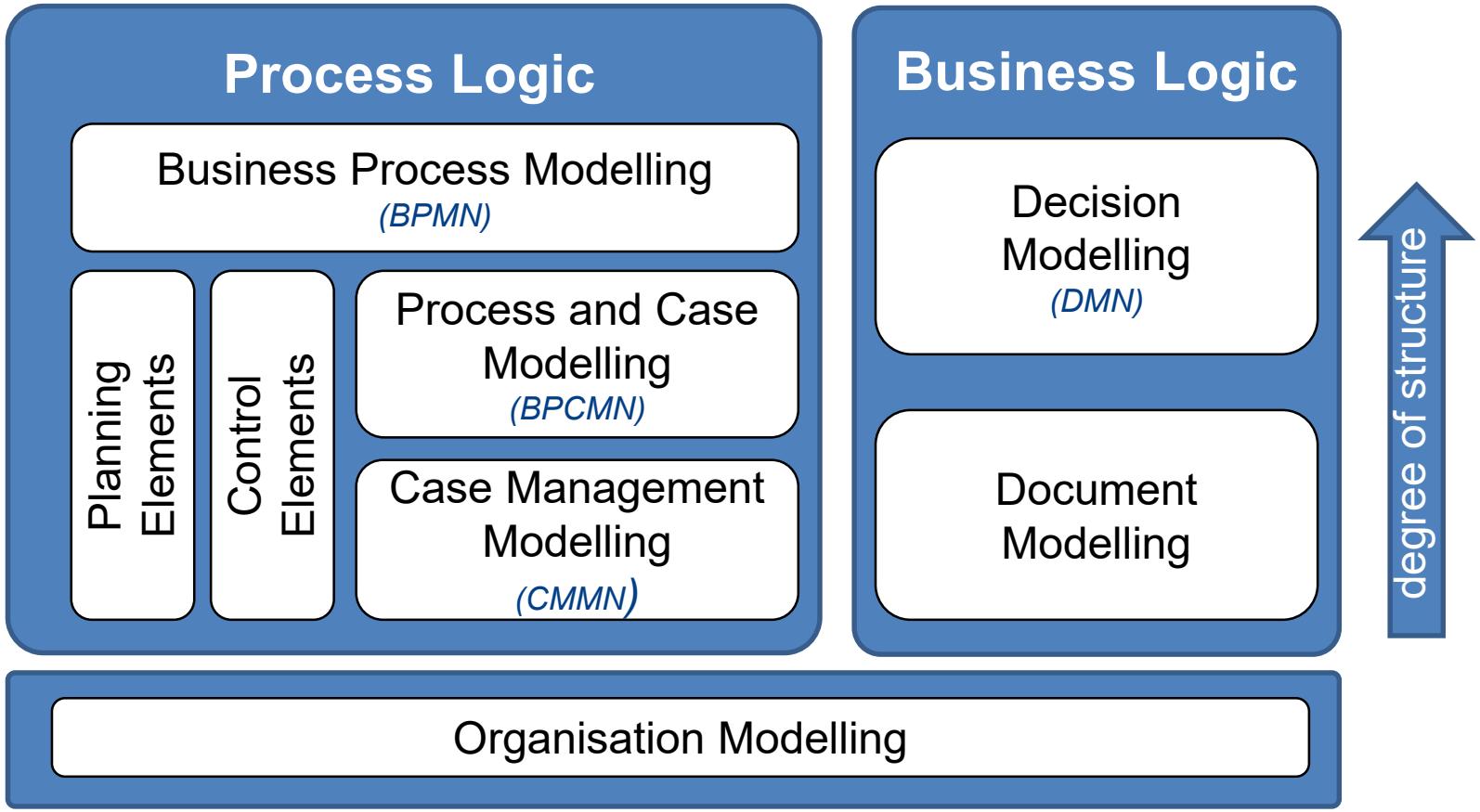
Bachelor Degree

FC	Bachelor Degree in *Information Systems, Business Administration, Information Technology, other, none*	Bachelor Degree *yes, no*
1	Information Systems	yes
2	Business Administration	yes
3	Information Technology	yes
4	other	yes
5	none	no

Details and Download: https://austria.omilab.org/psm/content/kwd/

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

Model types of the Knowledge Work Designer

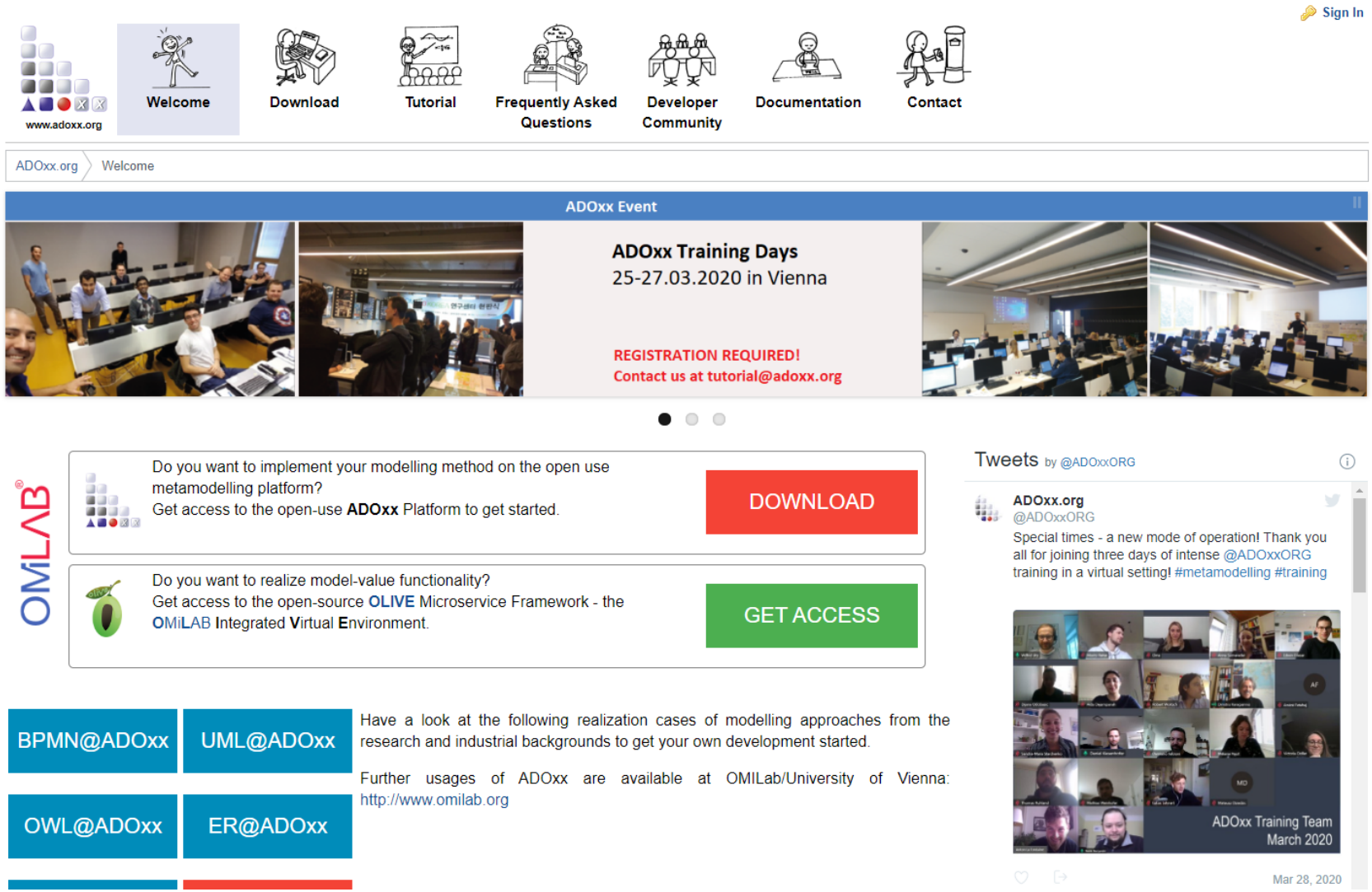


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Metamodelling with ADOxx

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

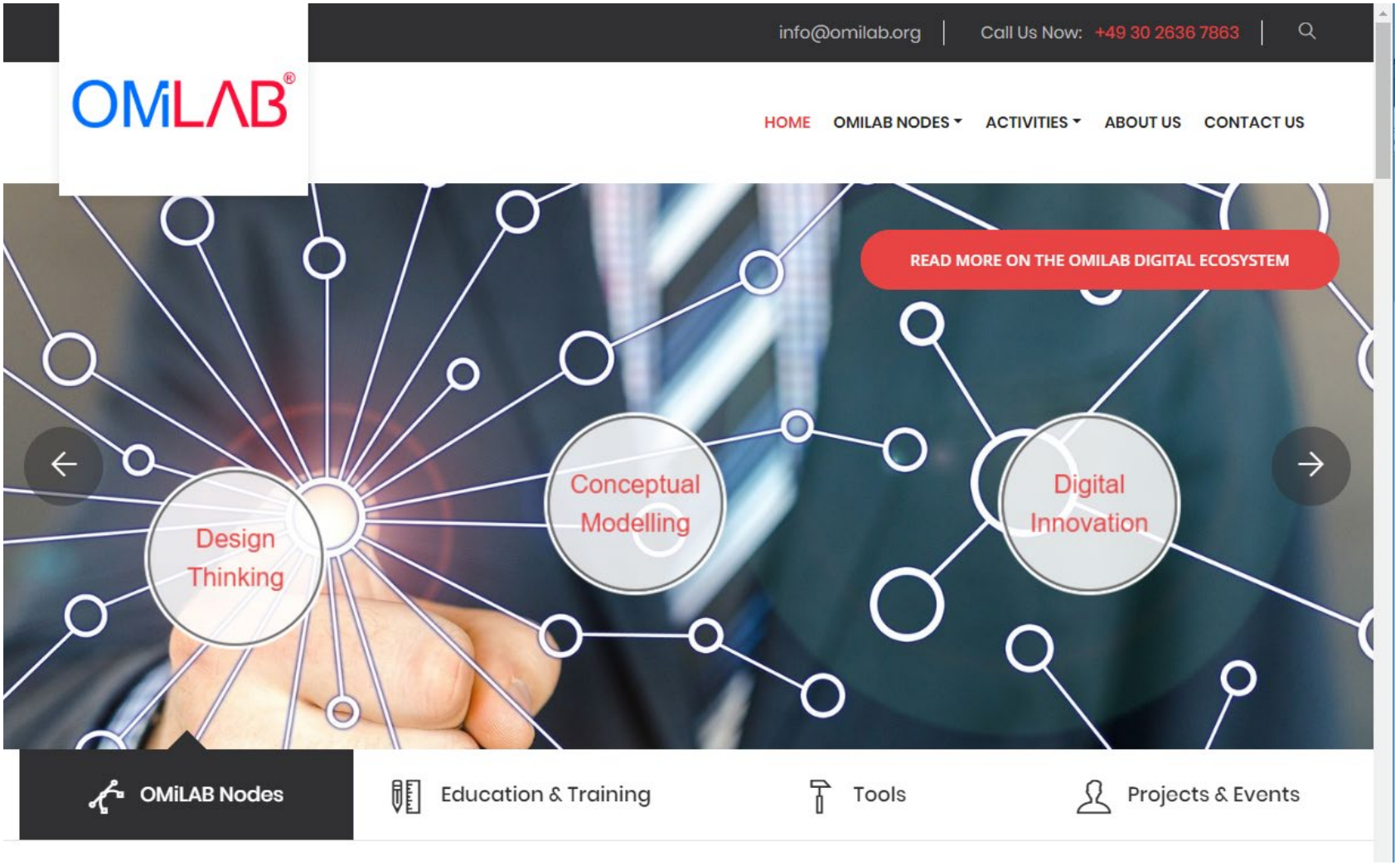
adoxx.org – Download, Tutorials, Community



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

OMiLAB – A Conceptual Modelling Commnity

ADOxx is the basis for OMiLAB

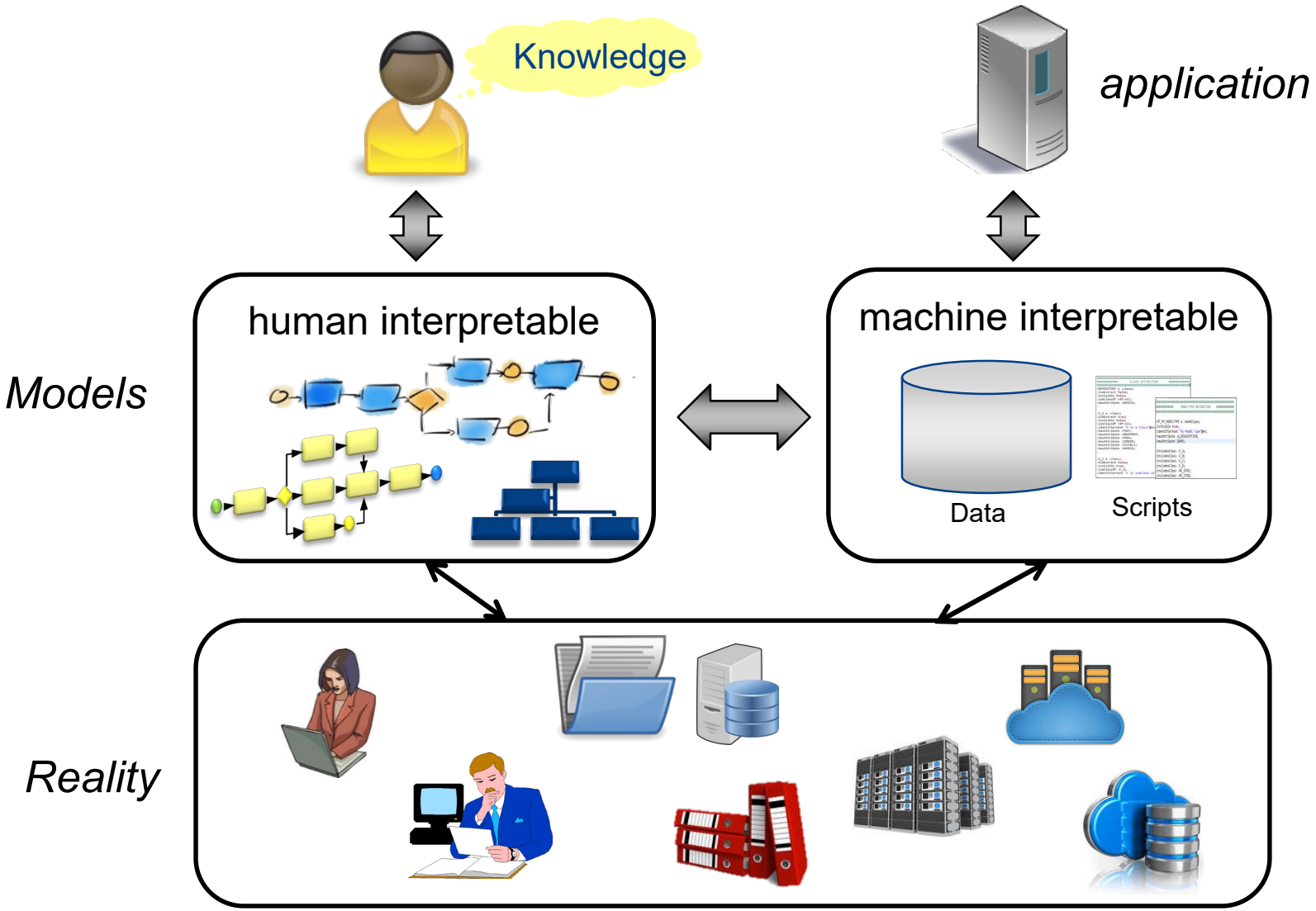


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

The ADOxx Environment

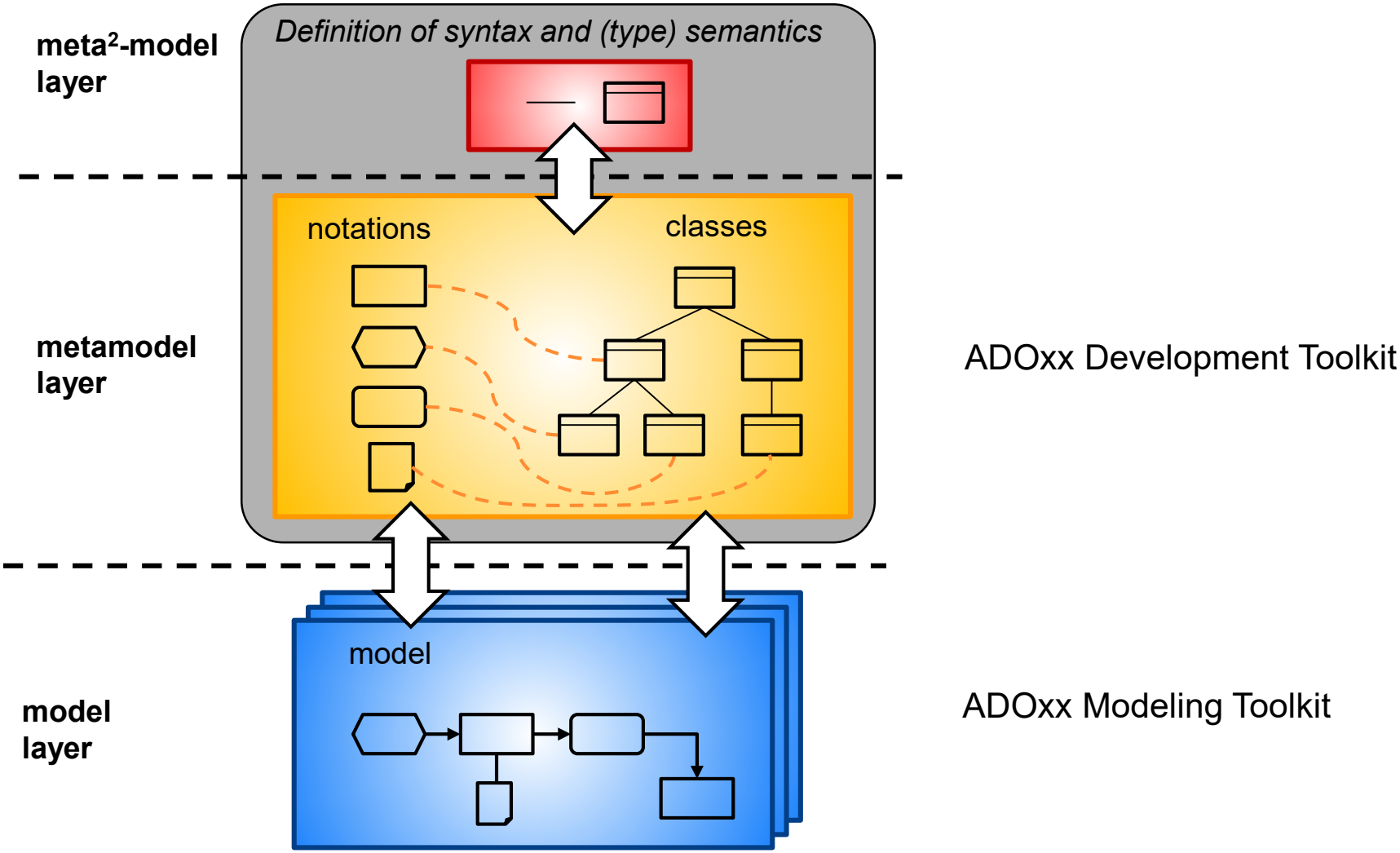
- ADOxx consists of …
  - ADOxx Development Toolkit
    - Defining Modelling languages – Library Management
    - Administration of users, models, components
  - ADOxx Modelling Toolkit
    - Creating models

Graphical Models are Represented in a Database



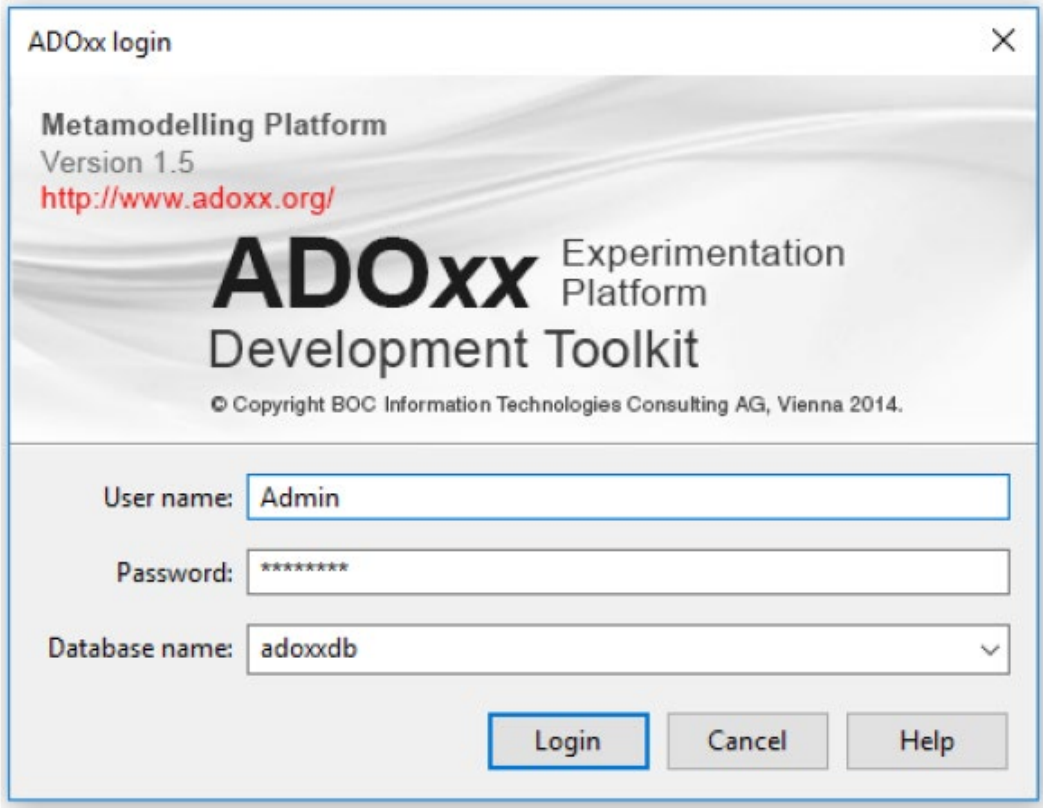
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Modeling Environment



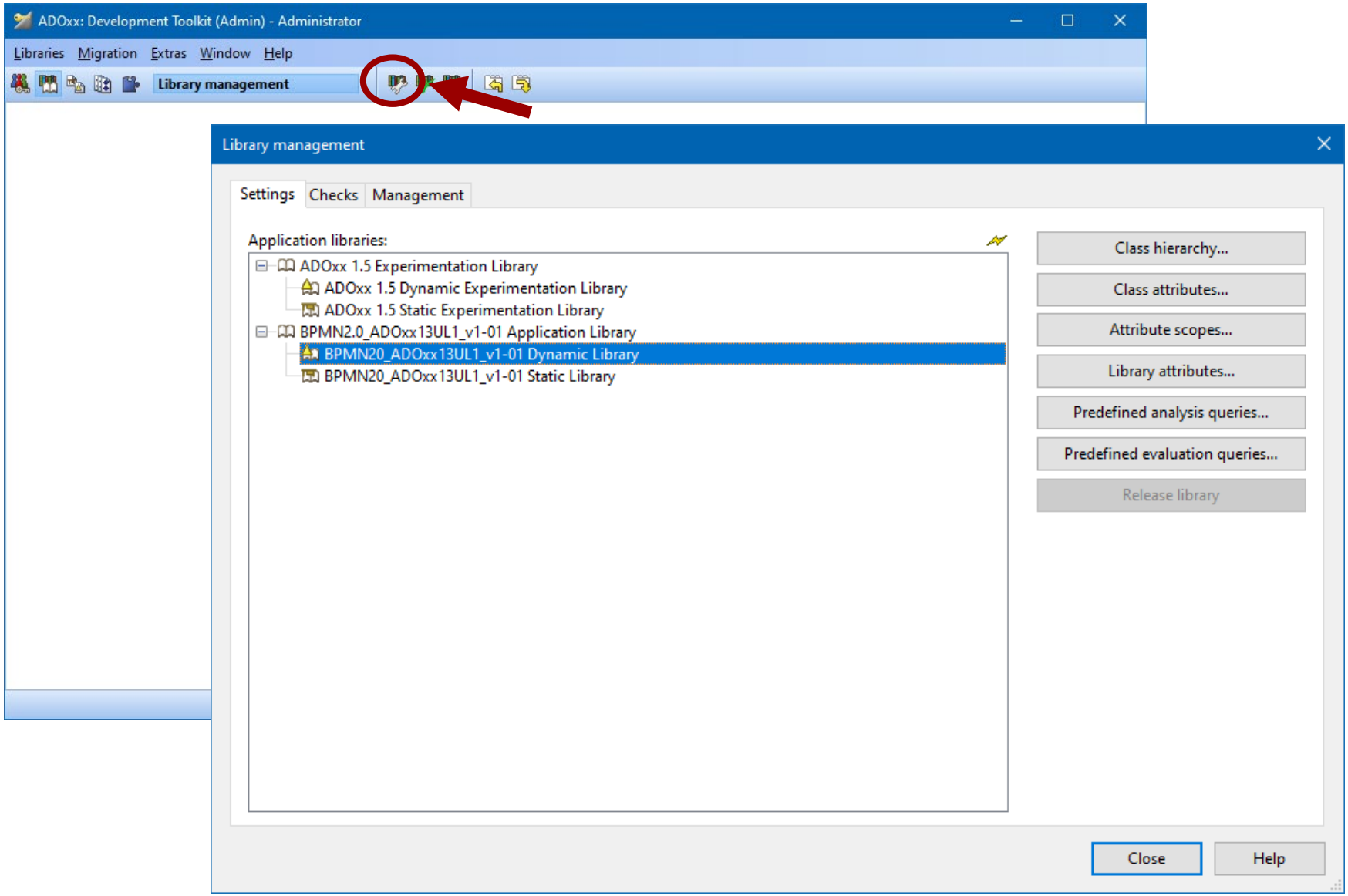
Development Toolkit

- 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

Metamodelling with ADOxx

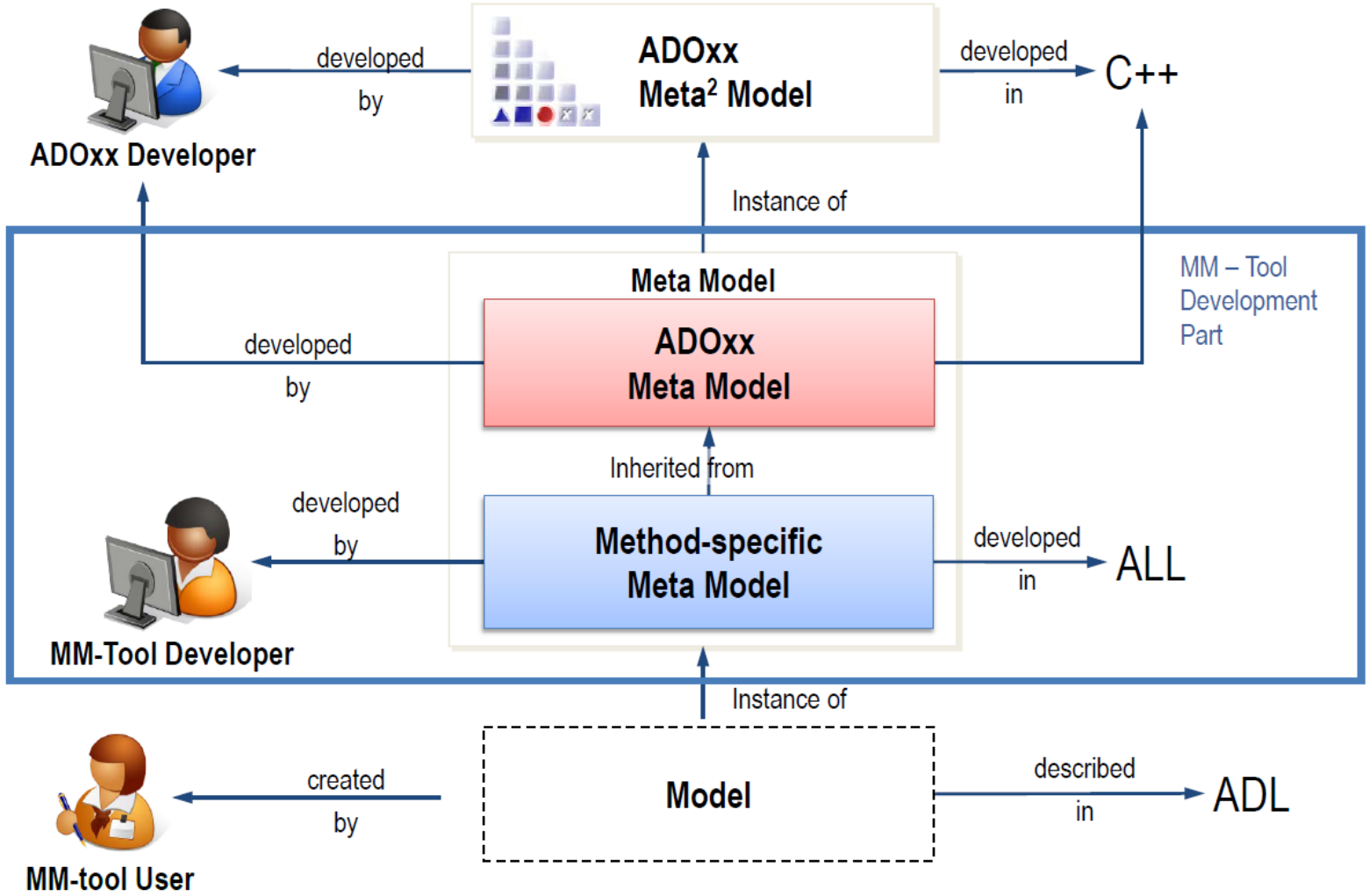


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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	

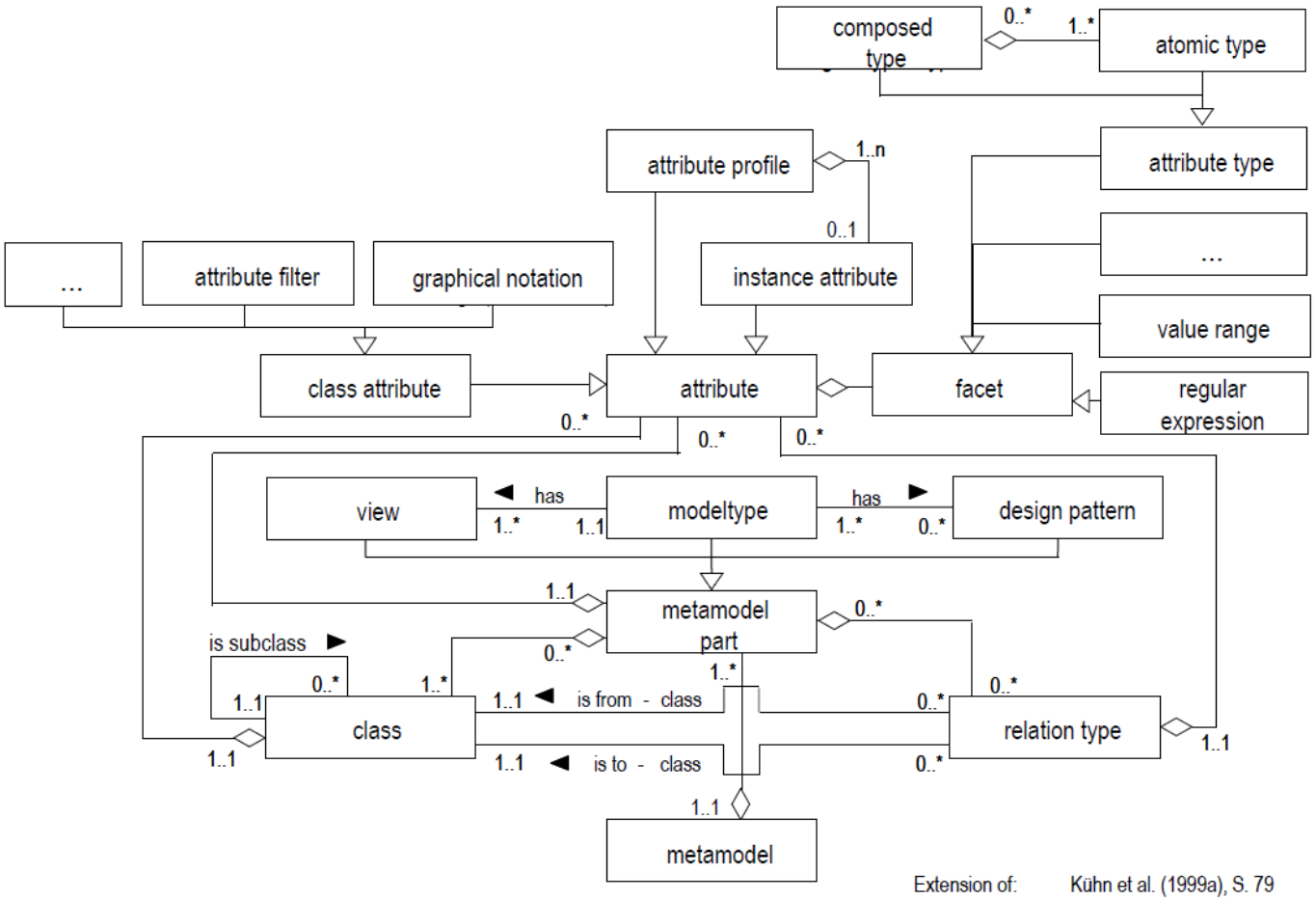
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Meta Modelling Platforms Hierarchyin ADOxx

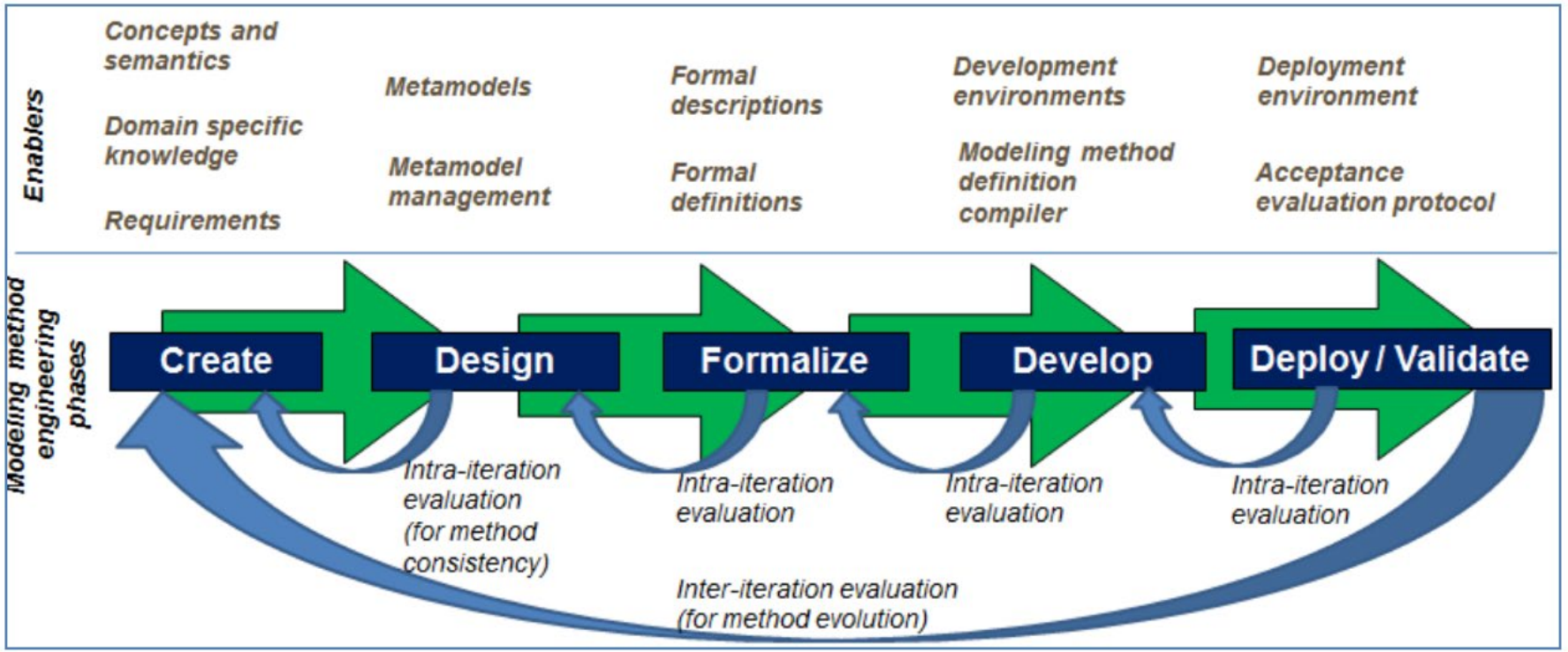


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Meta² Model: Meta Model of Meta Modelling Language

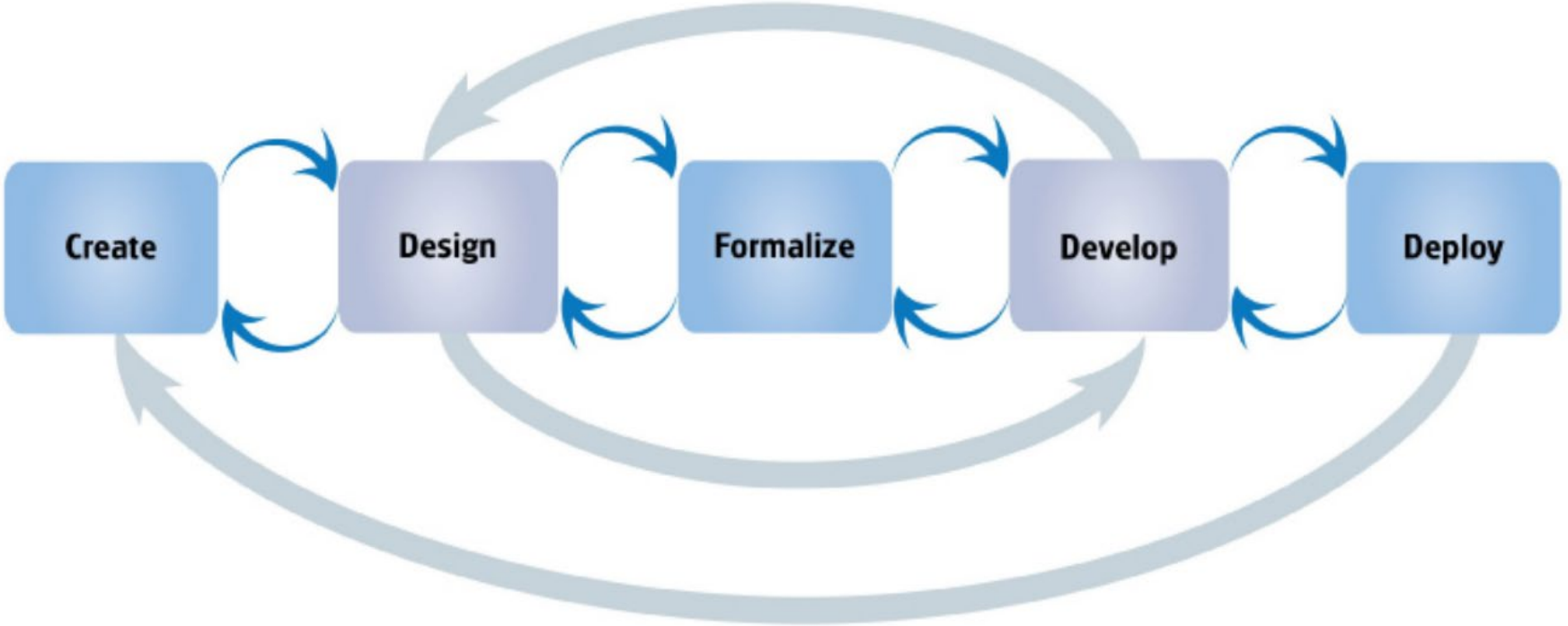


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch



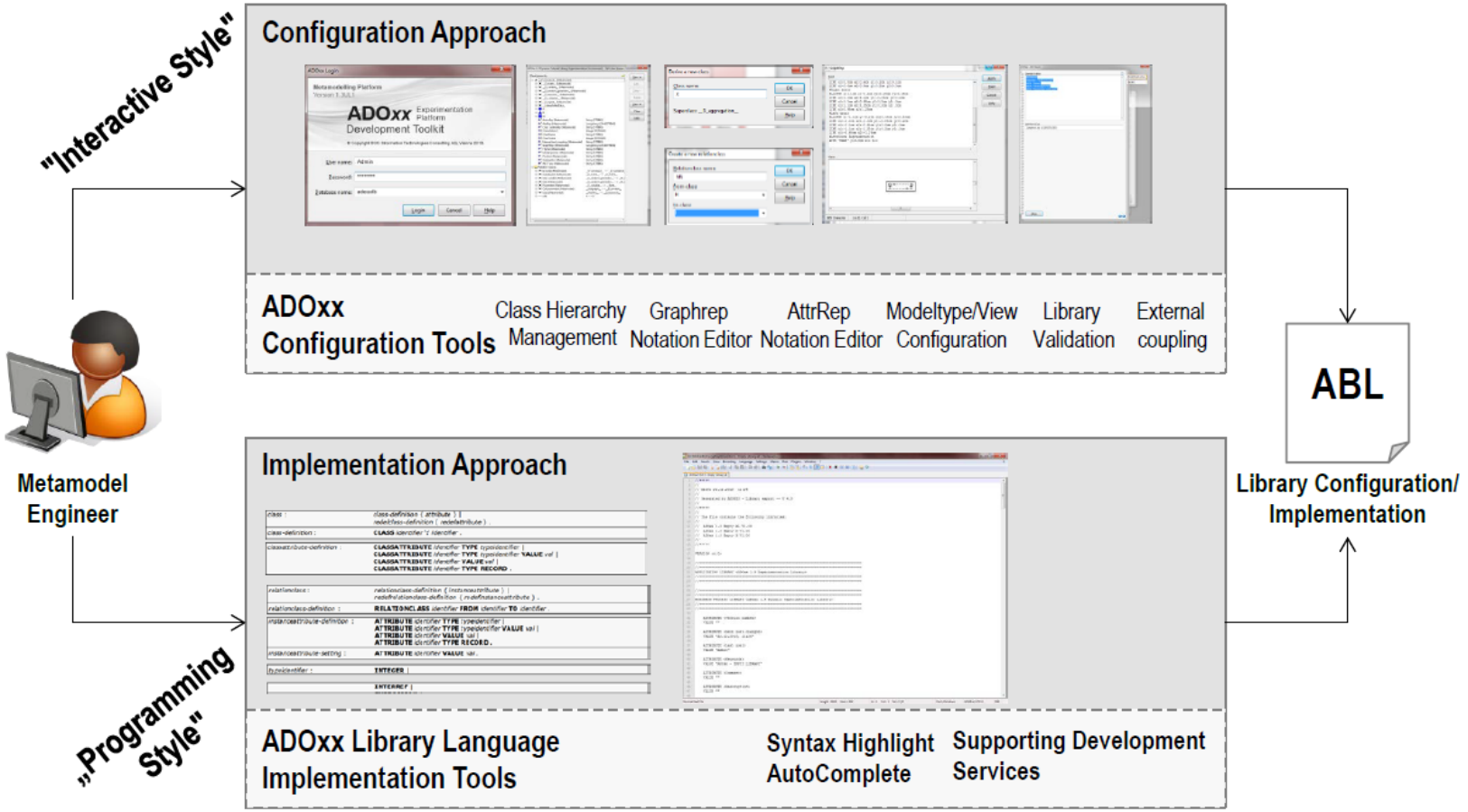
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

The AMME LifeCycle
Agile Meta Model Engineering



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

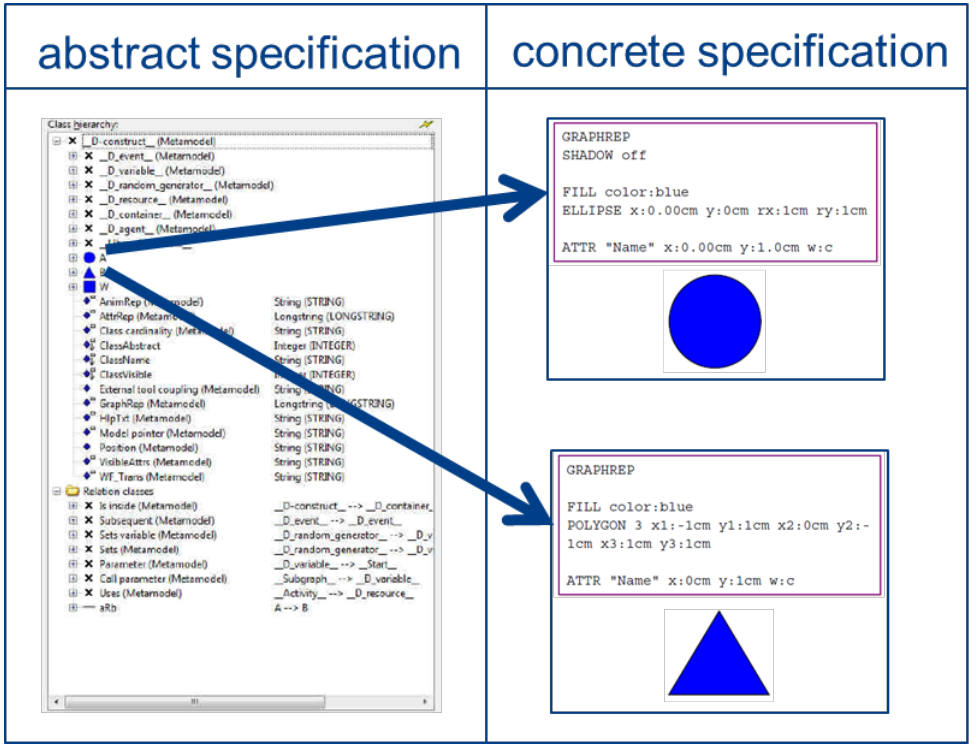
Development Appraoches in ADOxx – Configuration and Implementation



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

Abstract and Concrete Specification

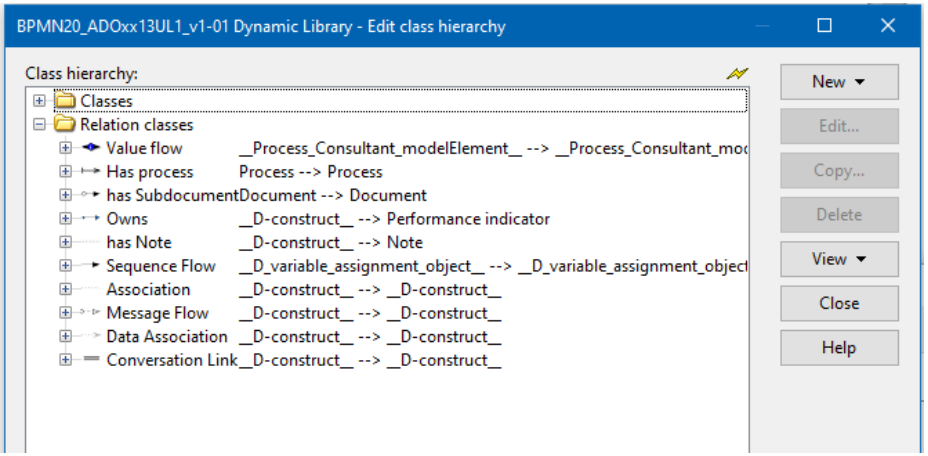
- The Semantics of a model language is defined by
  - Classes of elements and relations
  - Class hierarchy
  - Attributes of the elements
- The Syntax is defined by
  - special attribute GraphRep

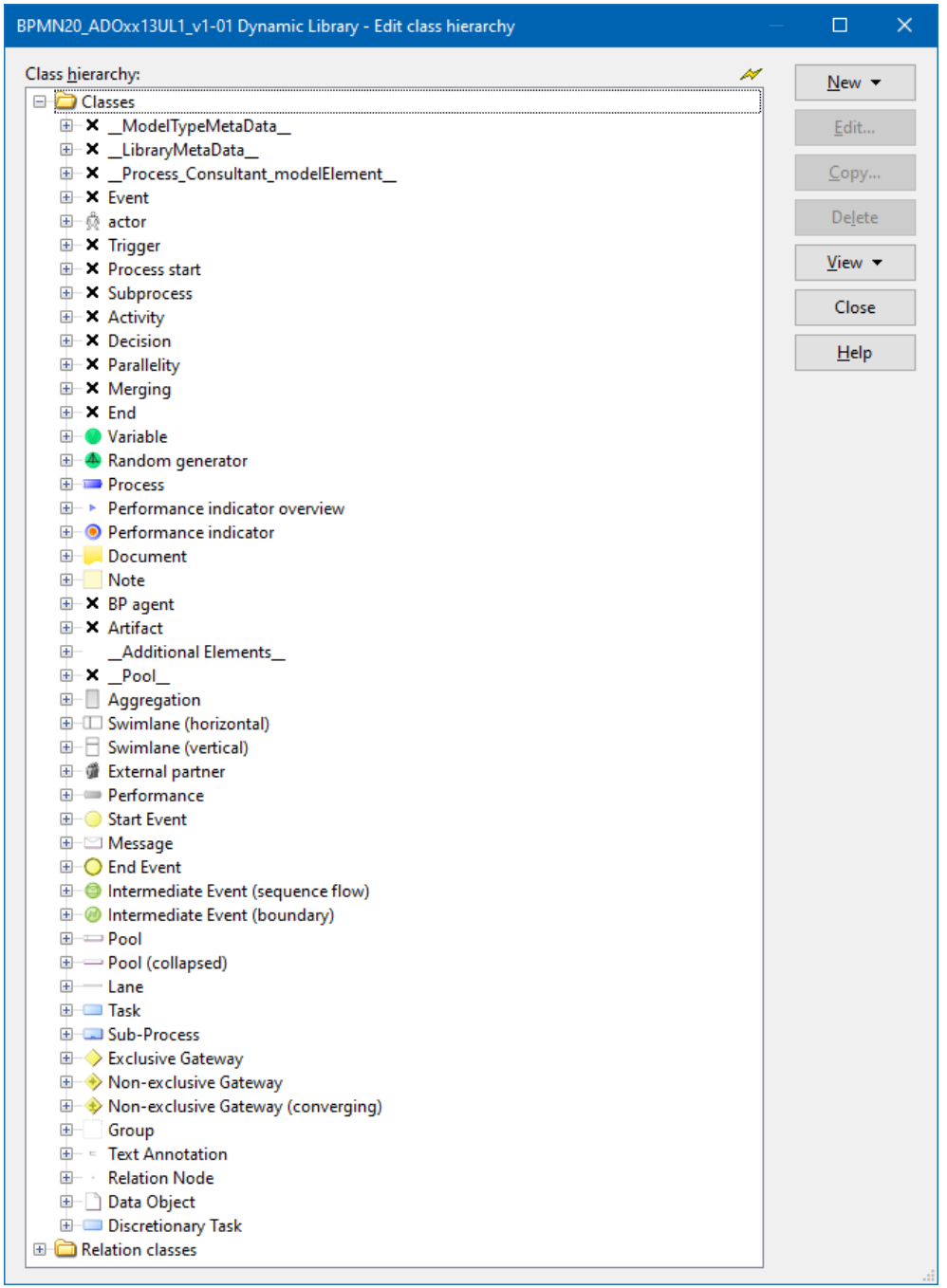


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Class Hierarchies

- ADOxx distinguishes
  - Classes
  - Relation classes

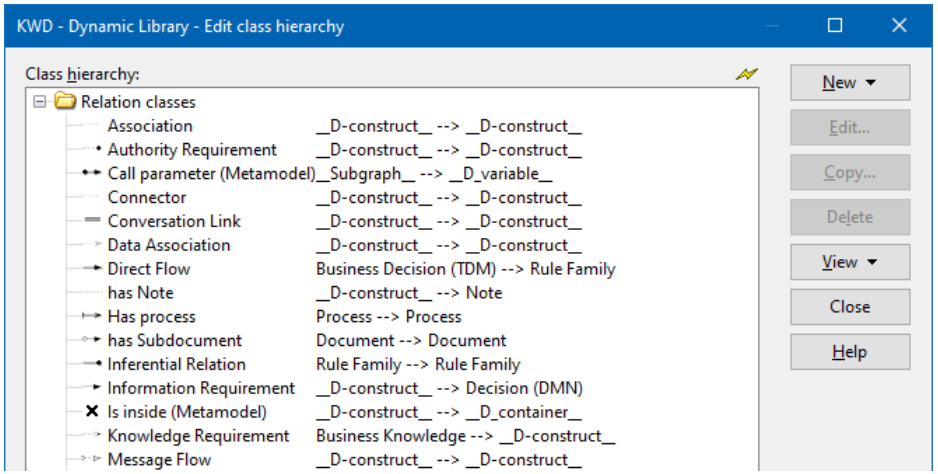


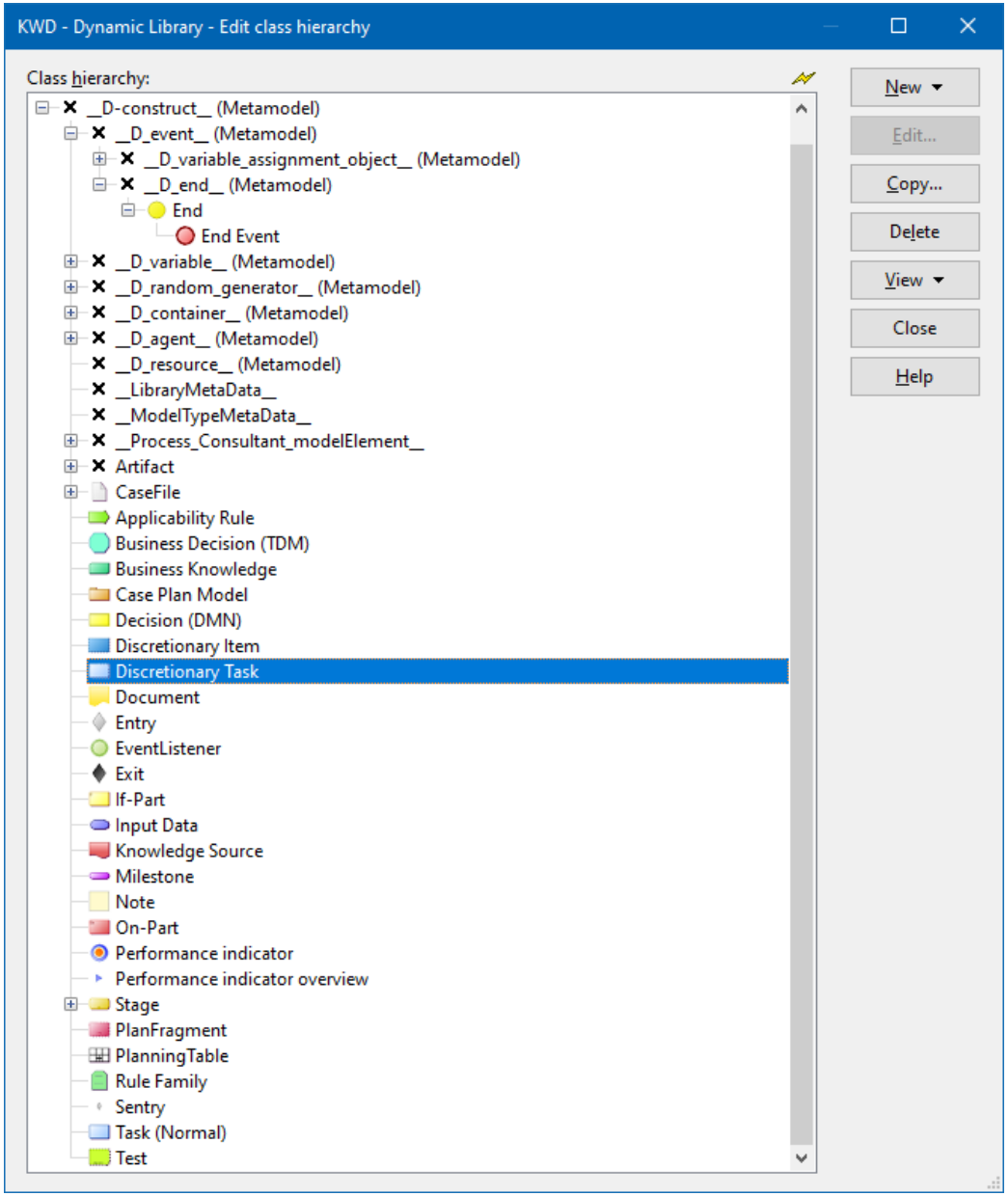


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Class Hierarchies

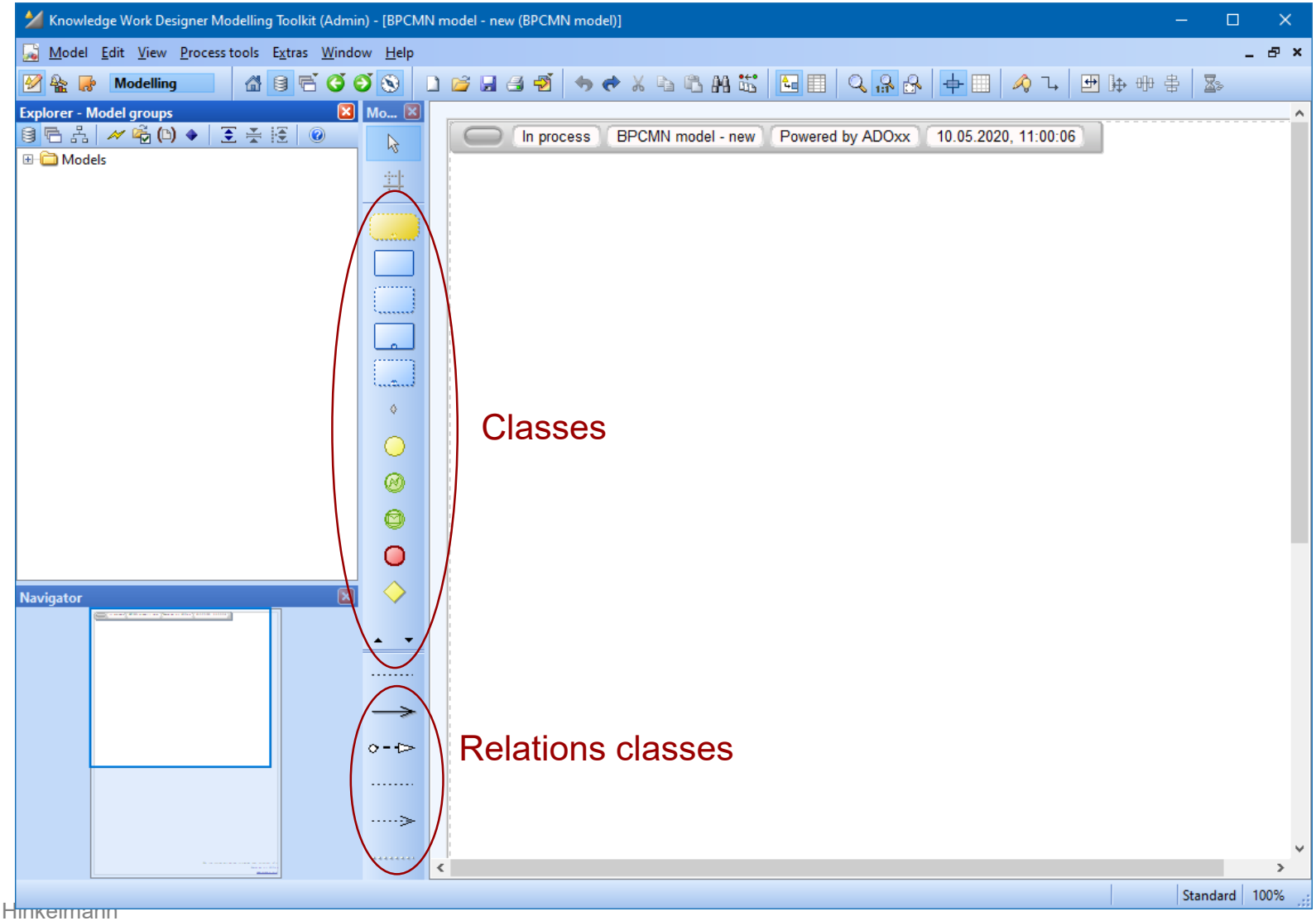
- ADOxx distinguishes
  - Classes
  - Relation classes





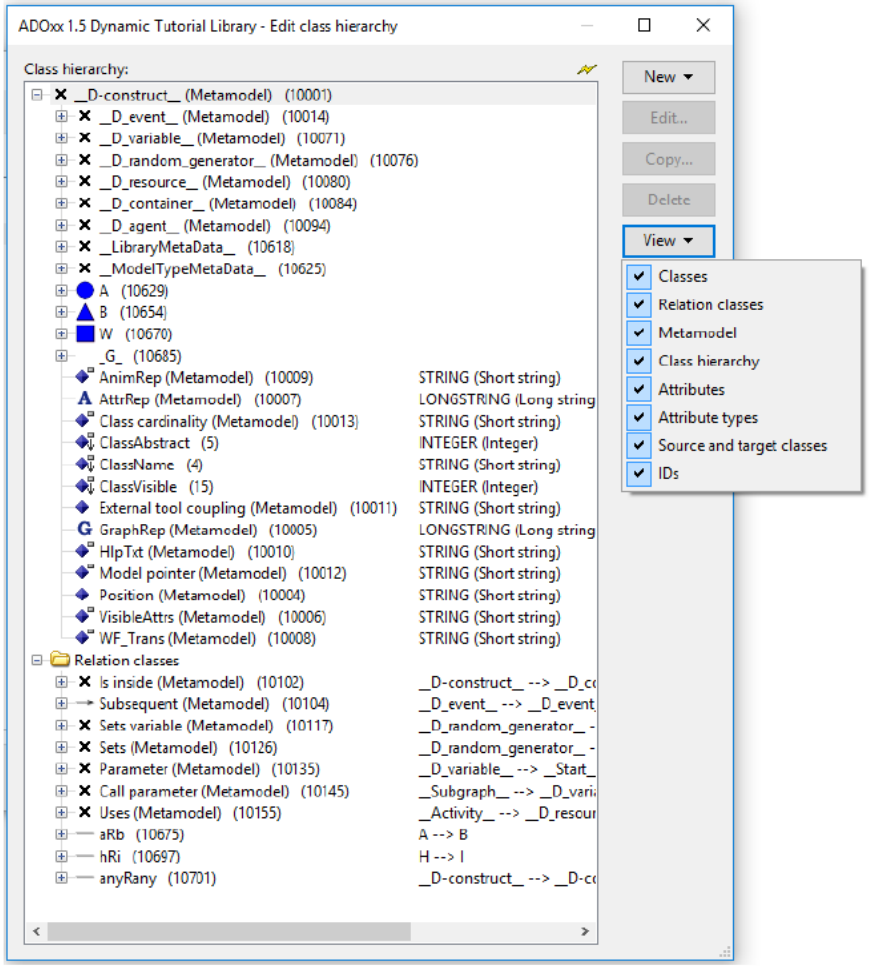
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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)

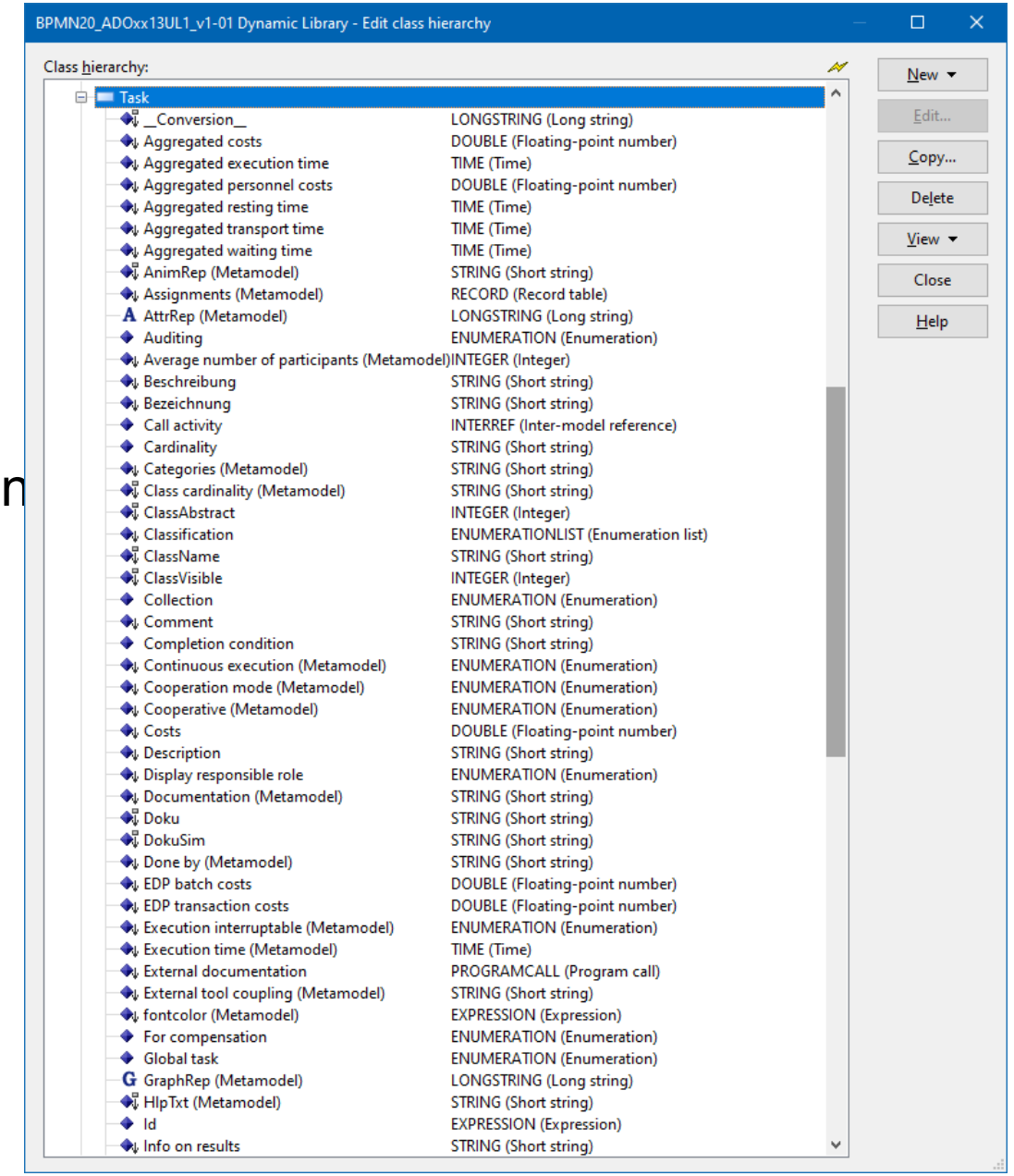
 **Class attribute**

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

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

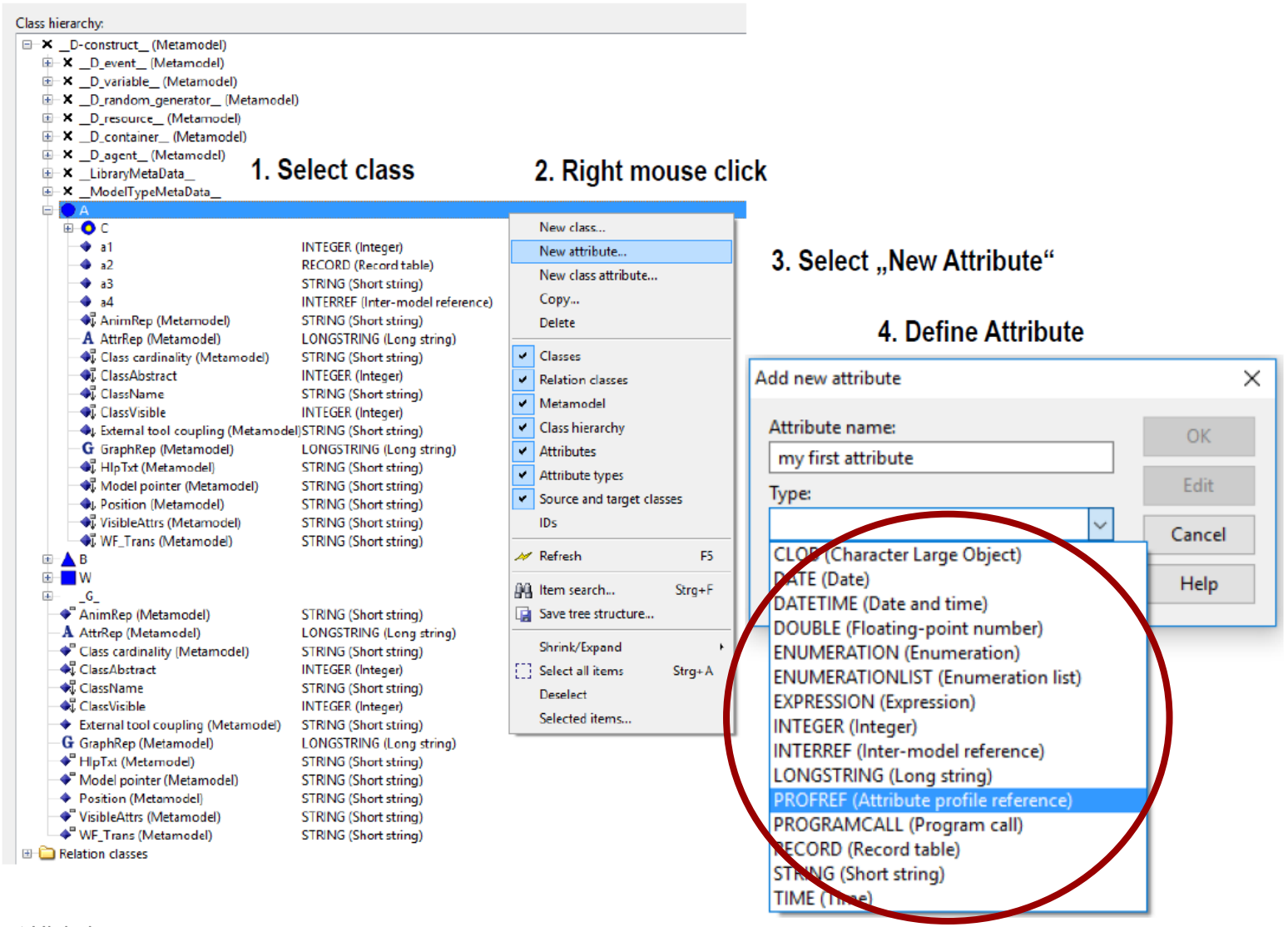
Attributes

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



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

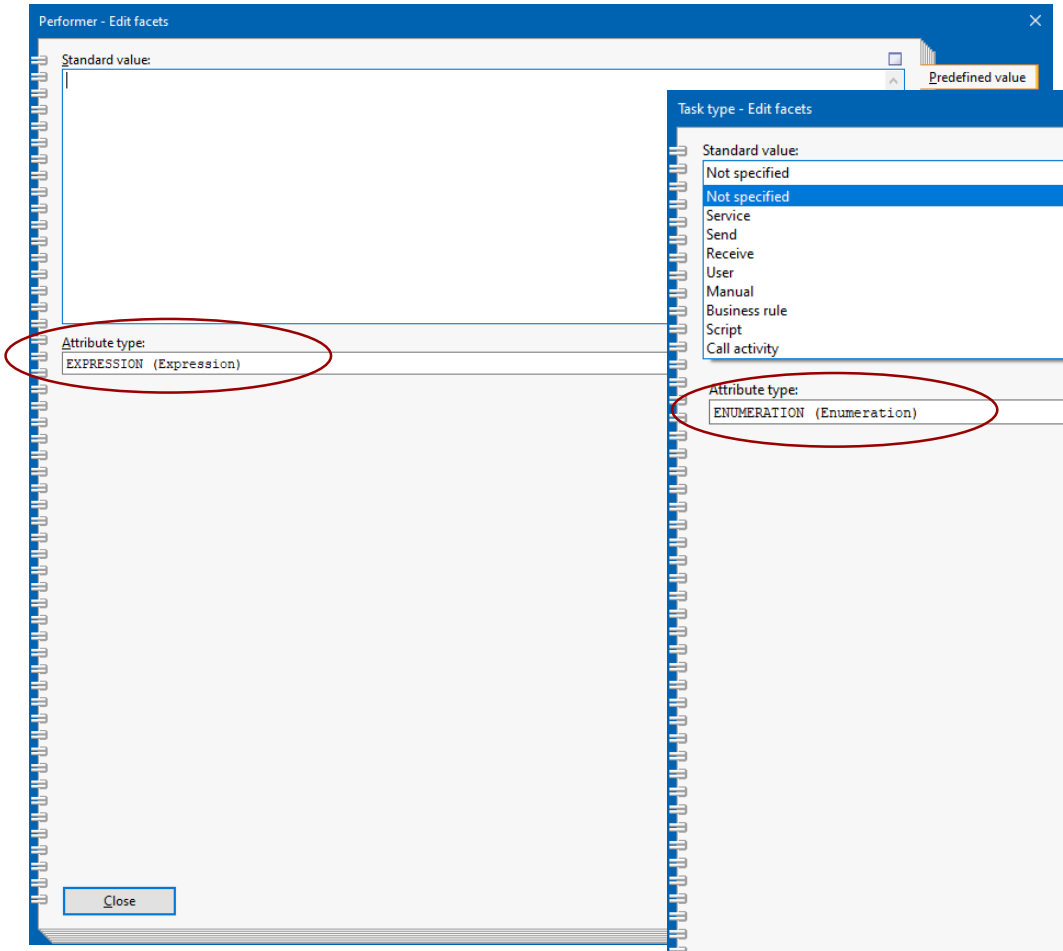
Defining a new Attribute



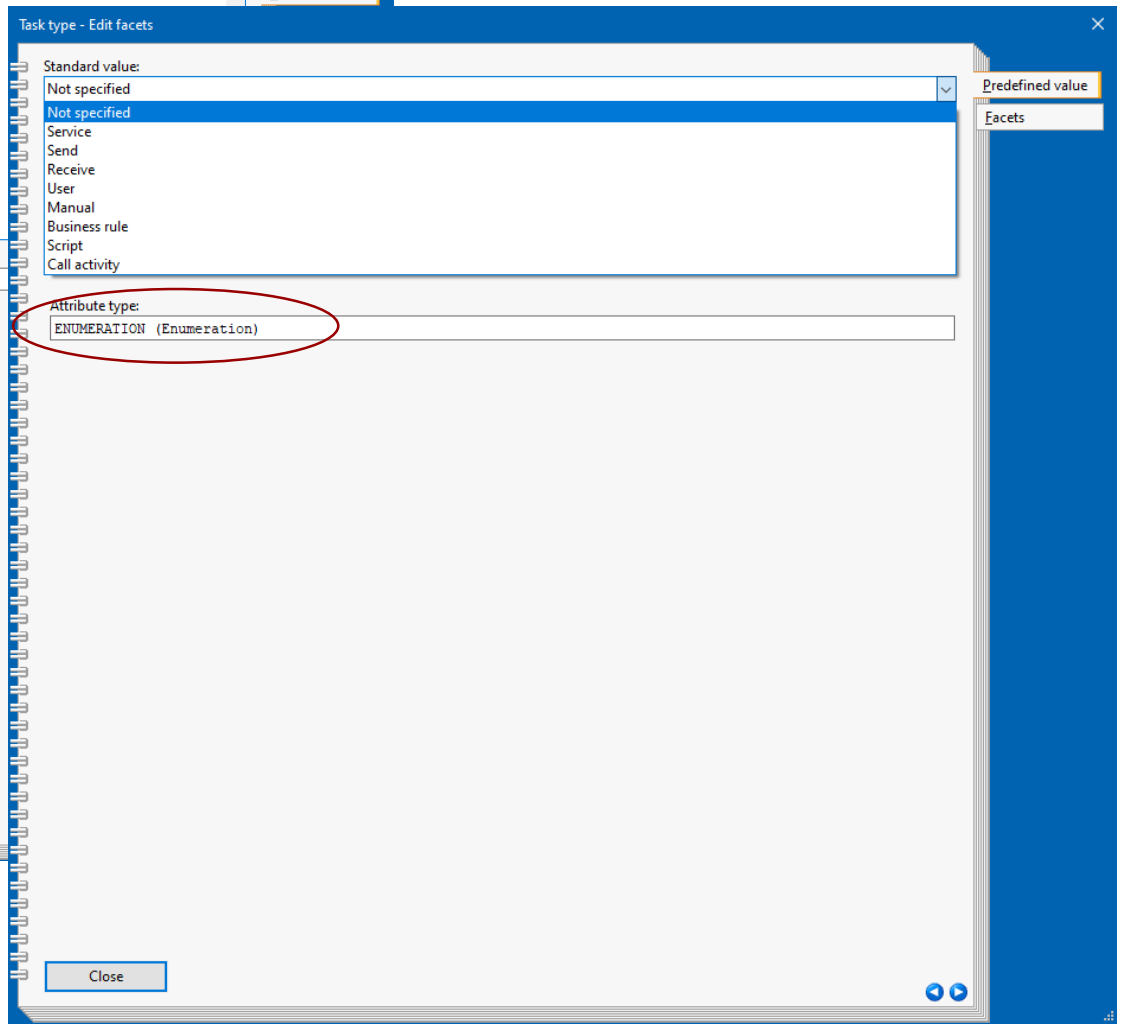
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Examples of Attributes

Performer



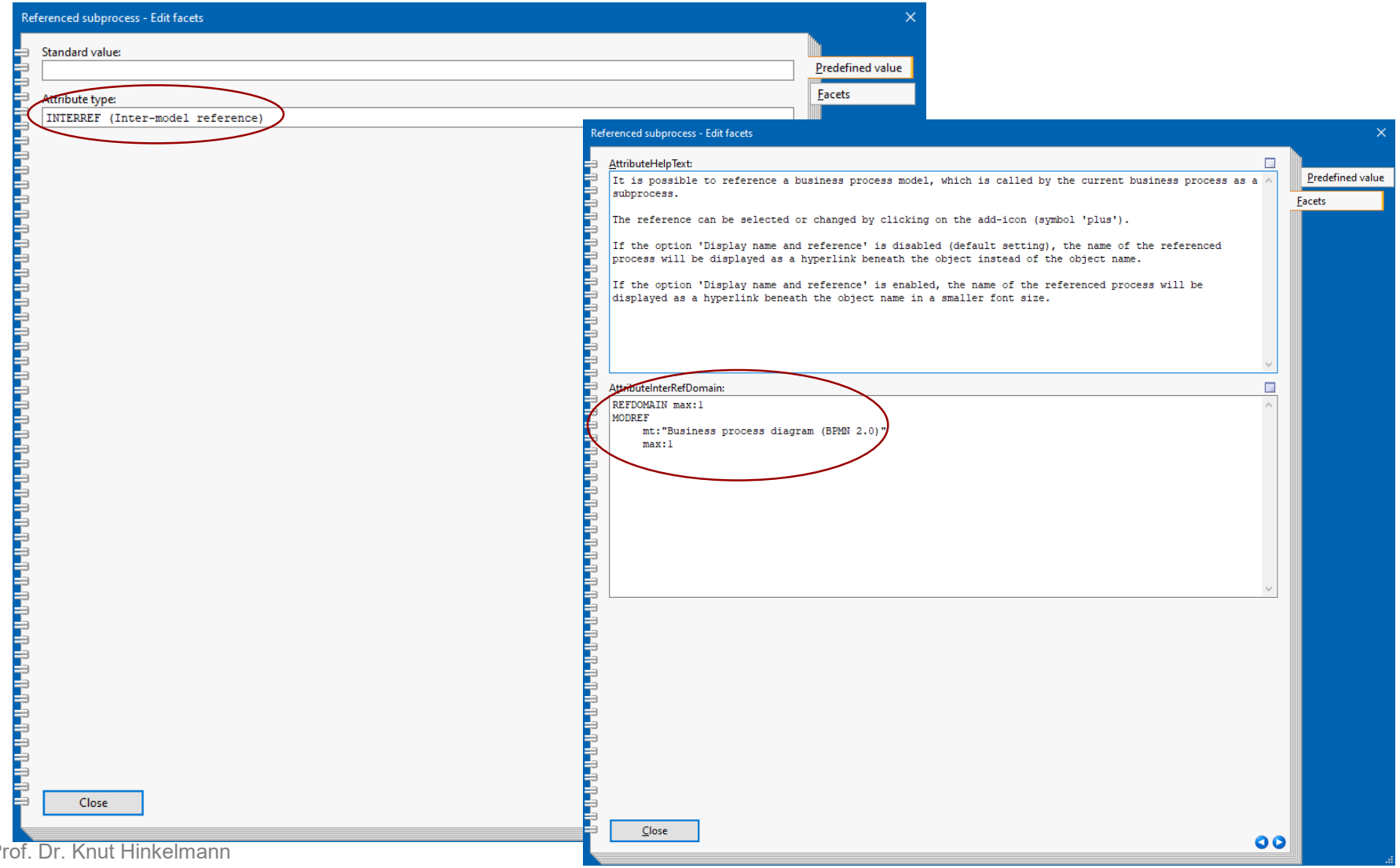
Task Type



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

References

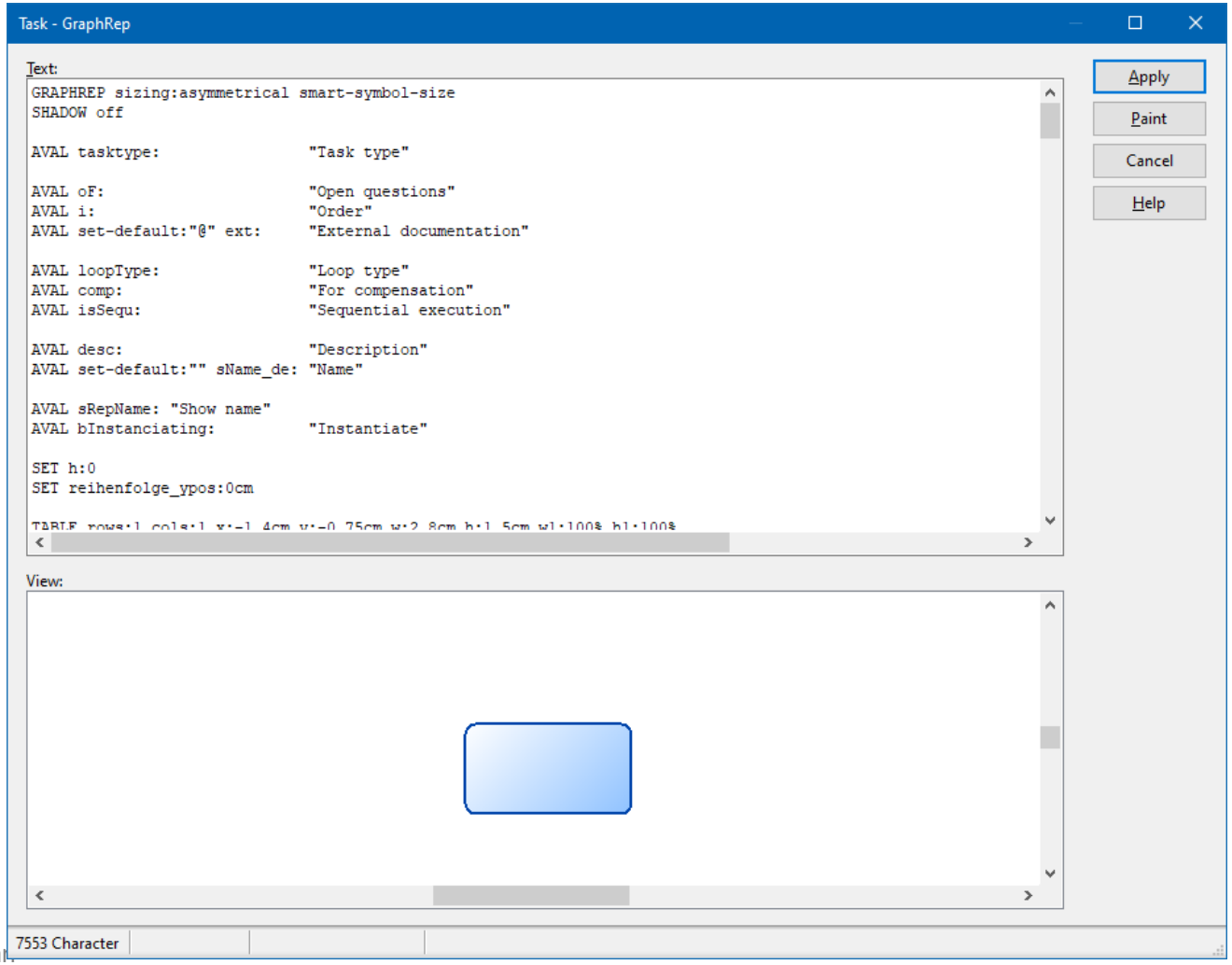
Referencing a Subprocess



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

Special Attribute GraphRep

GraphRep: A script language for the graphical representation



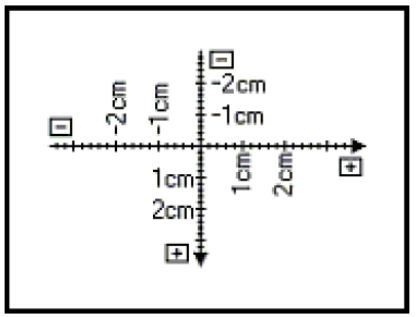
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

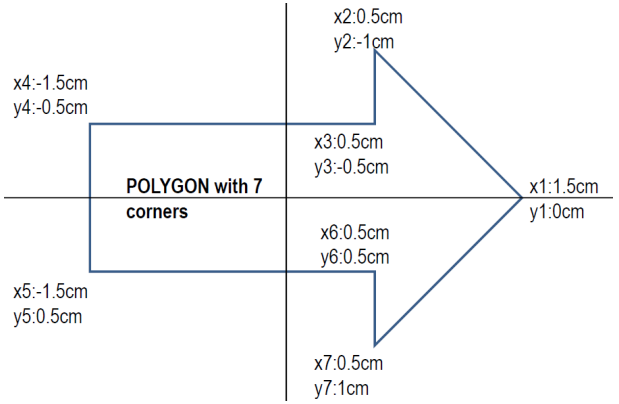
GraphRep Elements

- 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



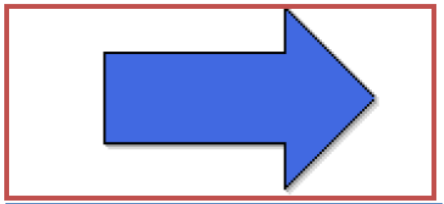


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

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
```

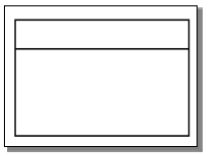
```
GRAPHREP
FILL color:royalblue
POLYGON 7 x1:1.5cm y1:0cm x2:0.5cm
y2:-1cm x3:0.5cm y3:-0.5cm x4:-1.5cm
y4:-0.5cm x5:-1.5cm y5:0.5cm
x6:0.5cm y6:0.5cm x7:0.5cm y7:1cm
ATTR "Name" y:1.4cm w:c h:c
```



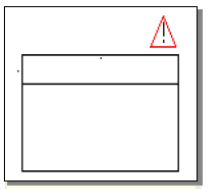
In case attribute name is available, it is shown here

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

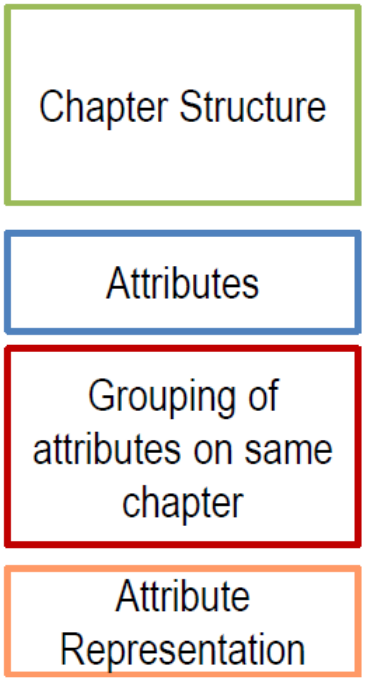
AttrRep

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

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

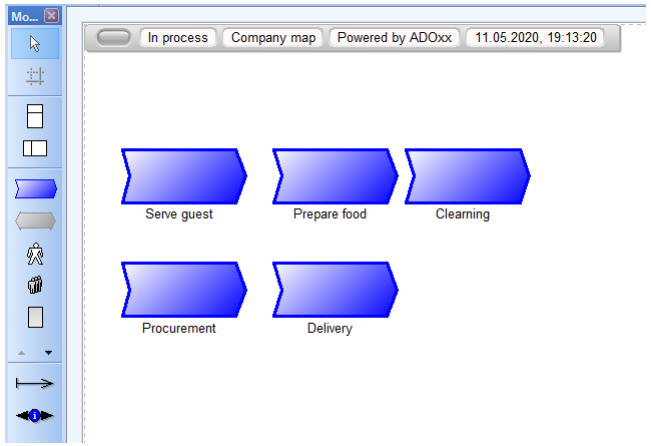
```
NOTEBOOK
CHAPTER "Definition"
ATTR "Name"
ATTR "Description"
CHAPTER "Dialectic Influence"
ATTR "Influencing dialectics" lines:10
```

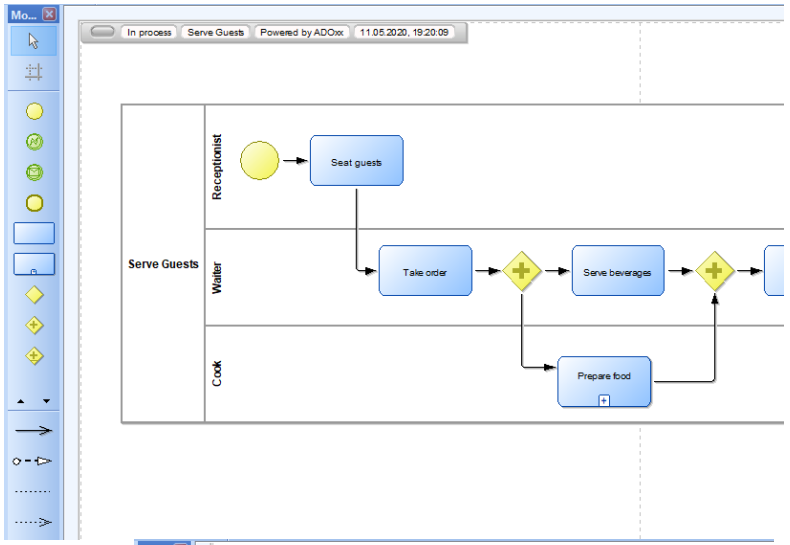
```
NOTEBOOK
CHAPTER "Definition"
ATTR "External graphic"
```

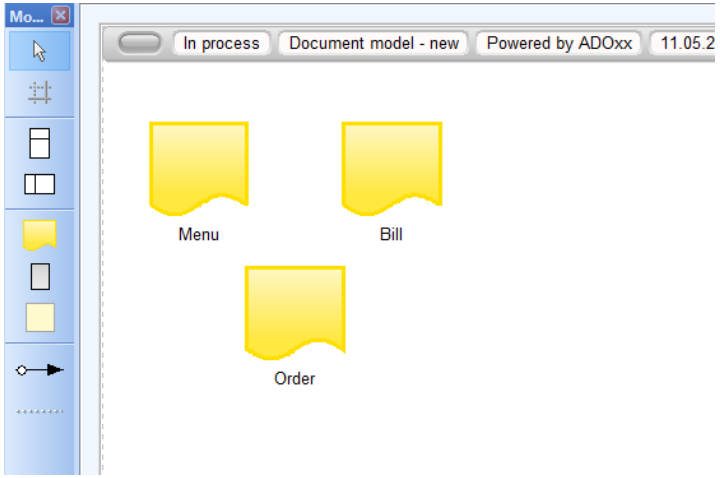


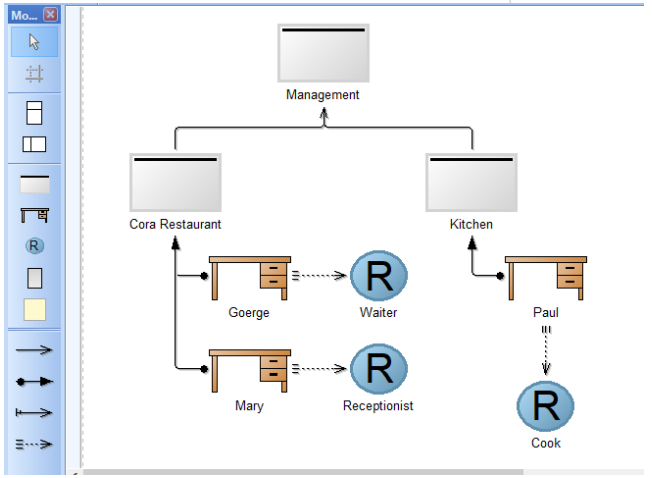
Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Model Types: Represention Views on the Knowledge



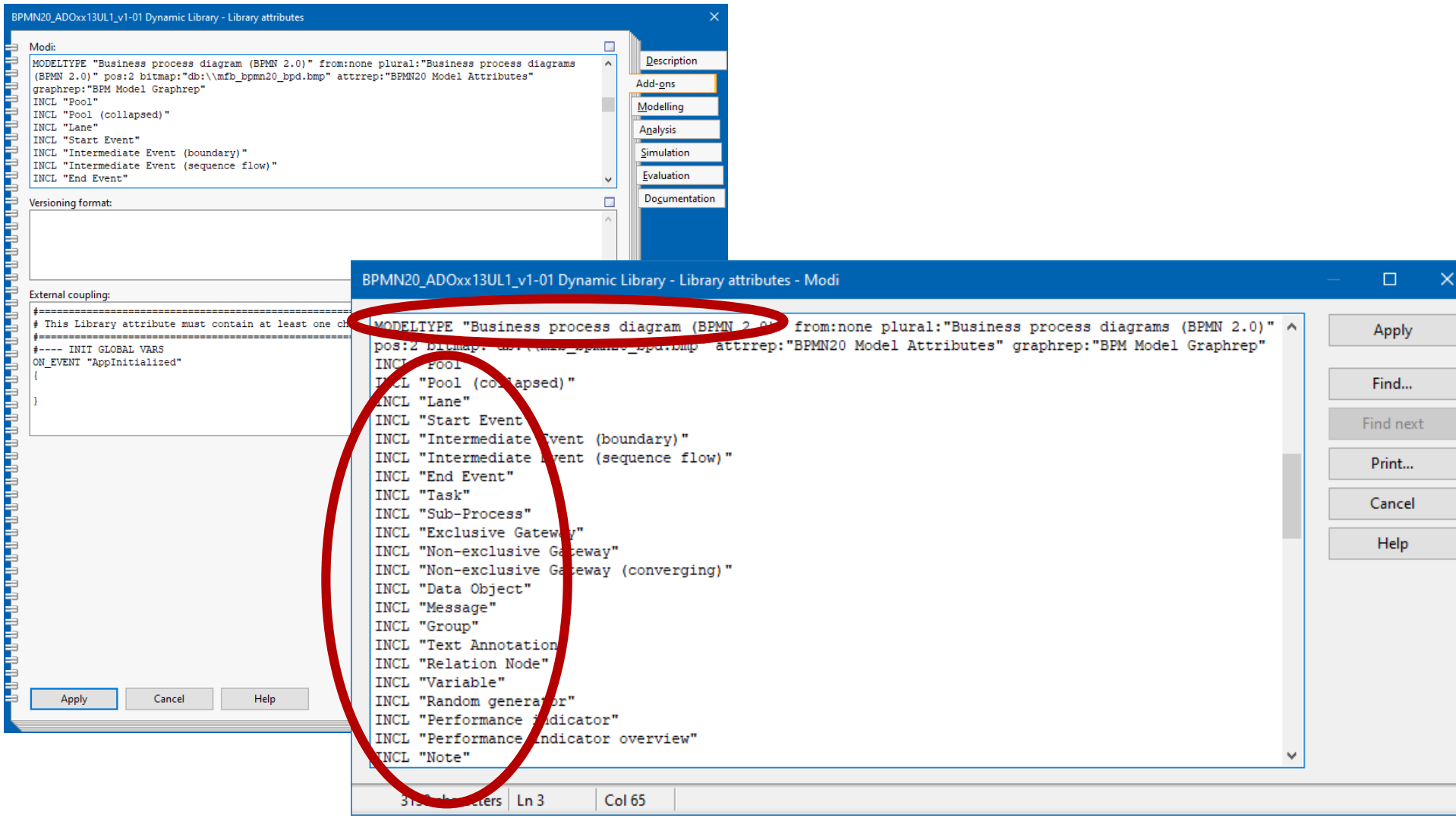






Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Classes are assigned to Model Types



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