

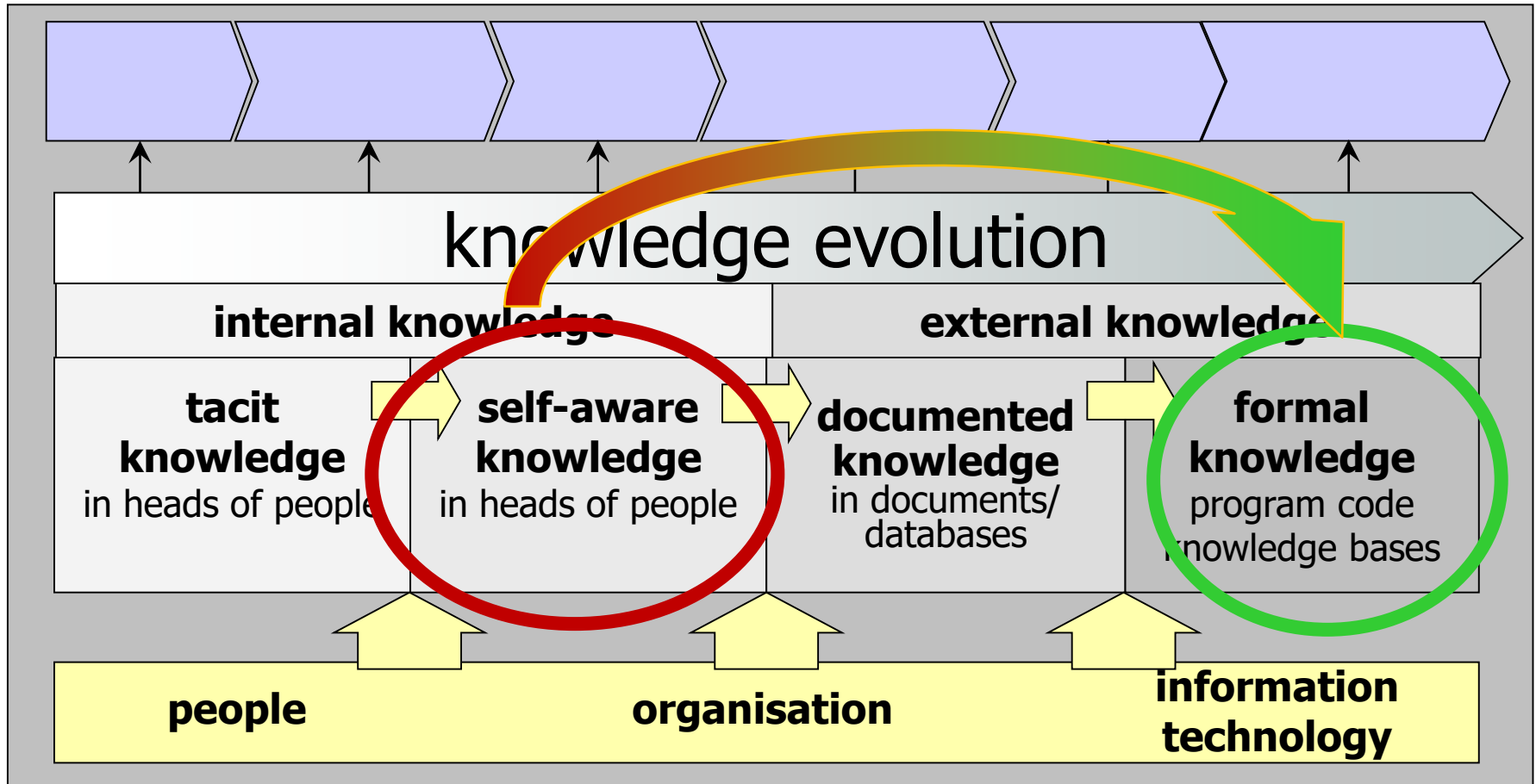


Modelling and Metamodelling

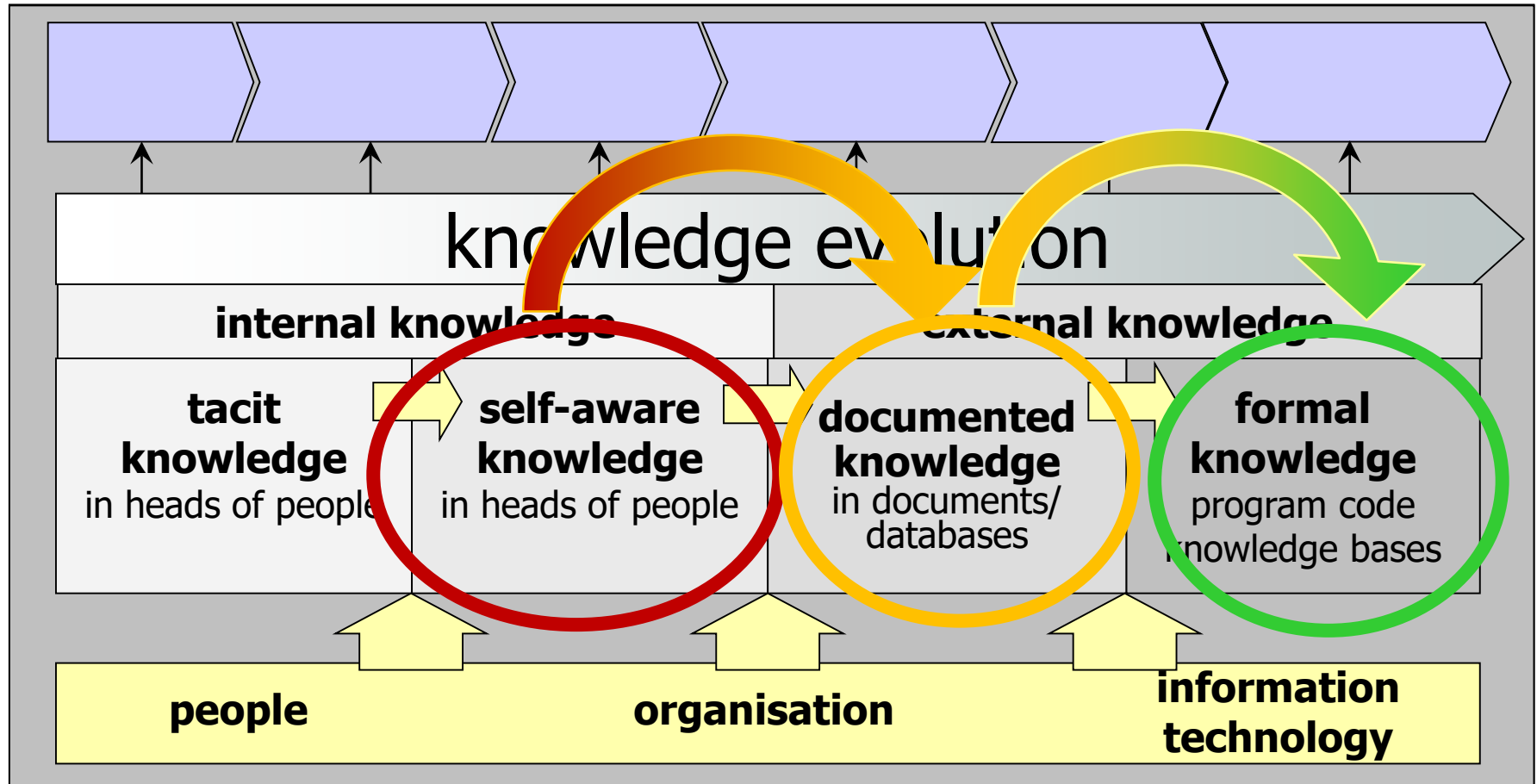
Knut Hinkelmann



Ontology Engineering



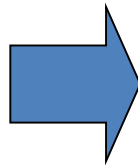
A Two-step Approach for Building a Knowledge Base



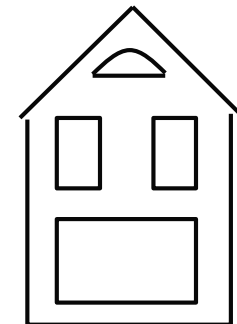
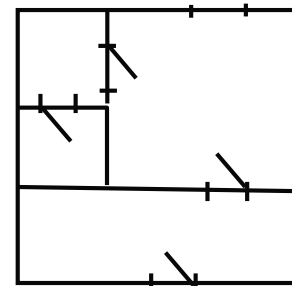
Models

- 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

real object



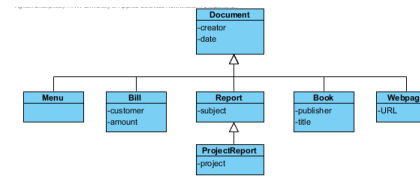
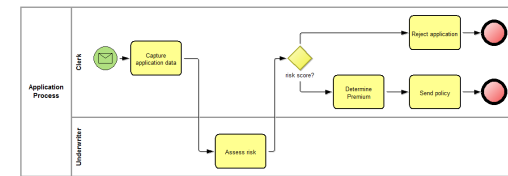
models (plan)



Models

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

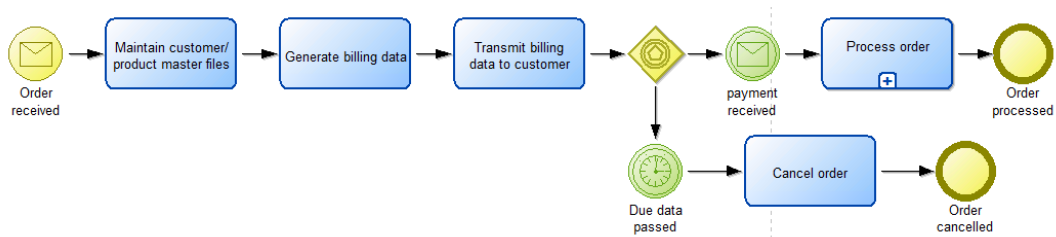
The Application Process
 In the business process for health insurance, first the application data are captured by the clerk. Then the risk assessment is made by the underwriter. Depending on the risk score, the clerk determines the premiums and sends the policy or the application is rejected.



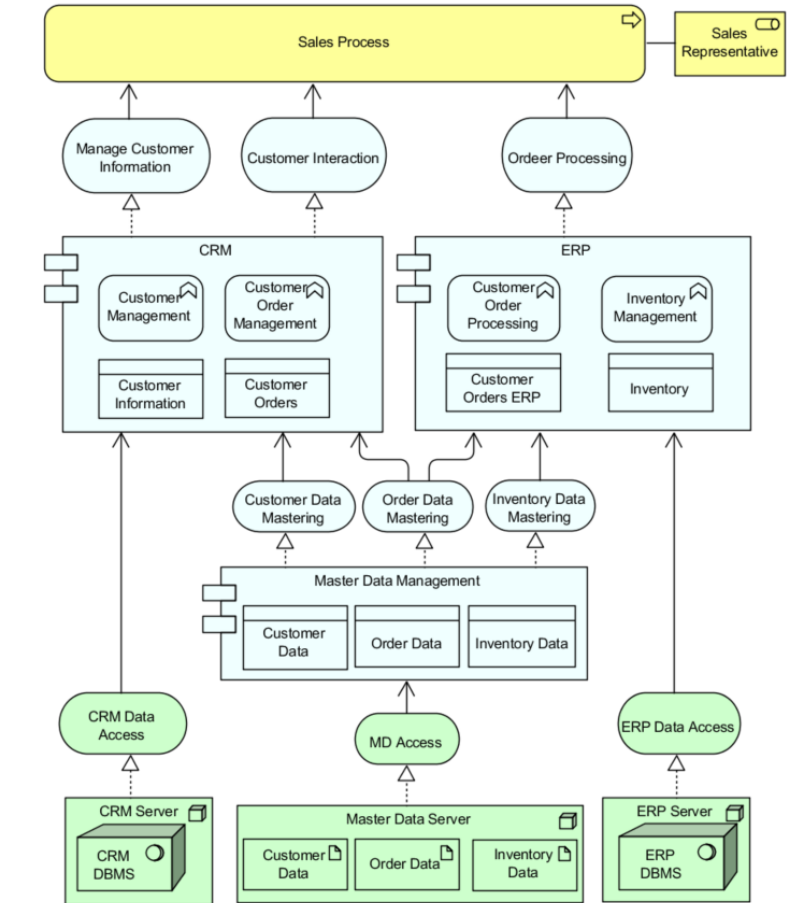
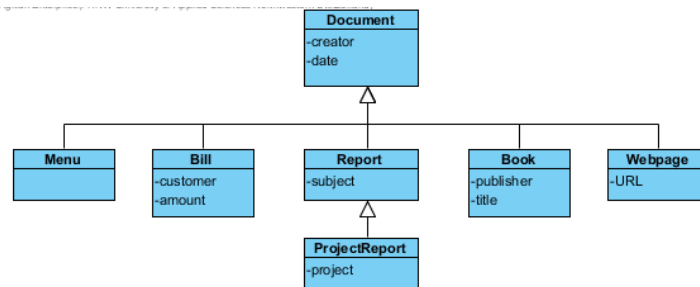
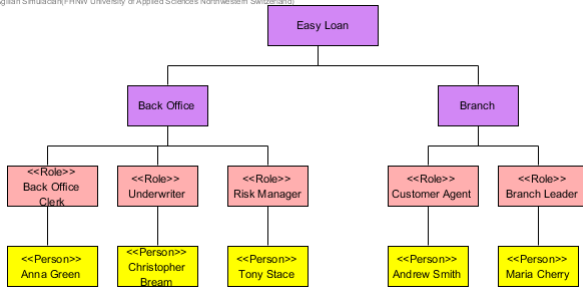
$$E = m c^2$$



Enterprise Models

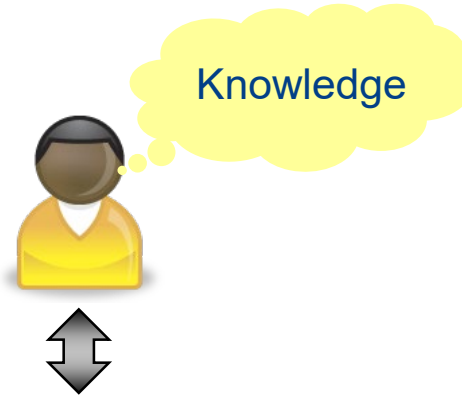


Agilan Simulacran/FHWM University of Applied Sciences Northwestern Switzerland

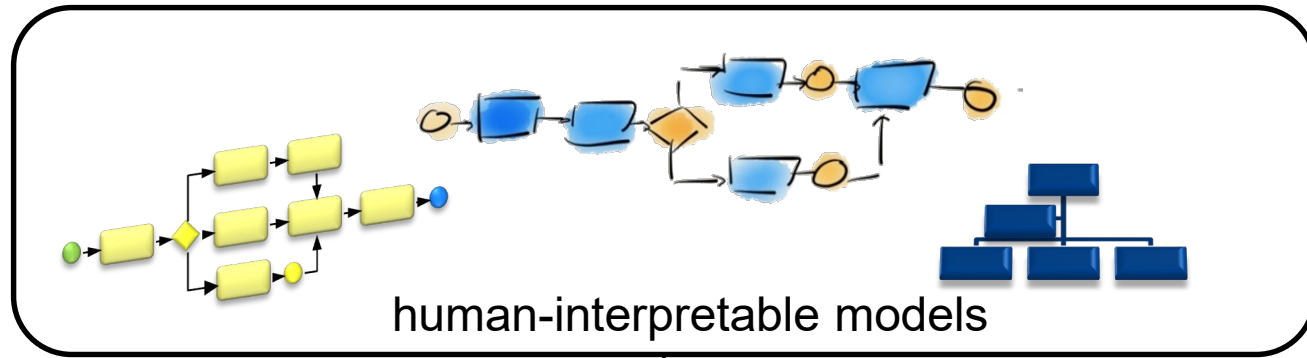


Human Problem Solving

*Communication/
Analysis/
Decision Making*



Models



Reality



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.

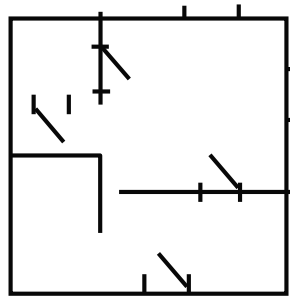
Model in Architecture

real object



house

model



architect's drawing
(plan)

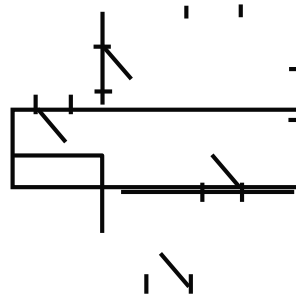
Model and Modelling Language in Architecture

real object



house

model



architect's drawing
(plan)

modelling language
(concrete syntax)

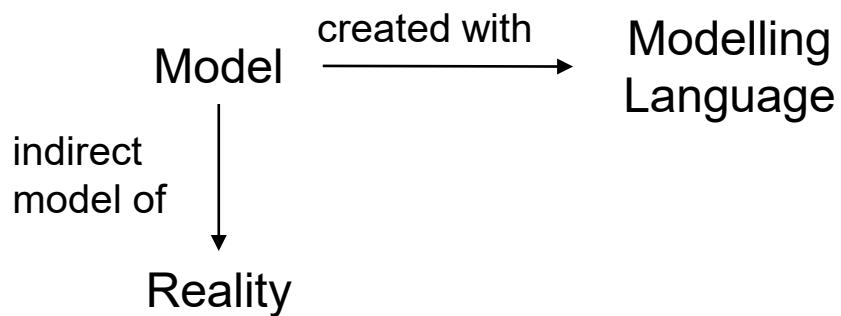
object types:

— wall

⊥ door

+—+ window

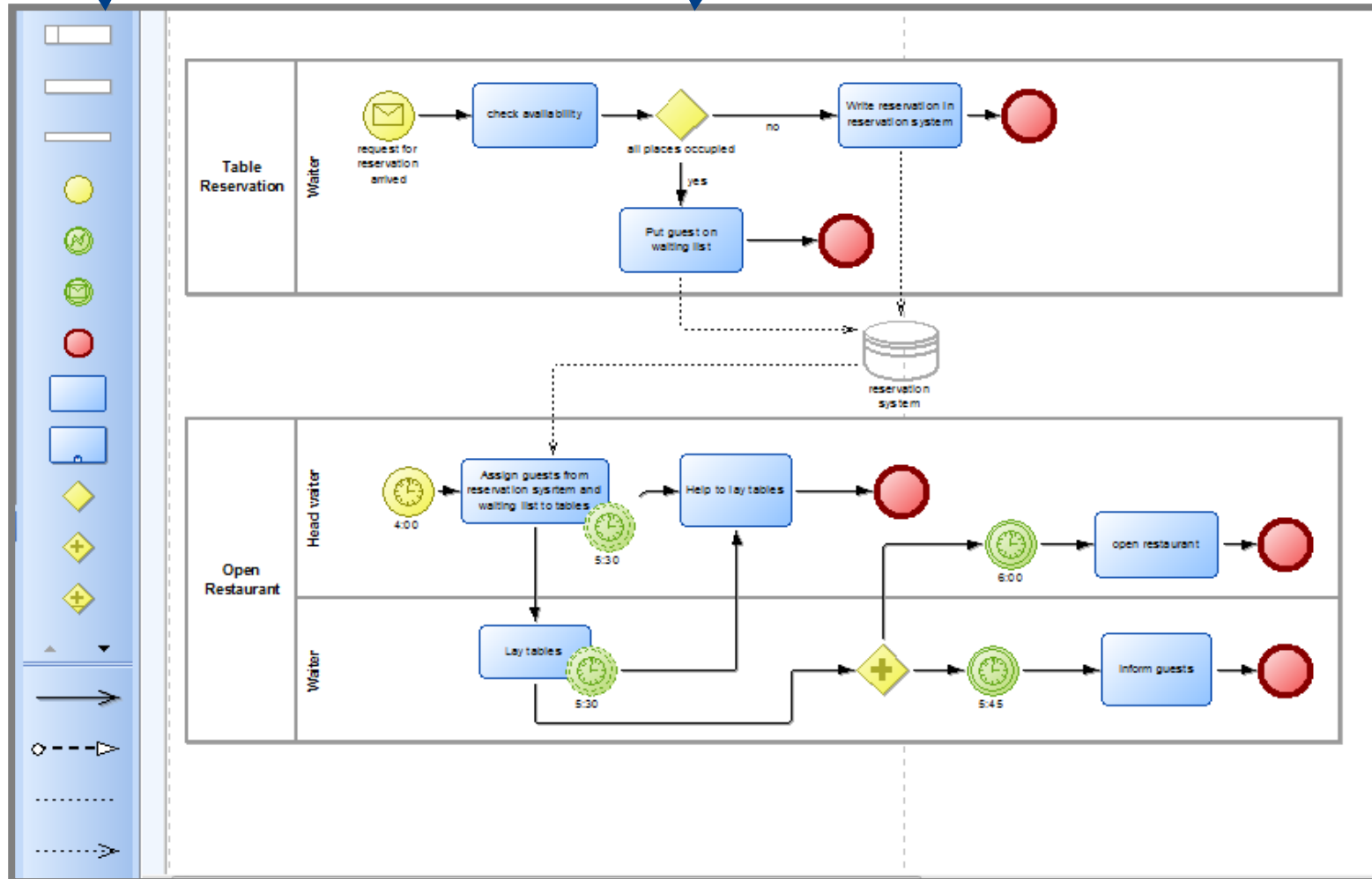
Modelling Language



- 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

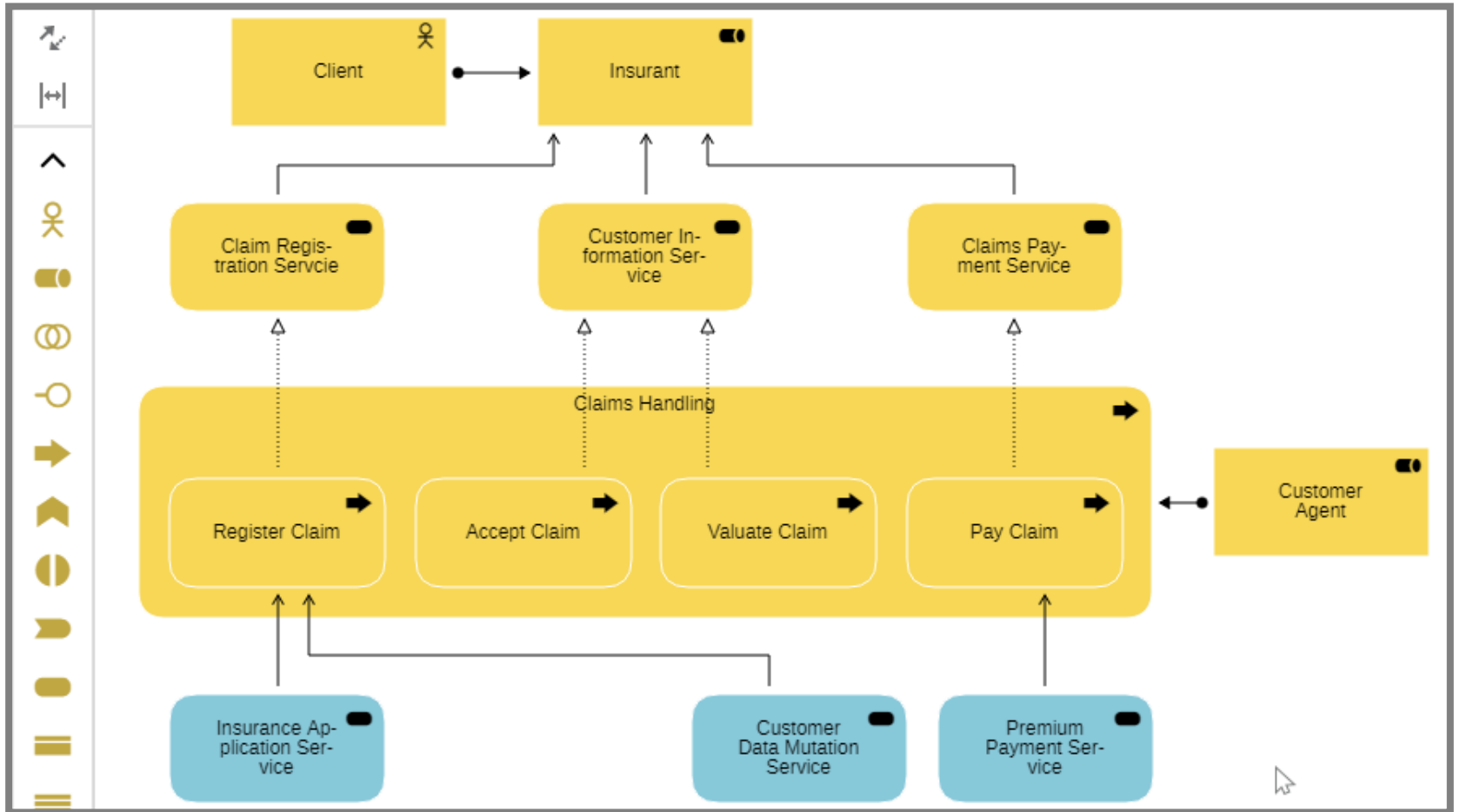
Modelling Language

Model



Modelling Language

Model



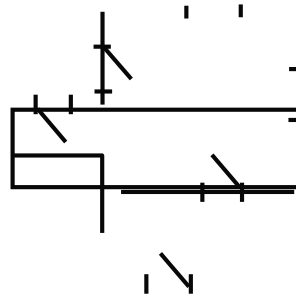
Model and Meta-Model in Architecture

real object



house

model



architect's drawing
(plan)

modelling language (concrete syntax)

object types:

— wall

⊥ door

+—+ window

meta-model (abstract syntax)

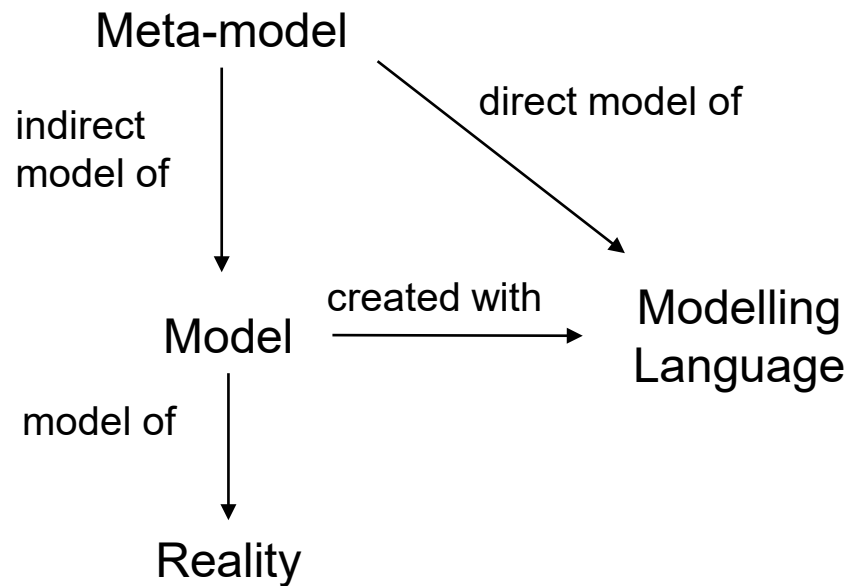
object types:

- wall
- door
- window

rules:

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

Meta-model



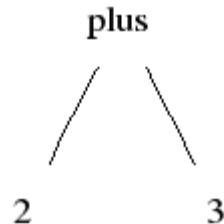
A meta-model defines the semantics of the modelling language, i.e. the building blocks that can be used to make a model. It defines the

- ◆ 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.

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



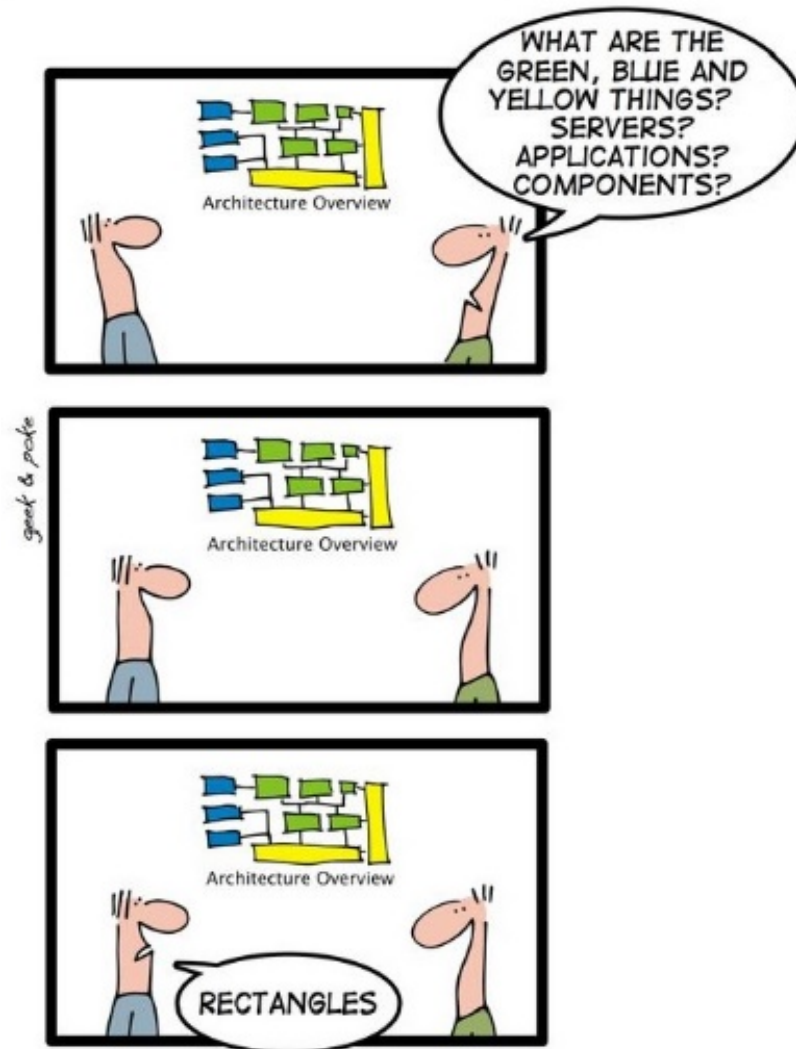
Concrete Syntax

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

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

<code>2 + 3</code>	-- infix
<code>(+ 2 3)</code>	-- prefix
<code>(2 3 +)</code>	-- postfix
<code>bipush 2 bipush 3 iadd</code>	-- JVM
<code>the sum of 2 and 3</code>	-- English

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



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*

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

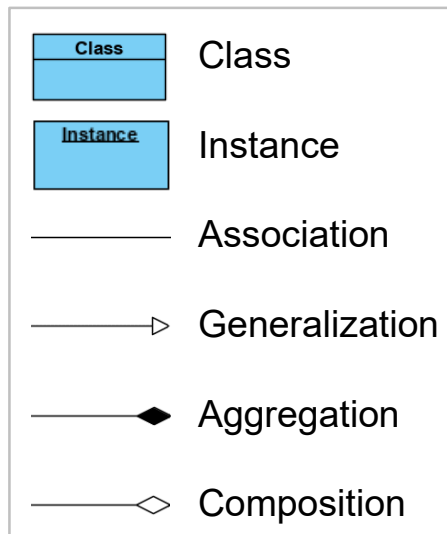
- «classes», «instances»,

and relations for

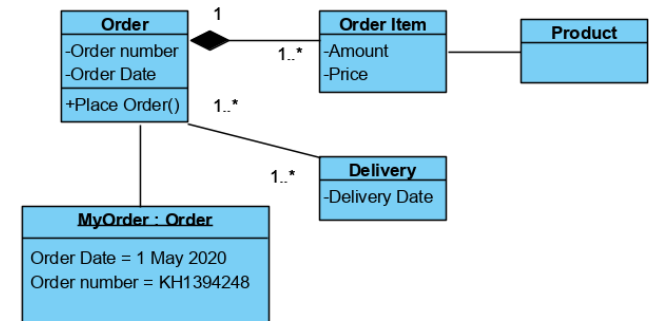
- «association», «generalization», «aggregation» and «composition»

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»

Illustration: Meta-model and Model for Processes

Metamodel:

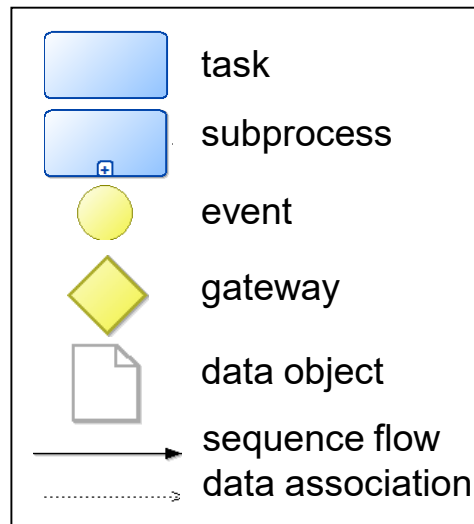
Abstract syntax:
Concepts and relations
which can be used to create
models.

Example: A process model
consists of concepts for

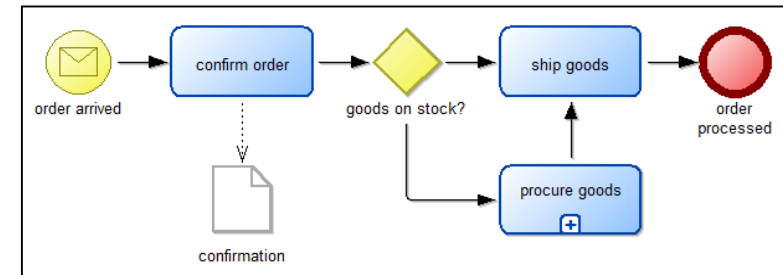
- «task», «subprocess»,
«event», «gateway»,
«data object»
and relations for
- «sequence flow»,
«data association».

Modelling Language:

Concrete syntax:
Notation/appearance of
meta-model elements



Model:

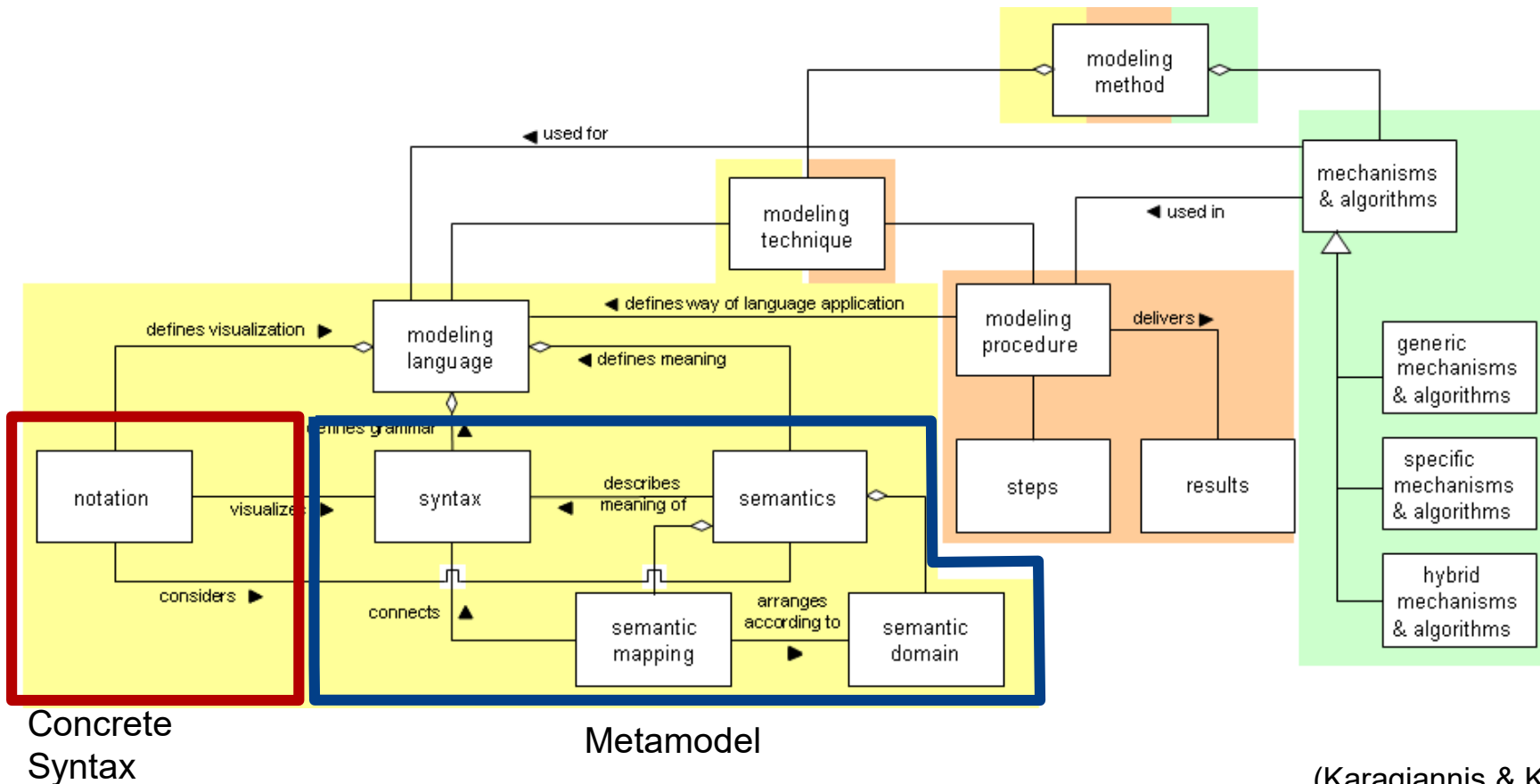


*A model contains instances of the
object types defined in the meta-
model, according to the concrete
syntax of the modelling language.
The object „confirm order“
represents a real entity; it is an
instance of the object type «task»*

Components of Modelling Methods

A Modelling Language is Part of a Modelling Modelling

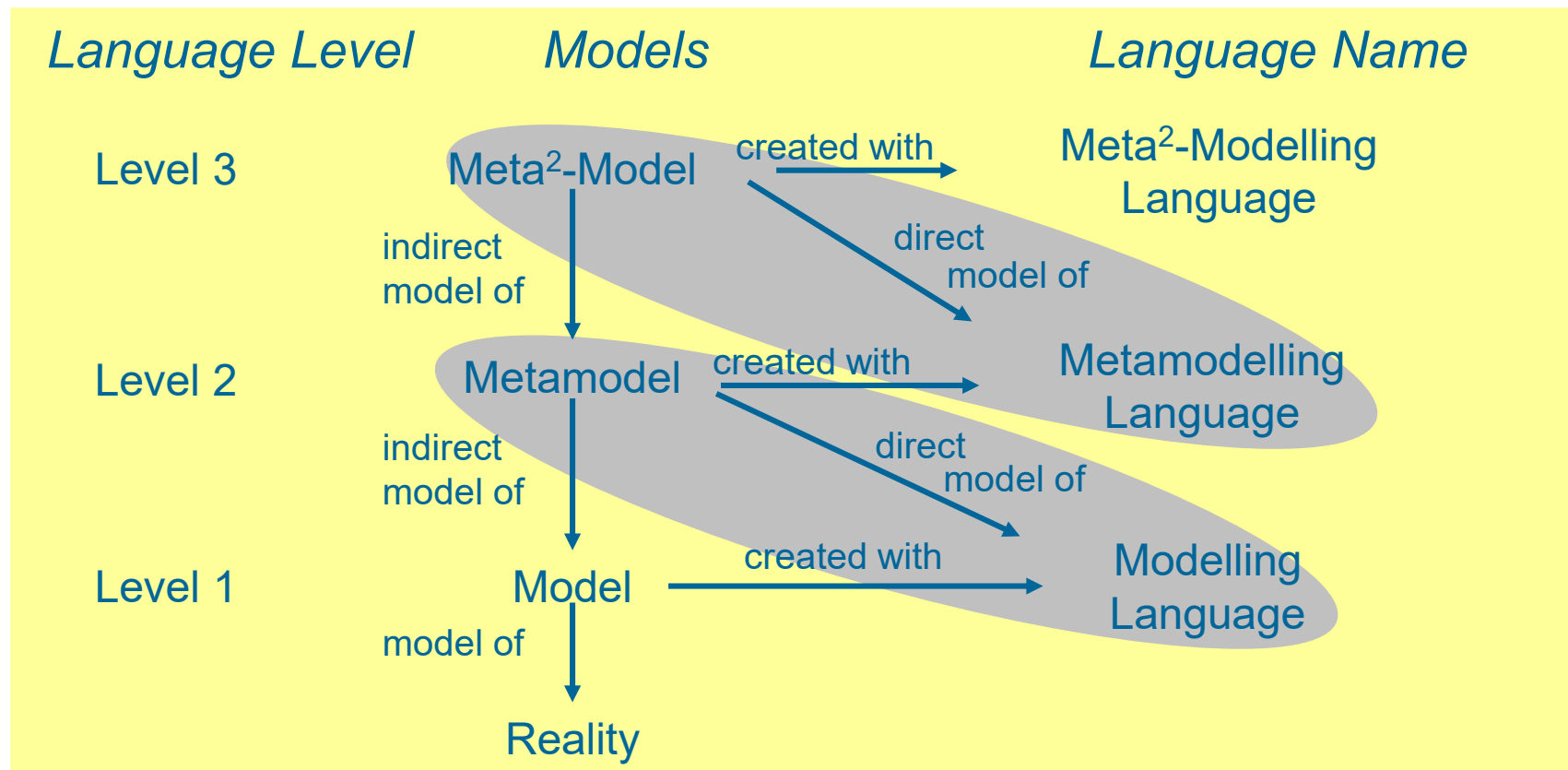
A Modelling Language consists of the Metamodel (Abstract Syntax and Semantics) and the Notation



(Karagiannis & Kühn 2002)

Meta Model Hierarchy

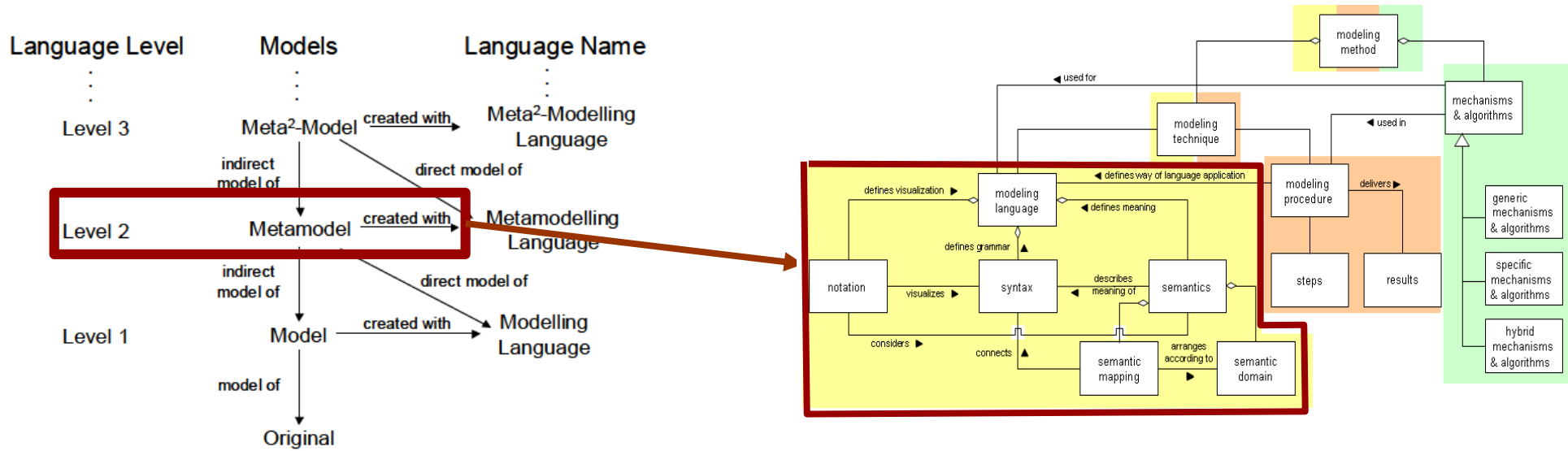
The meta-model must again be described in some language, which has to be specified in a meta-meta-model



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

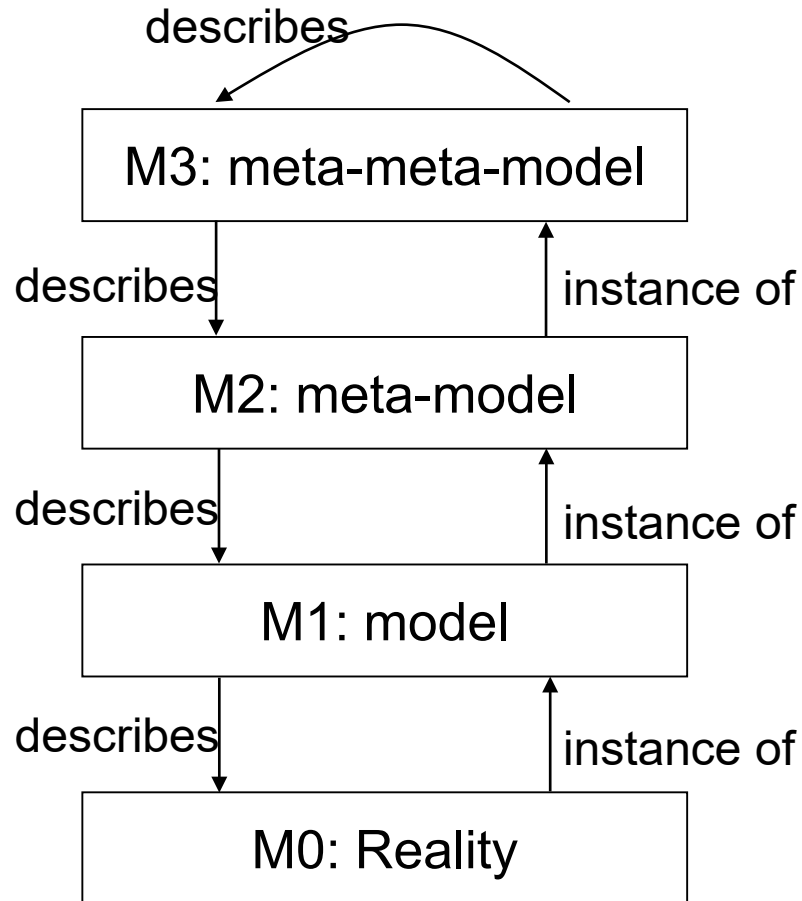
Modeling Language Definition



(Strahringer, 1996)

(Karagiannis & Kühn, 2002)

The Model Stack

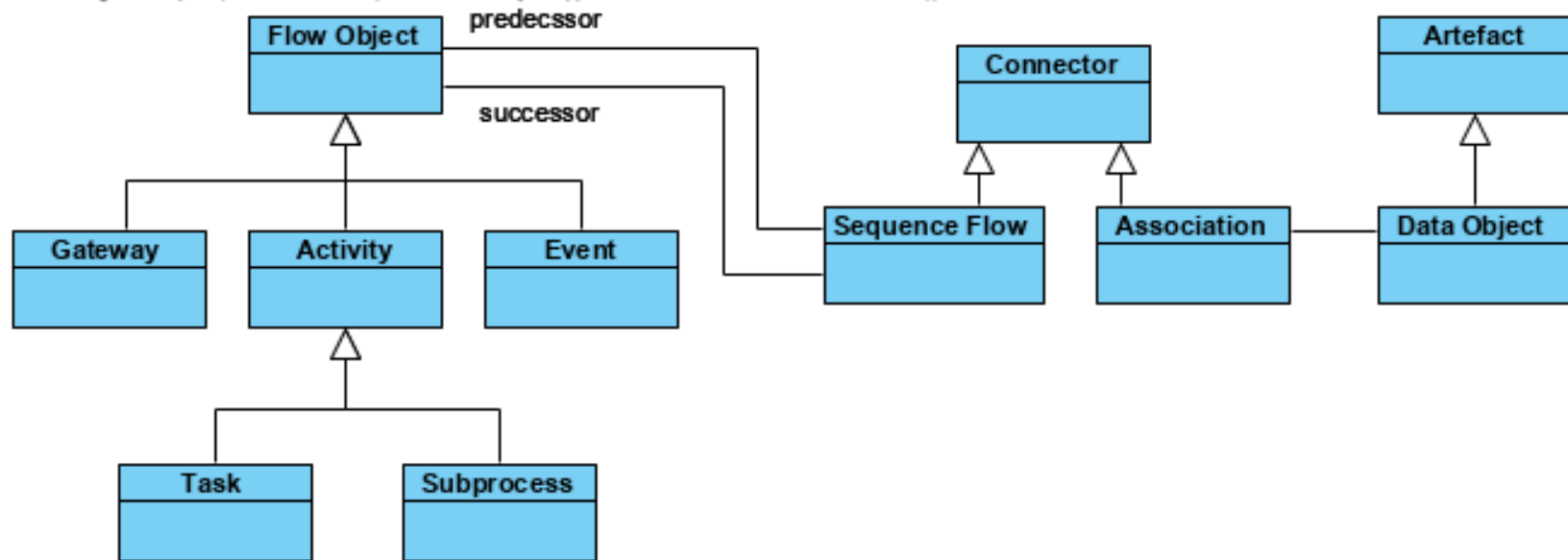


- A model is a **simplified representation of a reality**
- A meta-model defines a **modelling language** in which a model can be expressed.
- A meta-meta model defines the **language in which a meta-model** can be expressed.

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

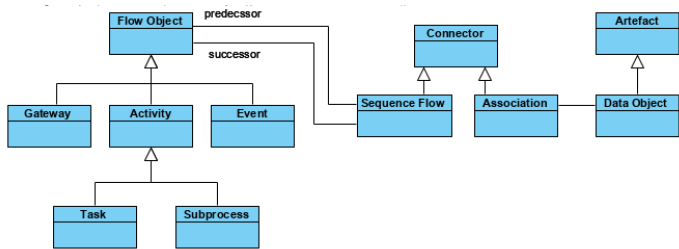


(UML Class diagrams were originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

A Metamodel for Processes

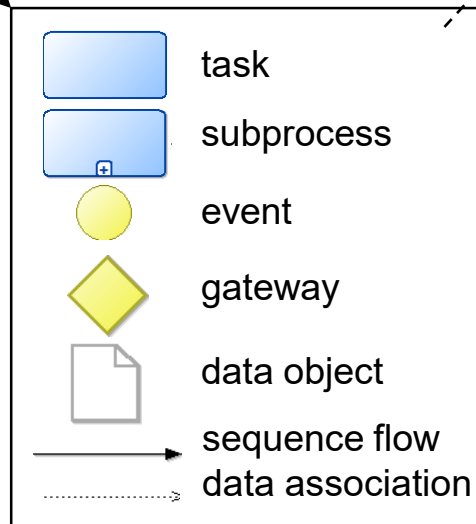
Meta-model:

- Classes and relations that can be used for modelling

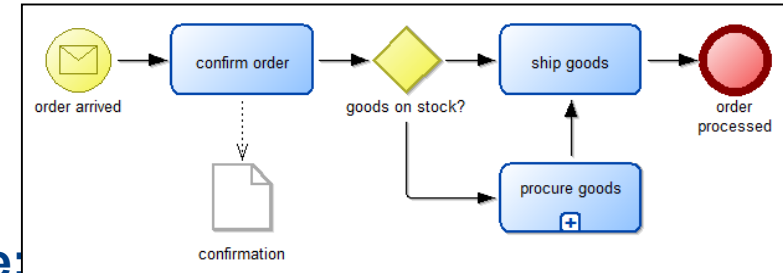


Modelling Language:

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

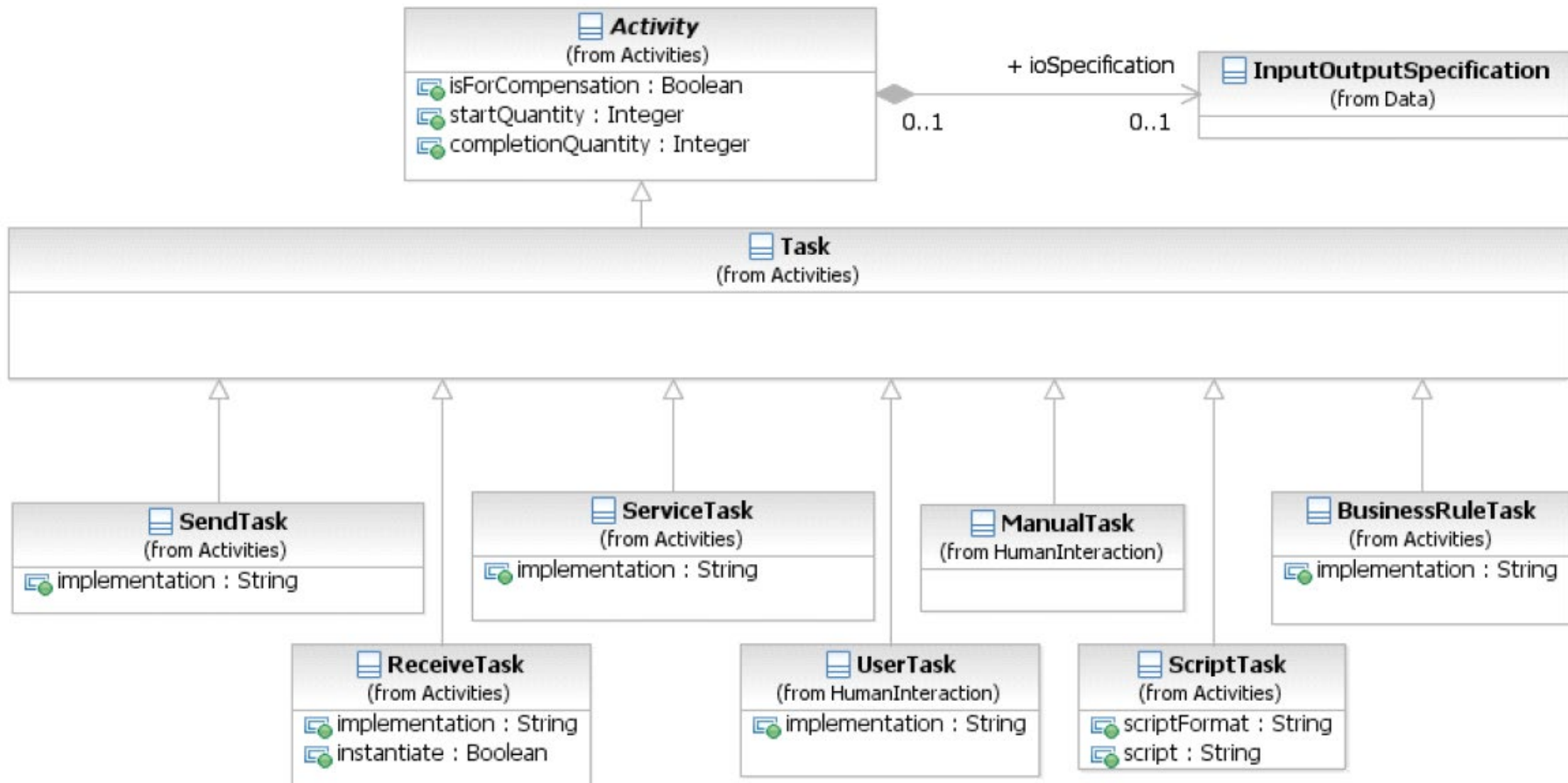


Model:



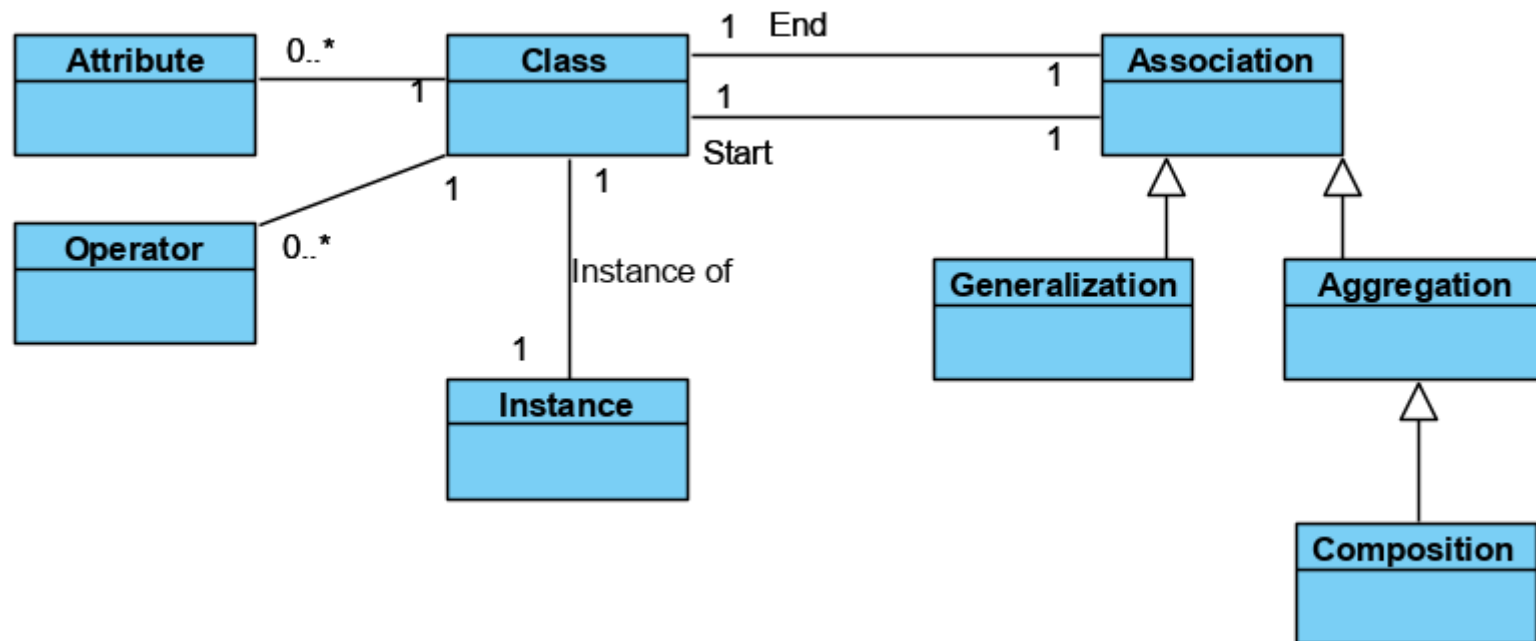
A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task»

Subset of the BPMN Metamodel in UML



A Metamodel for UML Class Diagrams

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

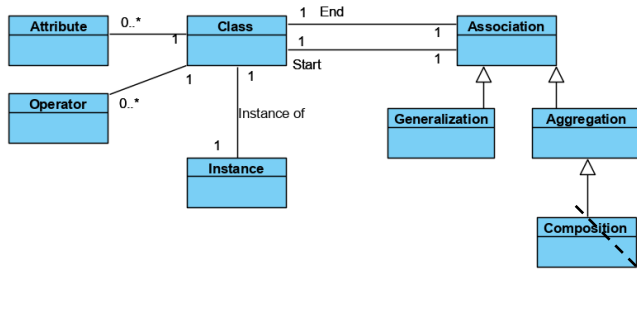


(UML Class diagrams were originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

A Metamodel for Class Diagrams

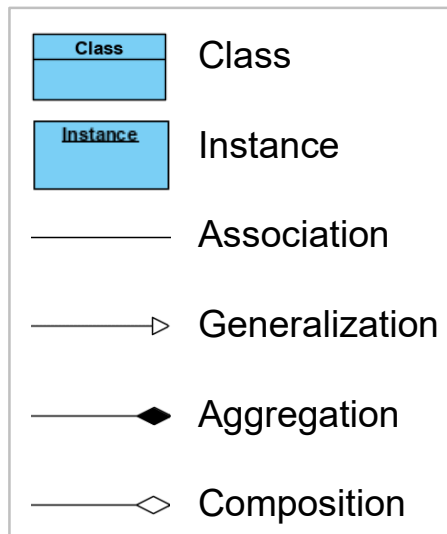
Meta-model:

- Classes and relations that can be used for modelling

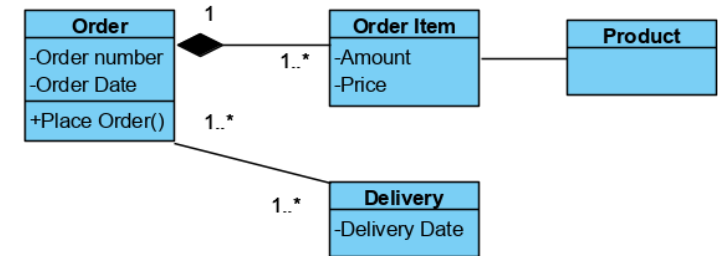


Modelling Language:

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»

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

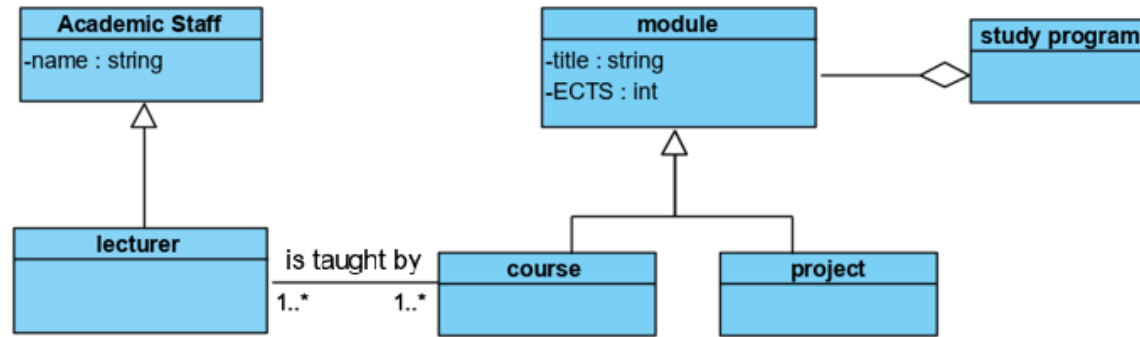
The Metamodel for a General-purpose Modelling Language

- 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

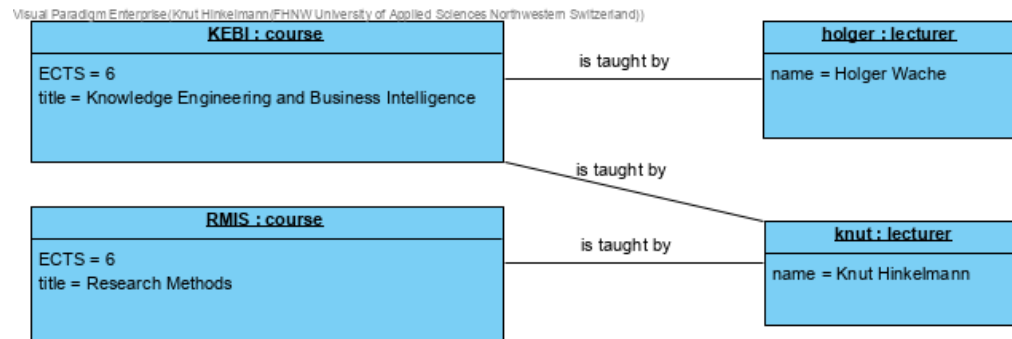
Modelling with a General-purpose Modelling Language

- 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

◆ Restricted reusability

- Different applications use different concepts

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 these 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, ...

Strengths and Weaknesses of Domain-specific Modelling Languages

■ Strengths

◆ Comprehensibility 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*

Knowledge Work Designer

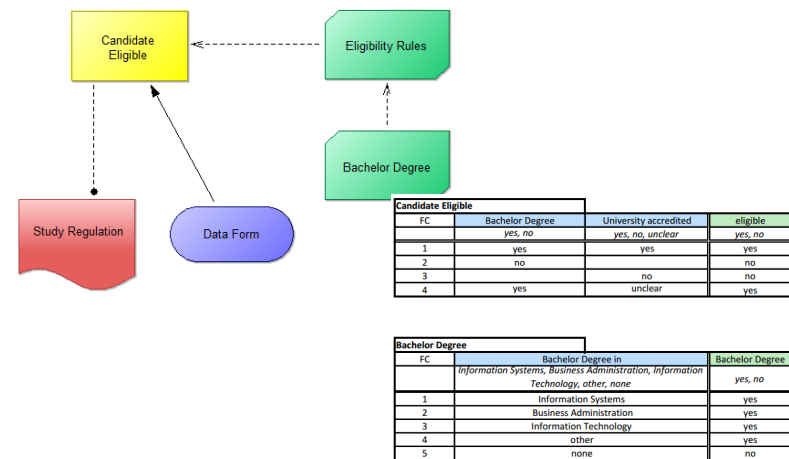
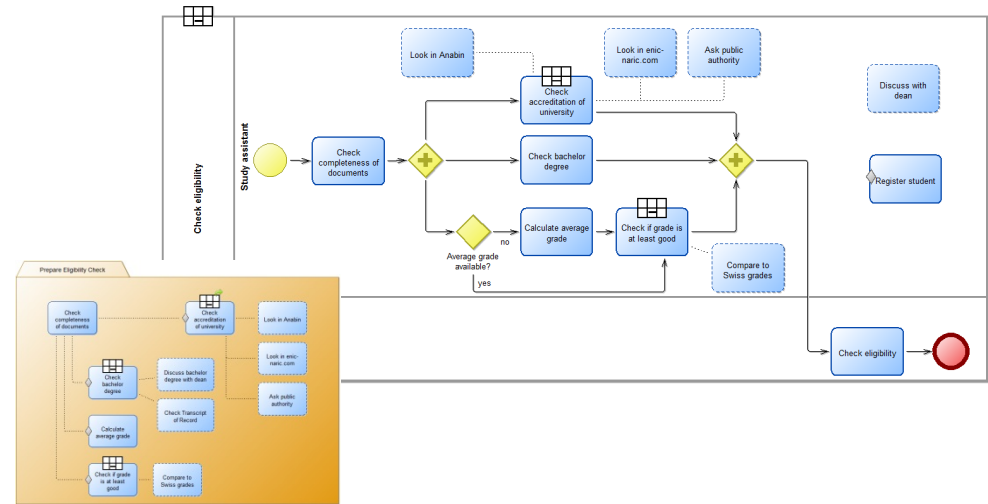
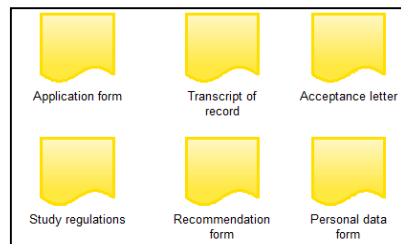
Modeling of Knowledge Work

■ Process Logic

- ◆ Structured Processes (BPM)
- ◆ Case Models (CMMN)
- ◆ Combination (BPCMN)

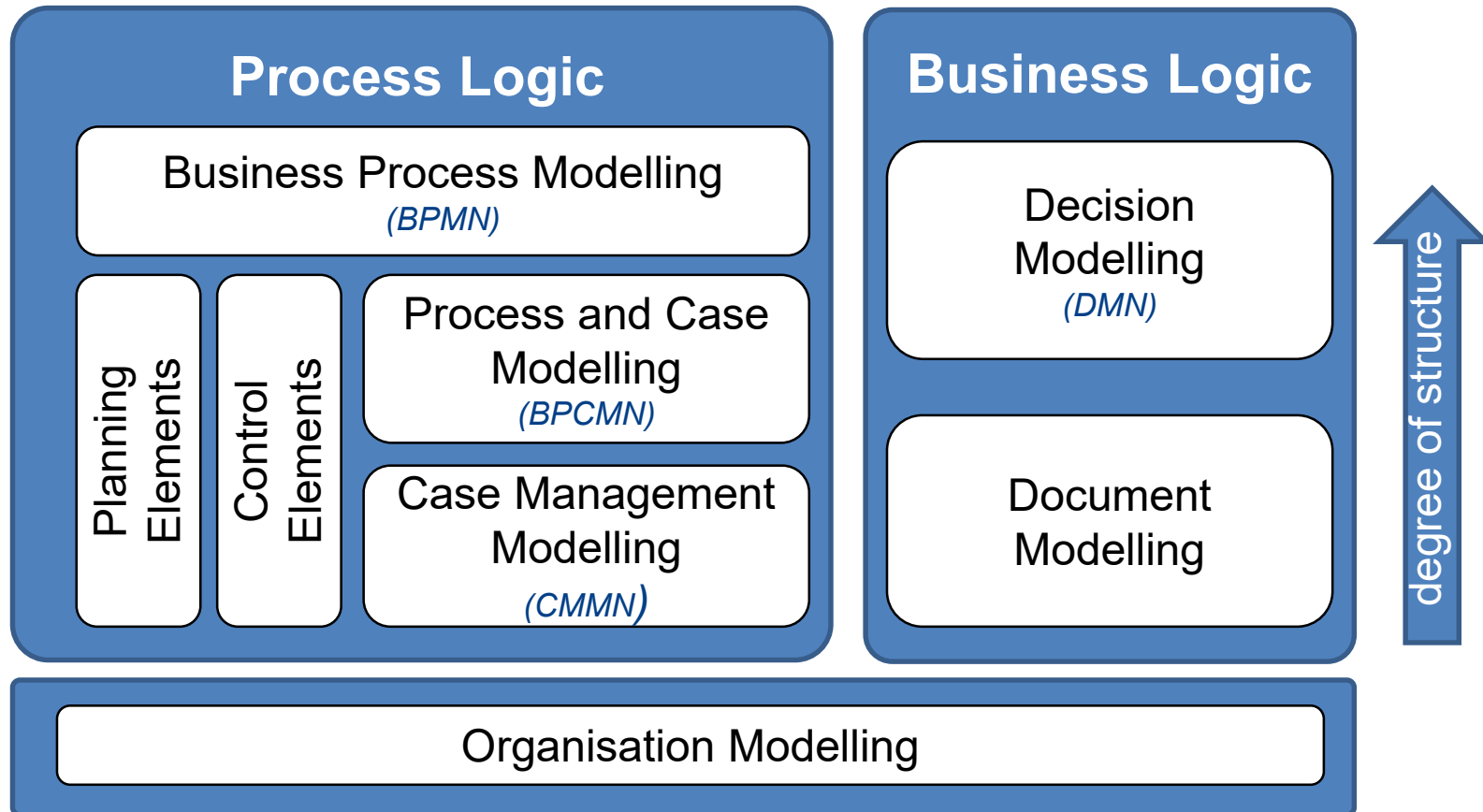
■ Decision Logic

- ◆ Decision Models (DMN)
- ◆ Document Model



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

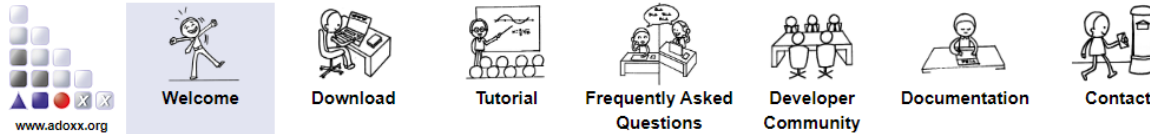
Model types of the Knowledge Work Designer



Metamodelling with ADOxx

adoxx.org – Download, Tutorials, Community

Sign In



ADOxx.org > Welcome



ADOxx Event




ADOxx Training Days
25-27.03.2020 in Vienna



REGISTRATION REQUIRED!
Contact us at tutorial@adoxx.org



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

DOWNLOAD

Do you want to realize model-value functionality?
Get access to the open-source **OLIVE** Microservice Framework - the **OMILAB** Integrated Virtual Environment.

GET ACCESS

Tweets by @ADOxxORG



ADOxx.org
@ADOxxORG

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



Mar 28, 2020

BPMN@ADOxx

UML@ADOxx

Have a look at the following realization cases of modelling approaches from the research and industrial backgrounds to get your own development started.

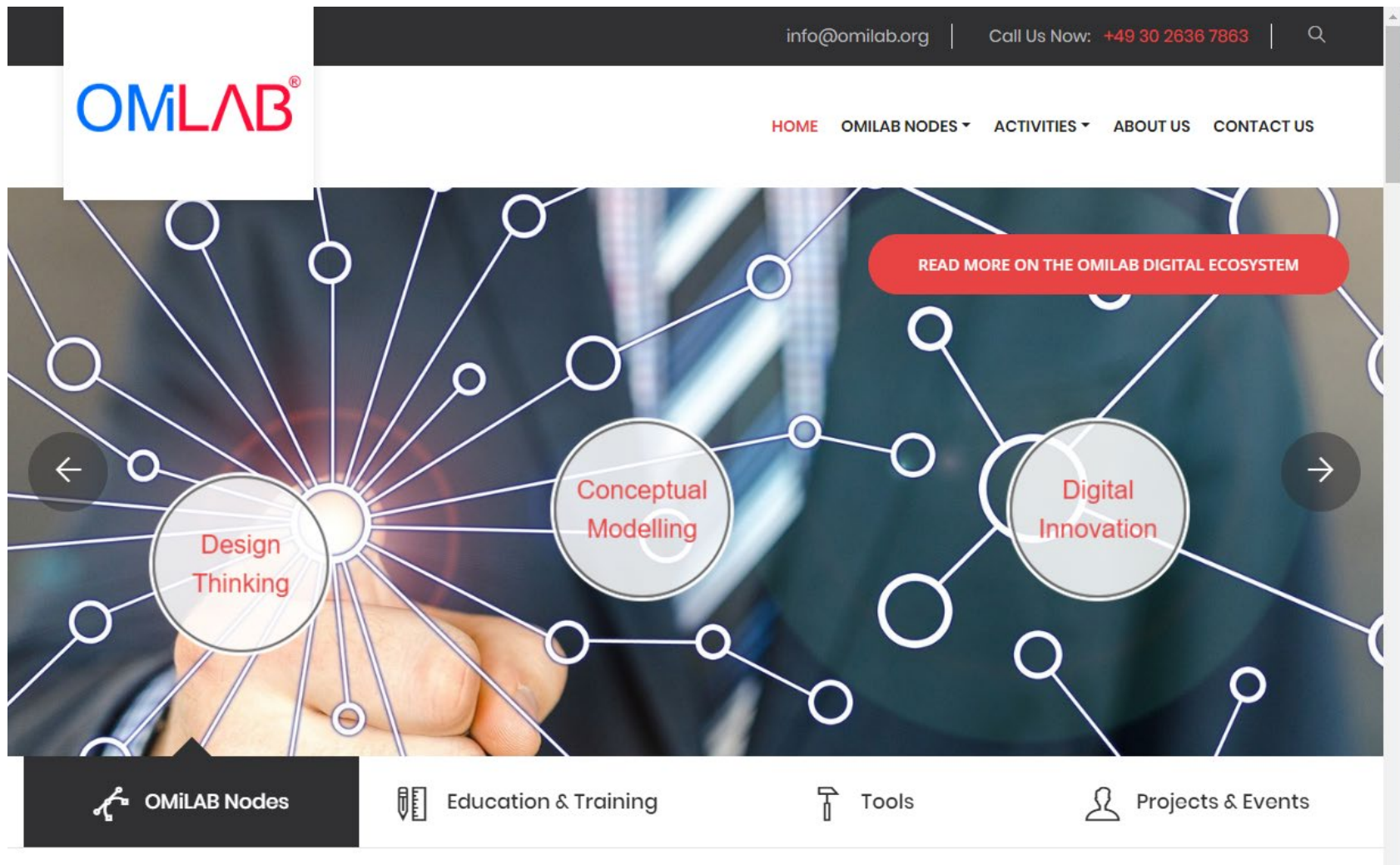
OWL@ADOxx

ER@ADOxx

Further usages of ADOxx are available at OMILab/University of Vienna:
<http://www.omilab.org>

OMiLAB – A Conceptual Modelling Community

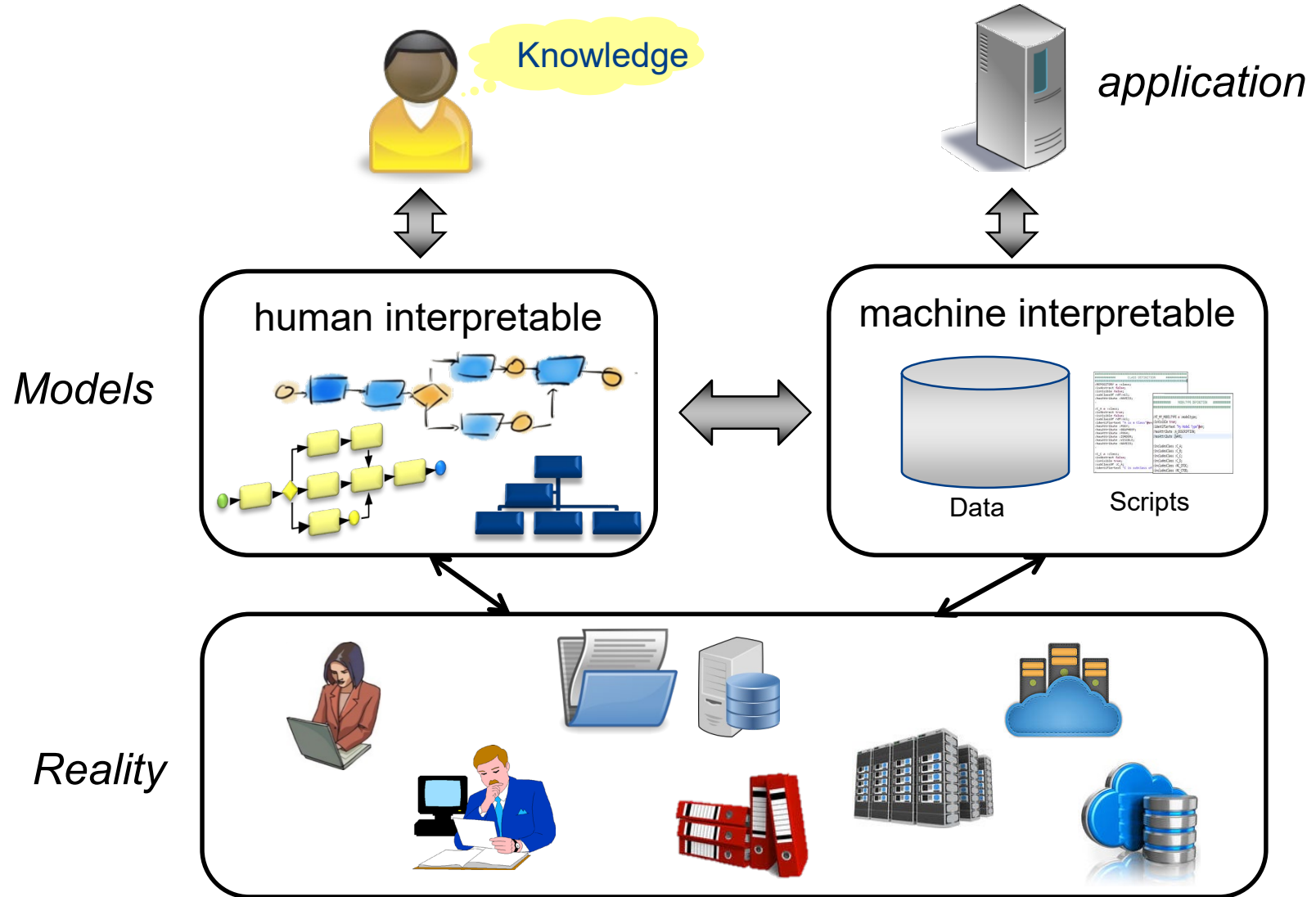
ADOxx is the basis for OMiLAB



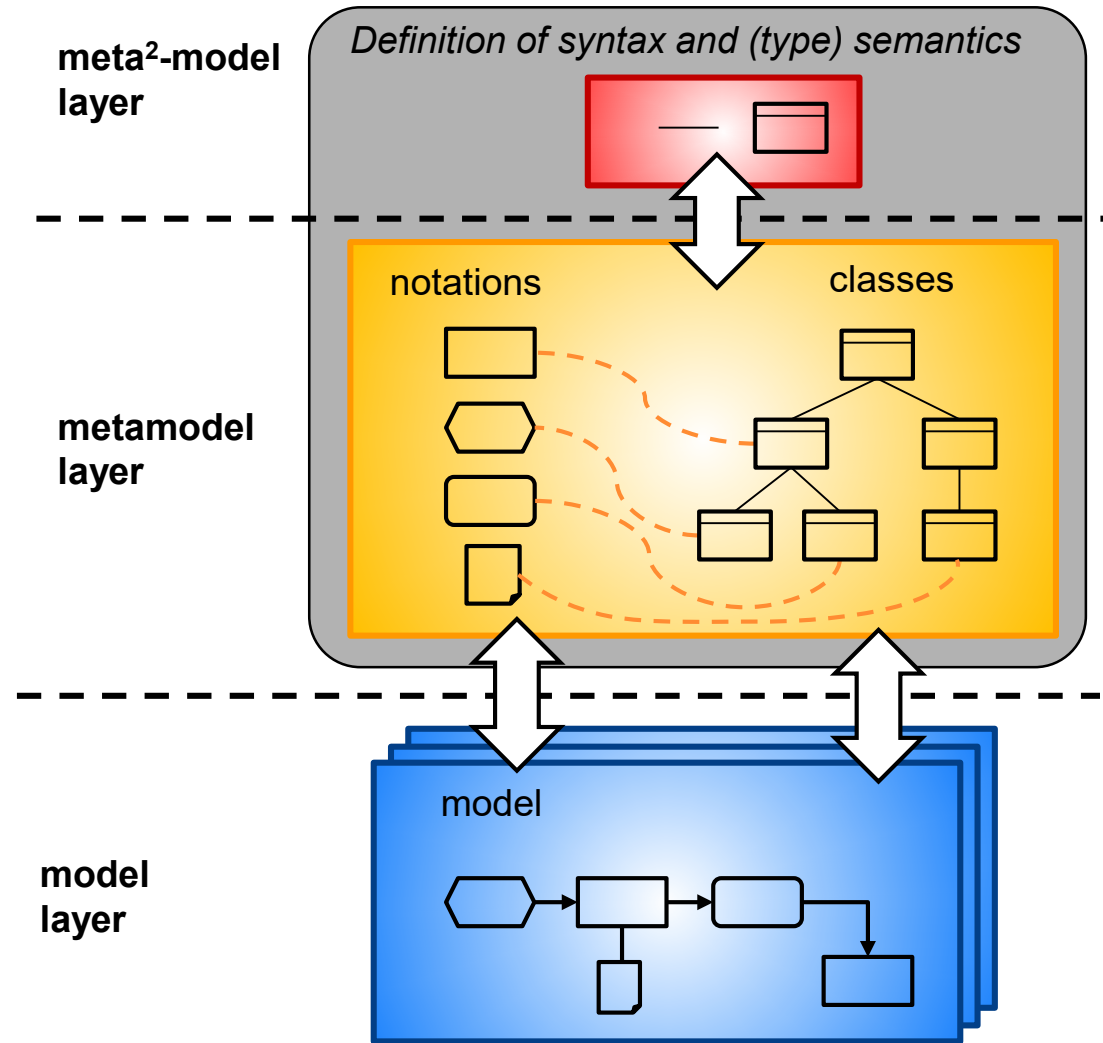
The ADOxx Environment

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

Graphical Models are Represented in a Database



Modeling Environment

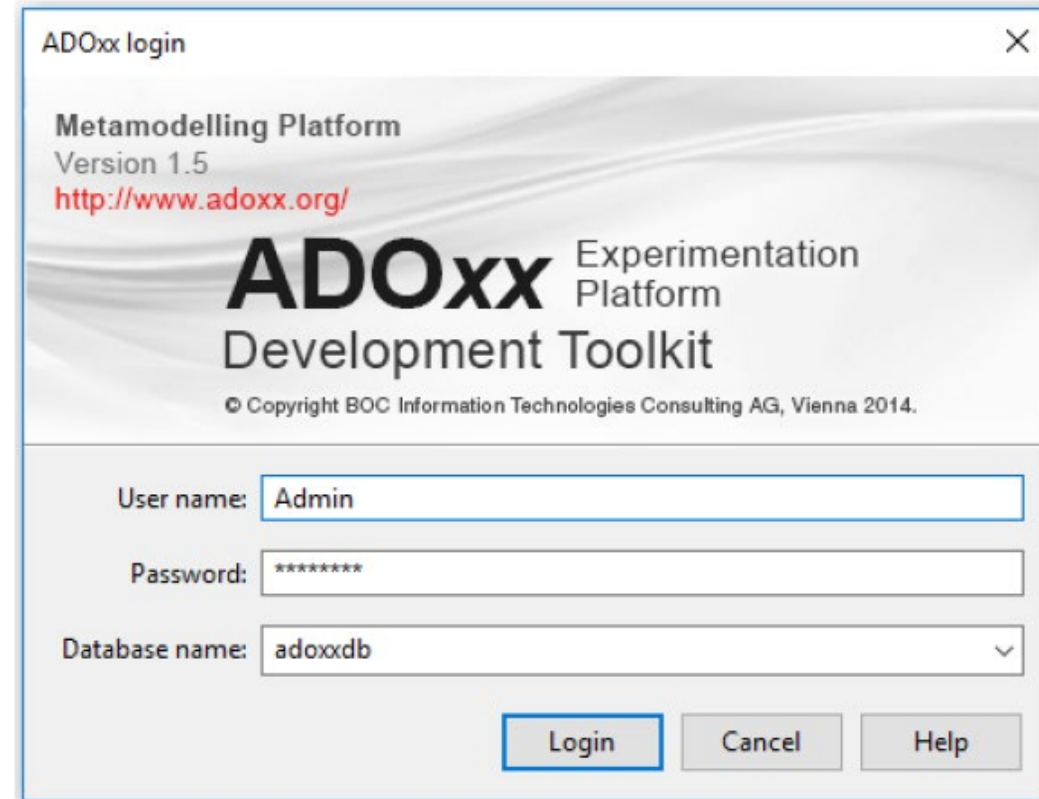


ADOxx Development Toolkit

ADOxx Modeling Toolkit

Development Toolkit

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



ADOxx login

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

ADOxx Experimentation Platform
Development Toolkit

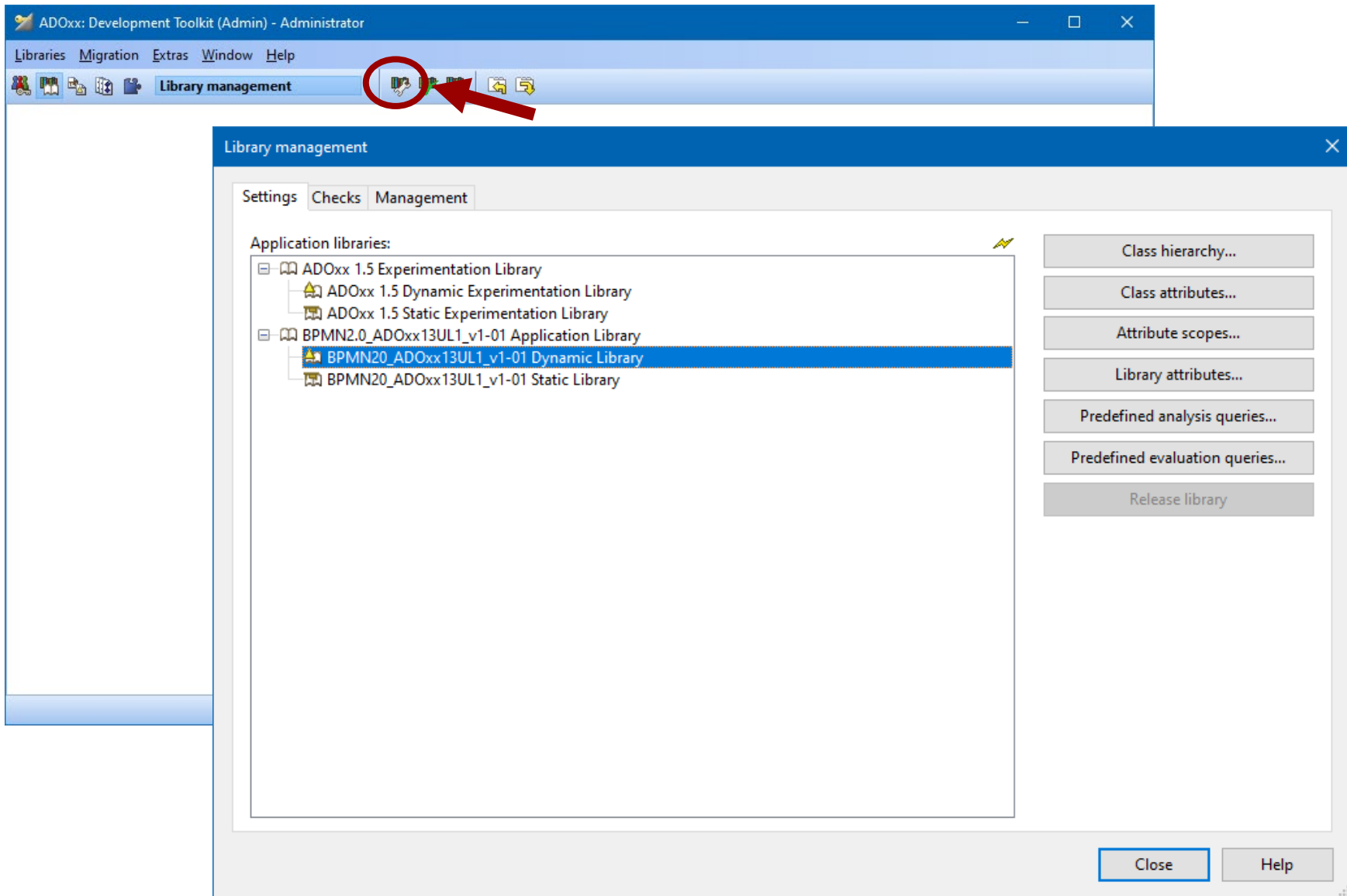
© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

User name:

Password:

Database name:

Metamodelling with ADOxx



The screenshot shows the ADOxx: Development Toolkit (Admin) - Administrator window. The 'Library management' dialog box is open, displaying a tree view of application libraries. The 'BPMN20_ADOxx13UL1_v1-01 Dynamic Library' is selected. The dialog box has tabs for 'Settings', 'Checks', and 'Management'. The 'Management' tab is active, showing a list of application libraries and a set of buttons for managing them.

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
- BPMN20_ADOxx13UL1_v1-01 Application Library
 - BPMN20_ADOxx13UL1_v1-01 Dynamic Library**
 - BPMN20_ADOxx13UL1_v1-01 Static Library

Class hierarchy...

Class attributes...

Attribute scopes...




Library attributes...

Predefined analysis queries...

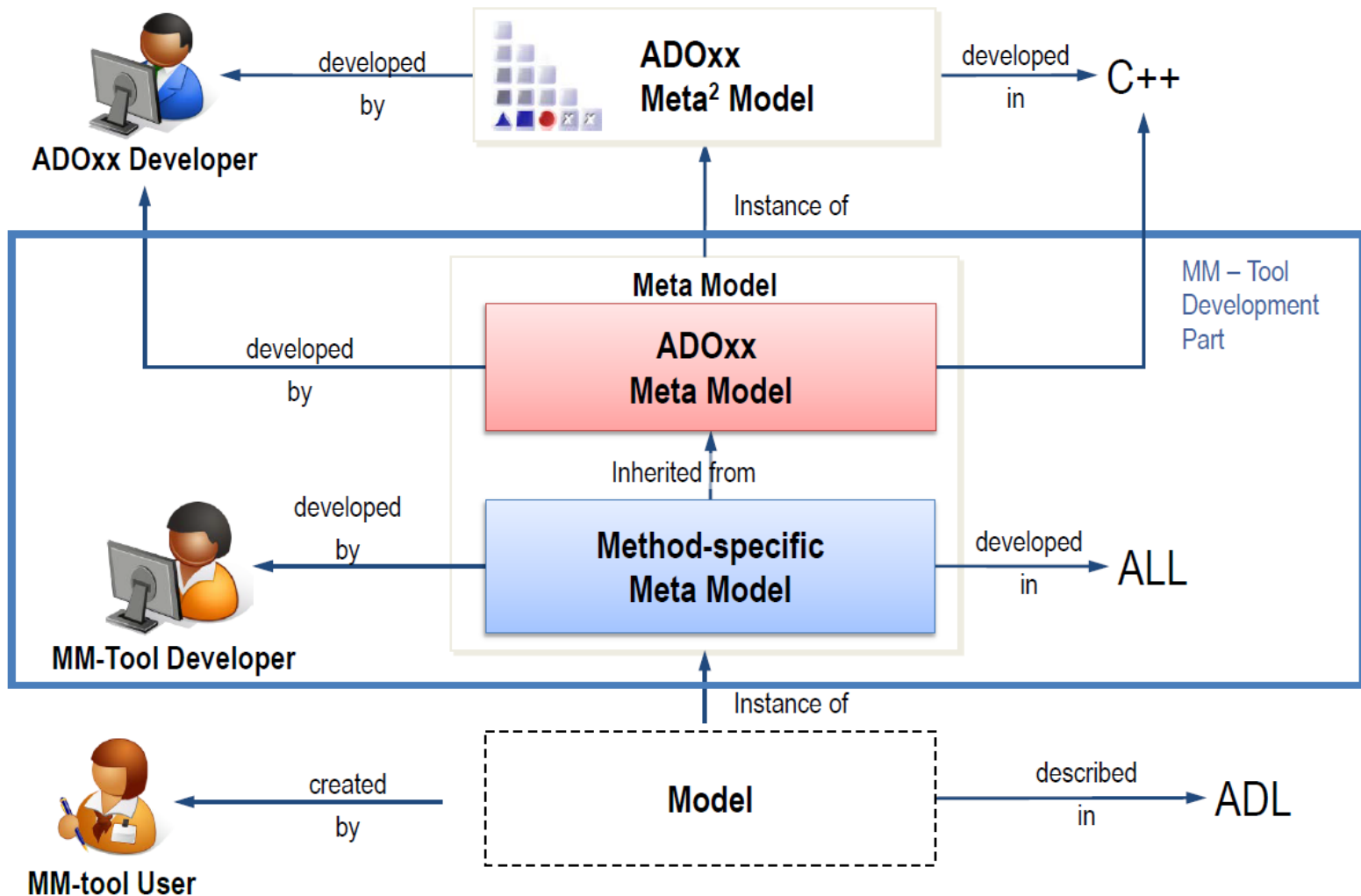
Predefined evaluation queries...

Release library

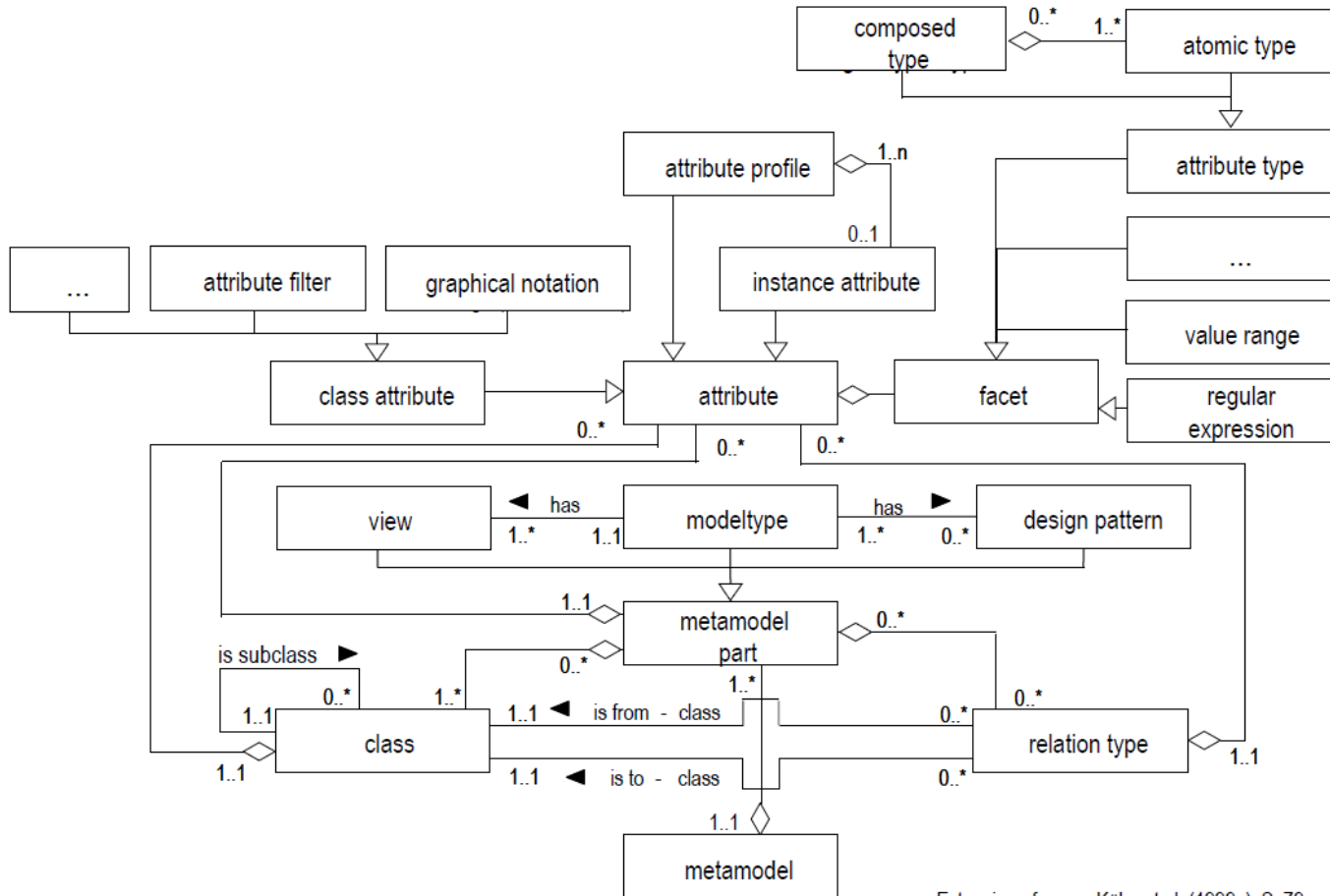
Close Help

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

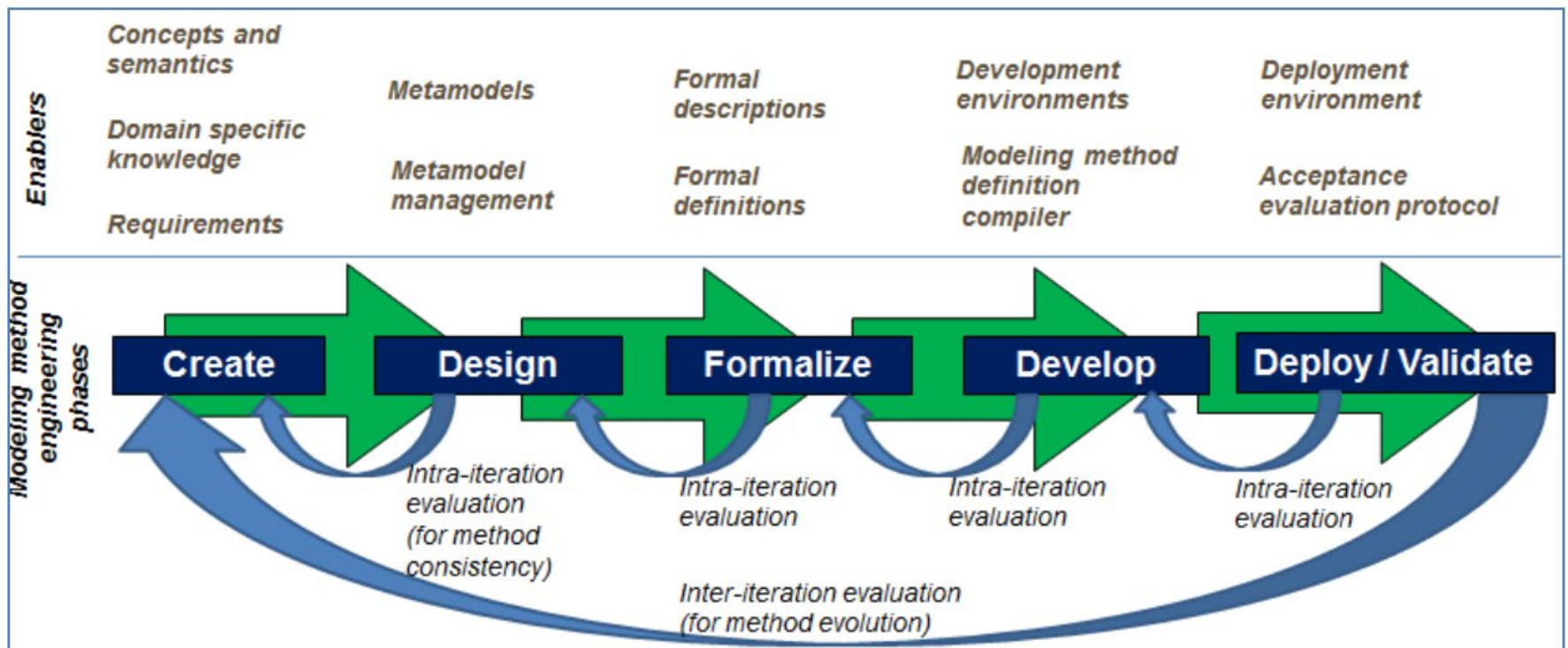
Meta Modelling Platforms Hierarchy in ADOxx



Meta² Model: Meta Model of Meta Modelling Language

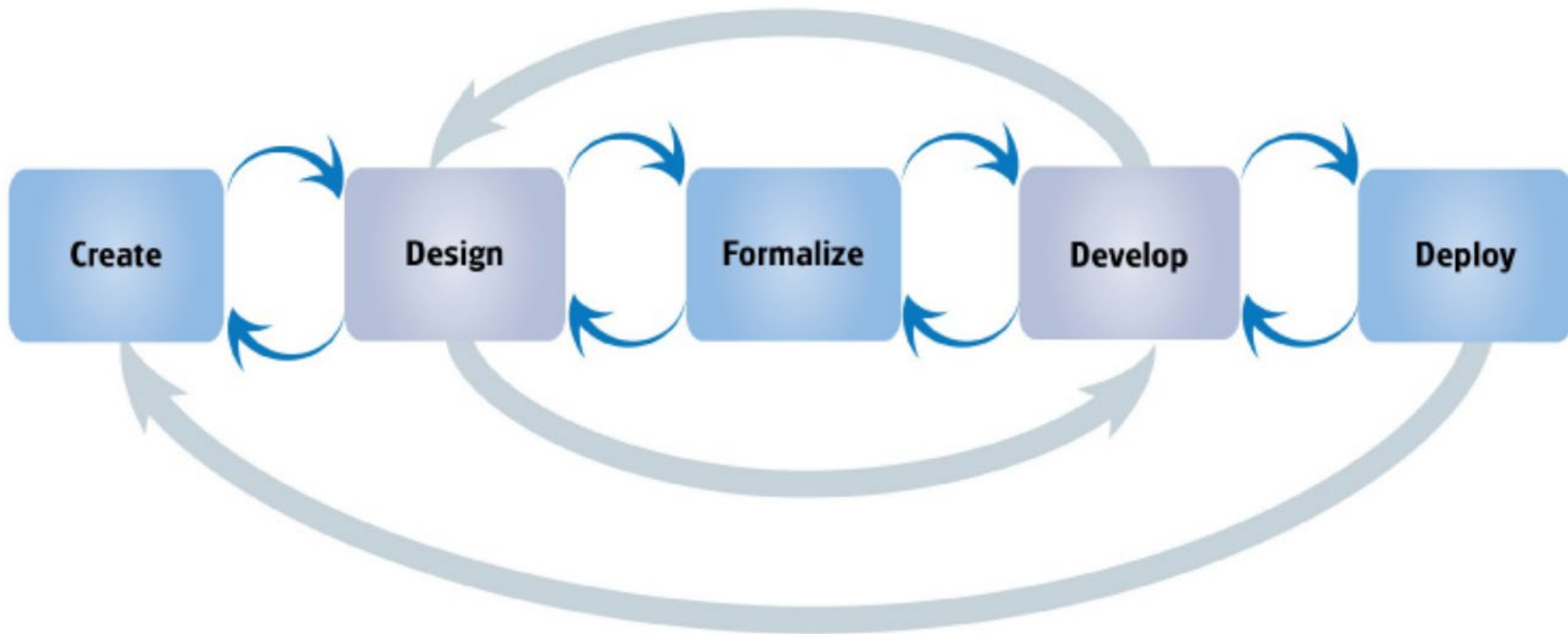


Extension of: Kühn et al. (1999a), S. 79

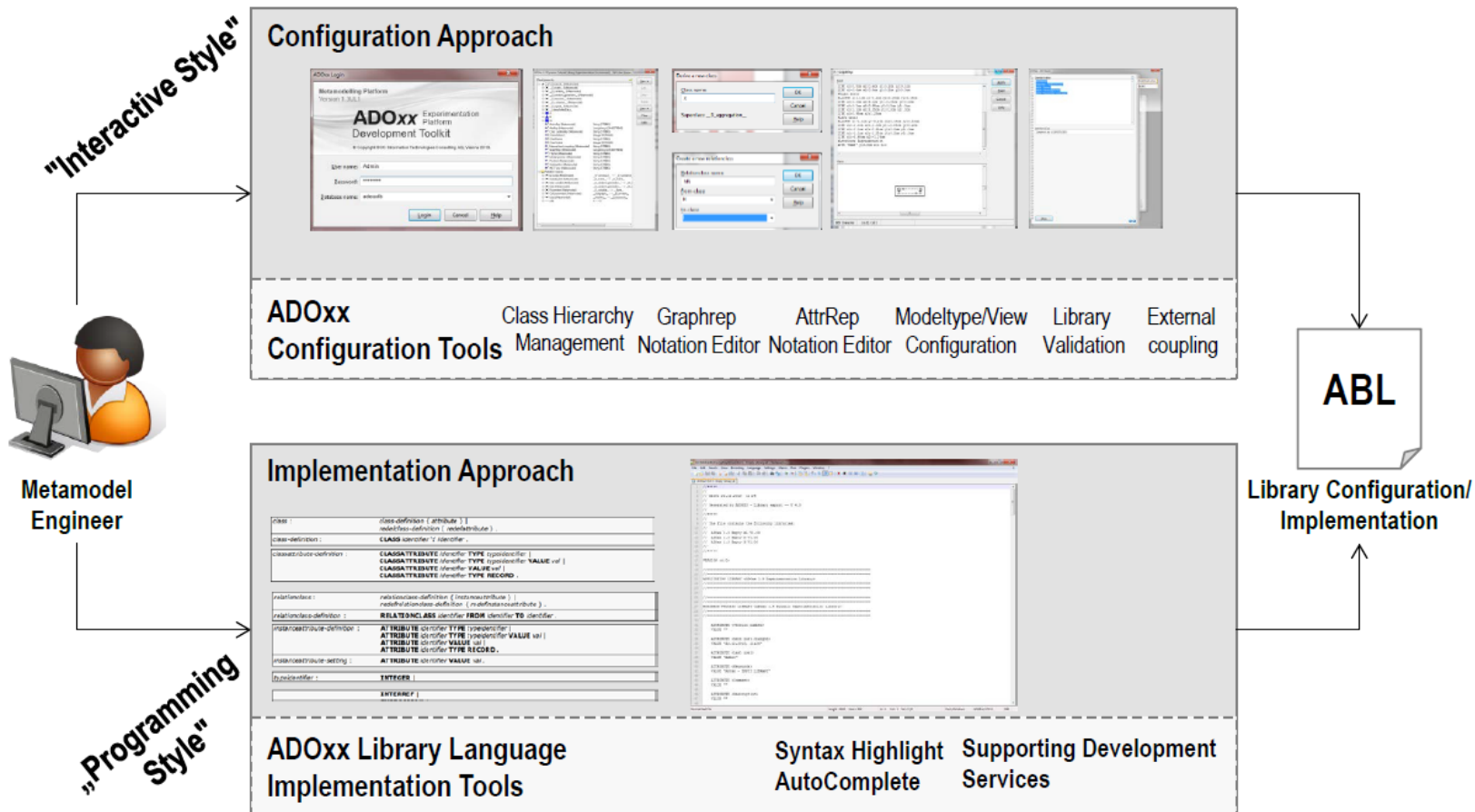


The AMME LifeCycle

Agile Meta Model Engineering

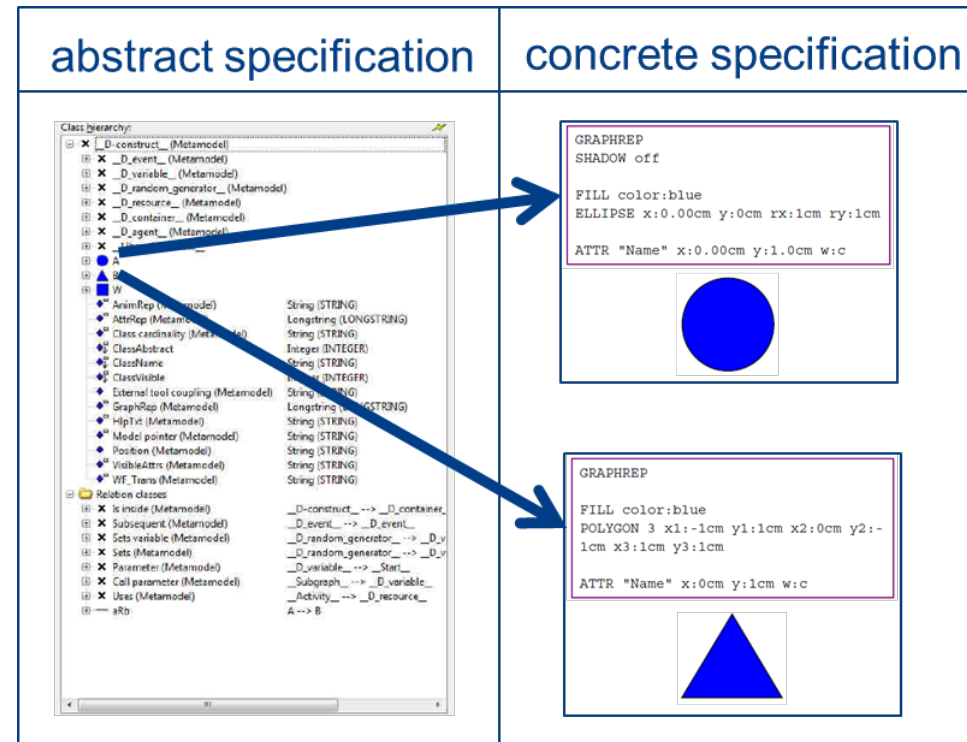


Development Approaches in ADOxx – Configuration and Implementation



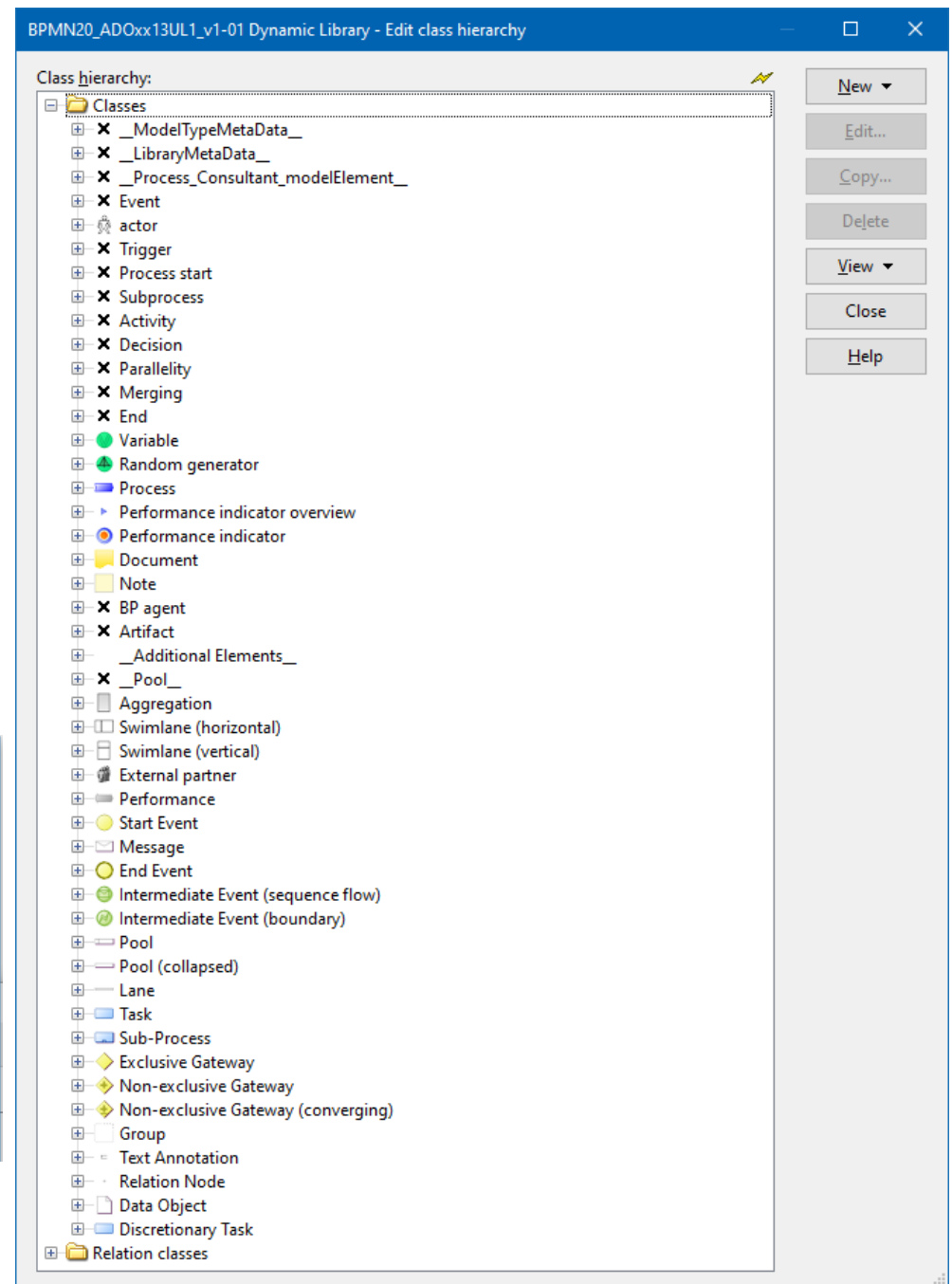
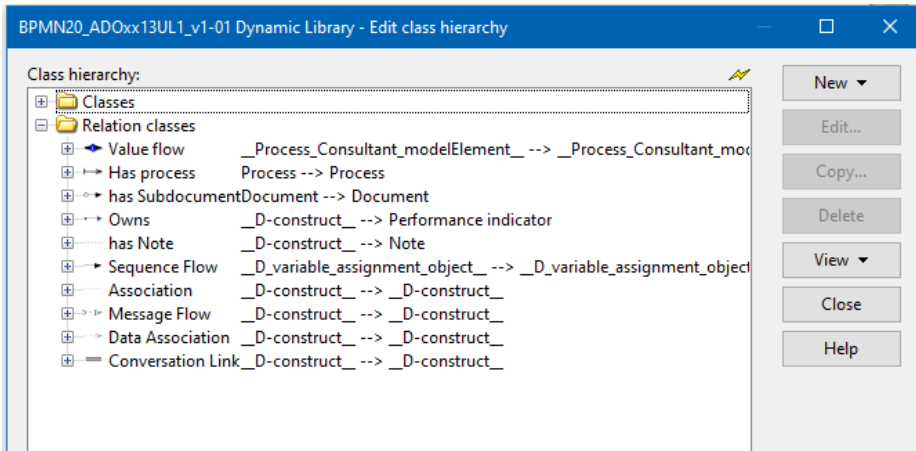
Abstract and Concrete Specification

- 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



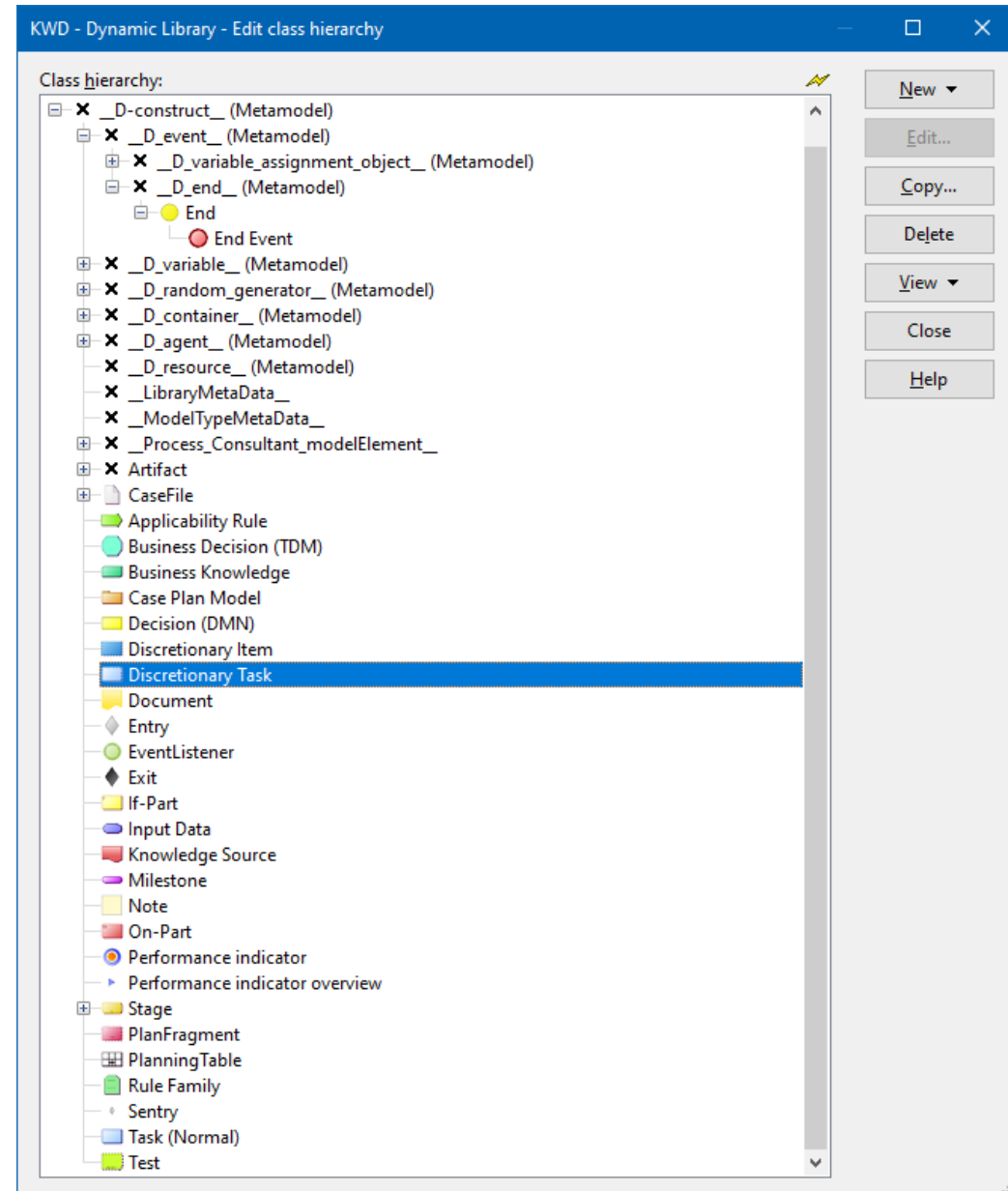
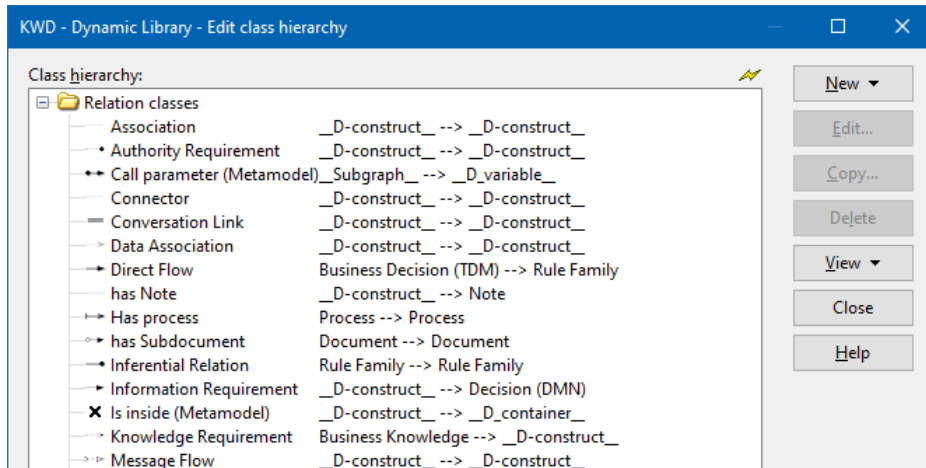
Class Hierarchies

- ADOxx distinguishes
 - ◆ Classes
 - ◆ Relation classes

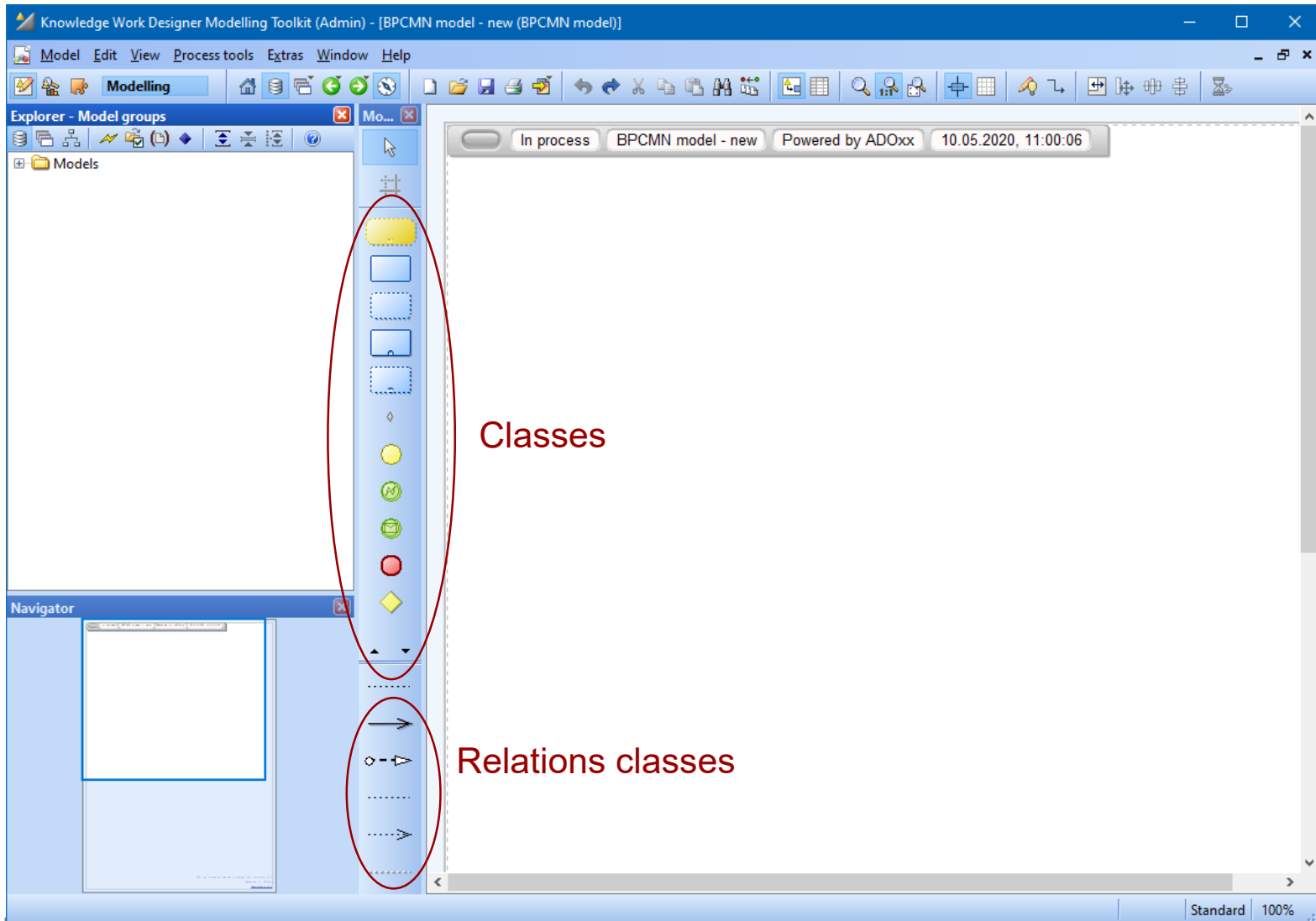


Class Hierarchies

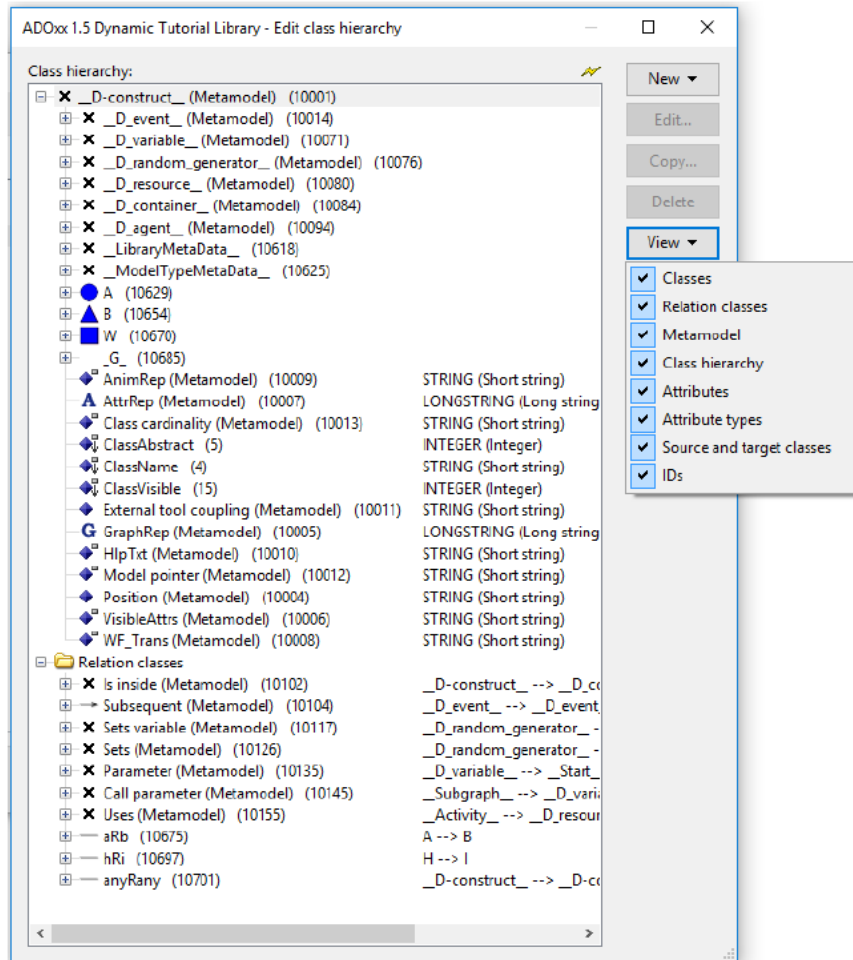
- ADOxx distinguishes
 - ◆ Classes
 - ◆ Relation classes



Appearance of Classes in the Modelling Toolkit



Views of the Class Hierarchy



Classes

All visible classes will be shown

Relation classes

All available relation classes will be shown

Metamodel

All classes will be shown

Class hierarchy

All classes will be shown with their inheritance in a hierarchy

Attributes

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

Attribute types

The type of each attribute will be shown







Source- and Target-classes

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

IDs

Shows ID numbers of classes and attributes

Icons in Class Hierarchy

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

Attributes

■ Kinds of Attributes

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

BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

Class hierarchy:

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

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

Defining a new Attribute

Class hierarchy:

[-] X	_D_construct_ (Metamodel)	
[+]	X	_D_event_ (Metamodel)
[+]	X	_D_variable_ (Metamodel)
[+]	X	_D_random_generator_ (Metamodel)
[+]	X	_D_resource_ (Metamodel)
[+]	X	_D_container_ (Metamodel)
[+]	X	_D_agent_ (Metamodel)
[+]	X	_LibraryMetaData_
[+]	X	_ModelTypeMetaData_
[+]	A	
[+]	[-] C	
[+]	[-] a1	INTEGER (Integer)
[+]	[-] a2	RECORD (Record table)
[+]	[-] a3	STRING (Short string)
[+]	[-] a4	INTERREF (Inter-model reference)
[+]	[-] 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)
[+]	[-] HipTxt (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)
[+]	B	
[+]	W	
[+]	[-] _G_	
[+]	[-] 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)
[+]	[-] HipTxt (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	

1. Select class **2. Right mouse click**

- New class...
- New attribute...**
- New class attribute...
- Copy...
- Delete

3. Select „New Attribute“

4. Define Attribute

Add new attribute

Attribute name:

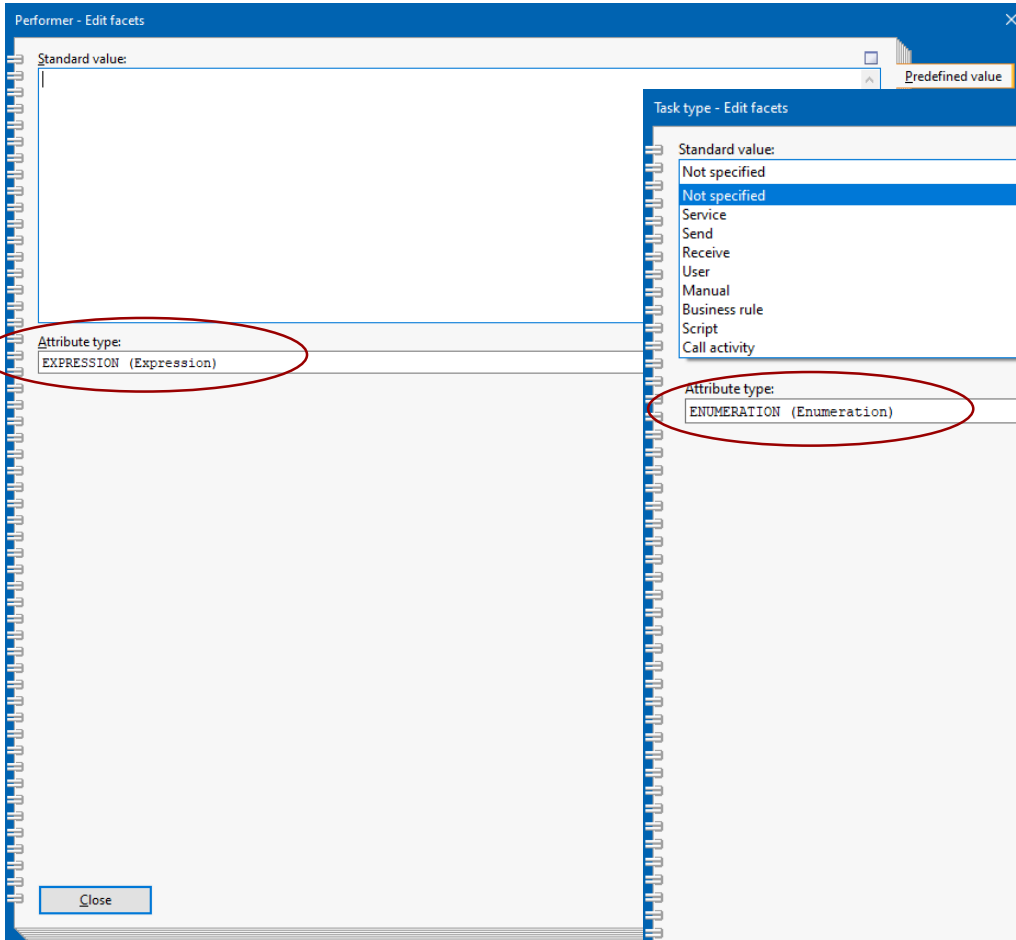
Type:

- 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)

OK Edit Cancel Help

Examples of Attributes

Performer



Performer - Edit facets

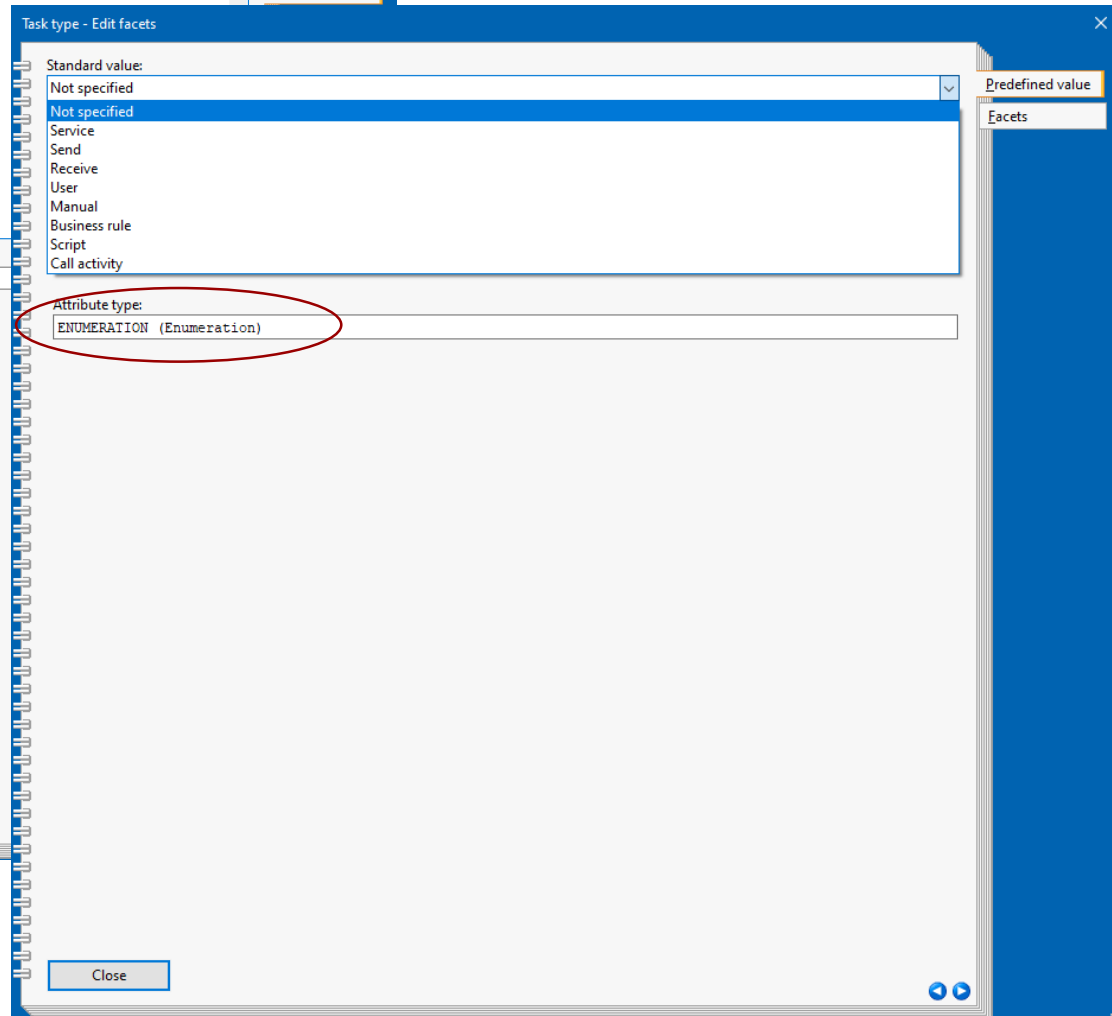
Standard value:

Predefined value

Attribute type:
EXPRESSION (Expression)

Close

Task Type



Task type - Edit facets

Standard value:

Not specified

Not specified

Service

Send

Receive

User

Manual

Business rule

Script

Call activity

Predefined value

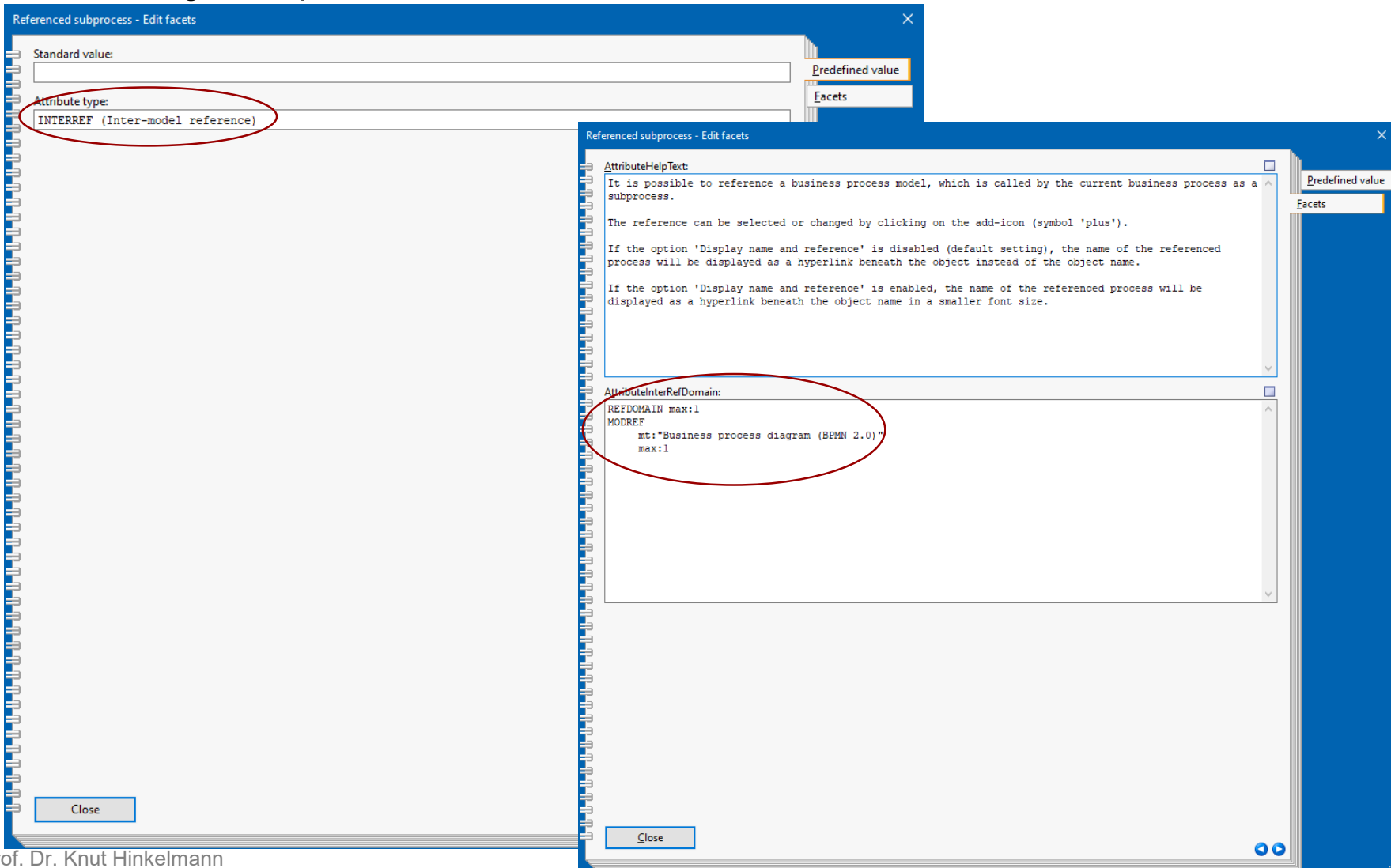
Facets

Attribute type:
ENUMERATION (Enumeration)

Close

References

Referencing a Subprocess



The image displays two screenshots of the 'Referenced subprocess - Edit facets' dialog box, illustrating the configuration for referencing a subprocess.

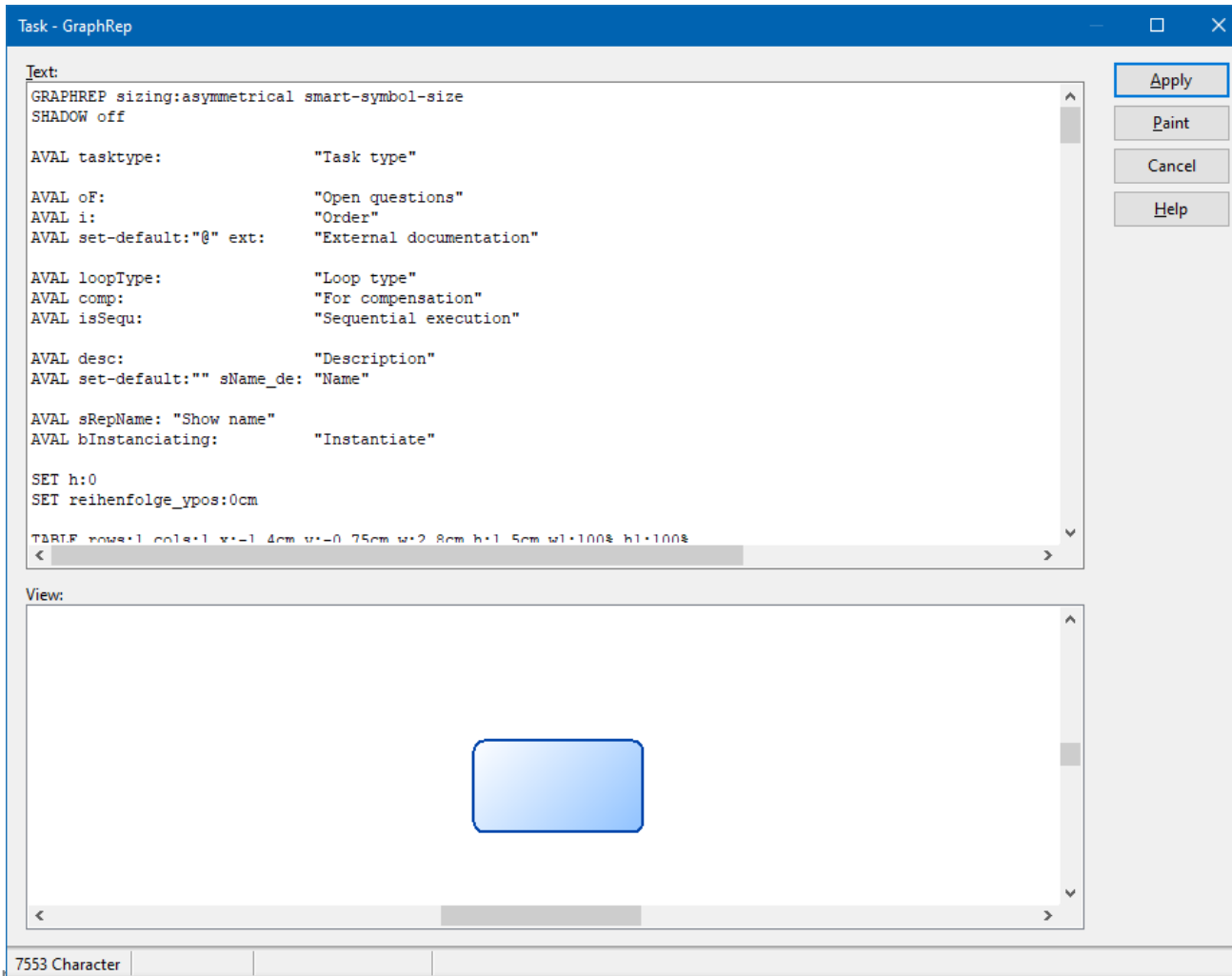
Left Screenshot: Shows the 'Attribute type' field with the value 'INTERREF (Inter-model reference)' selected. The 'Standard value' field is empty. The 'Predefined value' and 'Facets' buttons are visible on the right.

Right Screenshot: Shows the 'AttributeInterRefDomain' field with the value 'mt:Business process diagram (BPMN 2.0)' selected. The 'AttributeHelpText' field contains the following text:

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.

Special Attribute GraphRep

GraphRep: A script language for the graphical representation



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

Text:

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

AVAL tasktype:          "Task type"

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

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

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

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

SET h:0
SET reihenfolge_ypos:0cm

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

View:

The view window displays a single blue rounded rectangle centered on a white background.

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

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

GraphRep Elements

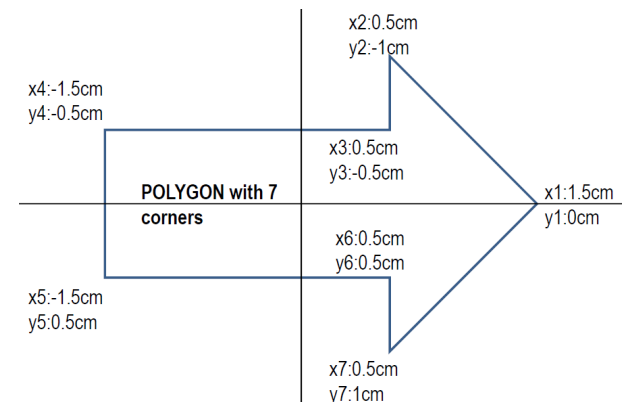
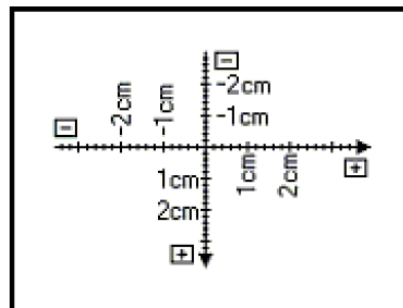
GraphRep Elements

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

```

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

- Elements are placed on x-y-axes



GraphRep Examples

```

GRAPHREP
SHADOW off

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

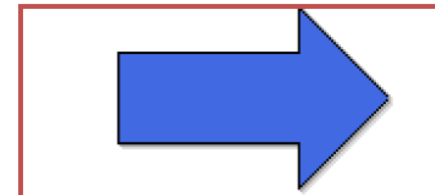
ATTR "Name" x:0.00cm y:1.0cm w:c
    
```



```

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

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

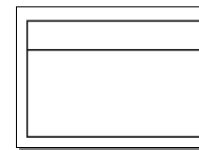


In case attribute name is available, it is shown here

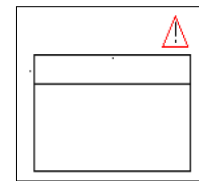
Conditional Representation

```

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



Condition fulfilled



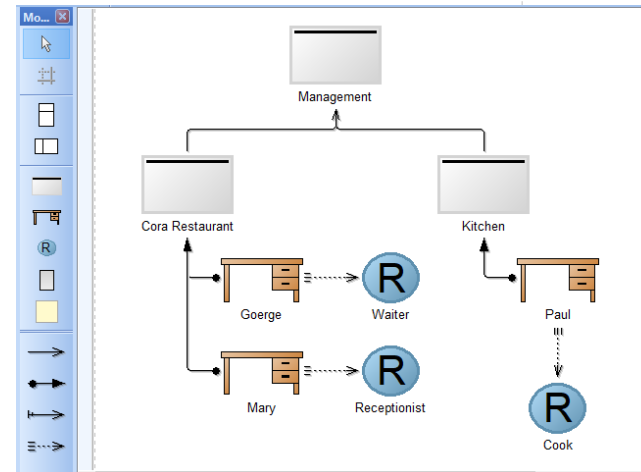
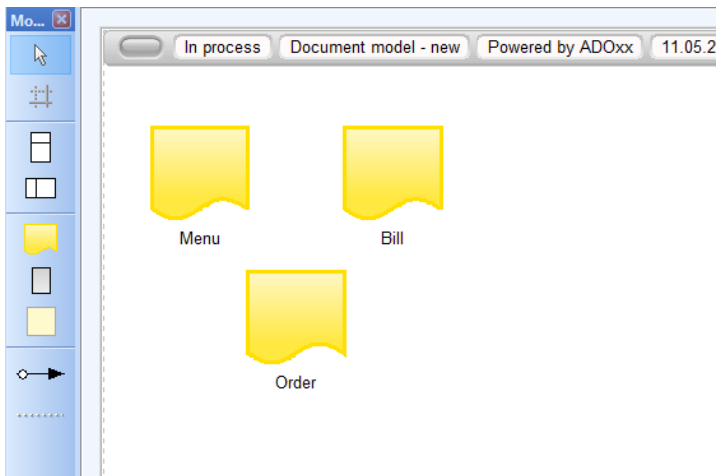
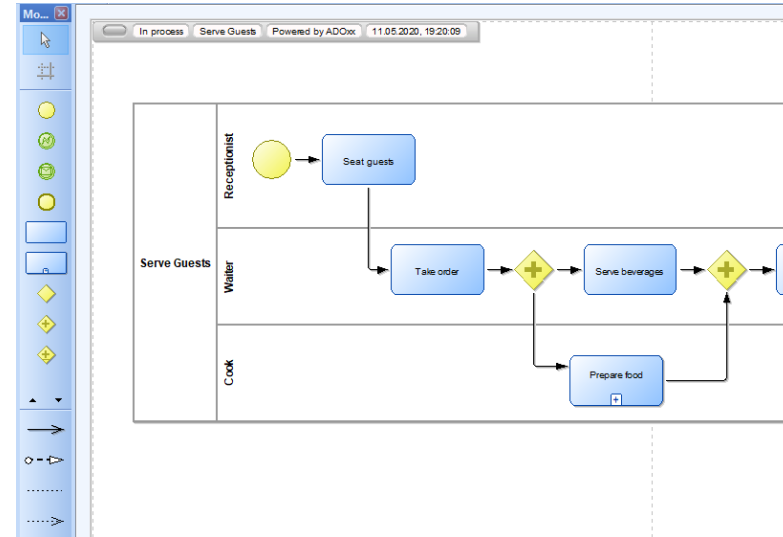
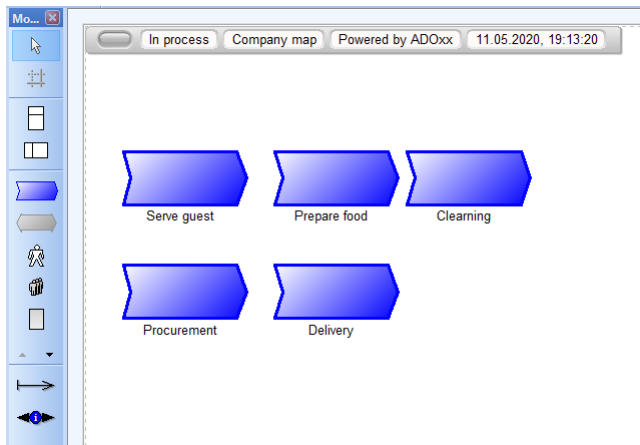
Condition not fulfilled

AttrRep

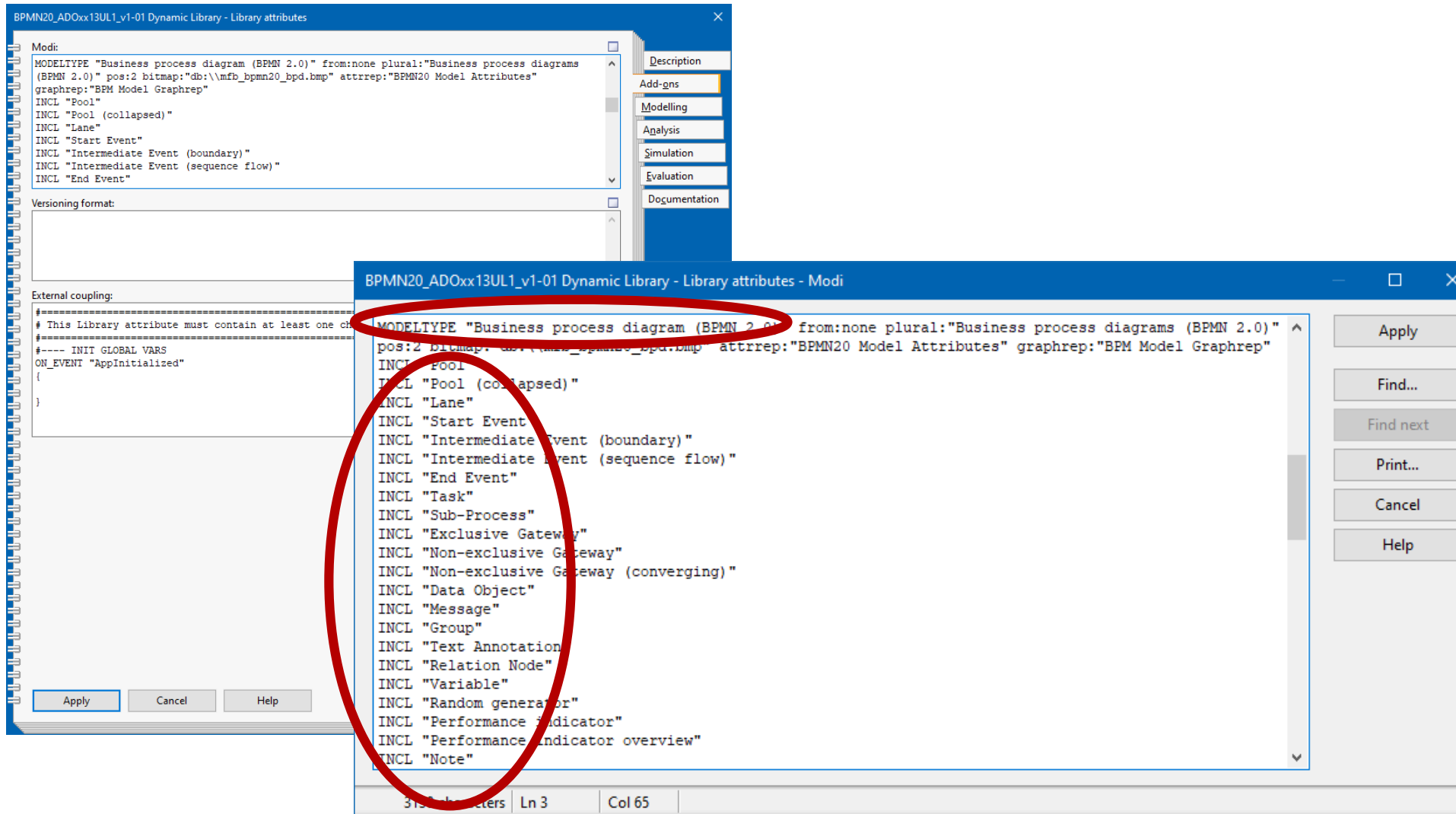
The class attribute „AttrRep“ controls the structure of the ADOxx-Notebook.



Model Types: Representation Views on the Knowledge



Classes are assigned to Model Types



The screenshot displays two overlapping dialog boxes from a software application. The background dialog, titled "BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes", shows a list of model types under the "Modi:" section. The foreground dialog, titled "BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes - Modi", shows a detailed list of model types with their corresponding class names. A red oval highlights the first line of the foreground dialog, which is: `MODELTYPE "Business process diagram (BPMN 2.0)" from:none plural:"Business process diagrams (BPMN 2.0)"`. The rest of the list in the foreground dialog includes: `pos:2 bitmap:"db:\\mf_bpmn20_bpd.bmp" attrrep:"BPMN20 Model Attributes" graphrep:"BPM Model Graphrep"`, `INCL "Pool"`, `INCL "Pool (collapsed)"`, `INCL "Lane"`, `INCL "Start Event"`, `INCL "Intermediate Event (boundary)"`, `INCL "Intermediate Event (sequence flow)"`, `INCL "End Event"`, `INCL "Task"`, `INCL "Sub-Process"`, `INCL "Exclusive Gateway"`, `INCL "Non-exclusive Gateway"`, `INCL "Non-exclusive Gateway (converging)"`, `INCL "Data Object"`, `INCL "Message"`, `INCL "Group"`, `INCL "Text Annotation"`, `INCL "Relation Node"`, `INCL "Variable"`, `INCL "Random generator"`, `INCL "Performance Indicator"`, `INCL "Performance Indicator overview"`, and `INCL "Note"`. The background dialog also shows a "Modi:" section with a list of model types and a "Versioning format:" section. The foreground dialog has a status bar at the bottom showing "315 characters Ln 3 Col 65".