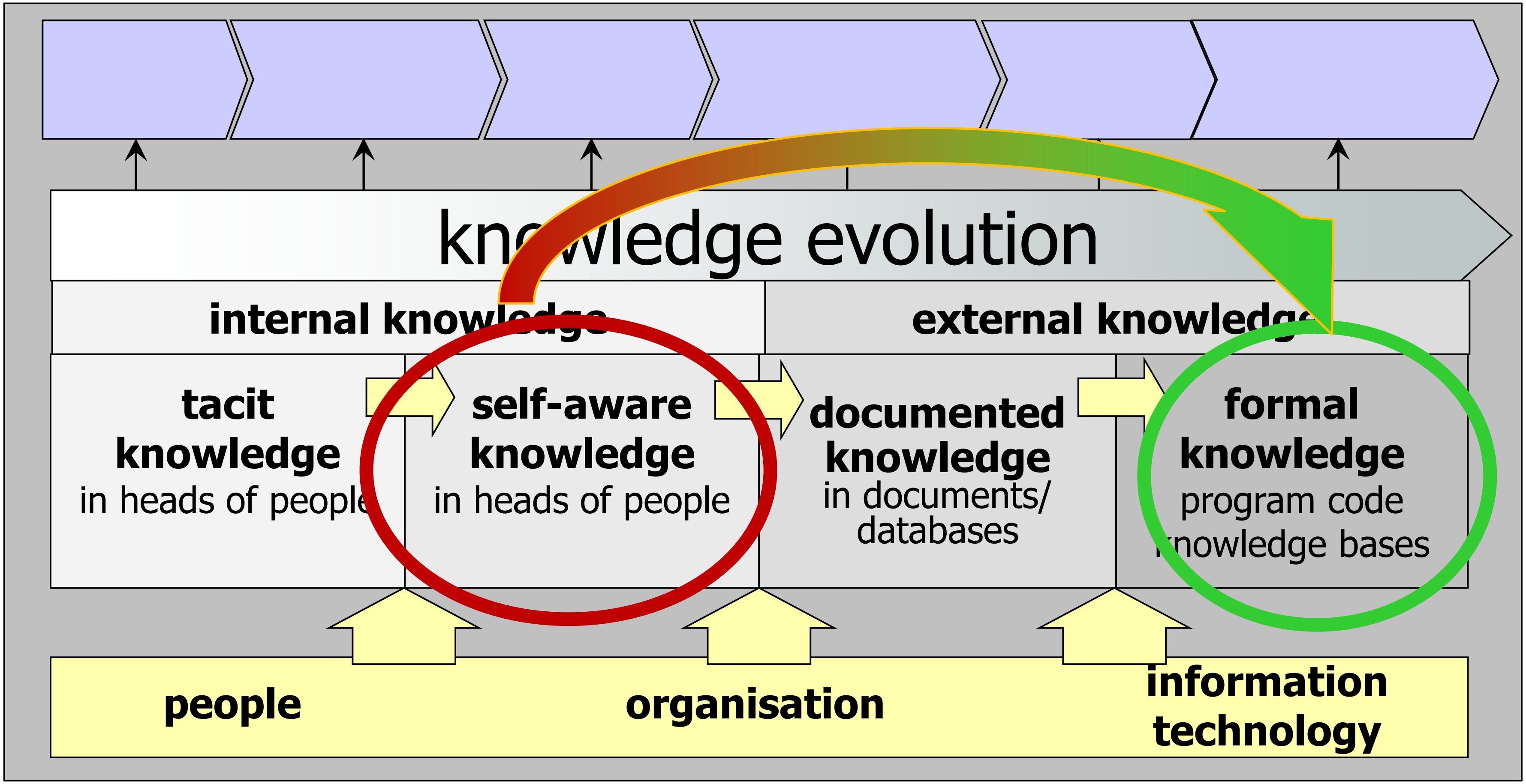
UNICAM
Università di Camerino
1336

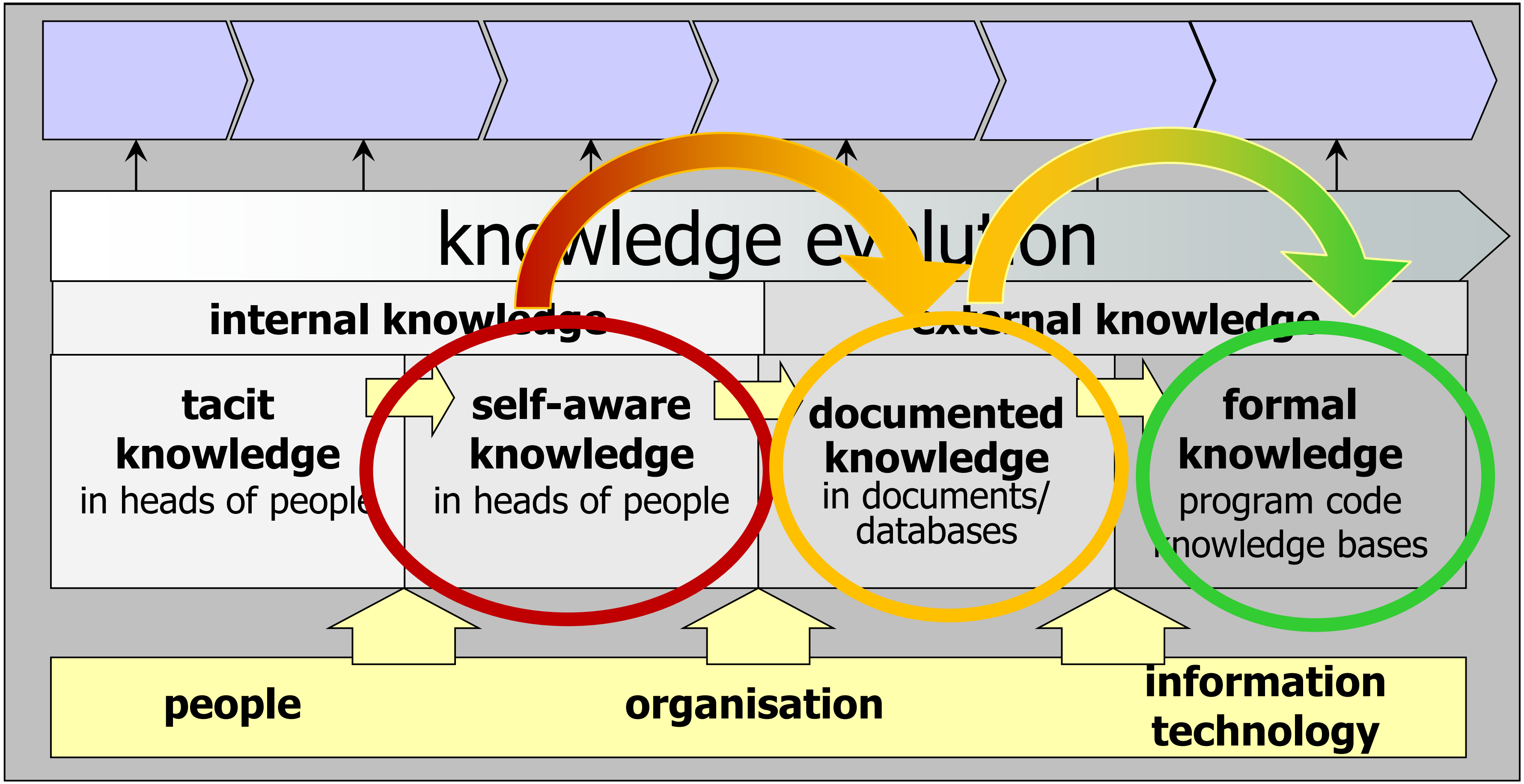
Modelling and Metamodelling

Knut Hinkelmann

Ontology Engineering

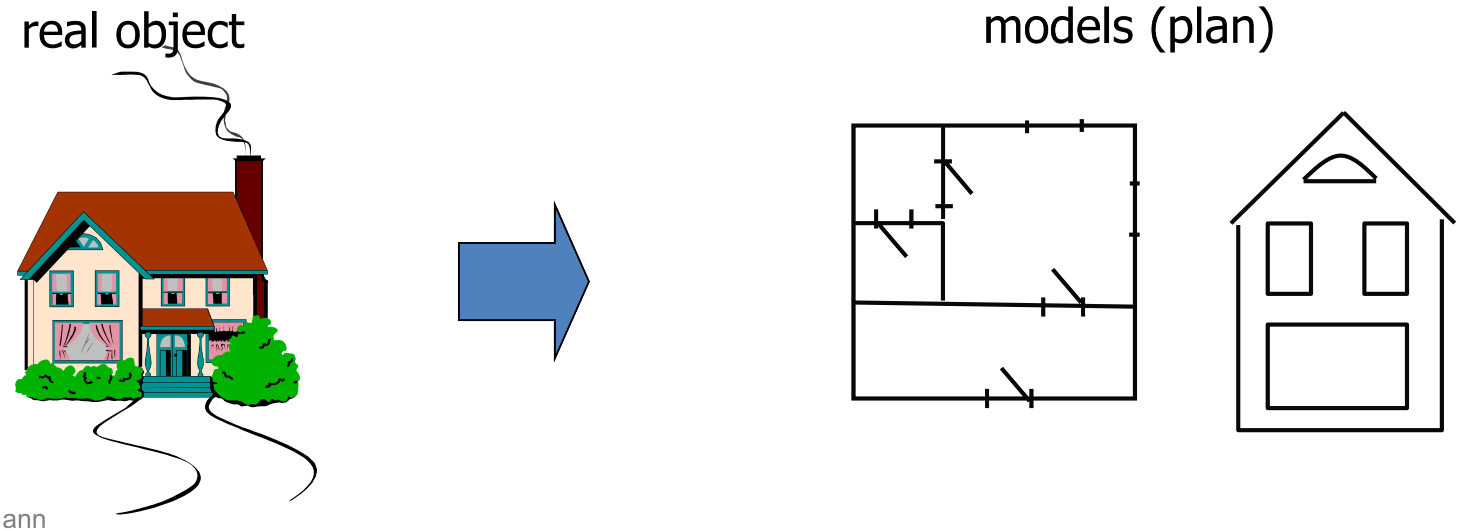


A Two-step Approach for Building a Knowledge Base



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

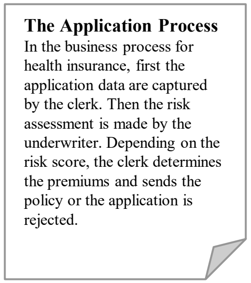


real object

models (plan)

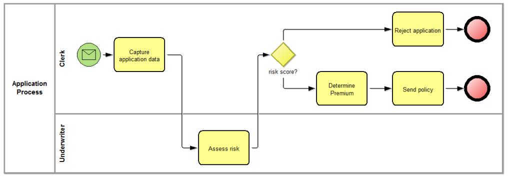
Models

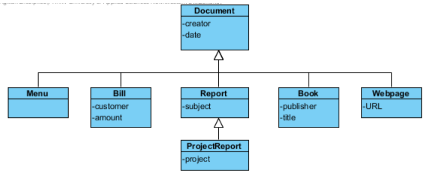
- There can be different kinds of models
  - textual model
  - graphical model
  - conceptual models
  - mathematical model
  - physical model



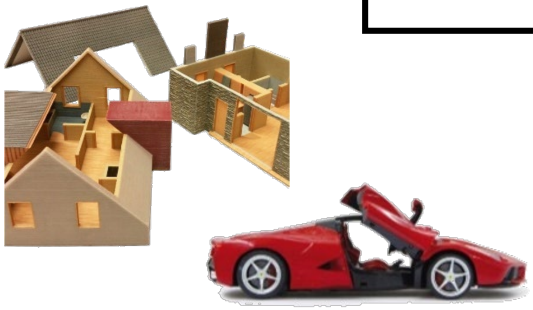
The Application Process

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.





$$\boxed{E = m c^2}$$



Visual Communication

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

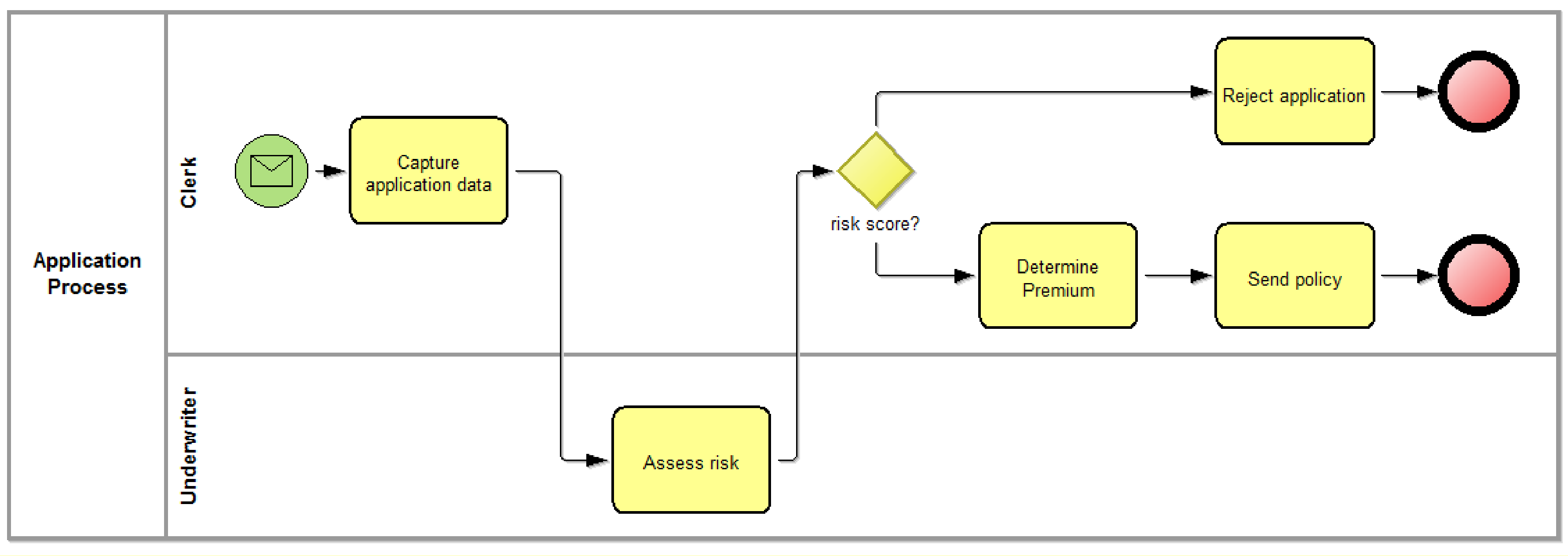
Experiment: Text vs. Model (1)

Process description:

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

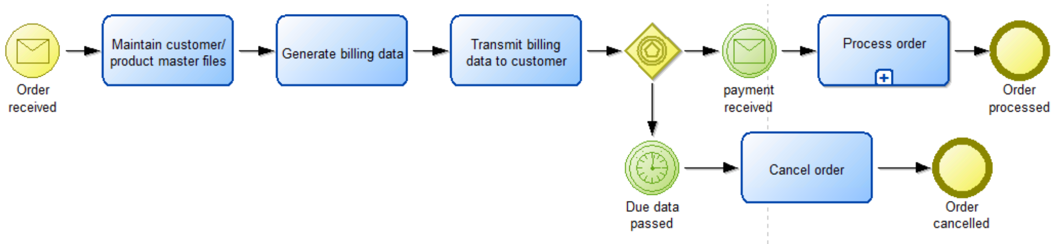
- Is «application is captured» a task or an event?
- Which tasks are executed in parallel?
- Who rejects the application?

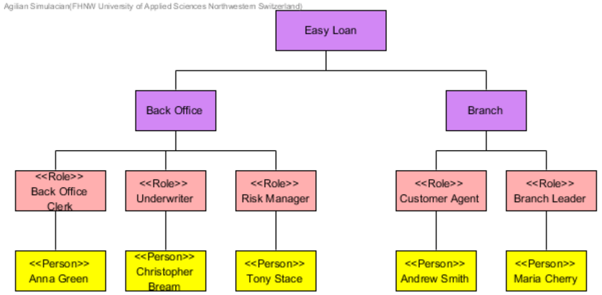
Experiment: Text vs. Model (2)

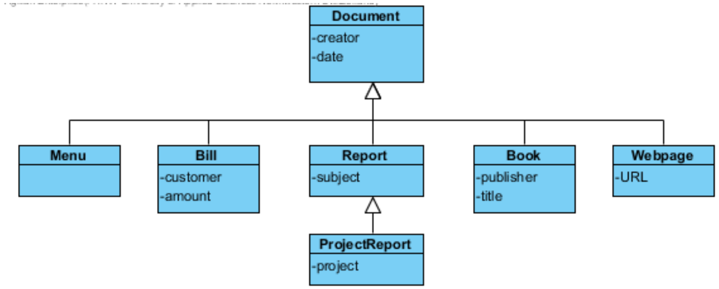


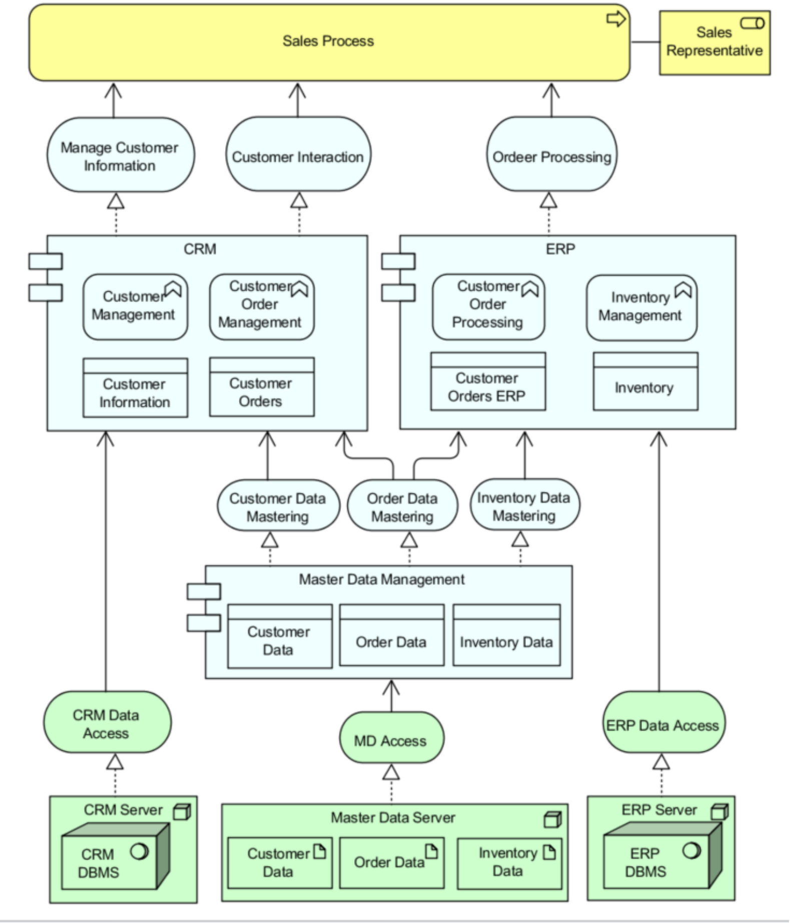
- Is «application is captured» a task or an event?
- Which tasks are executed in parallel?
- Who rejects the application?

Enterprise Models







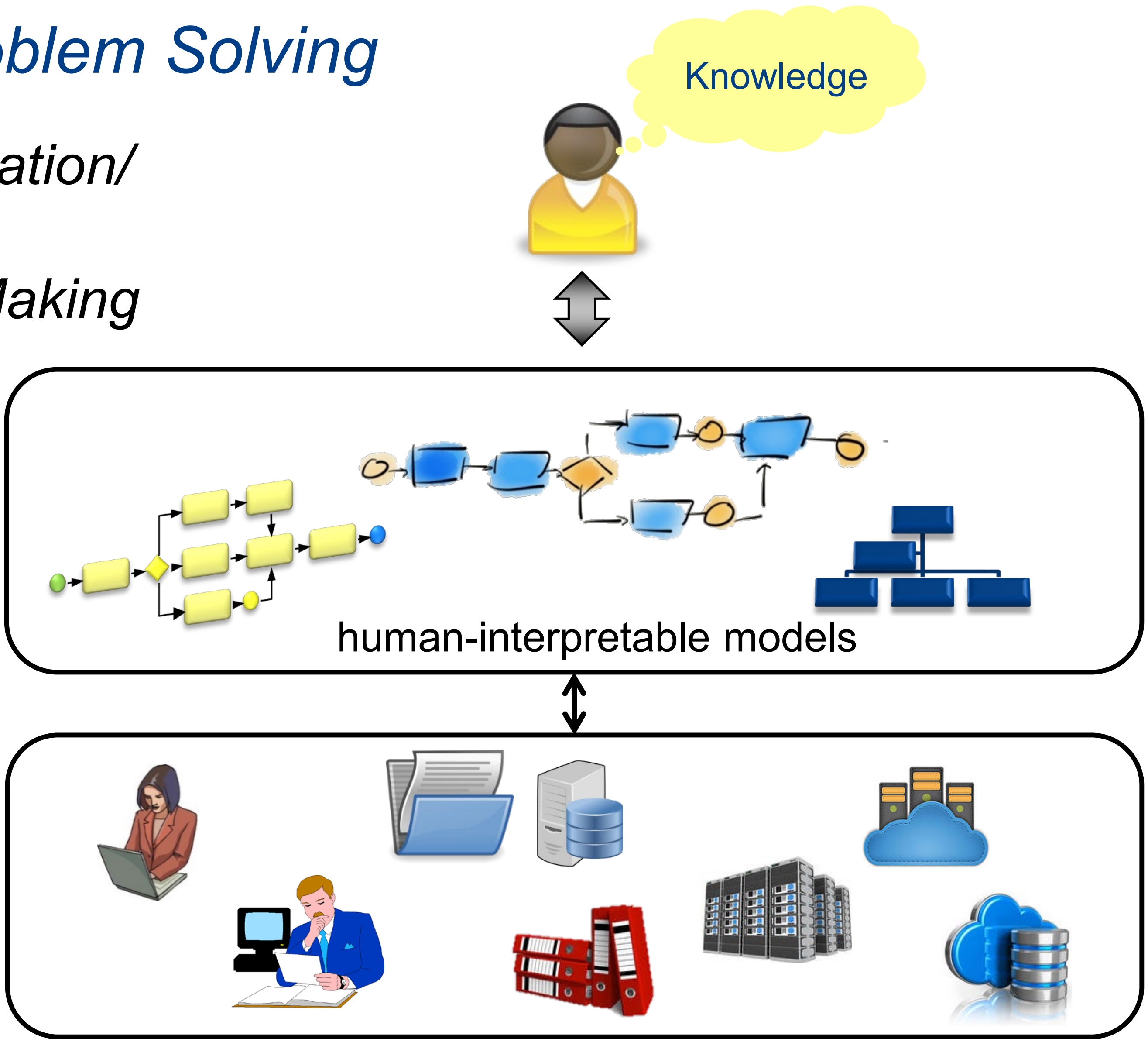


Human Problem Solving

Communication/
Analysis/
Decision Making

Models

Reality



Models and Modelling

Model

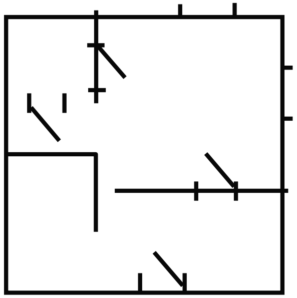
A reproduction of the part of reality which contains the essential aspects to be investigated.

Modelling

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

Result of modelling is a model.

Model in Architecture

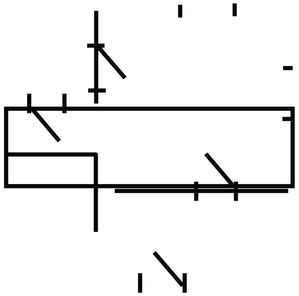
real object	**model**
	
house	architect’s drawing (plan)

Model and Modelling Language in Architecture

real object

house

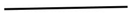
model



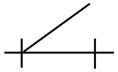
architect's drawing
(plan)

modelling language
(concrete syntax)

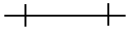
object types:



wall

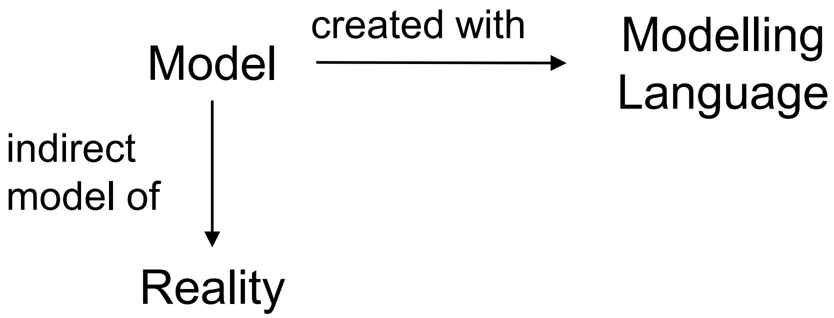


door



window

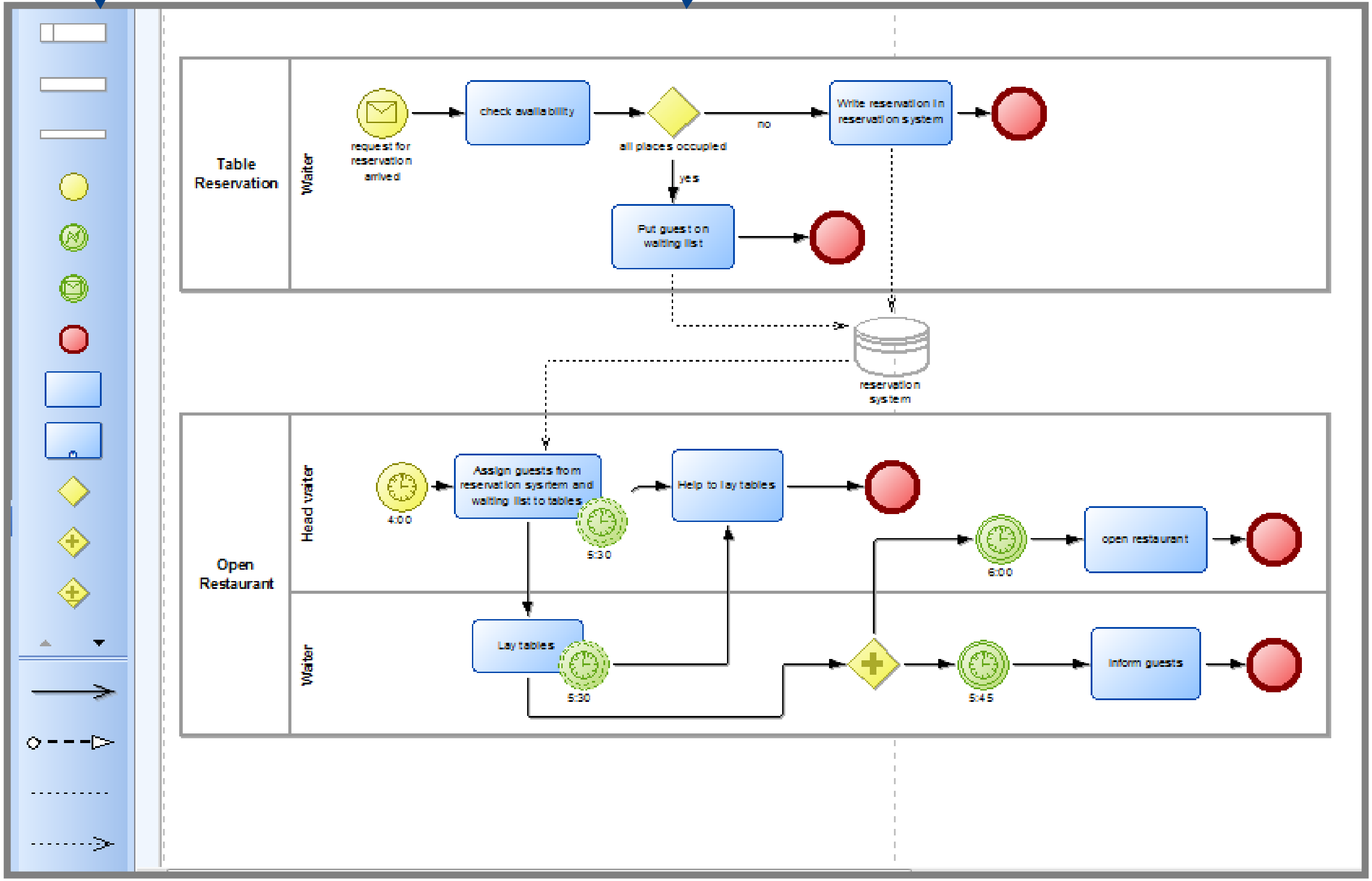
Modelling Language



- 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

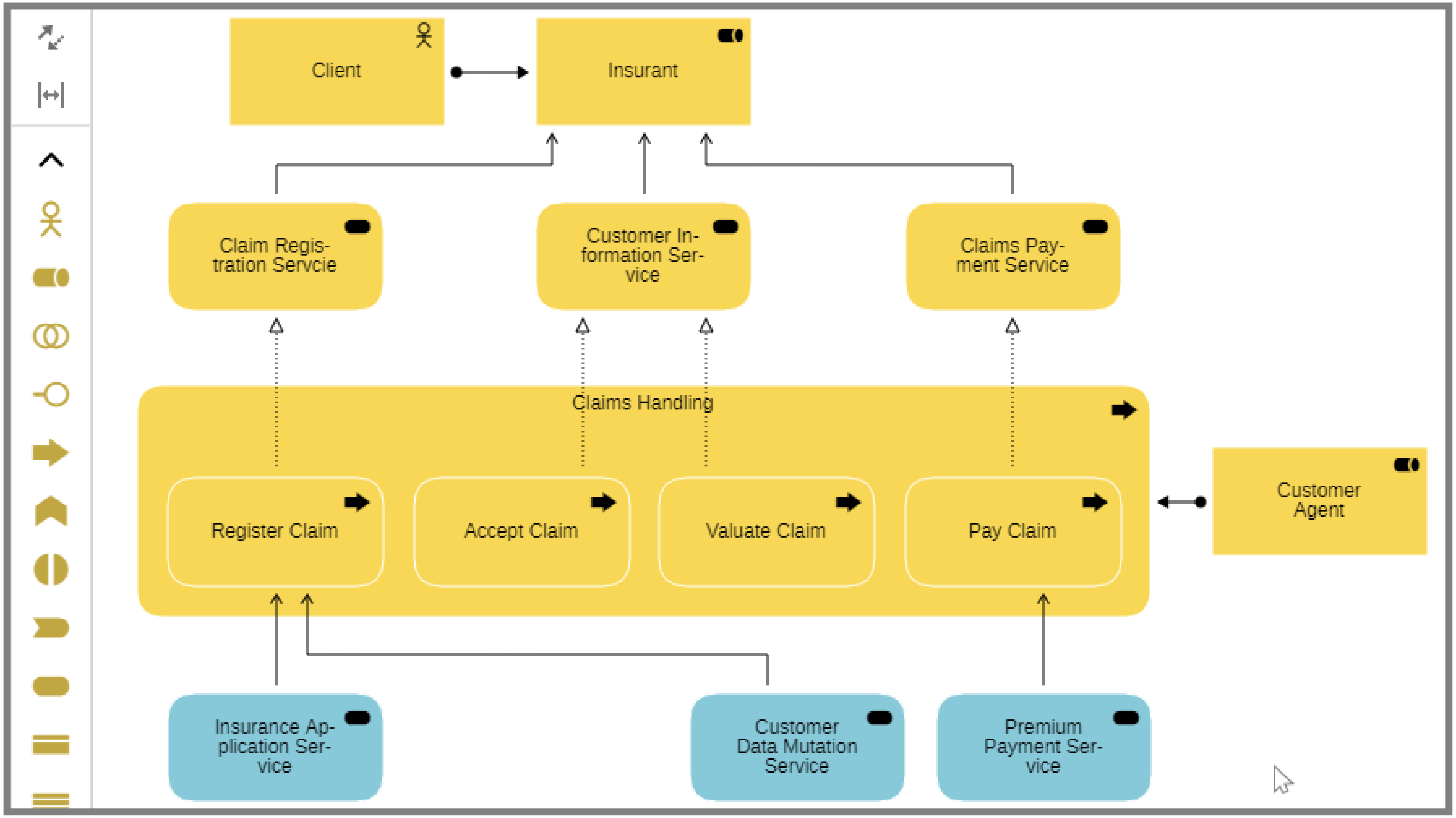
Modelling Language

Model



Modelling Language

Model

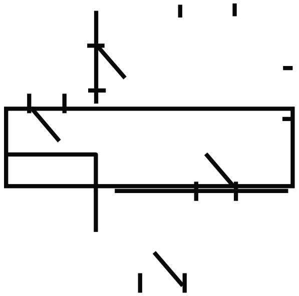


Model and Meta-Model in Architecture

real object

house

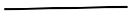
model



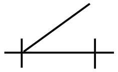
architect's drawing
(plan)

modelling language
(concrete syntax)

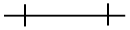
object types:



wall



door



window

meta-model
(abstract syntax)

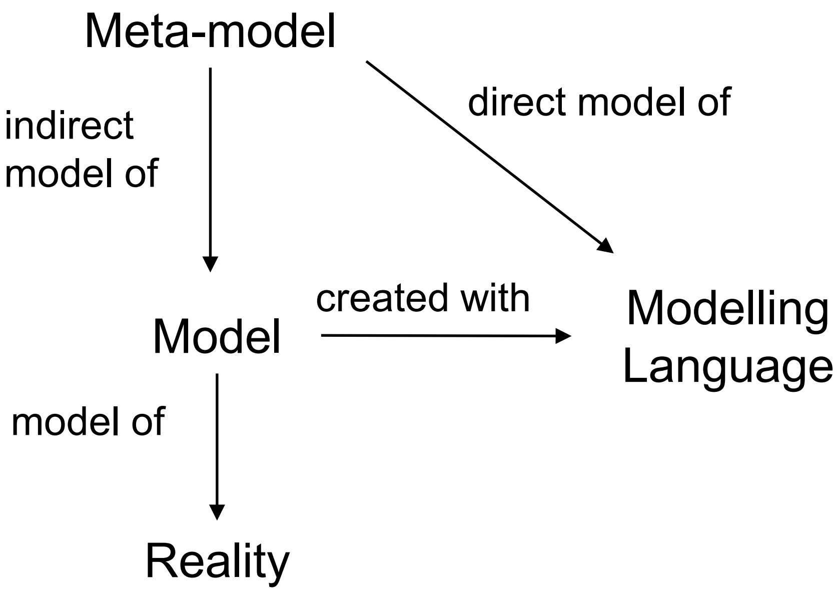
object types:

- 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

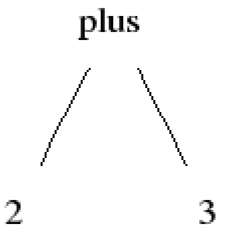
- ♦ object types that can be used to represent a model
- ♦ relations between object types
- ♦ attributes of the object types
- ♦ rules to combine object types and relations

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

Meta Model vs Model Language = Abstract vs. Concrete Syntax

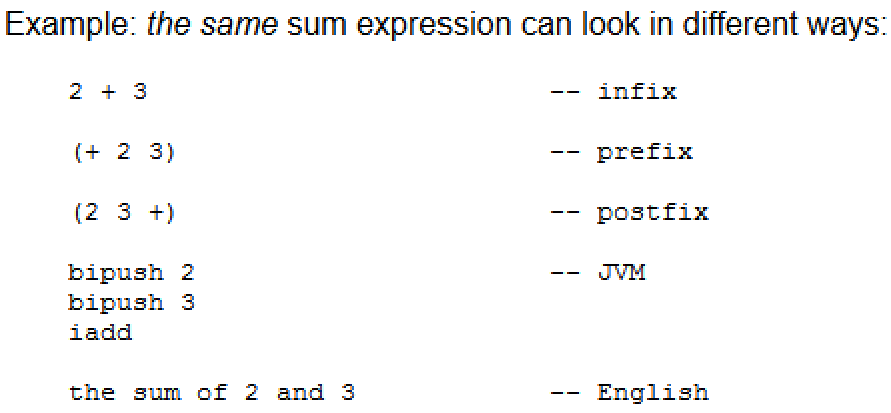
Abstract Syntax

- 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?

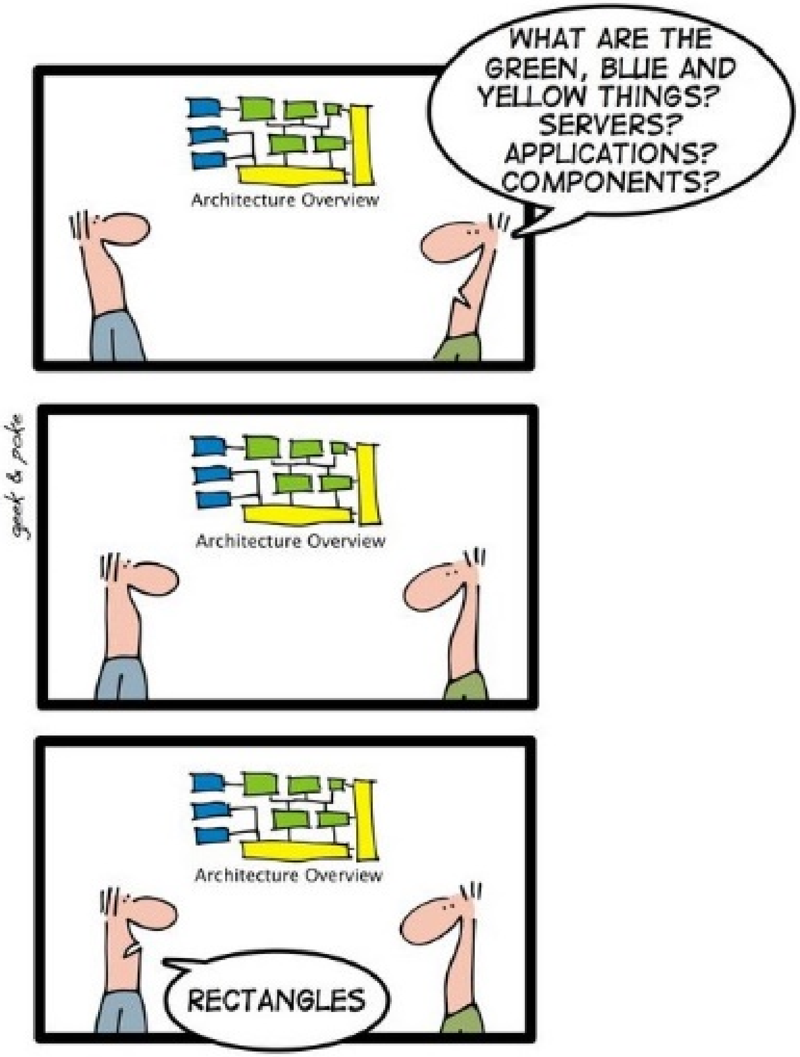


Example: *the same* sum expression can look in different ways:

```
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

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 process model consists of concepts for

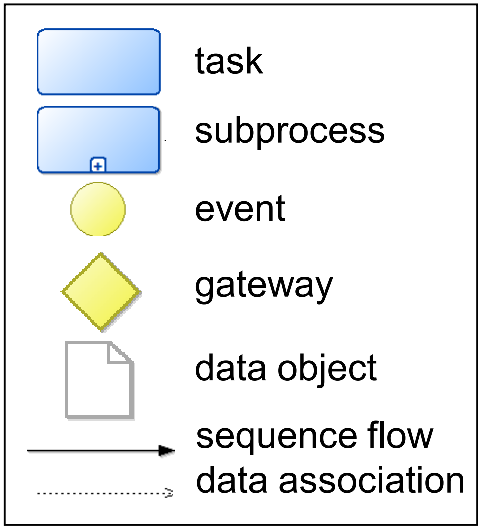
- «task», «subprocess», «event», «gateway», «data object»

and relations for

- «sequence flow», «data association».

Modelling Language:

Concrete syntax:
Notation/appearance of meta-model elements



task

subprocess

event

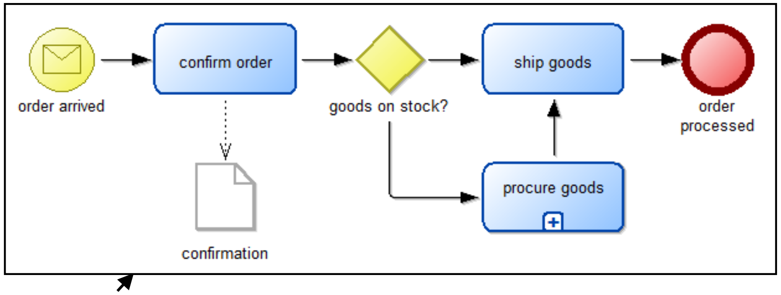
gateway

data object

sequence flow

data association

Model:

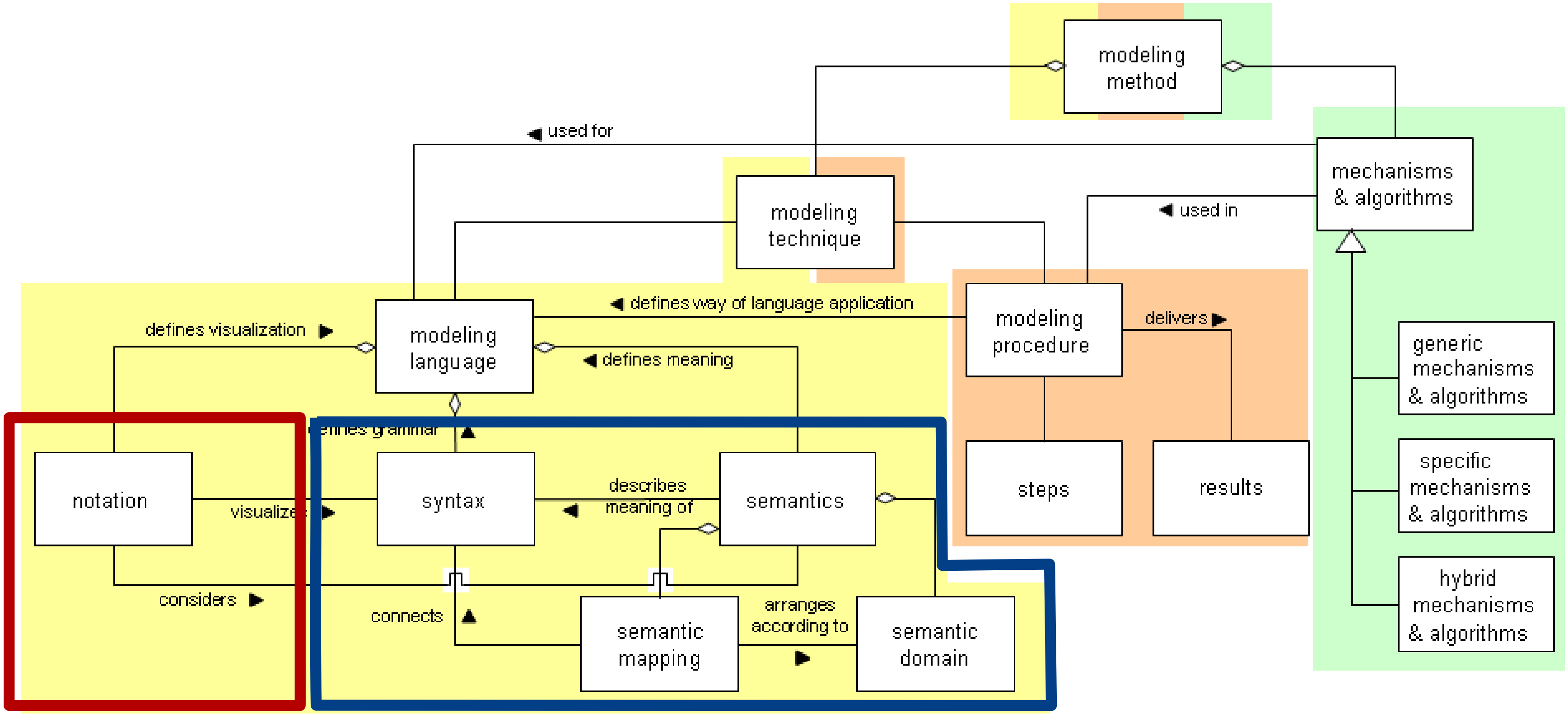


A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modelling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

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



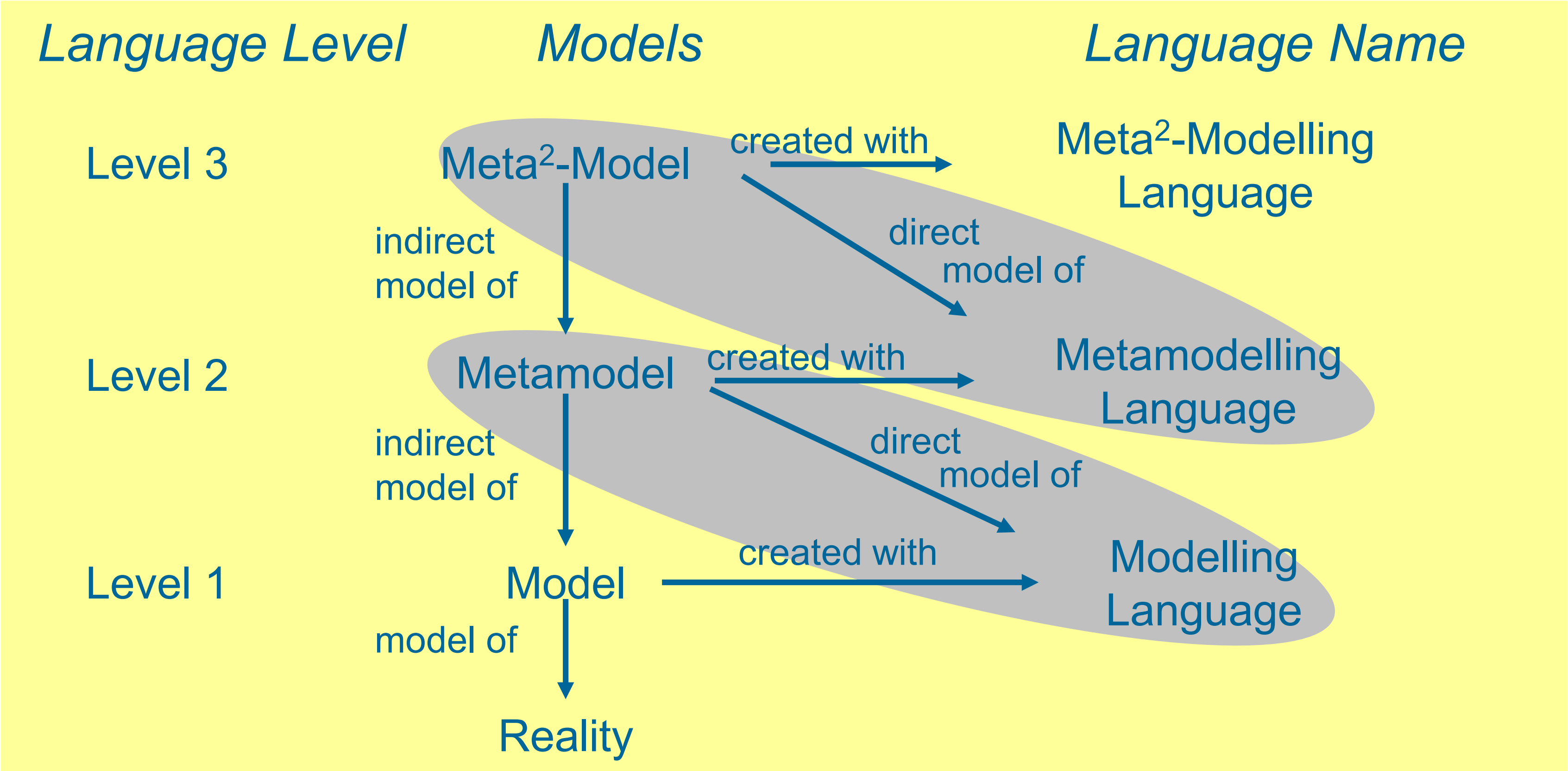
Concrete Syntax

Metamodel

(Karagiannis & Kühn 2002)

Meta Model Hierarchy

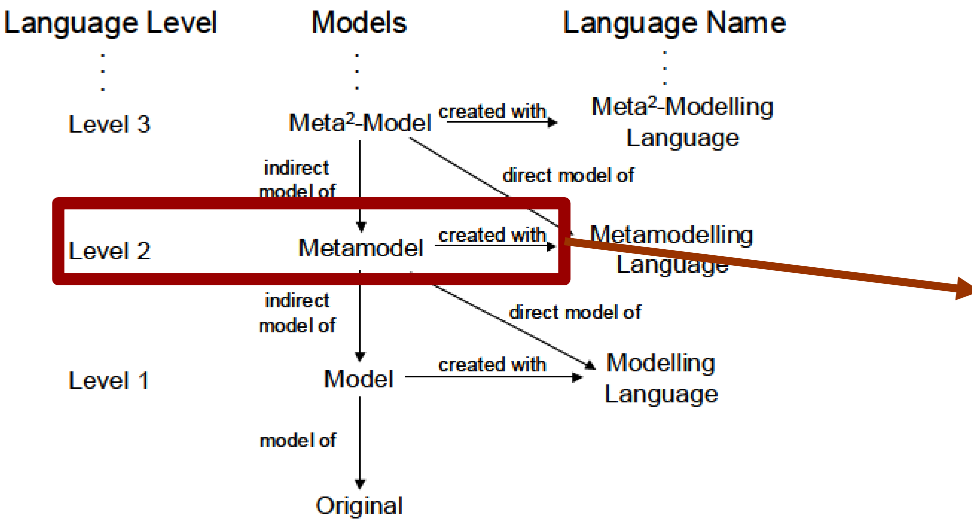
The meta-model must again be described in some language, which has to be specified in a meta-meta-model



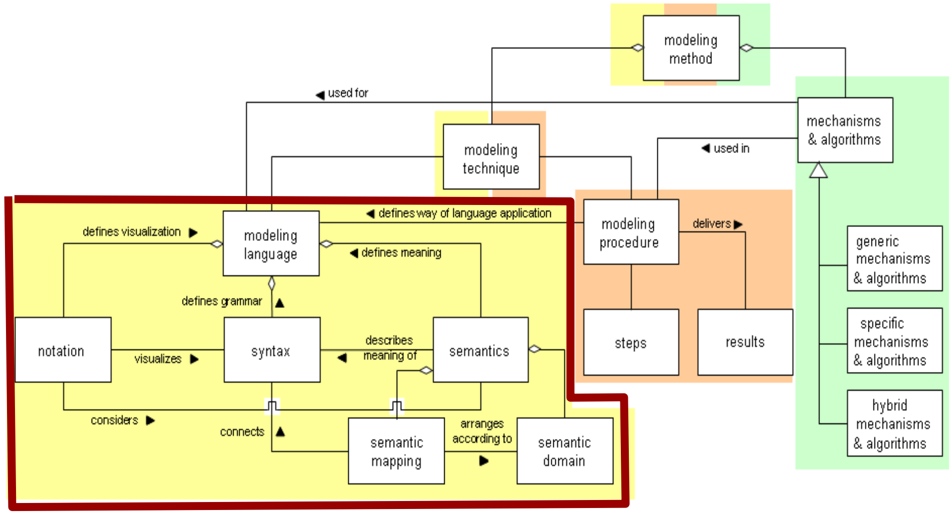
Karagiannis, D. & Kühn, H., 2002. Metamodelling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer, eds. *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.

Metamodelling

Modeling Language Definition

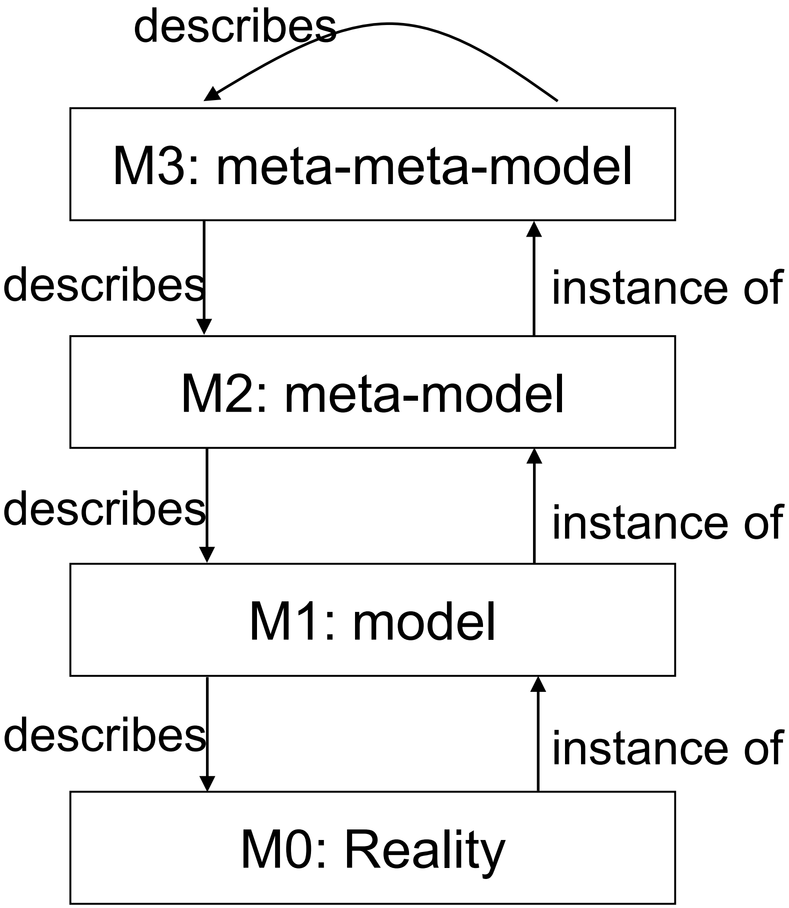


(Strahringer, 1996)



(Karagiannis & Kühn, 2002)

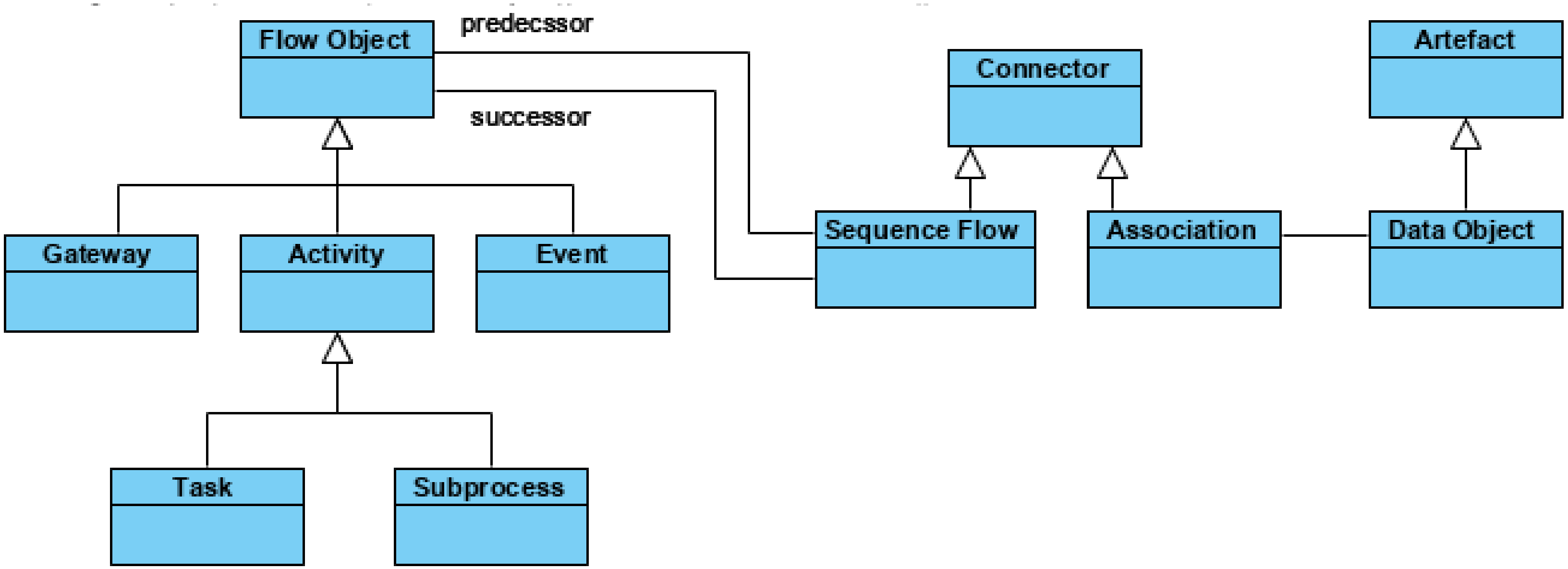
The Model Stack



- 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

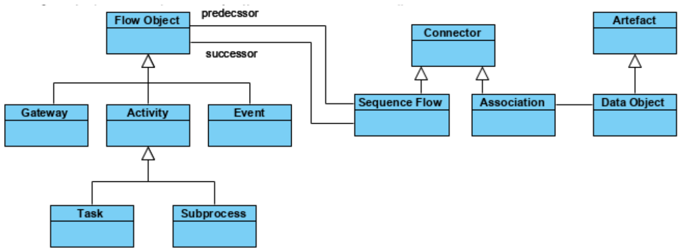


(UML Class diagrams where originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

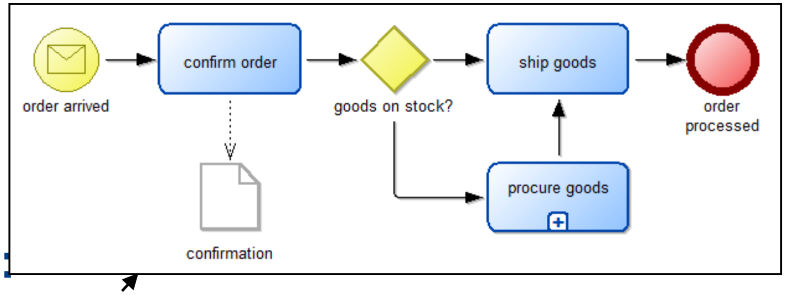
A Metamodel for Processes

Meta-model:

- Classes and relations that can be used for modelling

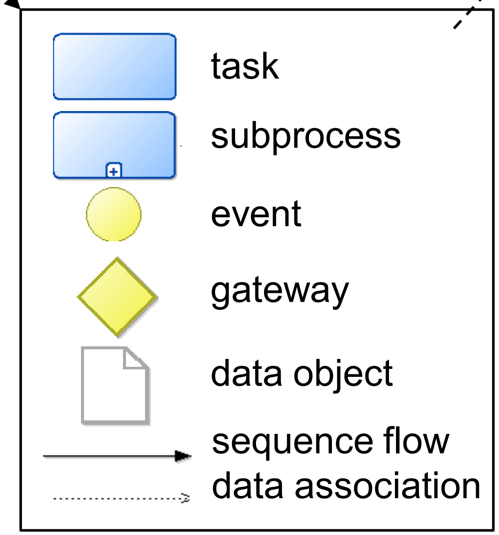


Model:



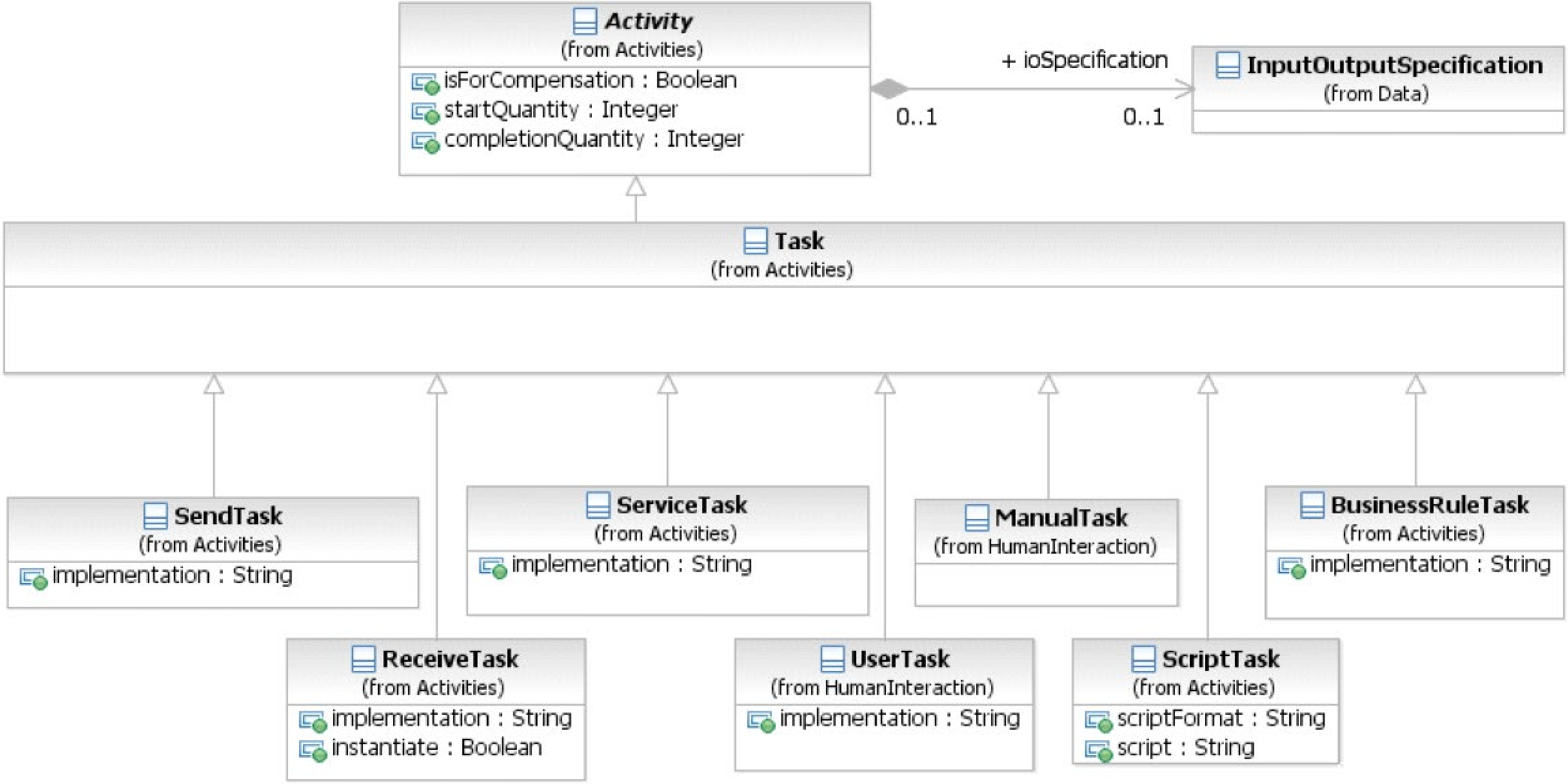
Modelling Language:

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"

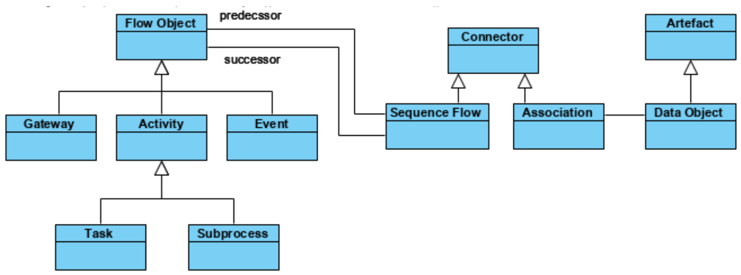
Subset of the BPMN Metamodel in UML



Source: BPMN 2.0 specification

Meta-Modelling Langauge has to be defined in a Meta-Meta Model

Meta-Model:



Meta^2 Model:

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

Example: A class and object diagram consists of concepts for

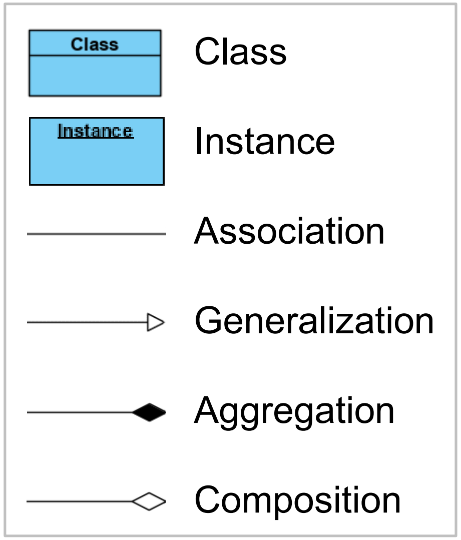
- «classes», «instances»,

and relations for

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

Meta- Modelling Language:

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



Class

Instance

Association

Generalization

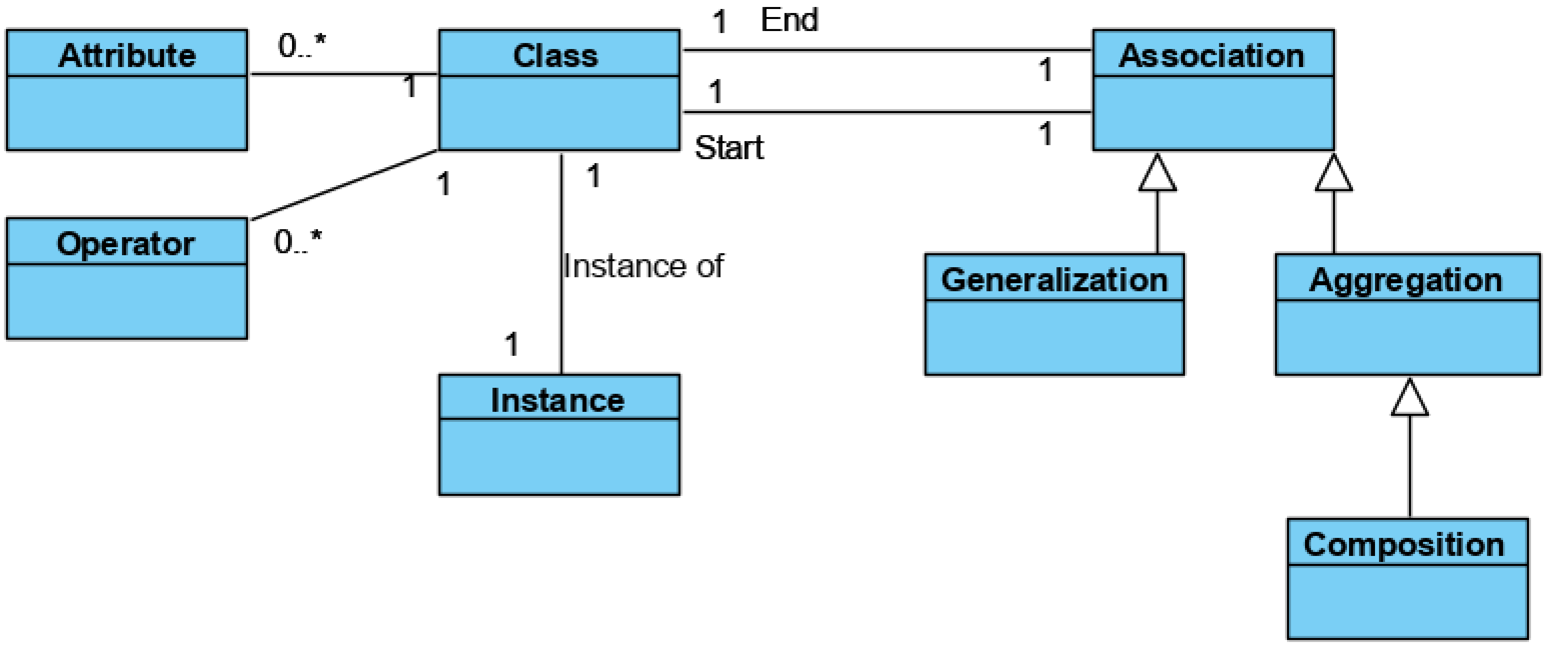
Aggregation

Composition

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"

A Metamodel for UML Class Diagrams

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

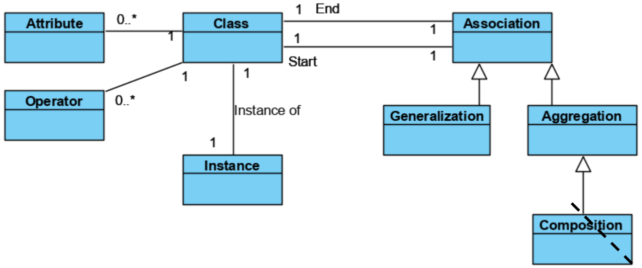


(UML Class diagrams where originally designed for modelling in object-oriented programming. This is why they contain operations and other features, which are not relevant for most modelling languages)

UML Class Diagrams can be used for Meta-Meta-Model

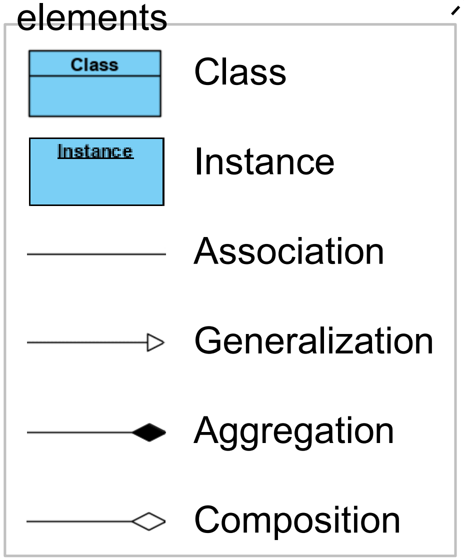
Meta(Meta)-model:

- Classes and relations that can be used for modelling



(Meta-) Modelling Language:

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



Class

Instance

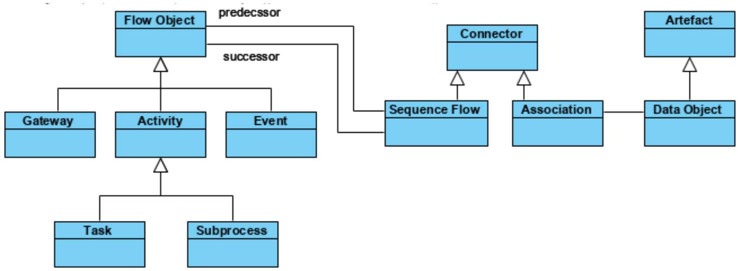
Association

Generalization

Aggregation

Composition

(Meta)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

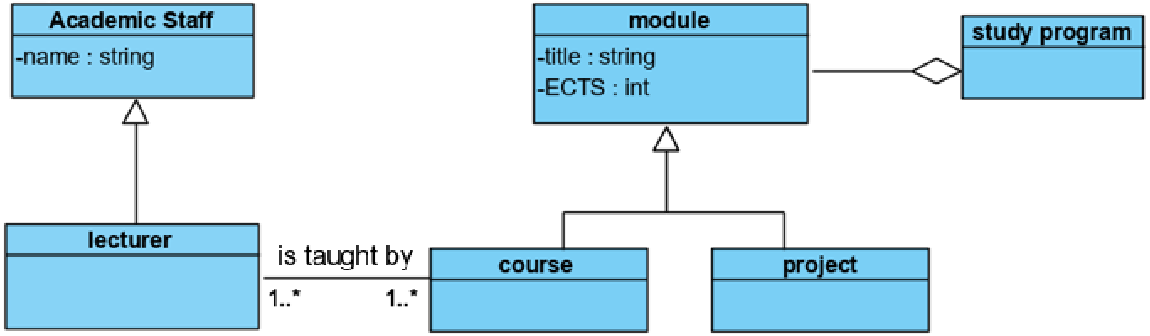
The Metamodel for a General-purpose Modelling Language

- 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

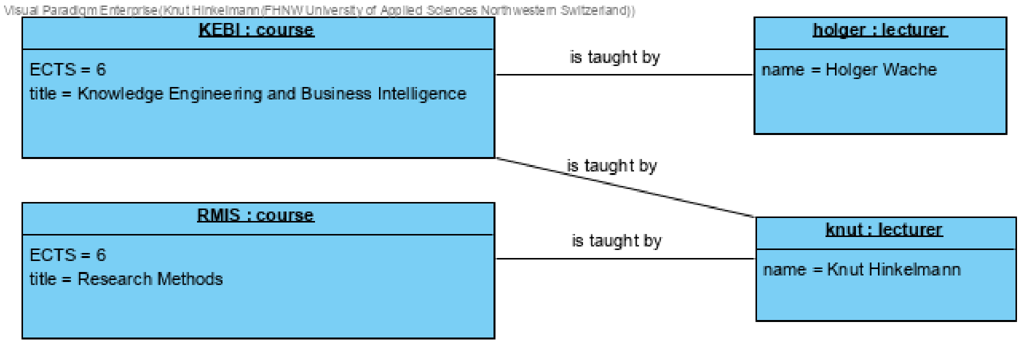
Modelling with a General-purpose Modelling Language

- 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
  - Restrictred 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 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, …

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

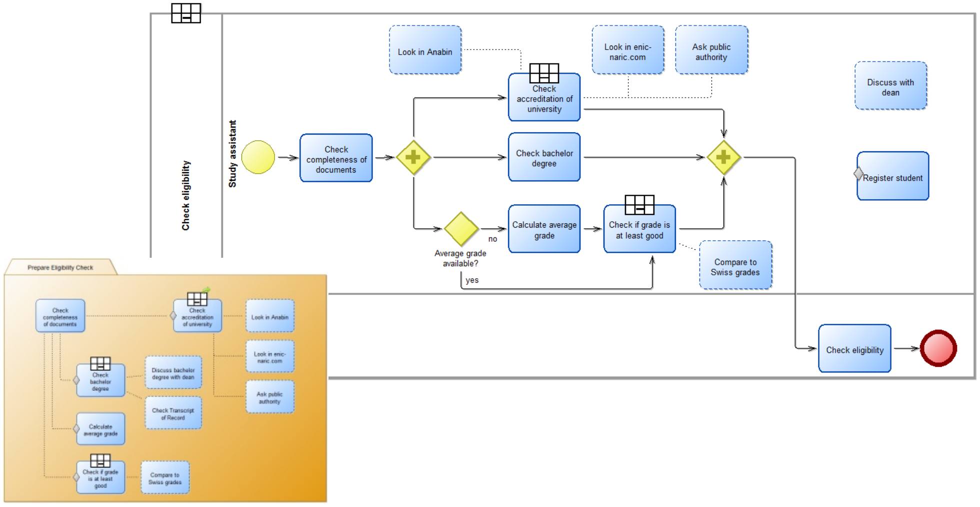
- If there is no domain-specific modelling language for a domain of interest, we can
  1. Use a general-purpose modelling language
  2. Define a new domain-specific modelling language
     - From scratch
     - By adapting an existing one

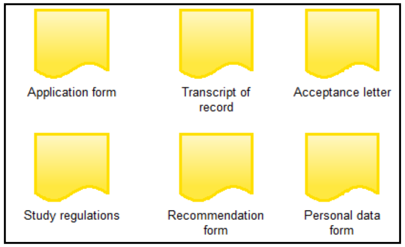
→ ***meta modelling***

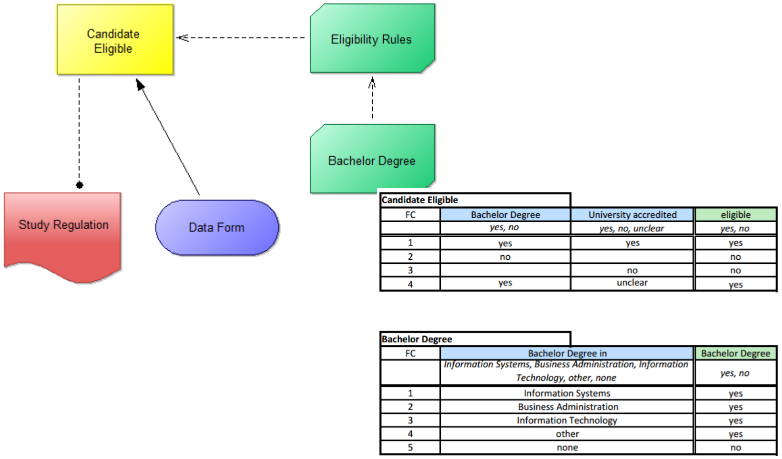
Knowledge Work Designer

Modeling of Knowledge Work

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







Candidate Eligible

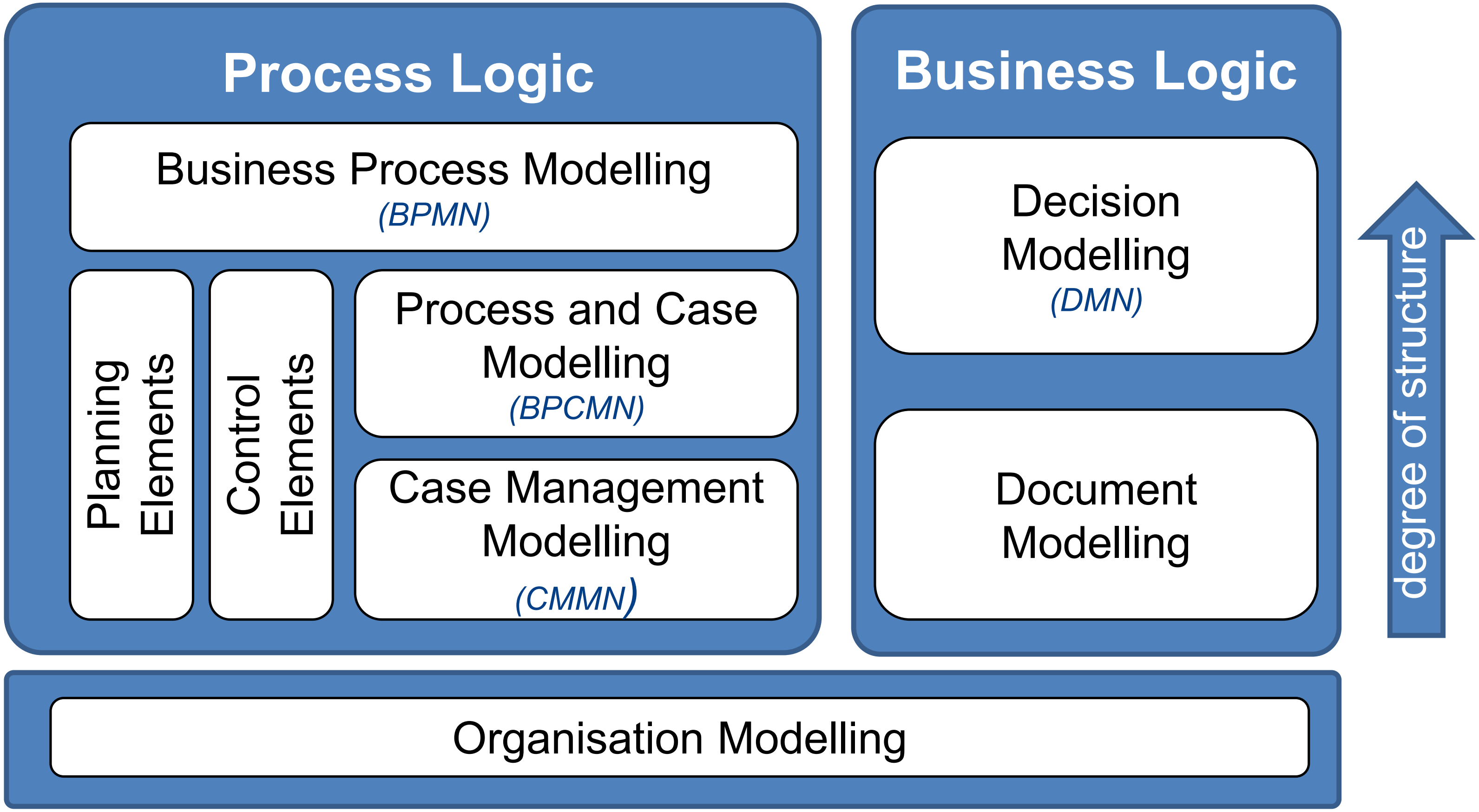
FC	Bachelor Degree	University accredited	eligible
	yes, no	*yes, no, unclear*	*yes, no*
1	yes	yes	yes
2	no		no
3		no	no
4	yes	unclear	yes

Bachelor Degree

FC	Bachelor Degree in	Bachelor Degree
	Information Systems, Business Administration, Information Technology, other, none	*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/

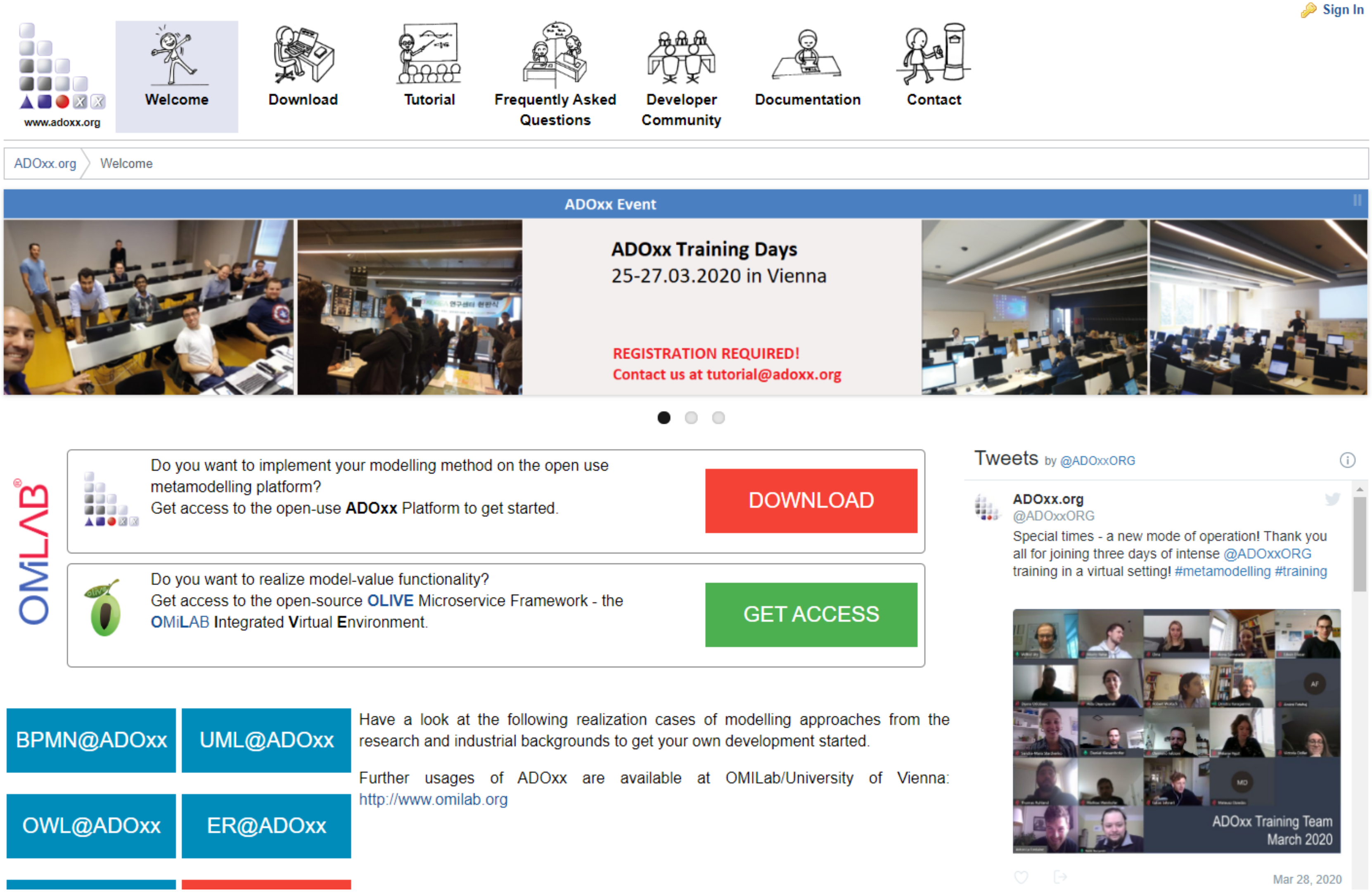
Model types of the Knowledge Work Designer



Metamodelling with ADOxx

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

adoxx.org – Download, Tutorials, Community



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 metamodelling platform?
Get access to the open-use **ADOxx** Platform to get started.

DOWNLOAD

Do you want to realize model-value functionality?
Get access to the open-source **OLIVE** Microservice Framework - the **O**MiLAB **I**ntegrated **V**irtual **E**nvironment.

GET ACCESS

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

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

Tweets by @ADOxxORG

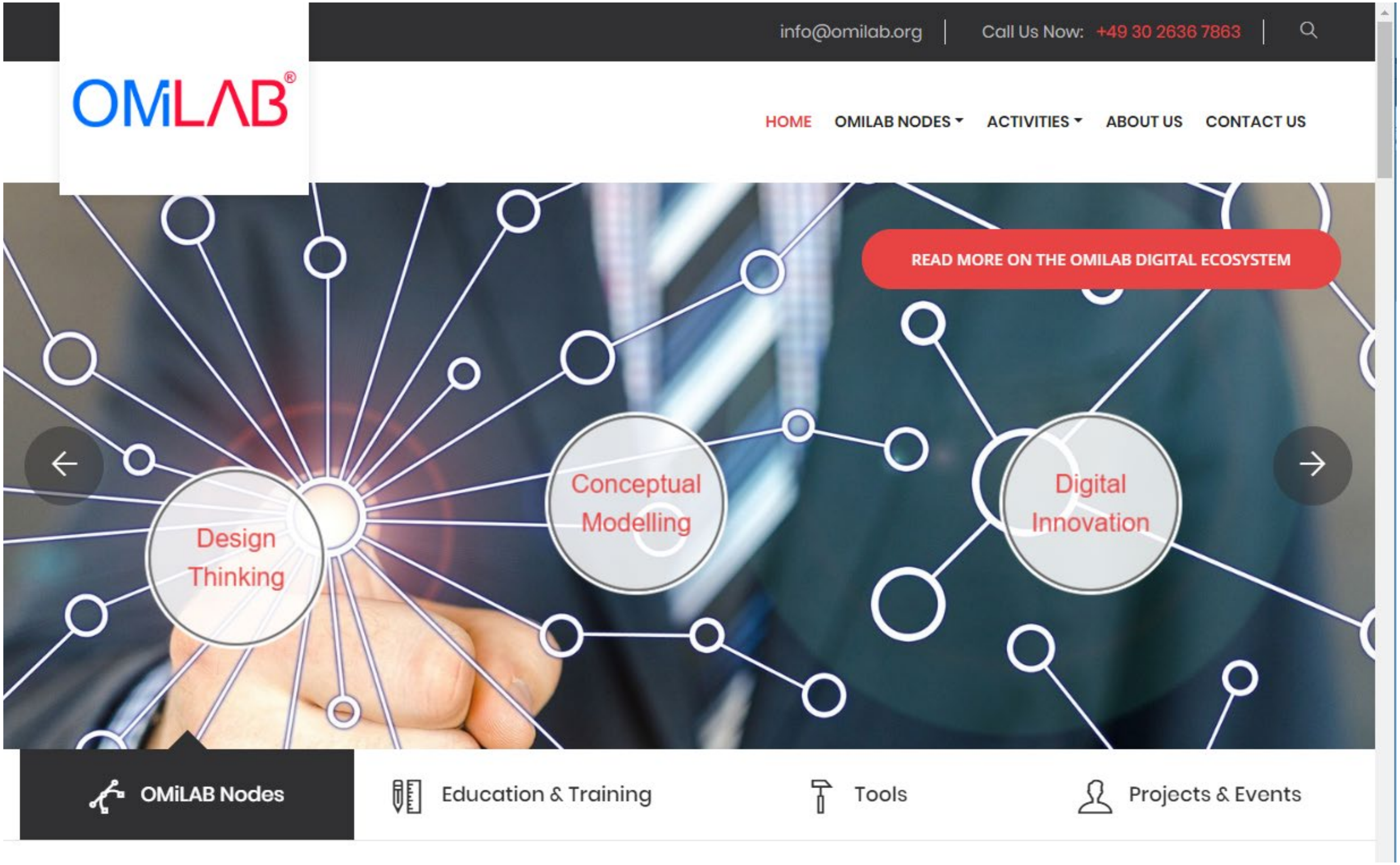
ADOxx.org
@ADOxxORG
Special times - a new mode of operation! Thank you all for joining three days of intense @ADOxxORG training in a virtual setting! #metamodelling #training

ADOxx Training Team
March 2020

Mar 28, 2020

OMiLAB – A Conceptual Modelling Commnity

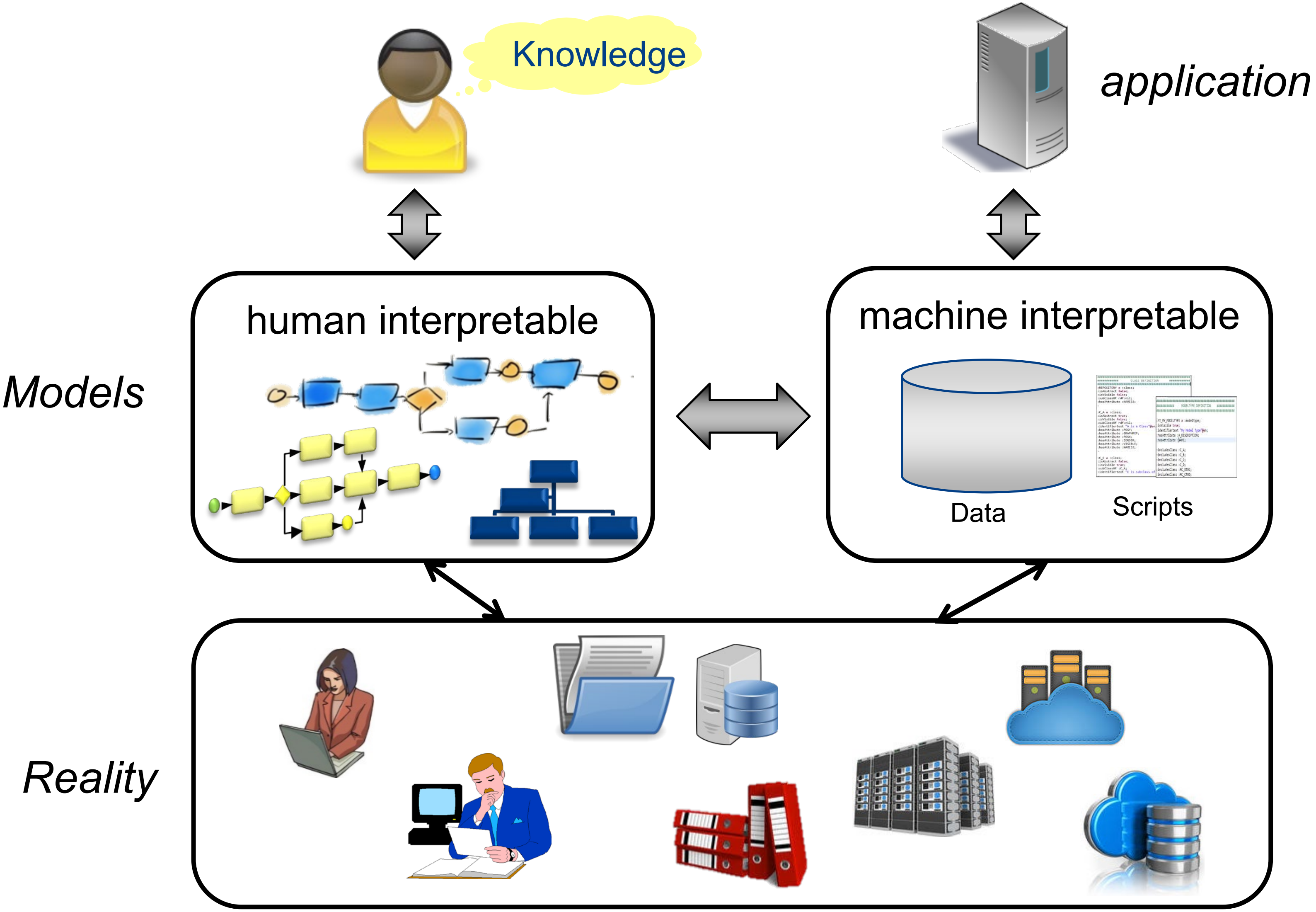
ADOxx is the basis for OMiLAB



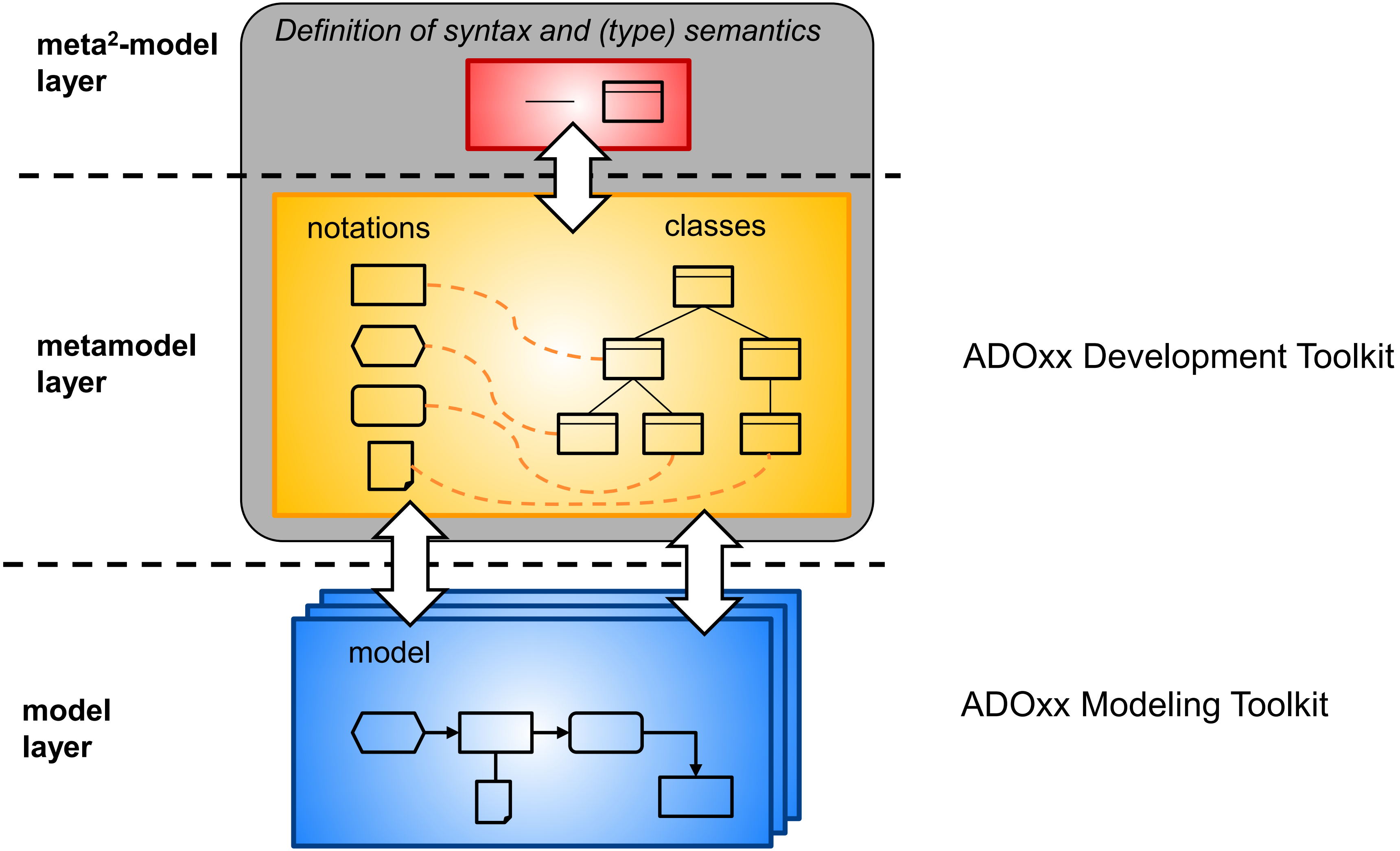
The ADOxx Environment

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

Graphical Models are Represented in a Database

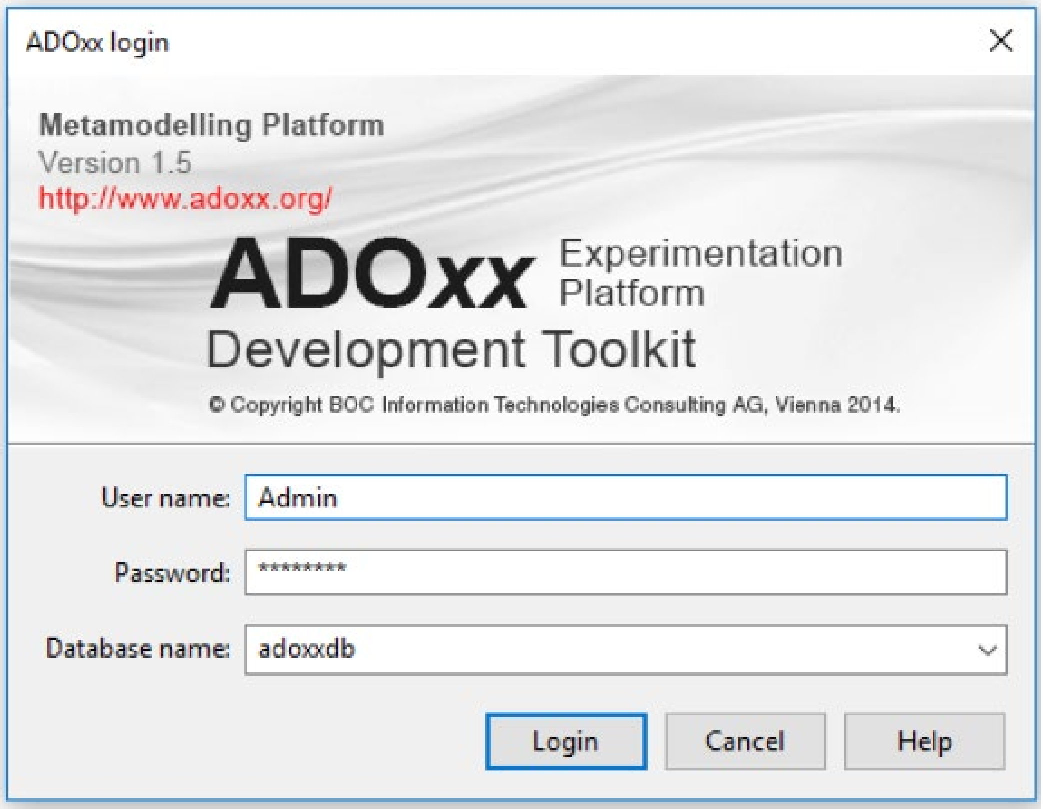


Modeling Environment



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

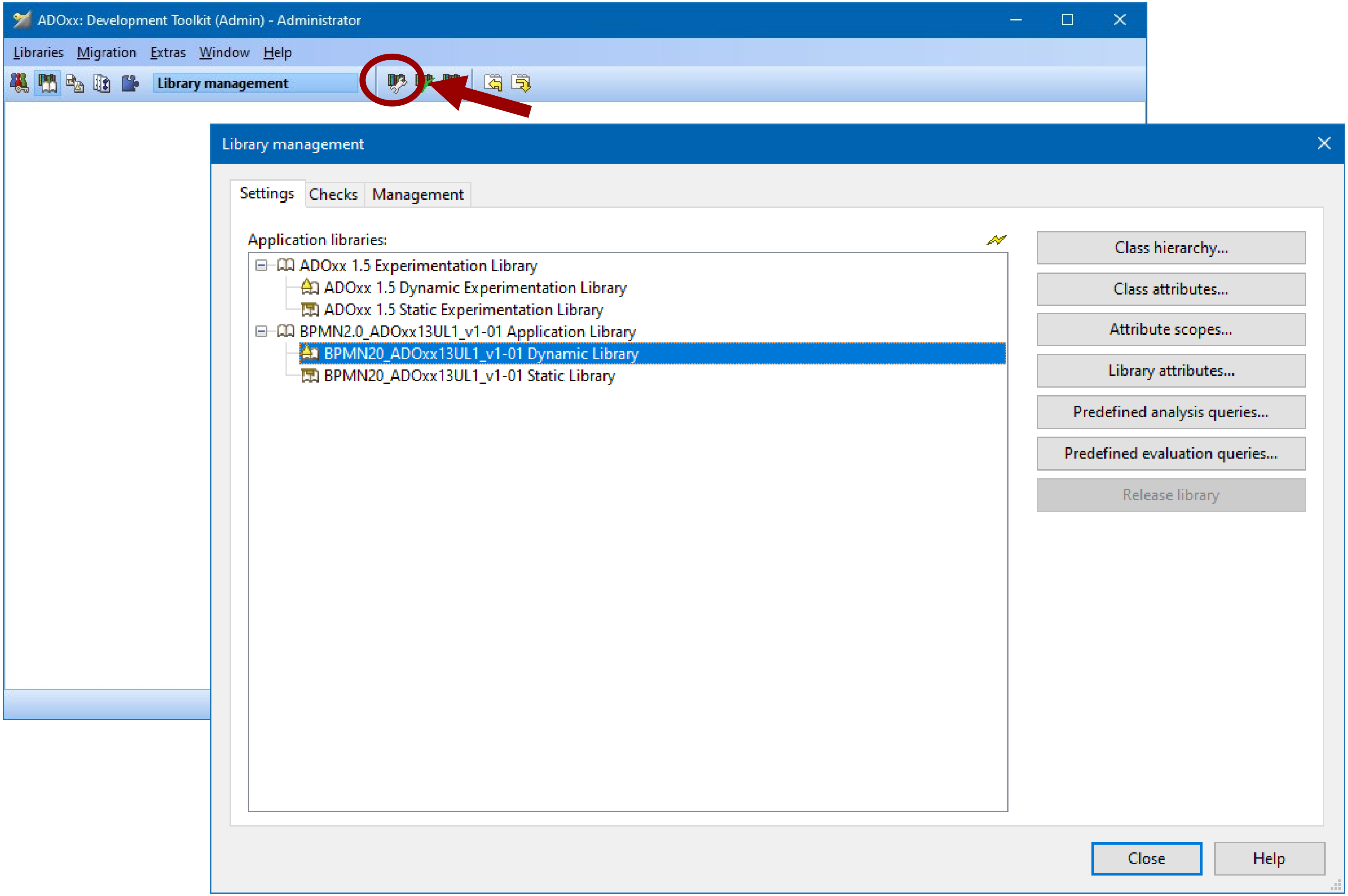
© Copyright BOC Information Technologies Consulting AG, Vienna 2014.

User name: Admin

Password: *********

Database name: adoxxdb

Metamodelling with ADOxx



ADOxx: Development Toolkit (Admin) - Administrator

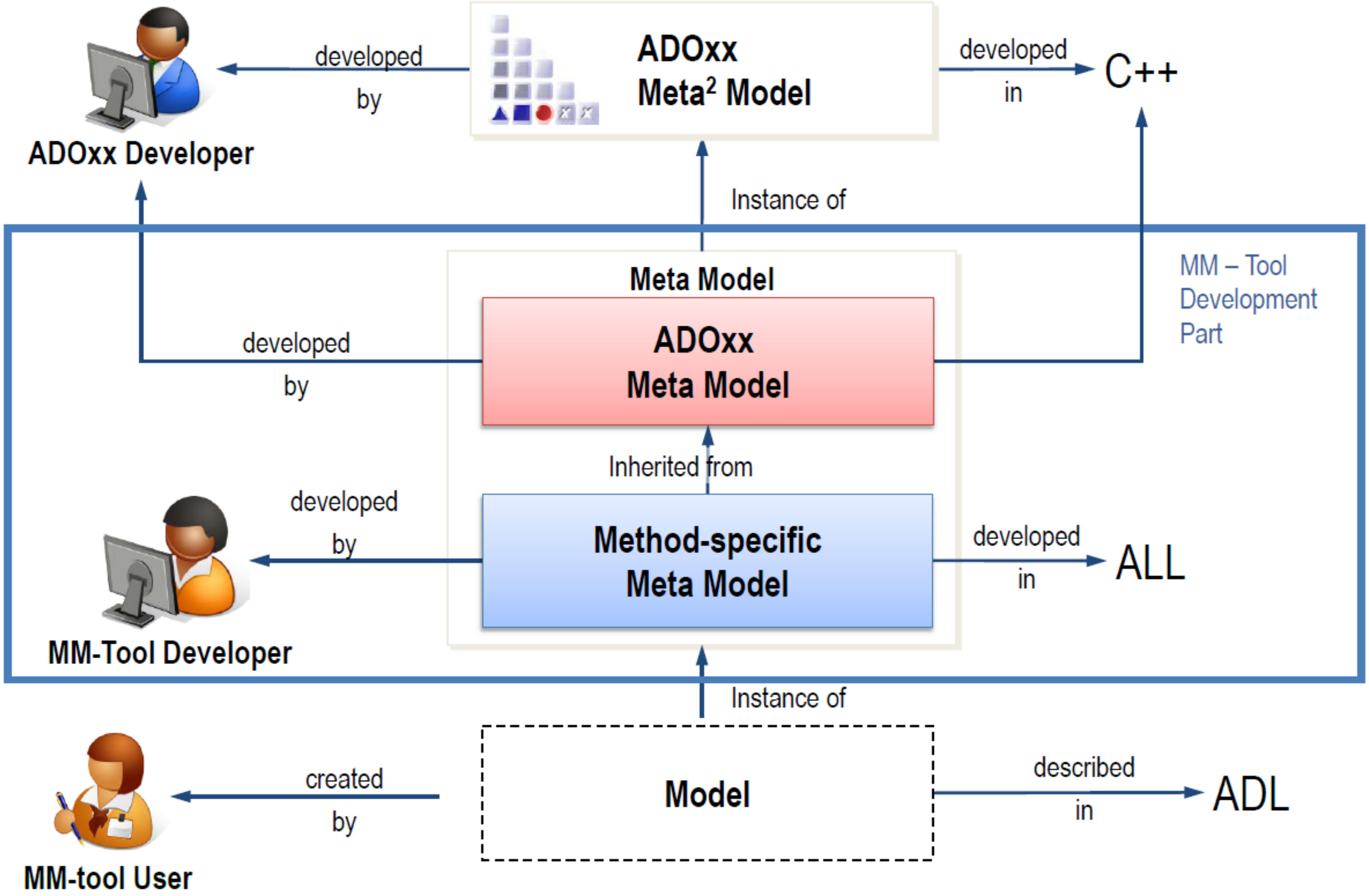
Library management

Application libraries:

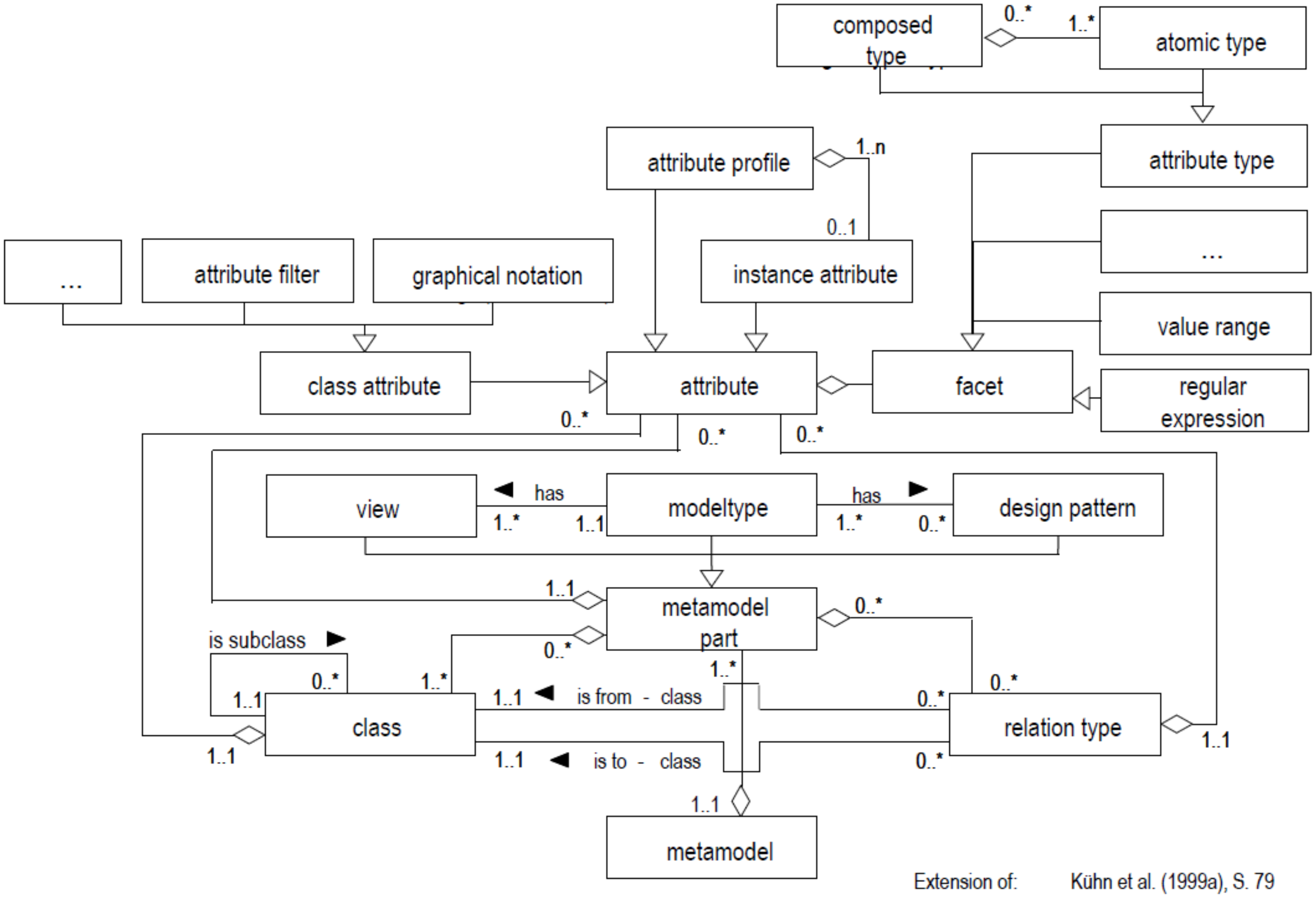
- ADOxx 1.5 Experimentation Library
  - ADOxx 1.5 Dynamic Experimentation Library
  - ADOxx 1.5 Static Experimentation Library
- BPMN2.0_ADOxx13UL1_v1-01 Application Library
  - BPMN20_ADOxx13UL1_v1-01 Dynamic Library
  - BPMN20_ADOxx13UL1_v1-01 Static Library

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

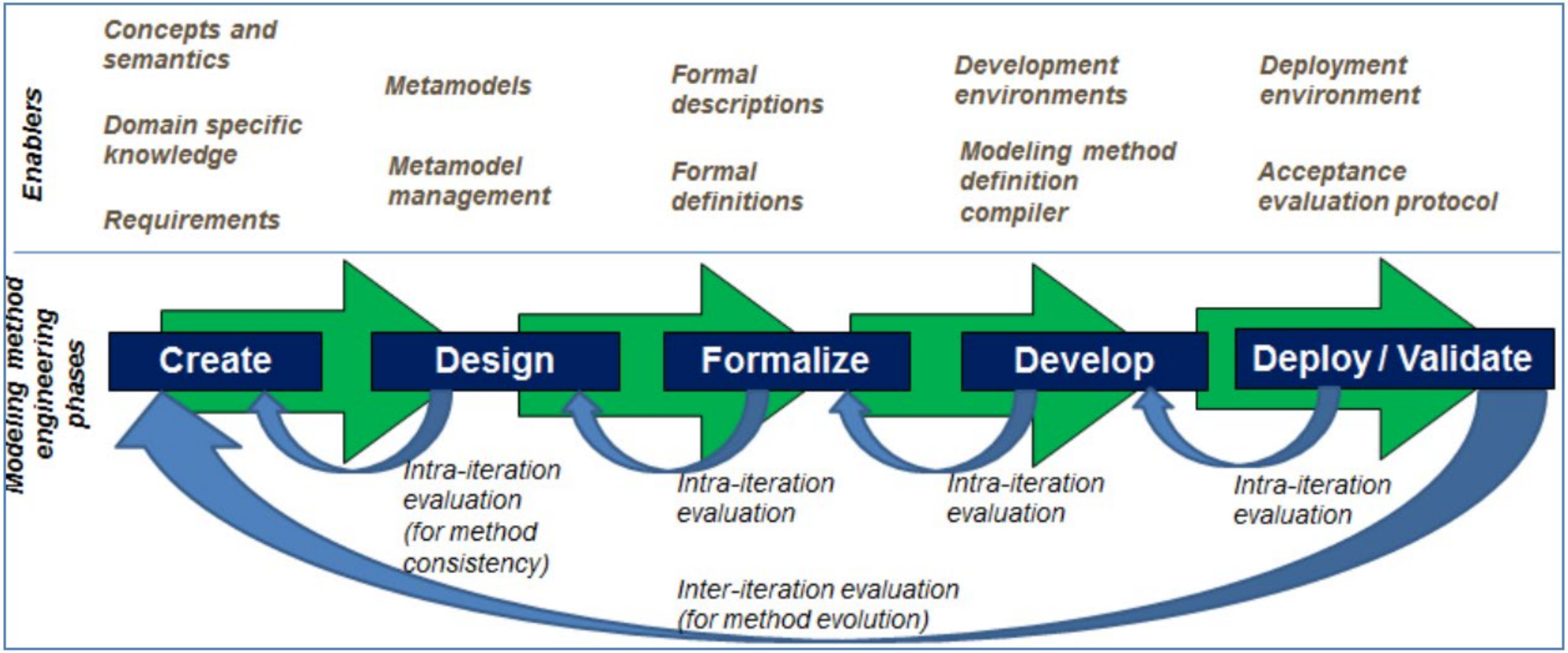
Meta Modelling Platforms Hierarchyin ADOxx



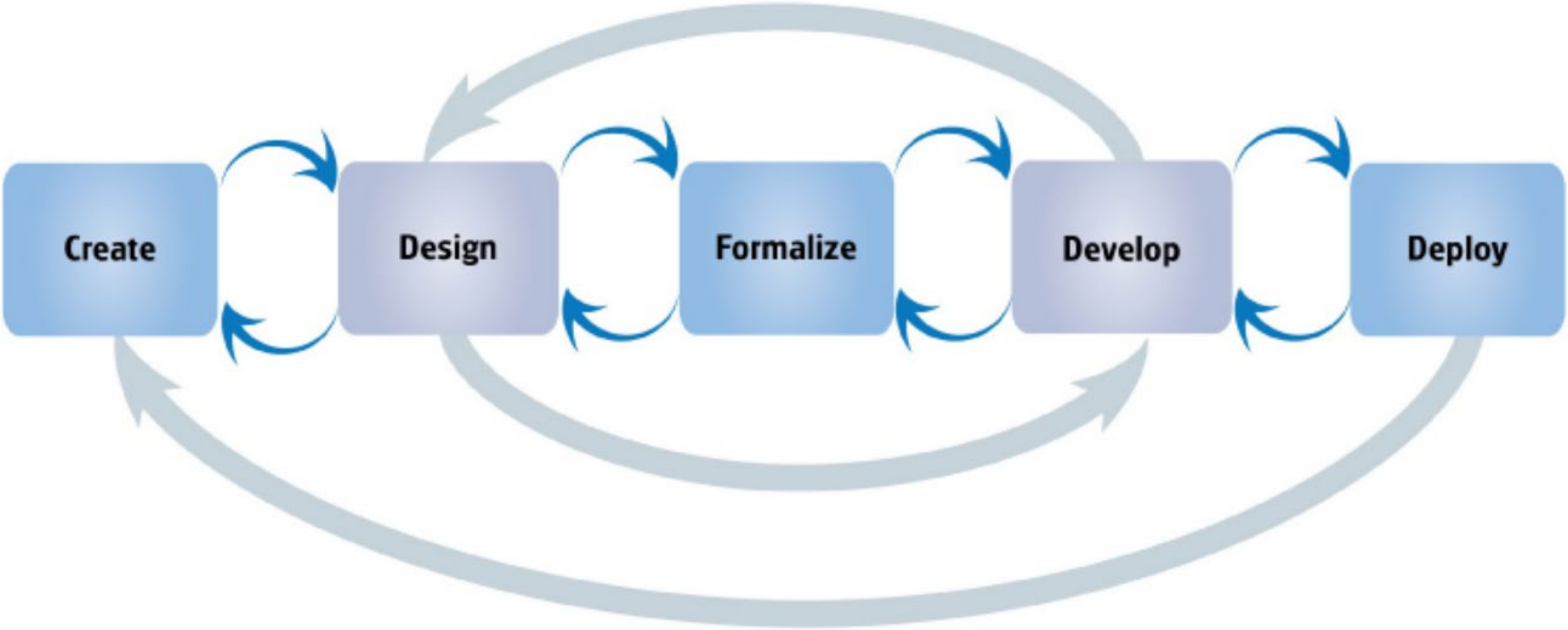
Meta^2 Model: Meta Model of Meta Modelling Language



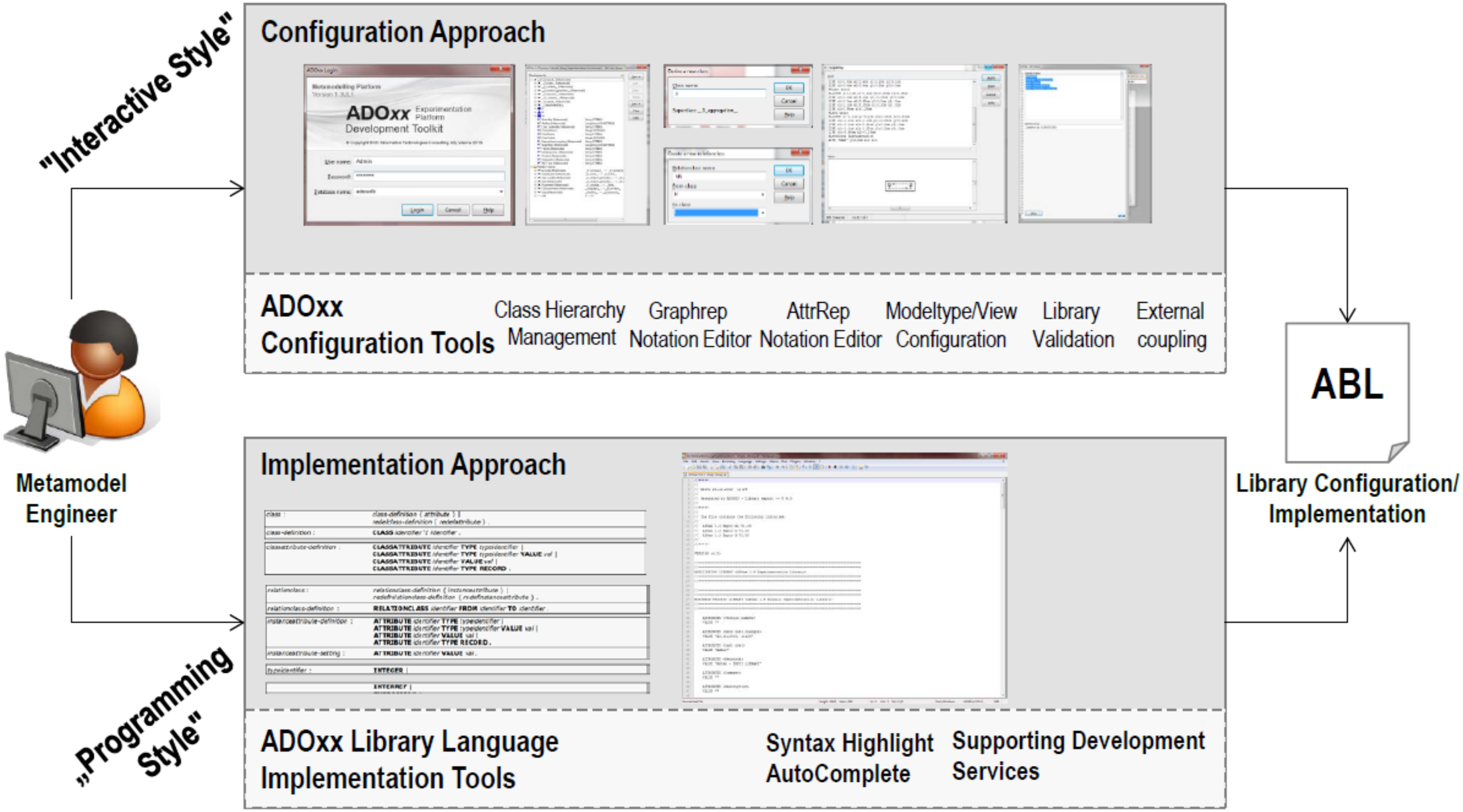
Extension of: Kühn et al. (1999a), S. 79


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

The AMME LifeCycle
Agile Meta Model Engineering



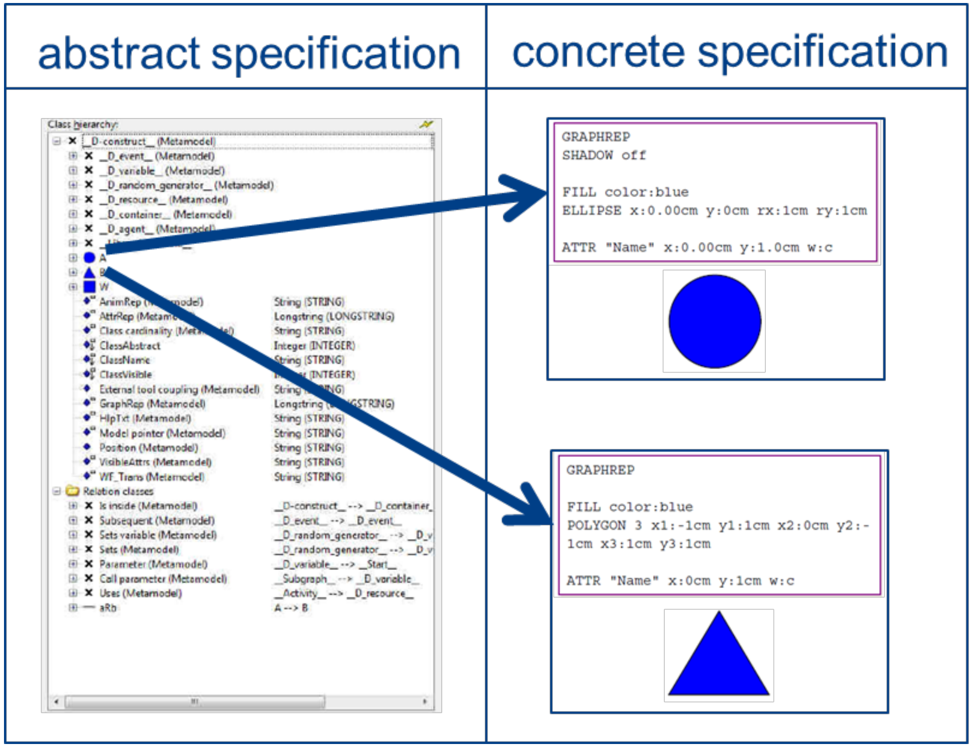
Development Appraoches in ADOxx – Configuration and Implementation



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

Abstract and Concrete Specification

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



abstract specification

concrete specification

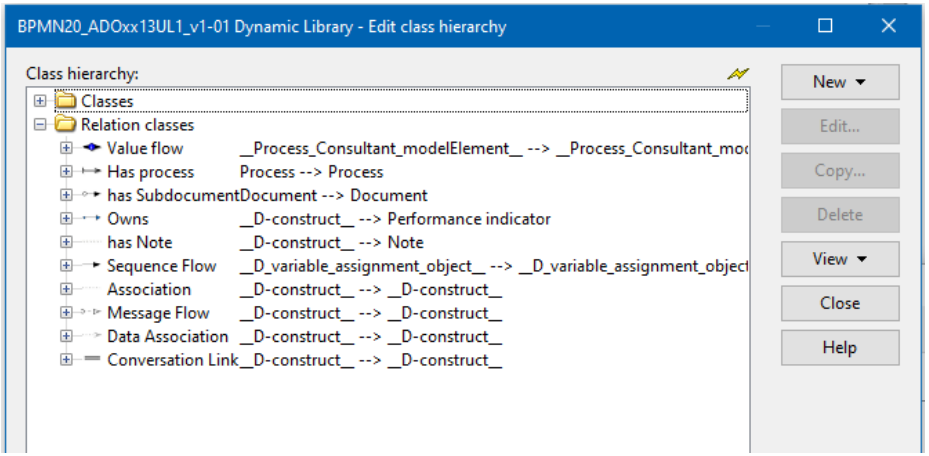
```
Class hierarchy:
⊟ ✕ __D-construct__ (Metamodel)
   ⊞ ✕ __D_event__ (Metamodel)
   ⊞ ✕ __D_variable__ (Metamodel)
   ⊞ ✕ __D_random_generator__ (Metamodel)
   ⊞ ✕ __D_resource__ (Metamodel)
   ⊞ ✕ __D_container__ (Metamodel)
   ⊞ ✕ __D_agent__ (Metamodel)
   ⊞ ✕ [illegible]
   ⊞ ● A
   ⊞ ▲ B
   ⊞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

```
GRAPHREP
SHADOW off

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

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

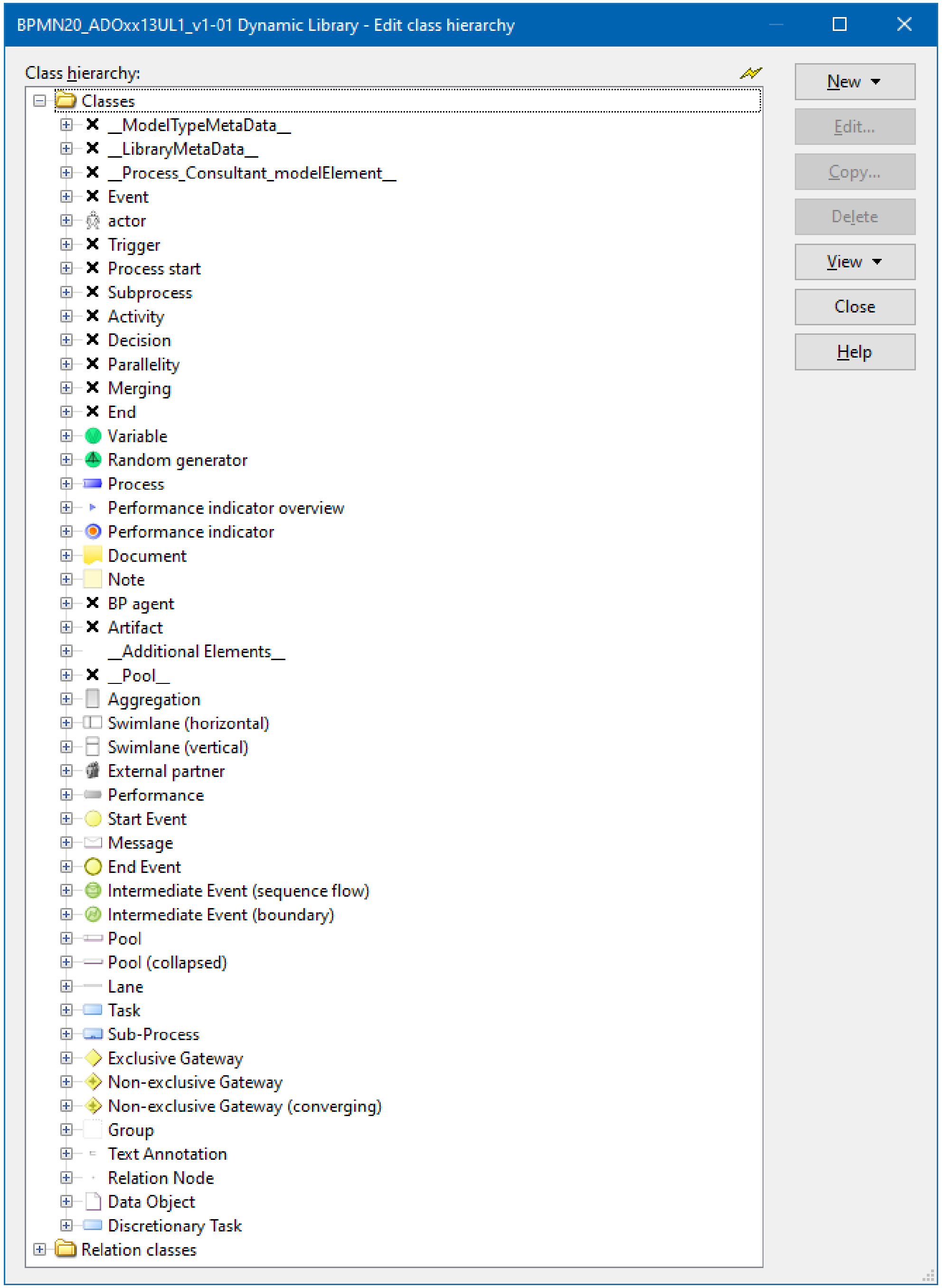
```
GRAPHREP

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

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

Class Hierarchies

- ADOxx distinguishes
  - Classes
  - Relation classes



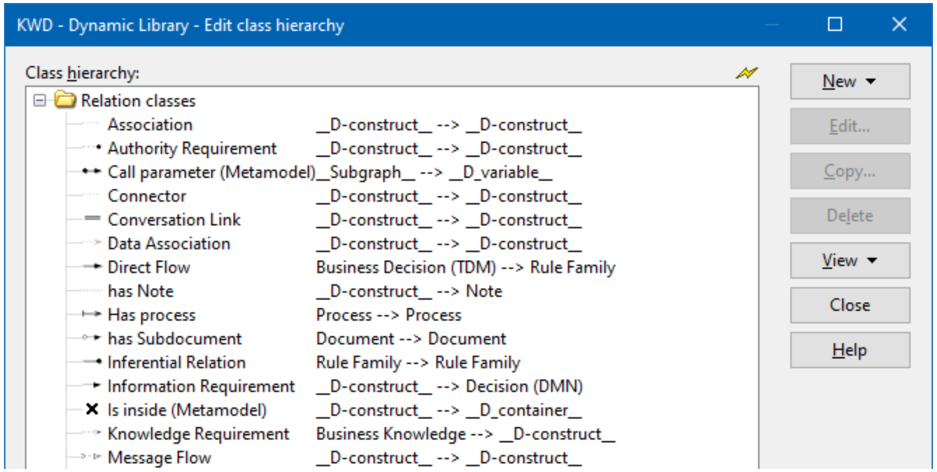
```
BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

Class hierarchy:
⊞ Classes
⊟ Relation classes
    ⊞ Value flow         __Process_Consultant_modelElement__ --> __Process_Consultant_moc
    ⊞ Has process        Process --> Process
    ⊞ has SubdocumentDocument --> Document
    ⊞ Owns               __D-construct__ --> Performance indicator
    ⊞ has Note           __D-construct__ --> Note
    ⊞ Sequence Flow      __D_variable_assignment_object__ --> __D_variable_assignment_object
    ⊞ Association        __D-construct__ --> __D-construct__
    ⊞ Message Flow       __D-construct__ --> __D-construct__
    ⊞ Data Association   __D-construct__ --> __D-construct__
    ⊞ Conversation Link__D-construct__ --> __D-construct__
```



```
BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

Class hierarchy:
⊟ Classes
    ⊞ __ModelTypeMetaData__
    ⊞ __LibraryMetaData__
    ⊞ __Process_Consultant_modelElement__
    ⊞ Event
    ⊞ actor
    ⊞ Trigger
    ⊞ Process start
    ⊞ Subprocess
    ⊞ Activity
    ⊞ Decision
    ⊞ Parallelity
    ⊞ Merging
    ⊞ End
    ⊞ Variable
    ⊞ Random generator
    ⊞ Process
    ⊞ Performance indicator overview
    ⊞ Performance indicator
    ⊞ Document
    ⊞ Note
    ⊞ BP agent
    ⊞ Artifact
    ⊞ __Additional Elements__
    ⊞ __Pool__
    ⊞ Aggregation
    ⊞ Swimlane (horizontal)
    ⊞ Swimlane (vertical)
    ⊞ External partner
    ⊞ Performance
    ⊞ Start Event
    ⊞ Message
    ⊞ End Event
    ⊞ Intermediate Event (sequence flow)
    ⊞ Intermediate Event (boundary)
    ⊞ Pool
    ⊞ Pool (collapsed)
    ⊞ Lane
    ⊞ Task
    ⊞ Sub-Process
    ⊞ Exclusive Gateway
    ⊞ Non-exclusive Gateway
    ⊞ Non-exclusive Gateway (converging)
    ⊞ Group
    ⊞ Text Annotation
    ⊞ Relation Node
    ⊞ Data Object
    ⊞ Discretionary Task
⊞ Relation classes
```

Class Hierarchies

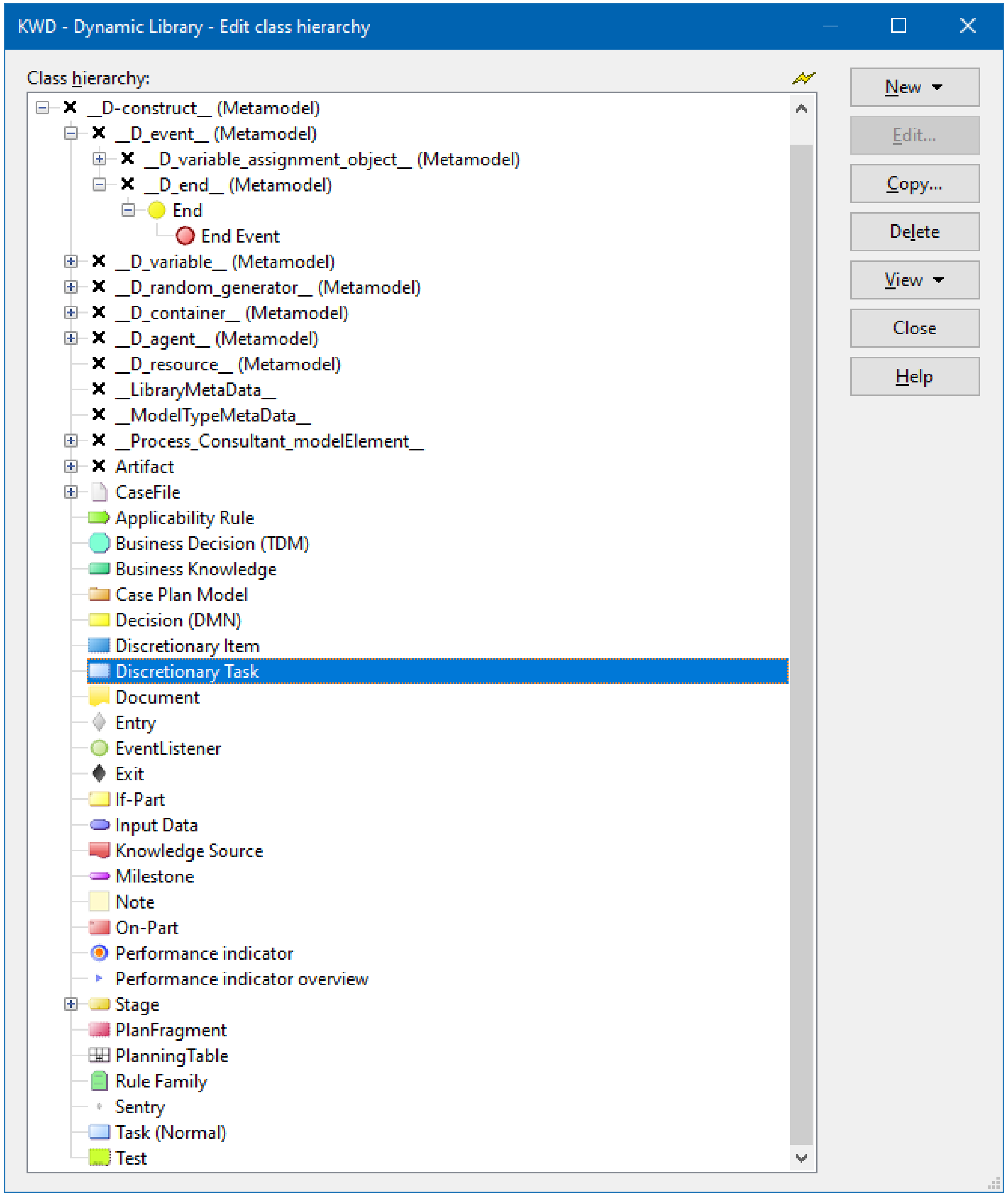
- ADOxx distinguishes
  - ♦ Classes
  - ♦ Relation classes


KWD - Dynamic Library - Edit class hierarchy

Class hierarchy:

⊟ Relation classes

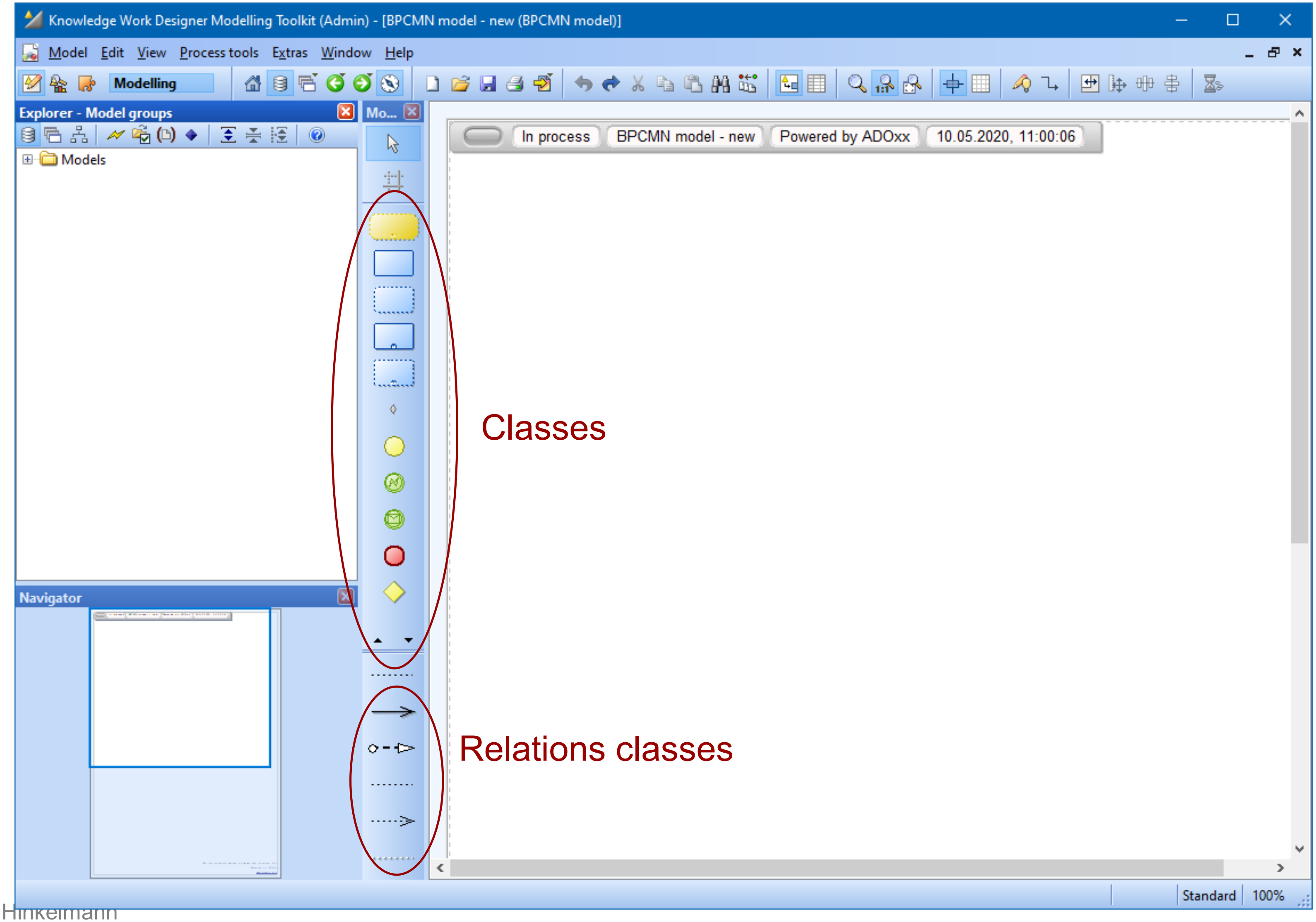
Association	_D-construct_ --> _D-construct_
Authority Requirement	_D-construct_ --> _D-construct_
Call parameter (Metamodel)	_Subgraph_ --> _D_variable_
Connector	_D-construct_ --> _D-construct_
Conversation Link	_D-construct_ --> _D-construct_
Data Association	_D-construct_ --> _D-construct_
Direct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


KWD - Dynamic Library - Edit class hierarchy

Class hierarchy:

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

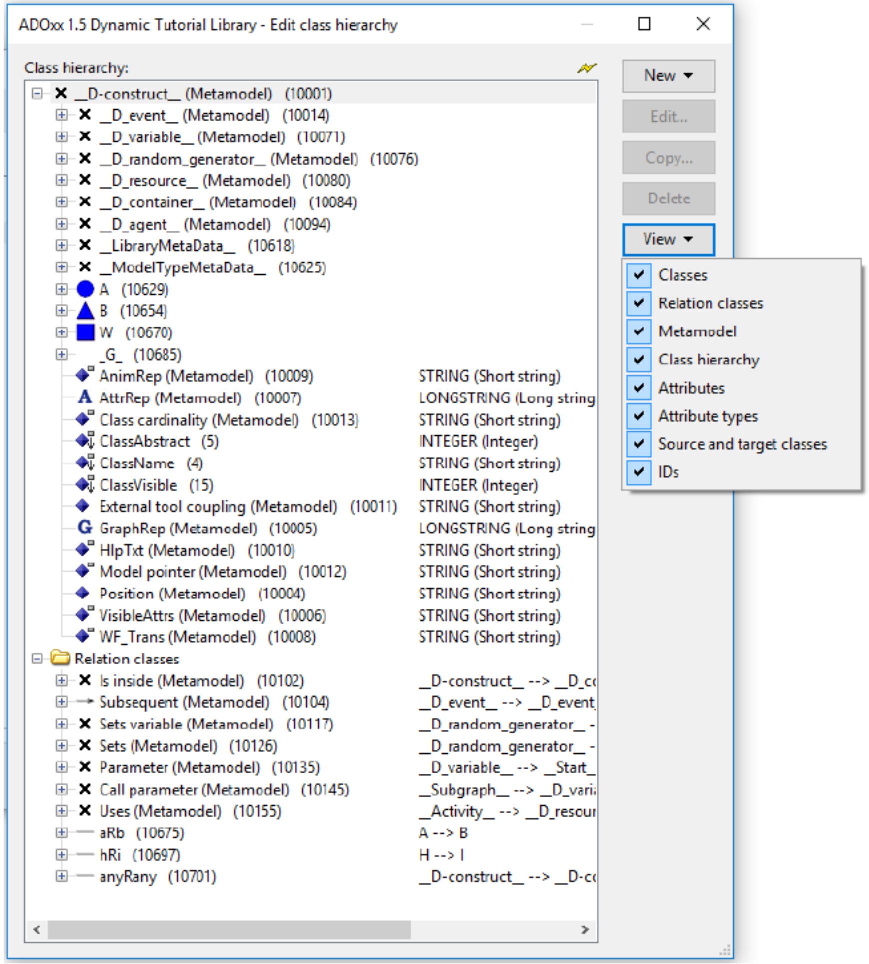
Appearance of Classes in the Modelling Toolkit


Knowledge Work Designer Modelling Toolkit (Admin) - [BPCMN model - new (BPCMN model)]
Model Edit View Process tools Extras Window Help
Modelling
Explorer - Model groups
Models
Navigator
Mo...
In process
BPCMN model - new
Powered by ADOxx
10.05.2020, 11:00:06
Standard
100%

Classes

Relations classes

Views of the Class Hierarchy



ADOxx 1.5 Dynamic Tutorial Library - Edit class hierarchy

Class hierarchy:

```
⊟ ✕ _D-construct_ (Metamodel)  (10001)
   ⊞ ✕ _D_event_ (Metamodel)  (10014)
   ⊞ ✕ _D_variable_ (Metamodel)  (10071)
   ⊞ ✕ _D_random_generator_ (Metamodel)  (10076)
   ⊞ ✕ _D_resource_ (Metamodel)  (10080)
   ⊞ ✕ _D_container_ (Metamodel)  (10084)
   ⊞ ✕ _D_agent_ (Metamodel)  (10094)
   ⊞ ✕ _LibraryMetaData_  (10618)
   ⊞ ✕ _ModelTypeMetaData_  (10625)
   ⊞ ● A  (10629)
   ⊞ ▲ B  (10654)
   ⊞ ■ W  (10670)
   ⊞   _G_  (10685)
     AnimRep (Metamodel)  (10009)                    STRING (Short string)
     AttrRep (Metamodel)  (10007)                    LONGSTRING (Long string
     Class cardinality (Metamodel)  (10013)          STRING (Short string)
     ClassAbstract  (5)                              INTEGER (Integer)
     ClassName  (4)                                  STRING (Short string)
     ClassVisible  (15)                              INTEGER (Integer)
     External tool coupling (Metamodel)  (10011)     STRING (Short string)
     GraphRep (Metamodel)  (10005)                   LONGSTRING (Long string
     HlpTxt (Metamodel)  (10010)                     STRING (Short string)
     Model pointer (Metamodel)  (10012)              STRING (Short string)
     Position (Metamodel)  (10004)                   STRING (Short string)
     VisibleAttrs (Metamodel)  (10006)               STRING (Short string)
     WF_Trans (Metamodel)  (10008)                   STRING (Short string)
⊟ Relation classes
   ⊞ ✕ Is inside (Metamodel)  (10102)                _D-construct_ --> _D_c
   ⊞ → Subsequent (Metamodel)  (10104)               _D_event_ --> _D_event
   ⊞ ✕ Sets variable (Metamodel)  (10117)            _D_random_generator_ -
   ⊞ ✕ Sets (Metamodel)  (10126)                     _D_random_generator_ -
   ⊞ ✕ Parameter (Metamodel)  (10135)                _D_variable_ --> _Start_
   ⊞ ✕ Call parameter (Metamodel)  (10145)           _Subgraph_ --> _D_vari
   ⊞ ✕ Uses (Metamodel)  (10155)                     _Activity_ --> _D_resour
   ⊞ — aRb  (10675)                                  A --> B
   ⊞ — hRi  (10697)                                  H --> I
   ⊞ — anyRany  (10701)                              _D-construct_ --> _D-c
```

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

Classes

All visible classes will be shown

Relation classes

All available relation classes will be shown

Metamodel

All classes will be shown

Class hierarchy

All classes will be shown with their inheritance in a hierarchy

Attributes

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

Attribute types

The type of each attribute will be shown

Source- and Target-classes

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

IDs

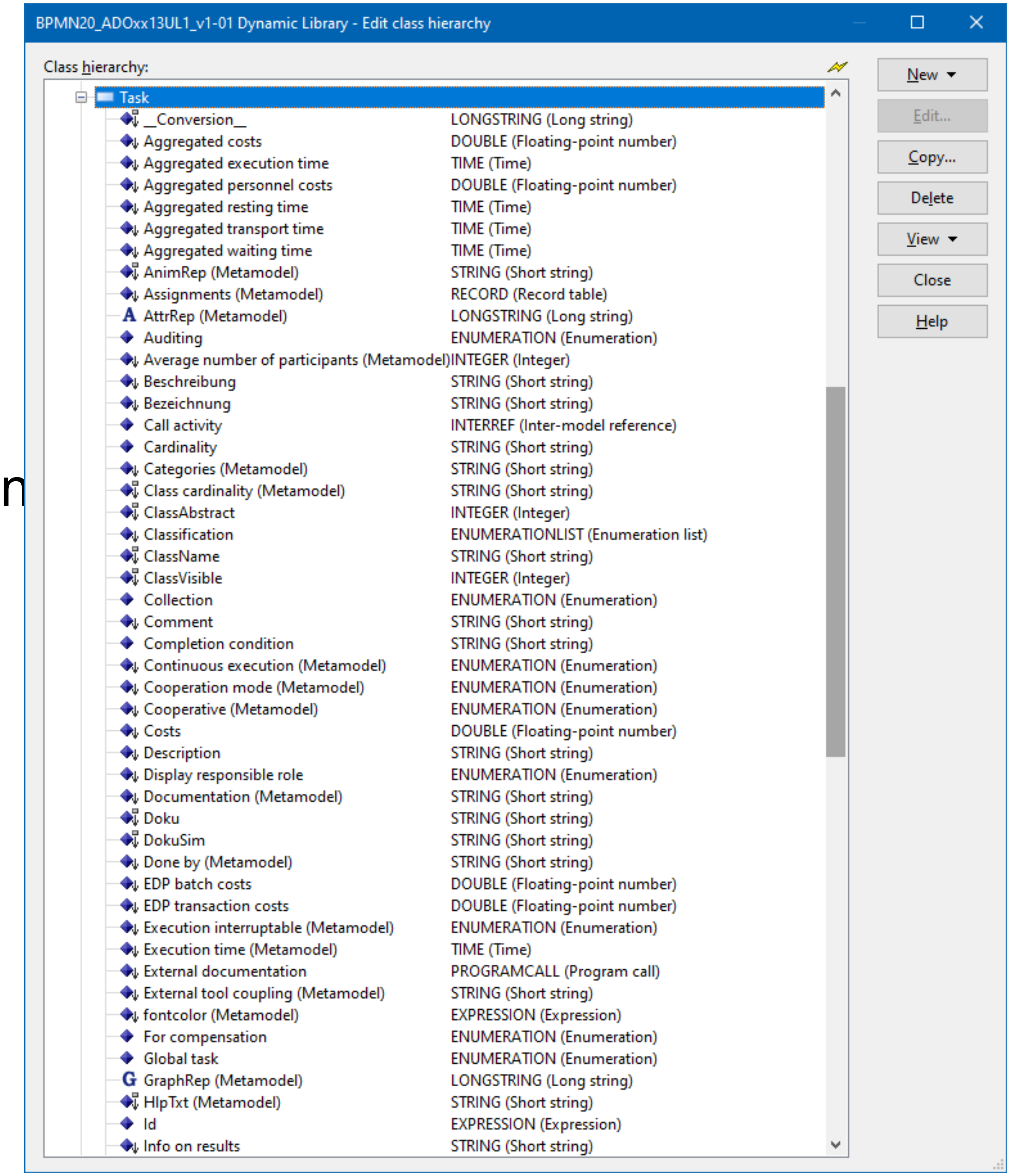
Shows ID numbers of classes and attributes

Icons in Class Hierarchy

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

Attributes

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

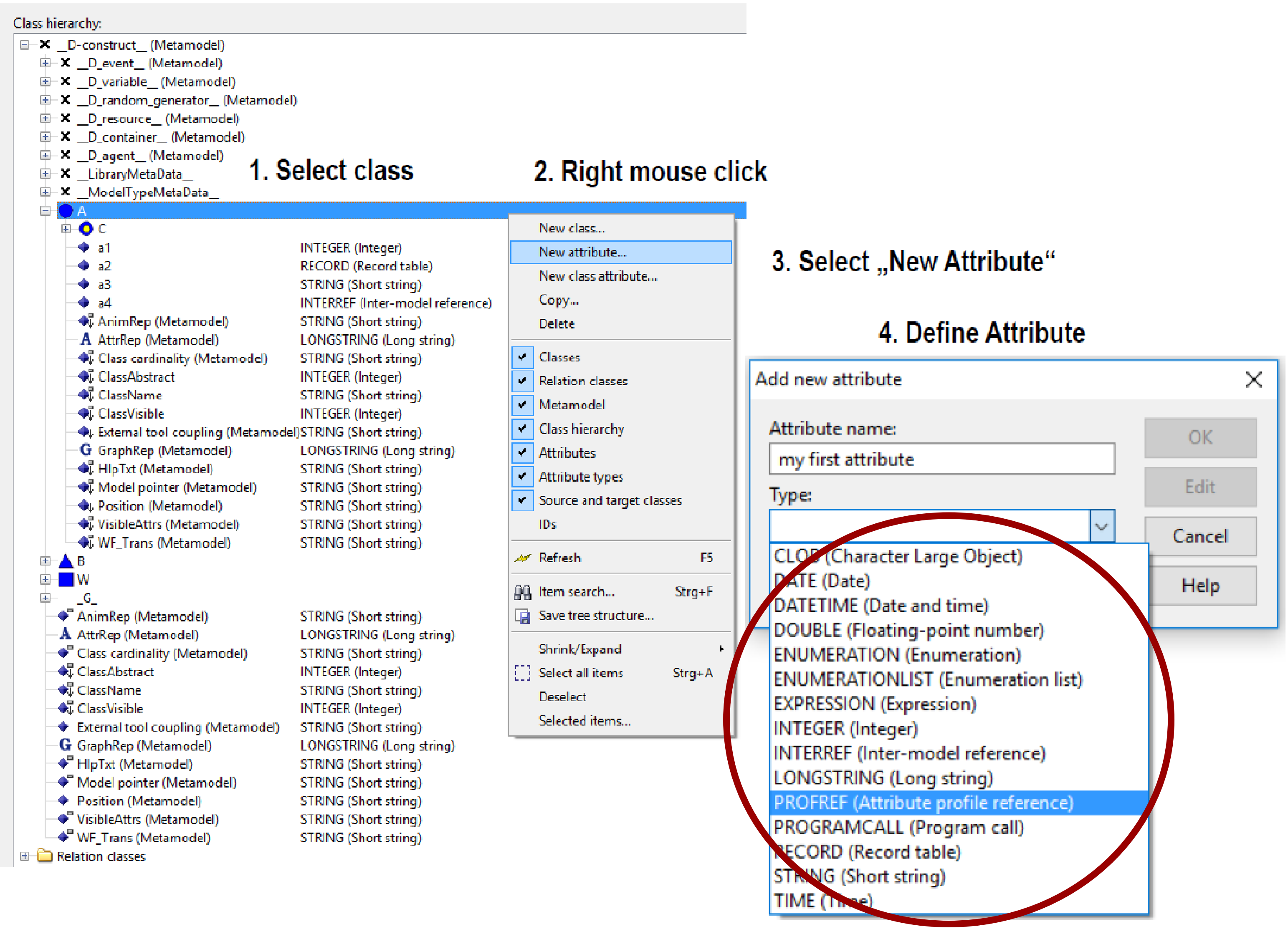


BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Edit class hierarchy

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)

Defining a new Attribute



1. Select class

2. Right mouse click

3. Select „New Attribute“

4. Define Attribute

Class hierarchy:

```
⊟ ✕ _D-construct_ (Metamodel)
  ⊞ ✕ _D_event_ (Metamodel)
  ⊞ ✕ _D_variable_ (Metamodel)
  ⊞ ✕ _D_random_generator_ (Metamodel)
  ⊞ ✕ _D_resource_ (Metamodel)
  ⊞ ✕ _D_container_ (Metamodel)
  ⊞ ✕ _D_agent_ (Metamodel)
  ⊞ ✕ _LibraryMetaData_
  ⊞ ✕ _ModelTypeMetaData_
  ⊟ ● A
      ⊞ C
        a1                                  INTEGER (Integer)
        a2                                  RECORD (Record table)
        a3                                  STRING (Short string)
        a4                                  INTERREF (Inter-model reference)
        AnimRep (Metamodel)                 STRING (Short string)
        A AttrRep (Metamodel)               LONGSTRING (Long string)
        Class cardinality (Metamodel)       STRING (Short string)
        ClassAbstract                       INTEGER (Integer)
        ClassName                           STRING (Short string)
        ClassVisible                        INTEGER (Integer)
        External tool coupling (Metamodel)STRING (Short string)
        G GraphRep (Metamodel)              LONGSTRING (Long string)
        HlpTxt (Metamodel)                  STRING (Short string)
        Model pointer (Metamodel)           STRING (Short string)
        Position (Metamodel)                STRING (Short string)
        VisibleAttrs (Metamodel)            STRING (Short string)
        WF_Trans (Metamodel)                STRING (Short string)
  ⊞ ▲ B
  ⊞ ■ W
  ⊞ _G_
    AnimRep (Metamodel)                     STRING (Short string)
    A AttrRep (Metamodel)                   LONGSTRING (Long string)
    Class cardinality (Metamodel)           STRING (Short string)
    ClassAbstract                           INTEGER (Integer)
    ClassName                               STRING (Short string)
    ClassVisible                            INTEGER (Integer)
    External tool coupling (Metamodel)      STRING (Short string)
    G GraphRep (Metamodel)                  LONGSTRING (Long string)
    HlpTxt (Metamodel)                      STRING (Short string)
    Model pointer (Metamodel)               STRING (Short string)
    Position (Metamodel)                    STRING (Short string)
    VisibleAttrs (Metamodel)                STRING (Short string)
    WF_Trans (Metamodel)                    STRING (Short string)
⊞ Relation classes
```

Add new attribute

Attribute name: my first attribute

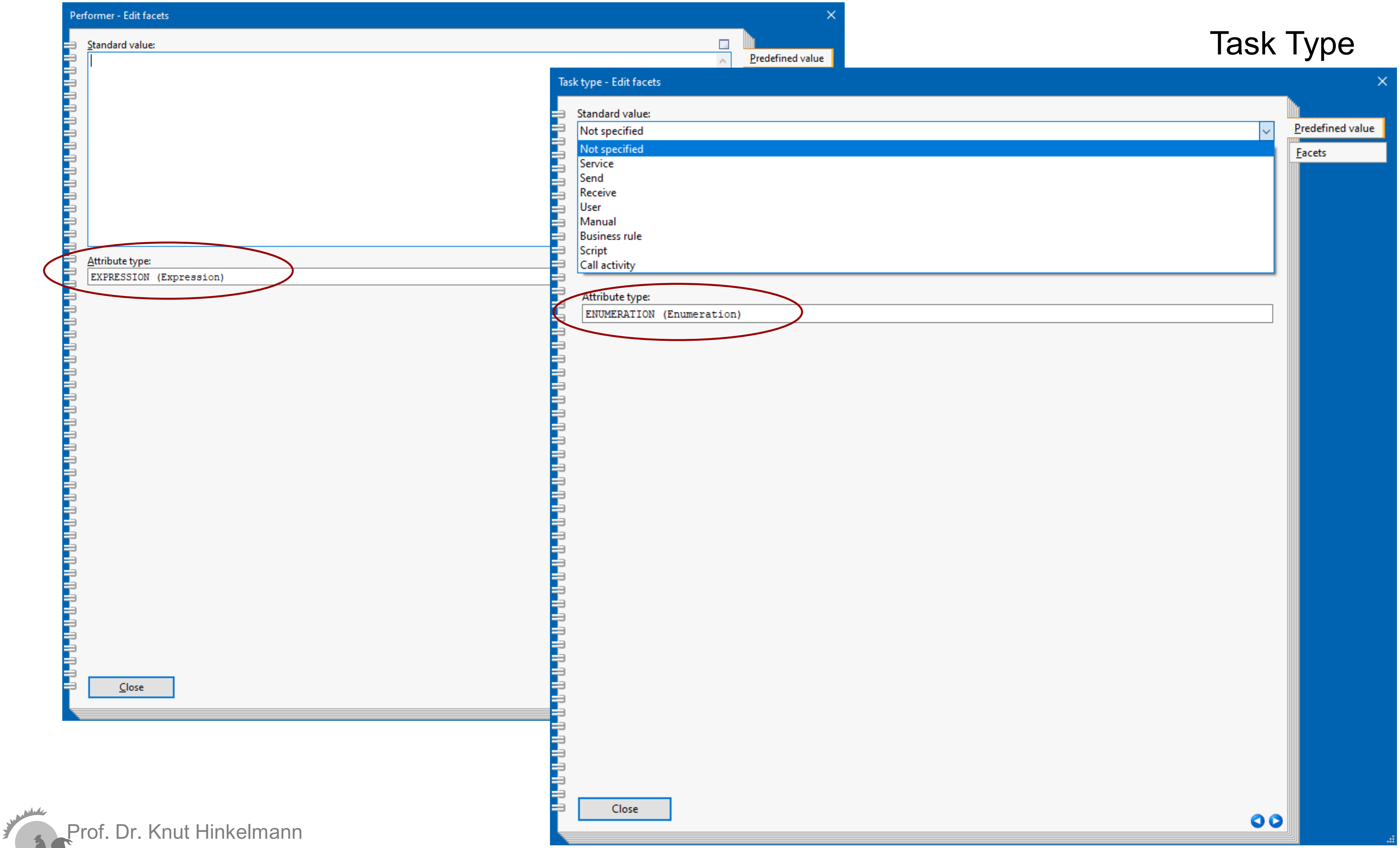
Type:

```
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)
```

Examples of Attributes

Performer

Task Type



Performer - Edit facets

Standard value:

Attribute type:
EXPRESSION (Expression)

Task type - Edit facets

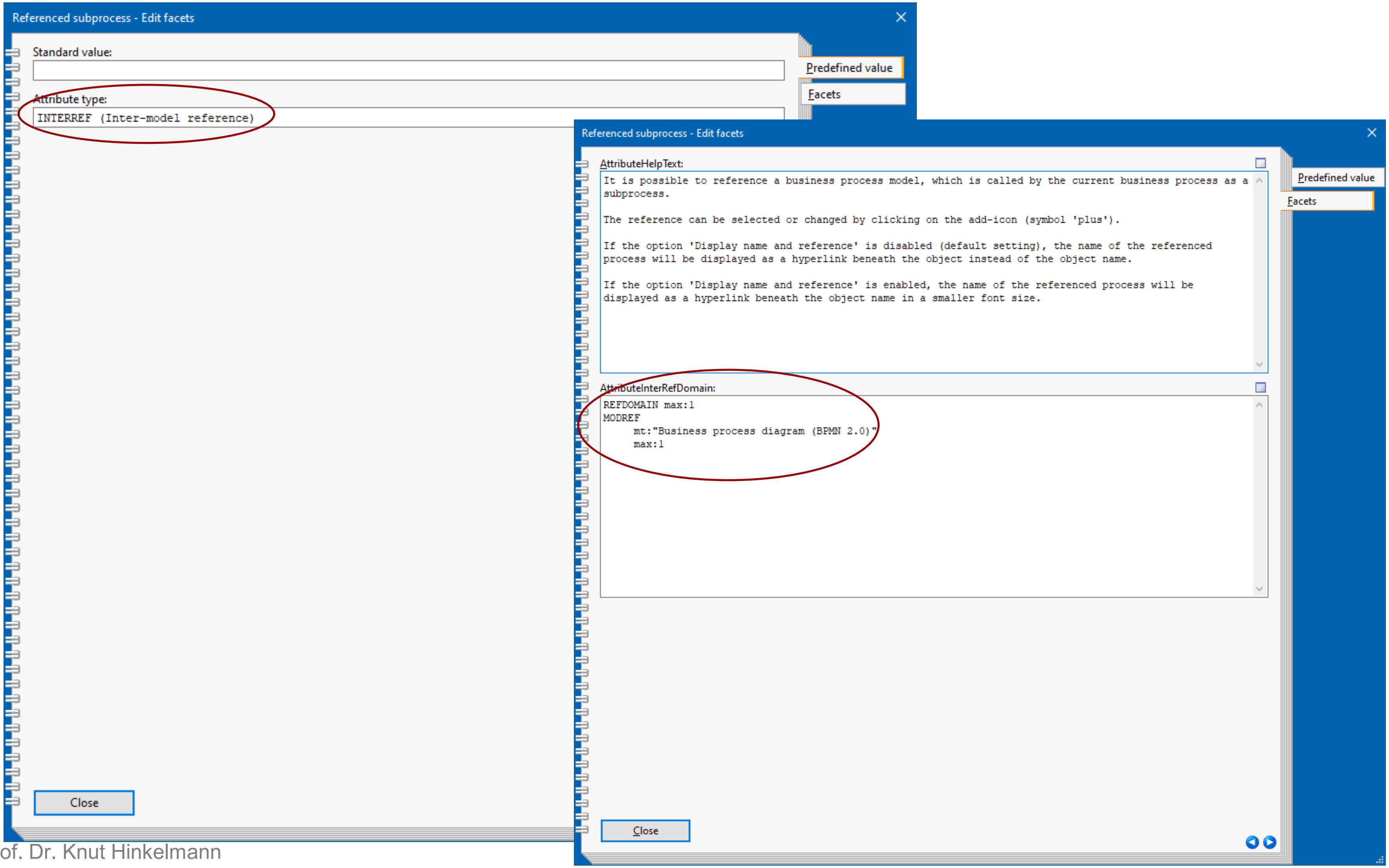
Standard value:
Not specified

- Not specified
- Service
- Send
- Receive
- User
- Manual
- Business rule
- Script
- Call activity

Attribute type:
ENUMERATION (Enumeration)

References

Referencing a Subprocess



Referenced subprocess - Edit facets
Predefined value
Facets
Close

Standard value:

Attribute type:

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

Referenced subprocess - Edit facets
Predefined value
Facets
Close

AttributeHelpText:

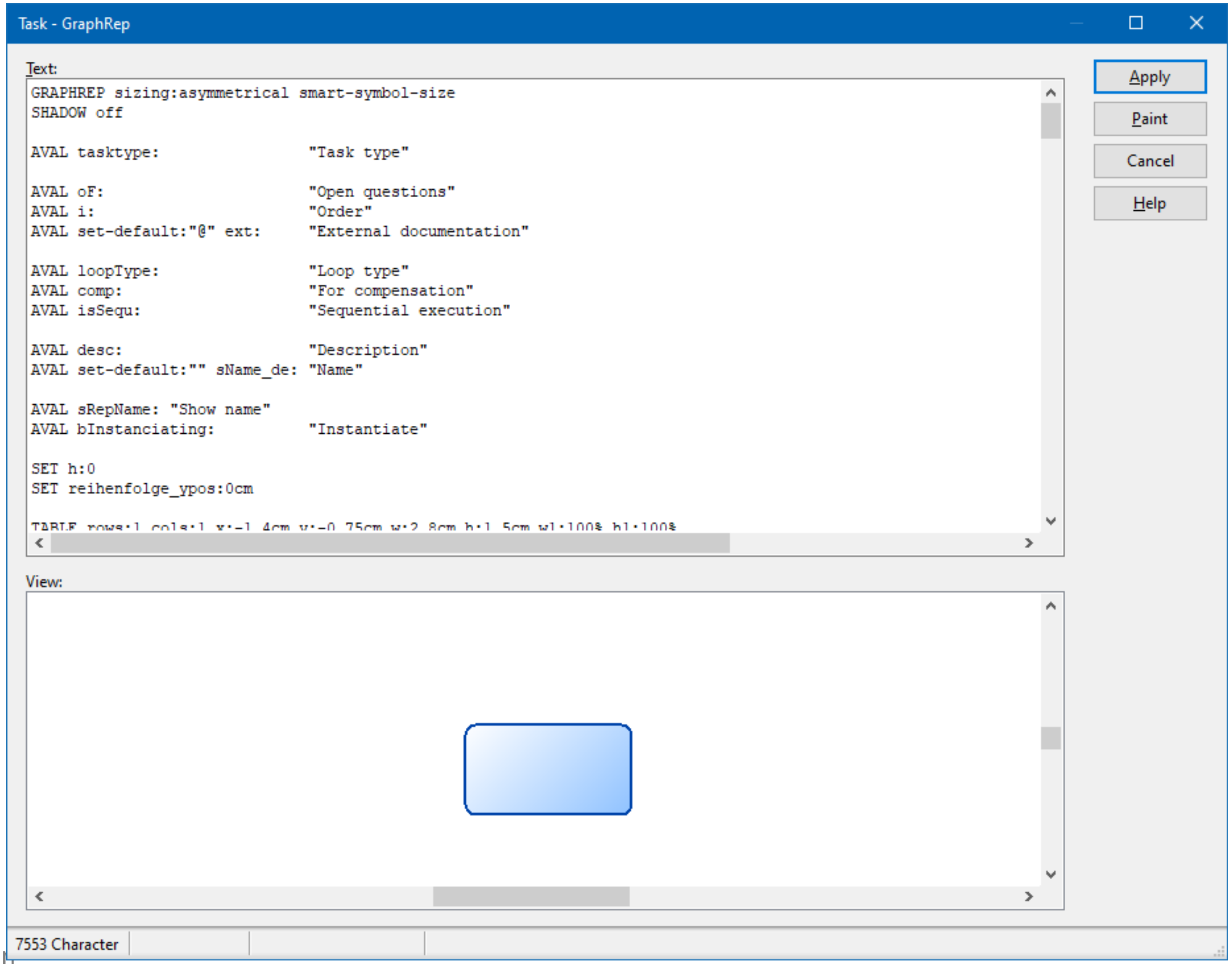
```
It is possible to reference a business process model, which is called by the current business process as a
subprocess.

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

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

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

AttributeInterRefDomain:

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

Special Attribute GraphRep

GraphRep: A script language for the graphical representation

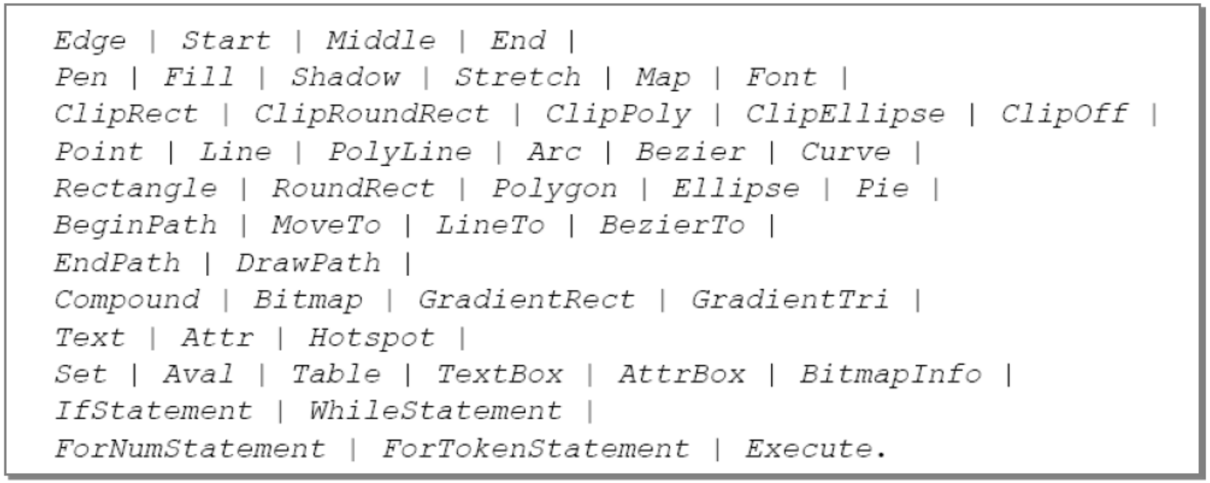


```
GRAPHREP sizing:asymmetrical smart-symbol-size
SHADOW off

AVAL tasktype:              "Task type"

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

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

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

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

SET h:0
SET reihenfolge_ypos:0cm

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

GraphRep Elements

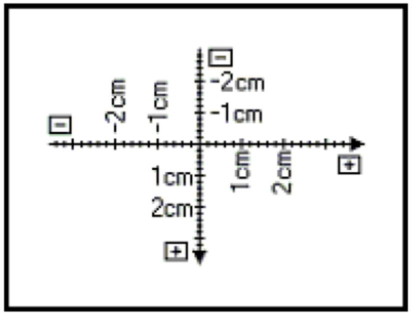
- Types of elements
  - Style elements
  - Shape elements
  - Variable assigning elements
  - Context elements
  - Control elements

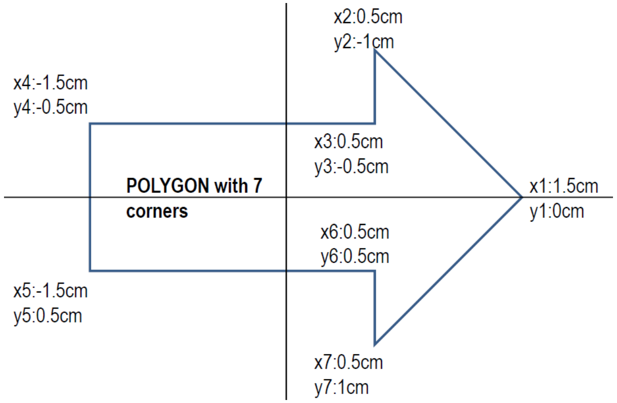
GraphRep Elements



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

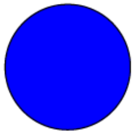
- Elements are placed on x-y-axes





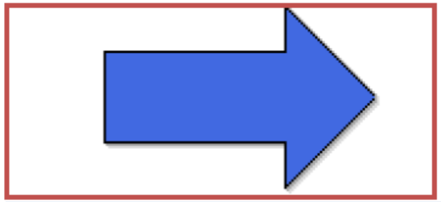
GraphRep Examples

```
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
```

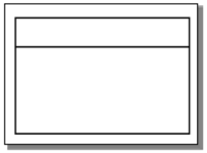
```
ATTR "Name" y:1.4cm w:c h:c
```



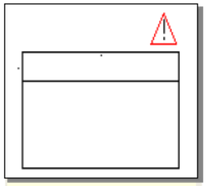
In case attribute name is available, it is shown here

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
```



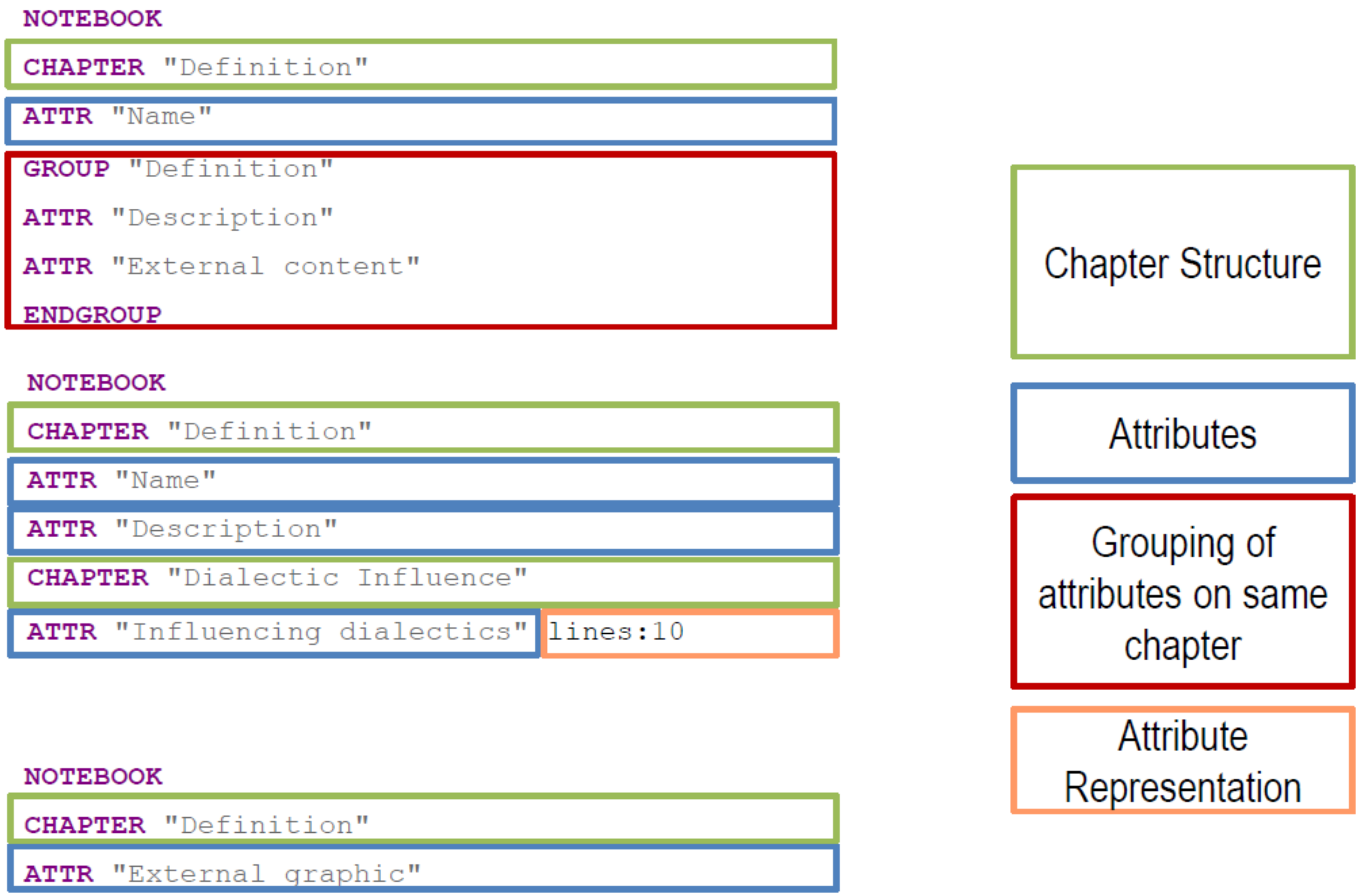
Condition fulfilled



Condition not fulfilled

AttrRep

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



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

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

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

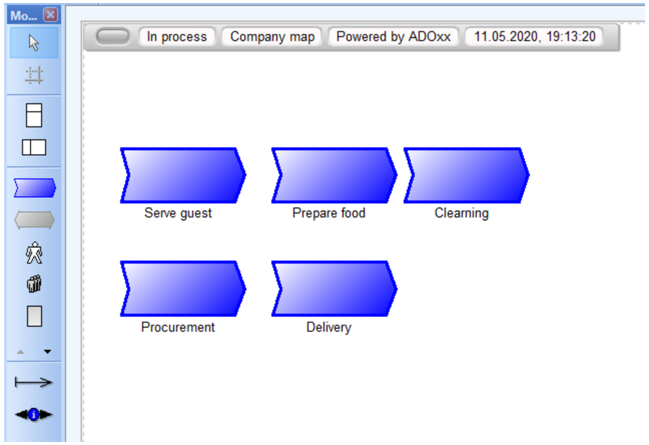
Chapter Structure

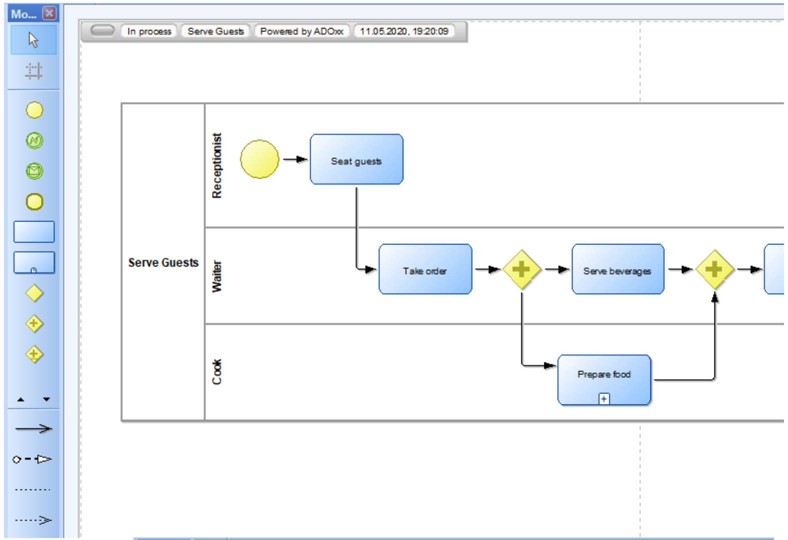
Attributes

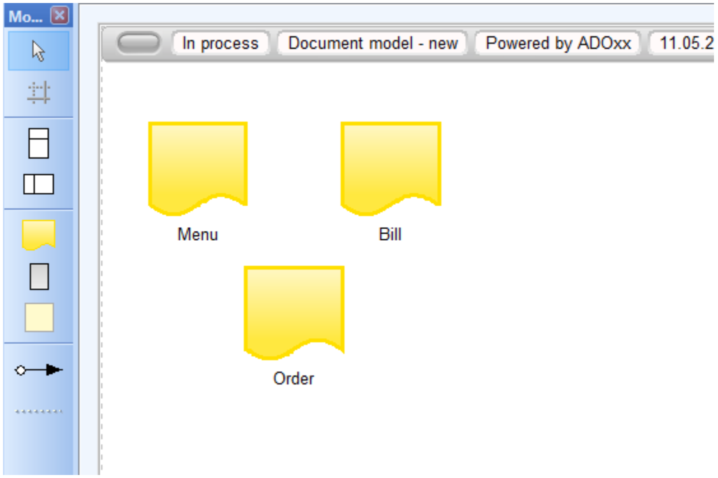
Grouping of attributes on same chapter

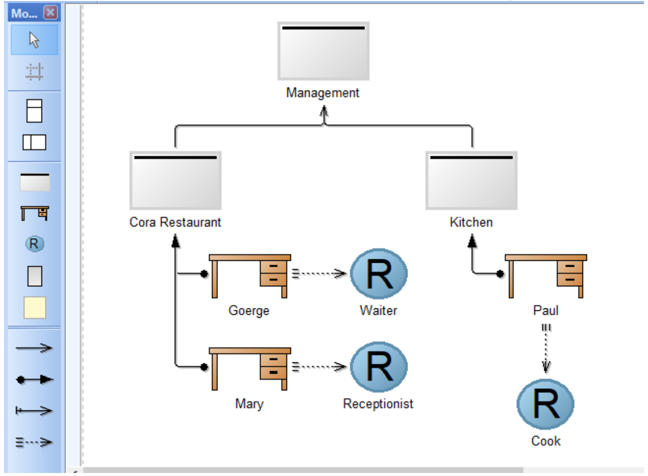
Attribute Representation

Model Types: Represention Views on the Knowledge

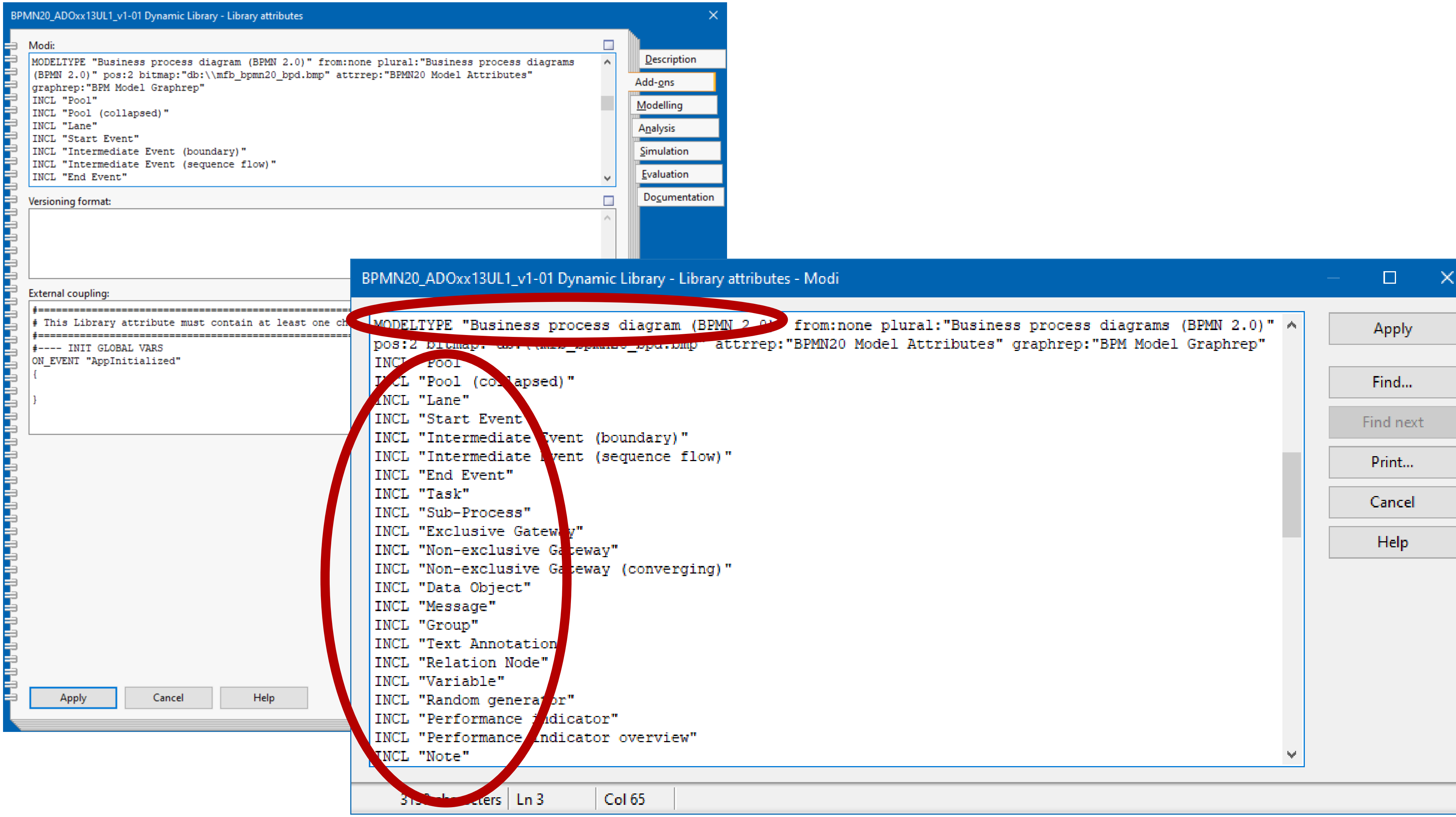








Classes are assigned to Model Types



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

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

}
```

BPMN20_ADOxx13UL1_v1-01 Dynamic Library - Library attributes - Modi
Apply
Find...
Find next
Print...
Cancel
Help

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

31[illegible] characters | Ln 3 | Col 65