Meta-Modeling and Modeling Languages

Prof. Dr. Knut Hinkelmann

Models and Modeling

Model

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

Modeling

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

Result of modeling is a model.

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

Metamodel

Metamodel

is a model of a model. It defines the modeling language, i.e. the constructs that can be used to express models.

Metamodeling

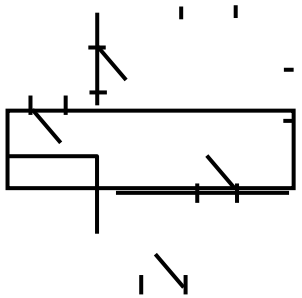
The process of generating metamodels

Model and Meta-Model in Architecture

real object

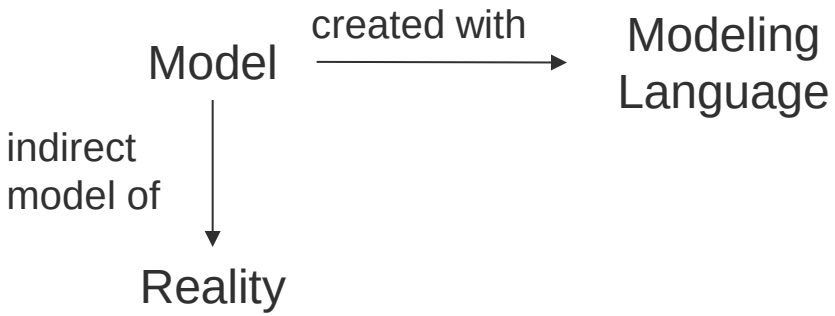
house

model



architect's drawing
(plan)

Modeling Language



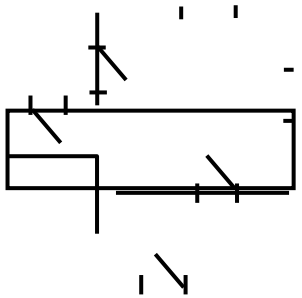
- A modeling "language" specifies the building blocks (elements) from which a model can be made.
- There can be different types of modeling languages, depending on the kind of model
  - graphical model
  - textual description
  - mathematical model
  - conceptual model
  - physical model

Model and Meta-Model in Architecture

real object

house

model

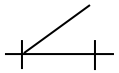


architect's drawing
(plan)

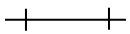
modeling language
(concrete syntax)

object types:

wall

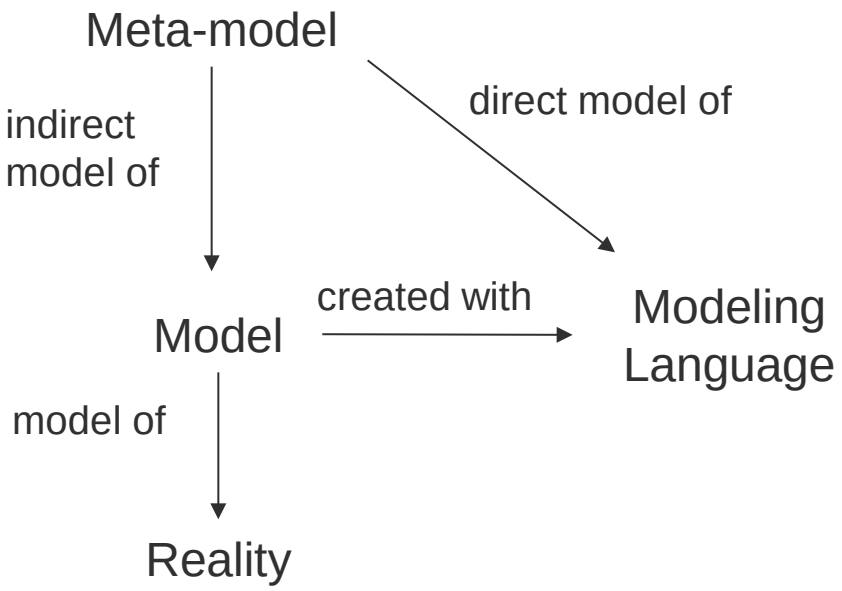


door



window

Meta-model



A meta-model defines the modeling 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

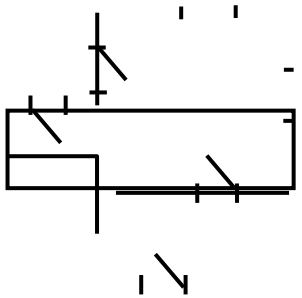
- The meta-model is the abstract syntax, the modeling language is the concrete syntax.

Model and Meta-Model in Architecture

real object

house

model

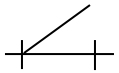


architect's drawing (plan)

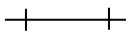
modeling language
(concrete syntax)

object types:

wall



door



window

meta-model
(abstract syntax)

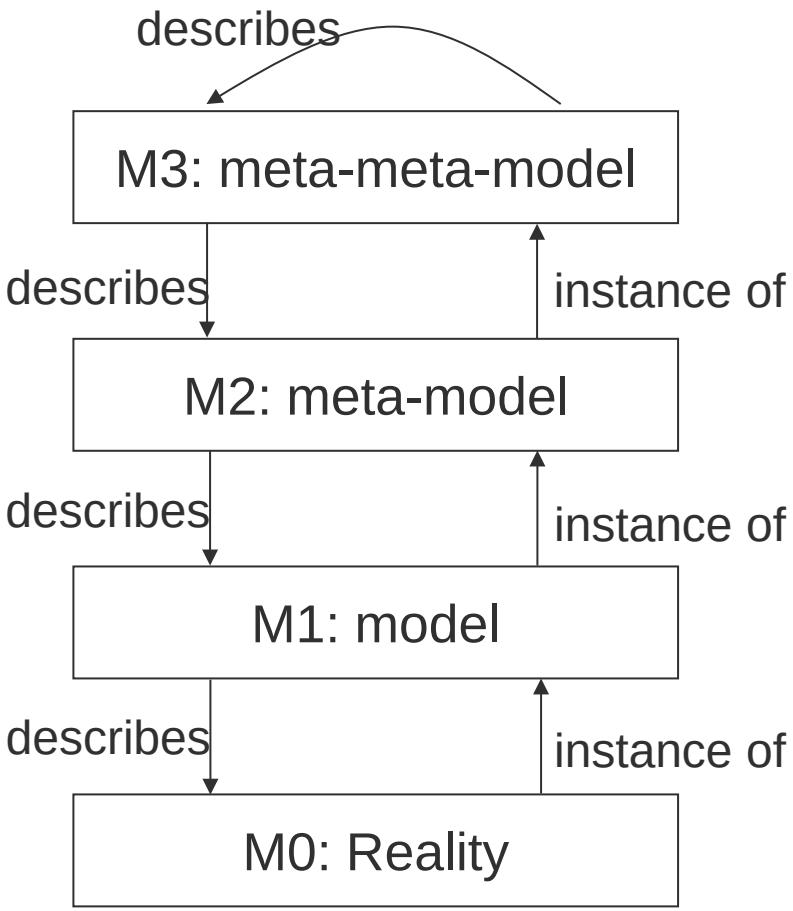
object types:

- wall
- door
- window

rules:

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

The Model Stack



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

Metamodel and Modeling Language

Metamodel

- The *metamodel* defines the modeling elements (concepts, relations, constraints) without specifying the layout and notation
- The *metamodel* corresponds to the *abstract syntax*

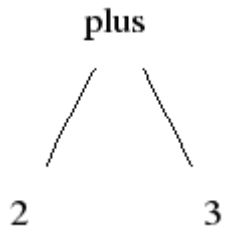
Modeling language

- The *modeling language* defines the notation/appearance of the modeling elements
- The *modeling language* corresponds to the *concrete syntax*

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

Illustration: Meta-model and Model for Processes

Metamodel:

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

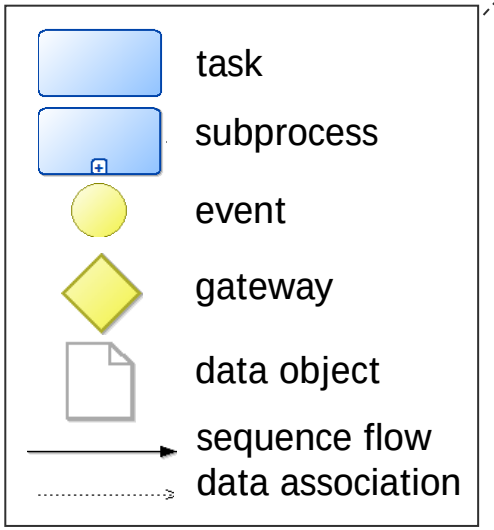
Example: A process model consists of concepts for

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

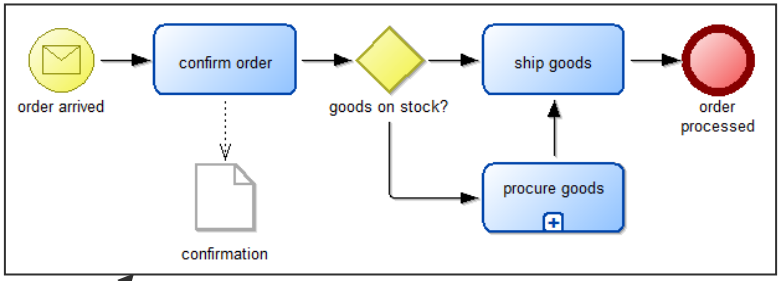
The elements have attributes and there are rules how the elements can be combined.

Modeling Language:

Concrete syntax:
Notation/appearance of meta-model elements



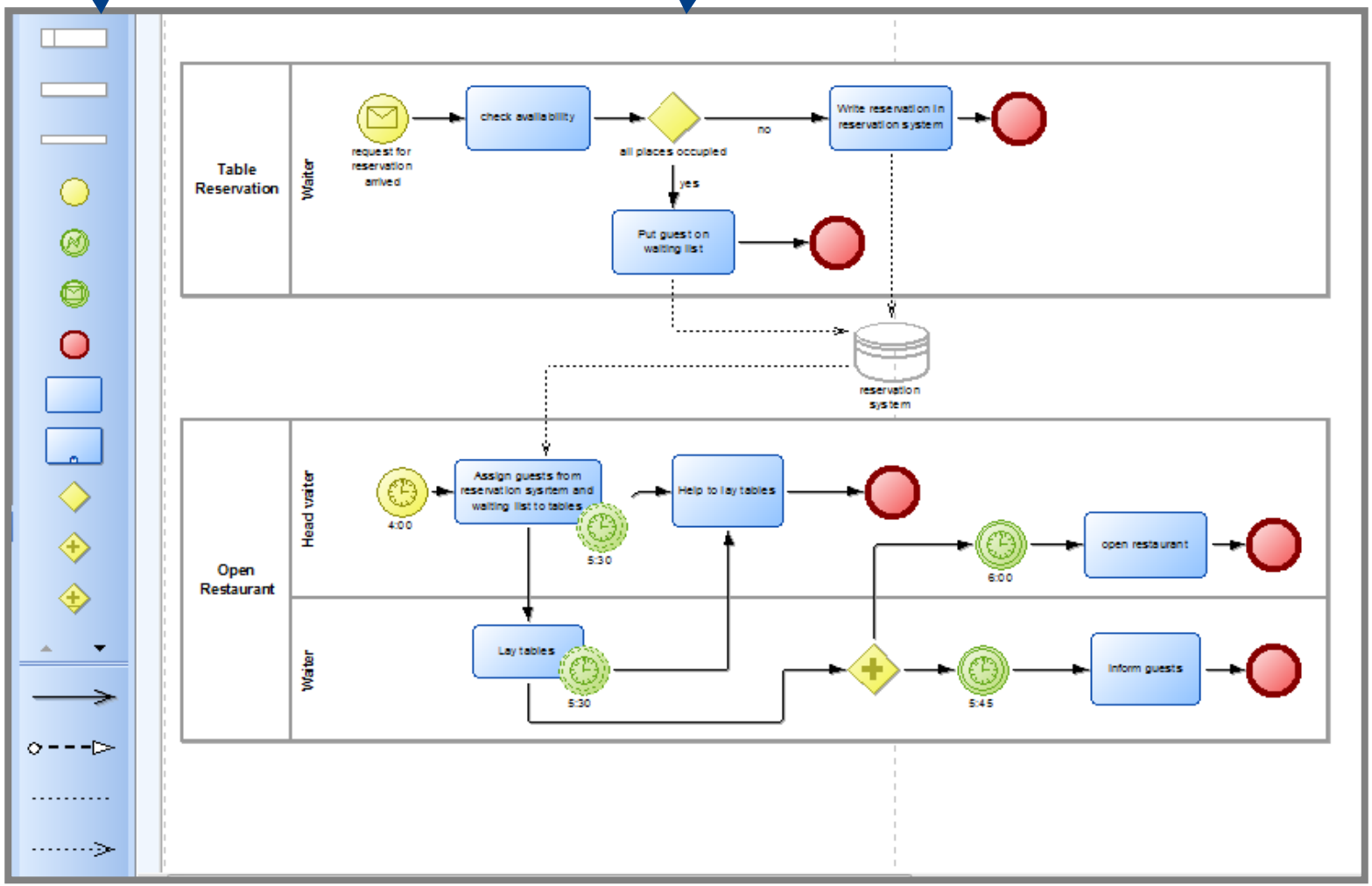
Model:



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

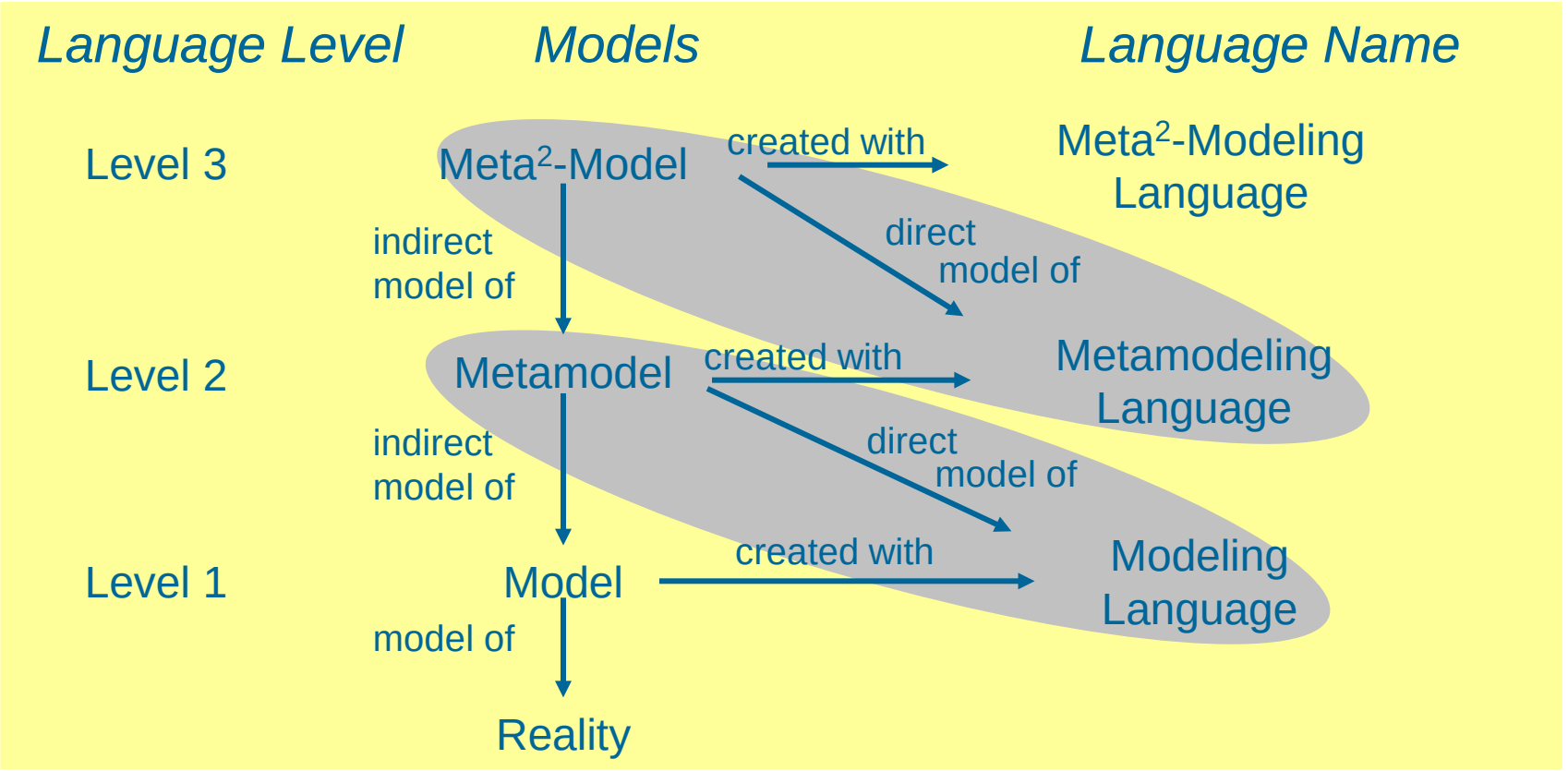
Modeling Language

Model



Meta Model Hierarchy

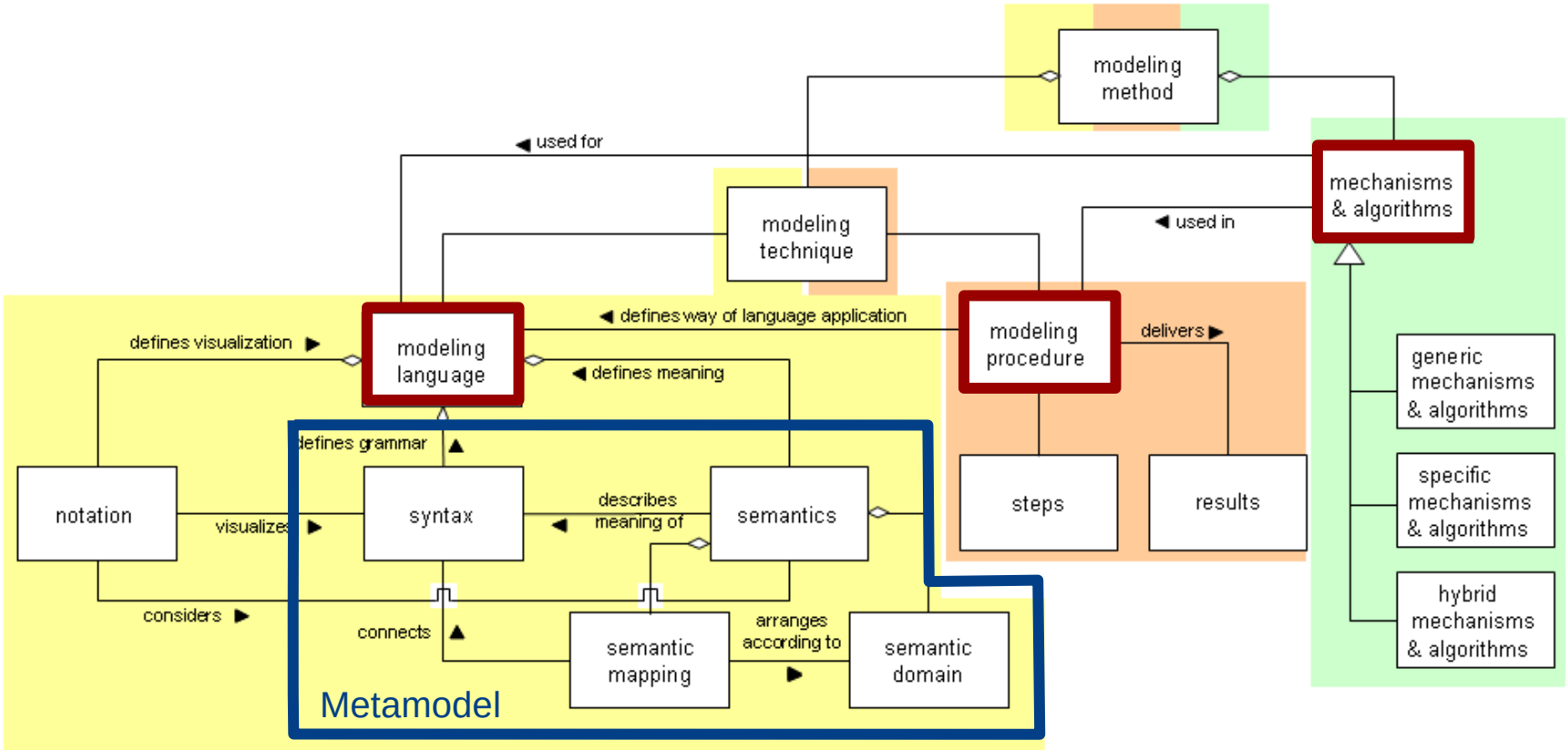
The meta-model must again be described in some language, which has to be specified in a meta-meta-model



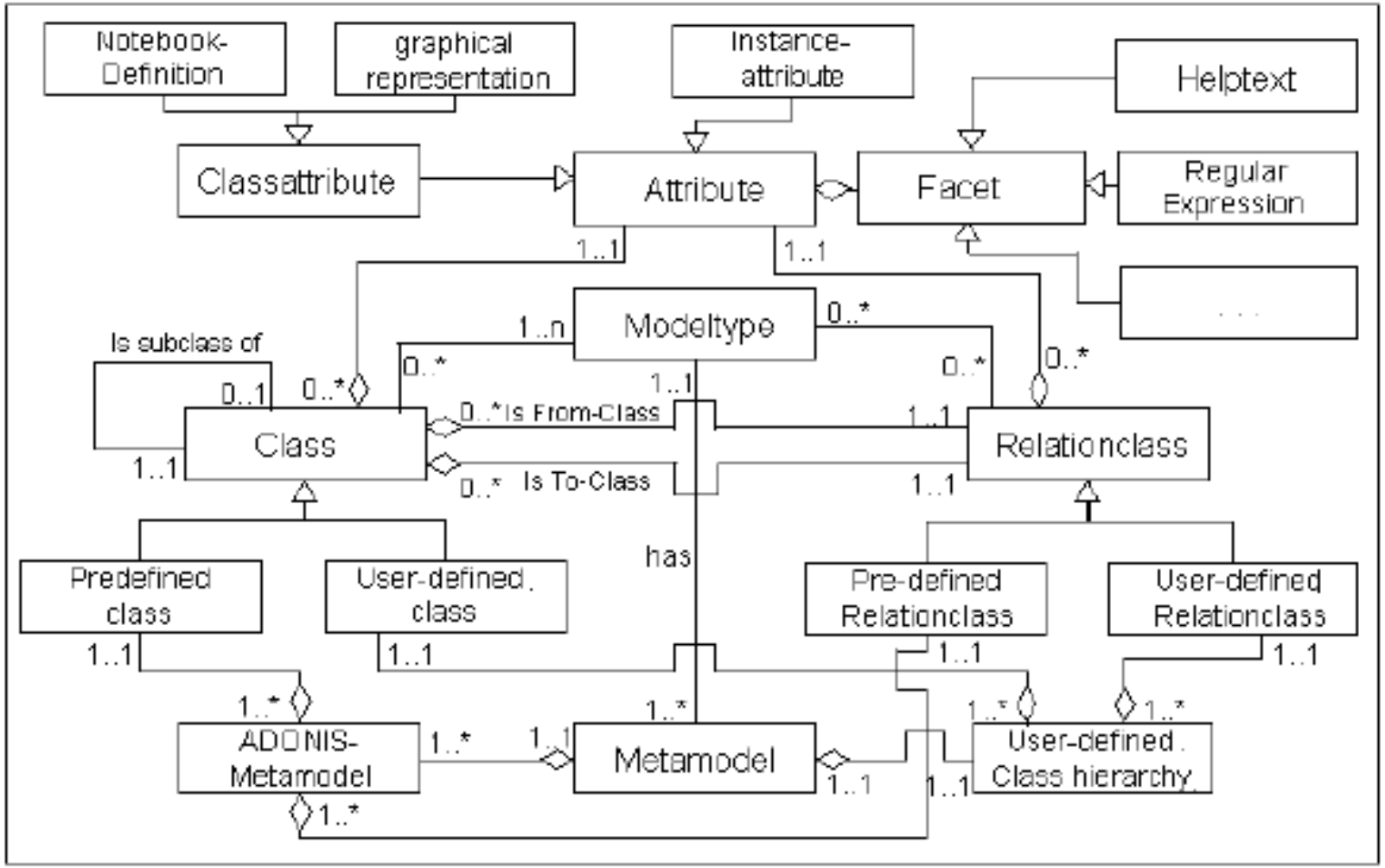
Karagiannis, D. & Kühn, H., 2002. Metamodeling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer, eds. *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.

Components of Modeling Methods

- A Modeling Language is part of a Modeling Method
- A Modeling Language consists of the Metamodel (Syntax and Semantics) and the Notation



ADONIS Meta-Meta Model



Domain-specific vs. Generic Modeling Languages

- **Domain-specific modelling languages** are notations which are defined to model knowledge about a specific domain

- **Generic modeling languages** can be used to represent any kind of knowledge

Domain-specific Modeling Languages

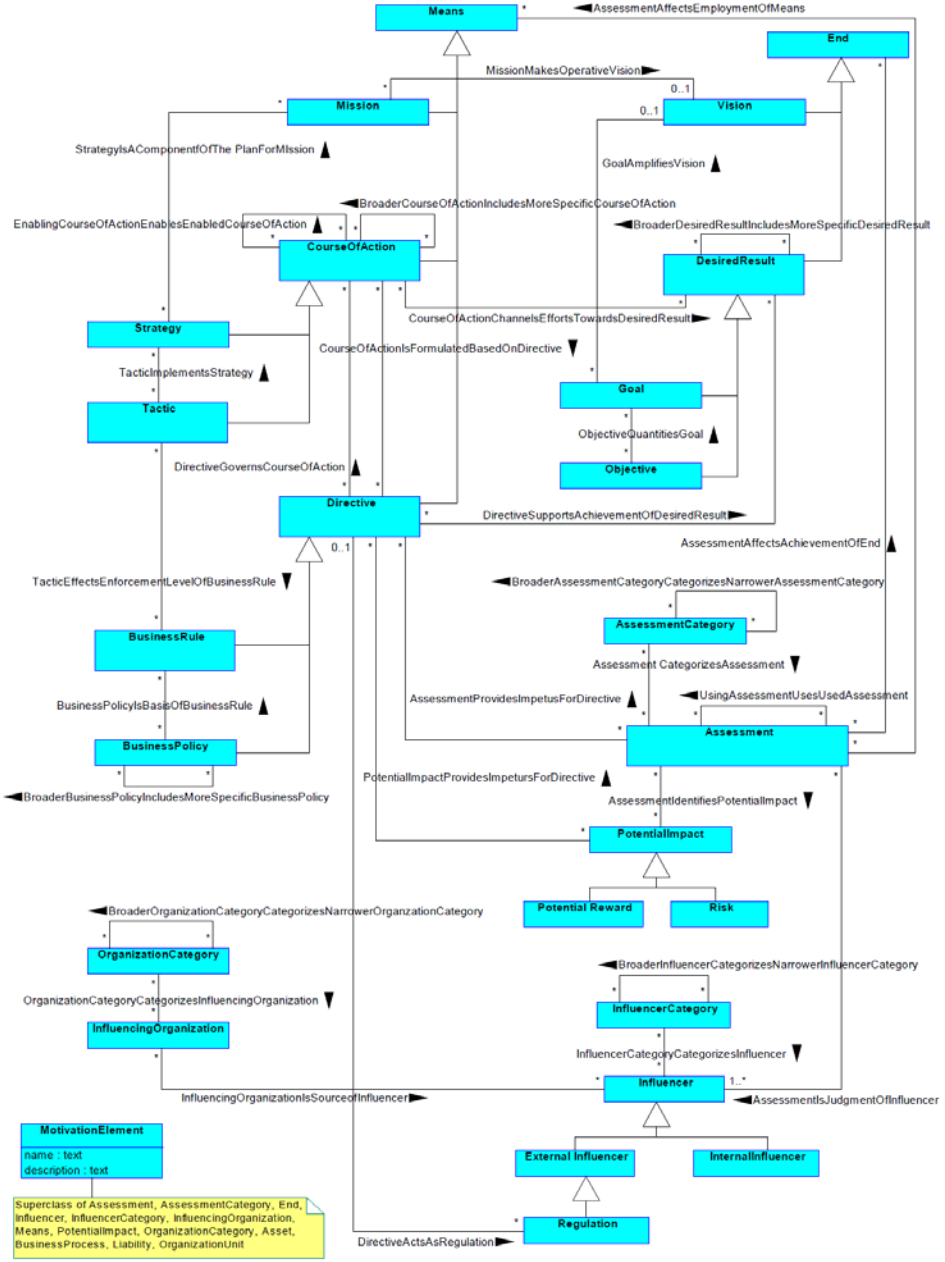
- Domain-specific modeling languages have modeling elements for typical concepts and relations of a domain of discourse
  - ♦ Predefined classes, associations and constraints
  - ♦ Specific shapes for modeling elements and relations
- Modeling means to create instances of theses classes and relations

Domain-specific Modeling Languages

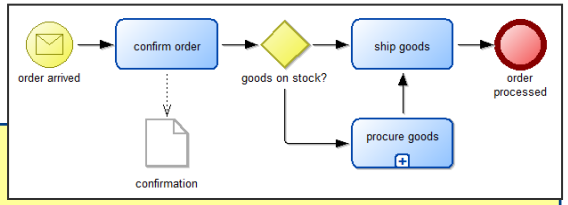
- Domain-specific modeling languages correspond to *model kinds* which have modeling elements for concepts and relations to represent specific *views*
- Examples of domain-specific modeling languages:
  - **BPMN** is a domain-specific language for business processes
    - Modeling elements: task, event, gateway, ….
    - relations: sequence flow, message flow, data association, …
  - **ArchiMate** is a domain-specific language for enterprise architectures
    - Modeling elements: process, actor, role, business object, …
    - relations: uses, realizes, …
  - **BMM** is a domain-specific language for business motivation
    - Modeling elements: vision, mission, goal, strategy, influencer, …
    - relations: judges, channels efforts, …

Meta-Meta Model: Modeling a Meta-Model

- OMG uses (UML) Class Diagrams as Meta-Modeling language
- Example: Business Motivation Meta-Model



A Business Process Model and Metamodel



Modeling Language

Flow object

event

activity

gateway

data object

subclass of

subclass of

subclass of

subclass of

start event

end event

task

subprocess

instance of

instance of

instance of

instance of

instance of

instance of

instance of

Model

confirmation

order arrived

order processed

confirm order

ship goods

procure goods

goods on stock?

triggers

has name

output

"Confirm order"

Metamodels can be defined as Class Diagrams

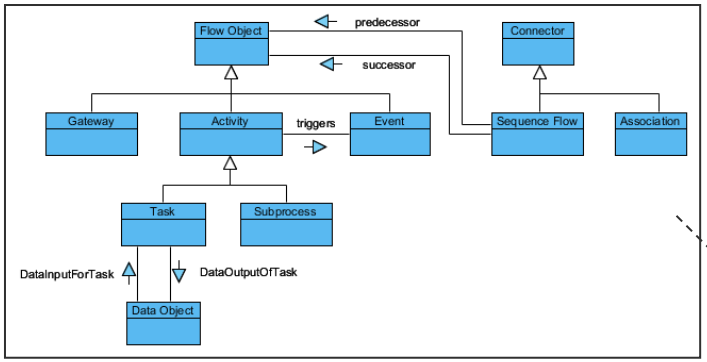
To model a metamodel one can use (a subset of) UML class diagrams
This class diagrams contains the elements of the the sample model

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

A Domain-specific Metamodel for Processes

Meta-model:

- Classes and relations that can be used for modeling
- Abstract syntax and semantics



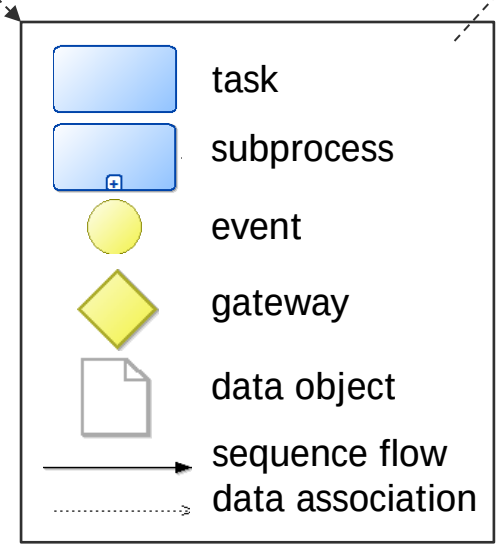
Example: A process model consists of object types for

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

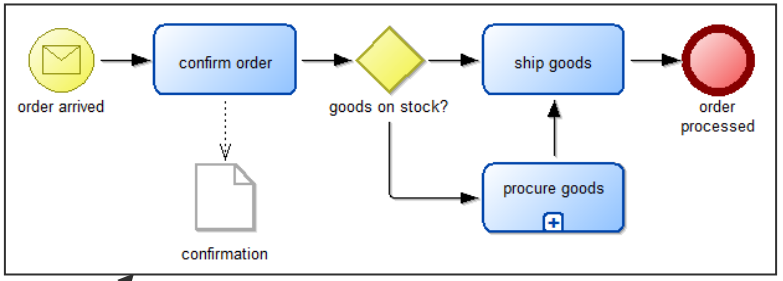
The elements have attributes and there are rules how the elements can be combined.

Modeling Language:

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

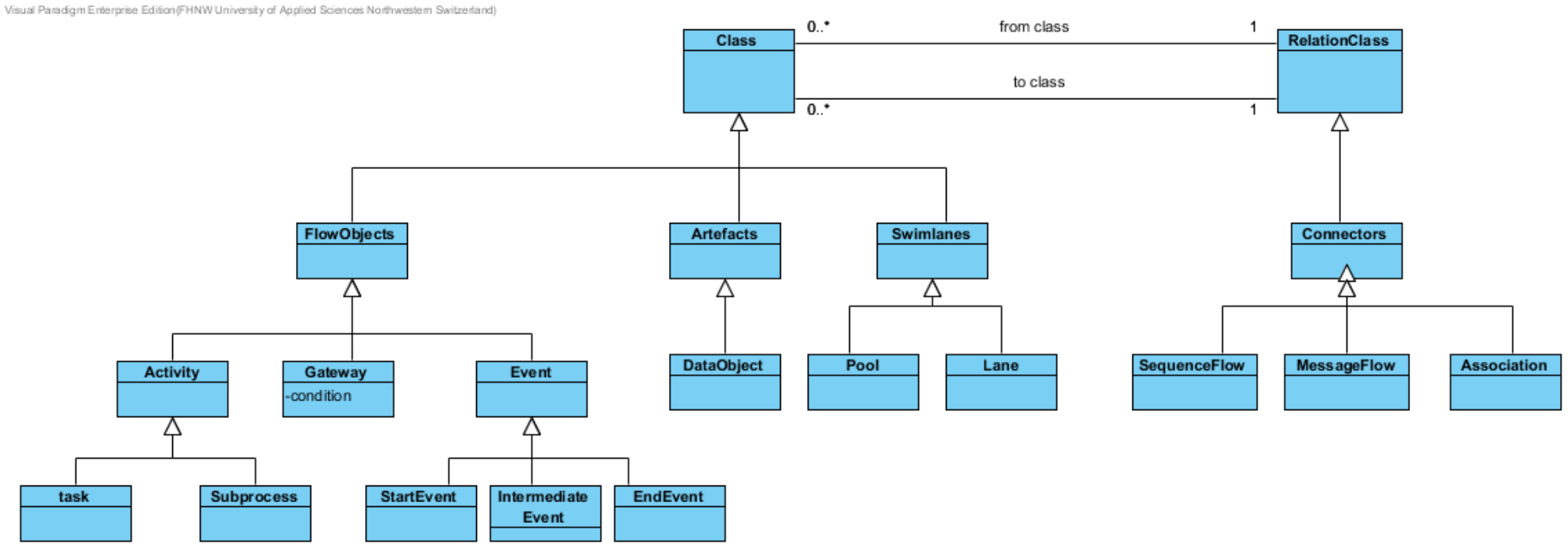


Model:

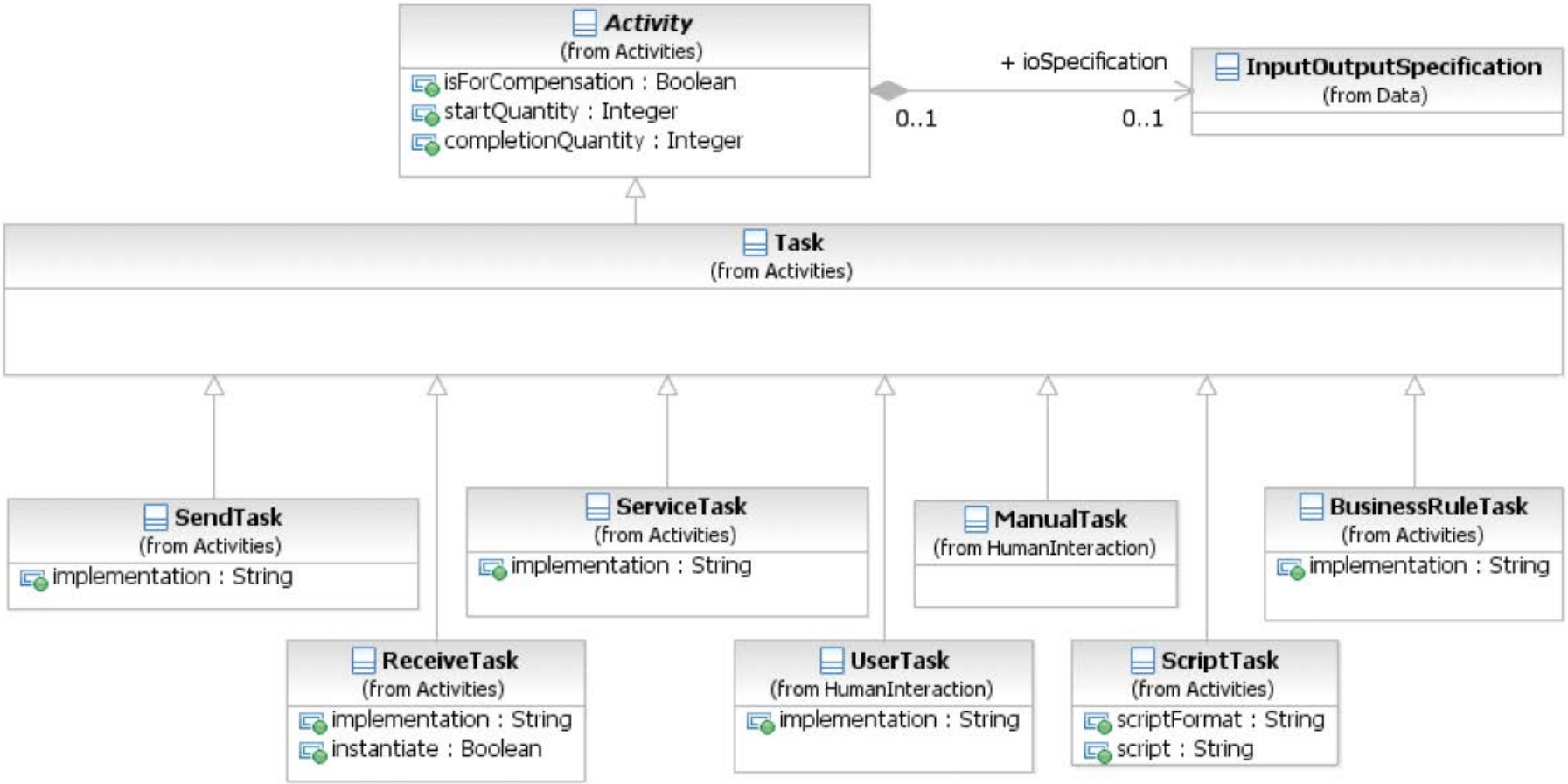


A model contains instances of the object types defined in the meta-model, according to the concrete syntax of the modeling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task"

More Details of the BPMN Metamodel



Subset of the BPMN Metamodel in UML



Source: BPMN 2.0 specification

Strengths and Weaknesses of Domain Modeling Languages

- Strength
  - ♦ Reuse and comprehensiblity of models is possible
    - standard model elements (classes)
  - ♦ Easy to understand
    - Element-specific shapes
- Weaknesses
  - ♦ Restricted to a specific domain
    - Only what can be expressed with the modeling elements can be modeled

Generic Modeling Languages

- Generic modeling languages can be used to represent any kind of knowledge
- They can be used, if no domain-specific modeling language is available (for a view)
- There are a wide range of generic modeling languages
  - Natural language allows to express any knowledge
  - Formal languages: Typically a subset of Logic
  - Graphical Diagrams

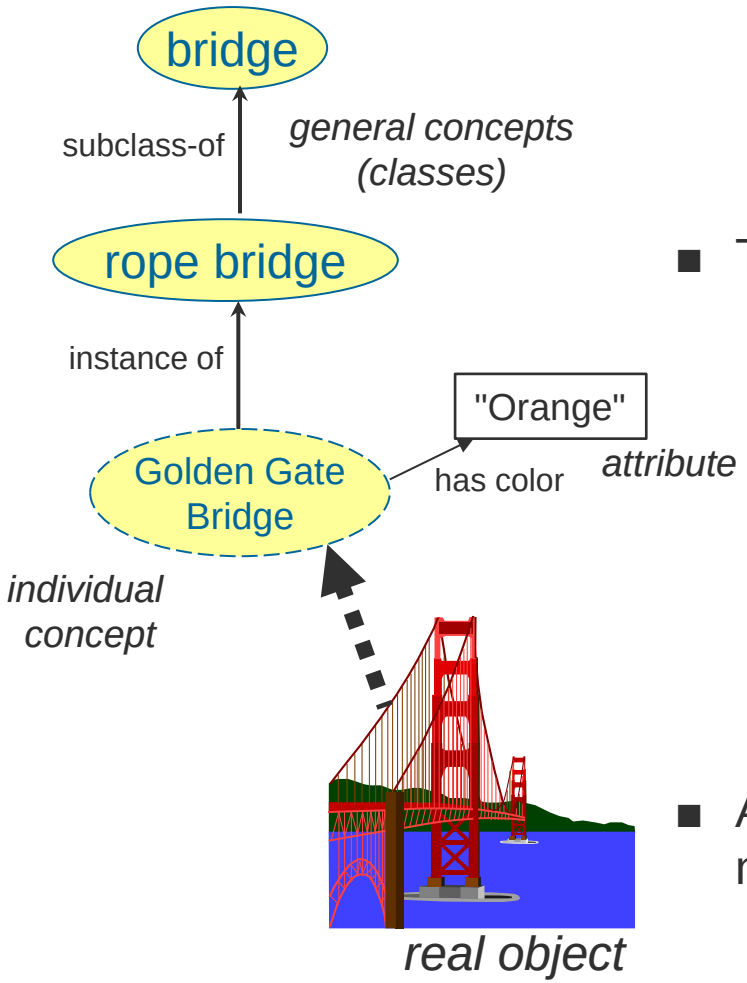
Graphical Diagrams for Generic Modeling

- Generic graphical modeling languages have been developed in a many difference fields:
  - Artificial Intelligence: Semantic networks, Description Logics
  - Data Modeling: Entity Relationship Diagrams
  - Object-Oriented Programming: UML Class Diagrams
- Although having different notations these generic modeling languages typically allow to represent
  - **concepts** (general and individual)
  - **properties** of concepts
  - **relations** between concepts
  - **constraints** (e.g. cardinalities, type restrictions)

The Metamodel for a Generic Modeling Language

- The metamodel for ageneric modeling language has only few modeling elements
  - class
  - association
  - attribute
  - object
- This can be modelled with Class Diagrams, e.g.
  - (a subset of) UML Class Diagrams
  - Ontology Languages
- Modeling means to
  - define classes
  - create instances of these classes

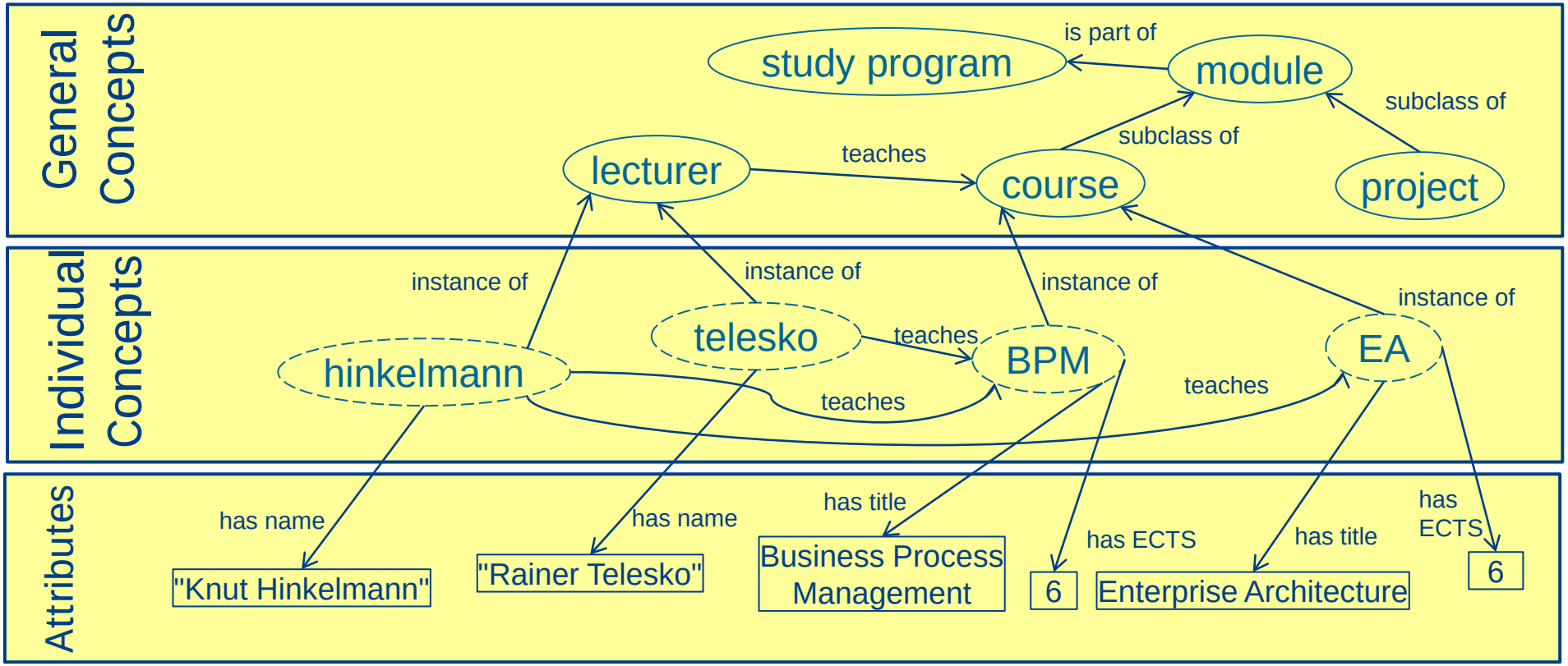
Concepts and Relations



- There are two kinds of concepts:
  - **general concepts** (also called classes)
  - **individual concepts** (also called objects, individuals or instances)
- There are different kinds of relationships
  - **generalisation** ("is a")
    - between classes **(subclass of)**
    - between individual and class **(instance of)**
  - **aggregation and composition**
    - "part-of" relationship
  - **associations**
    - any other kind of relationship
- Attributes can be regarded as associations whose value is not node but is of a primitive type (number, string).

Modeling with a Generic Modeling Language

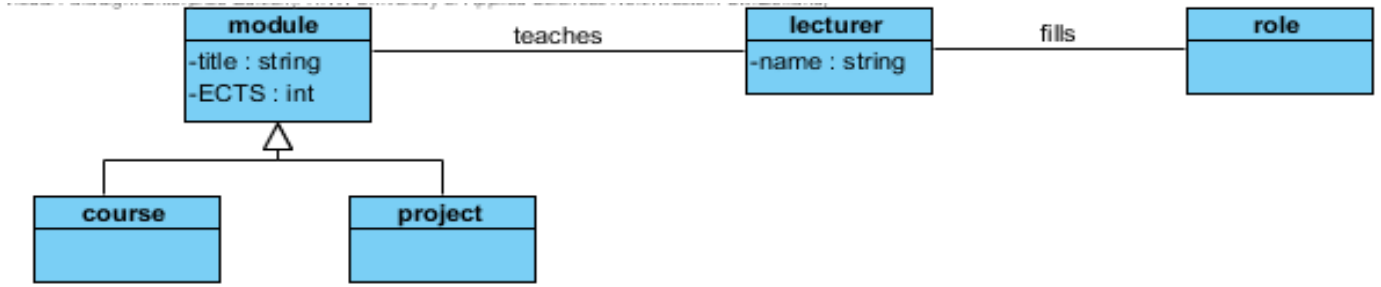
With a generic metamodel, knowledge of any domain can be modeled.
This is a model for modules of a study program.



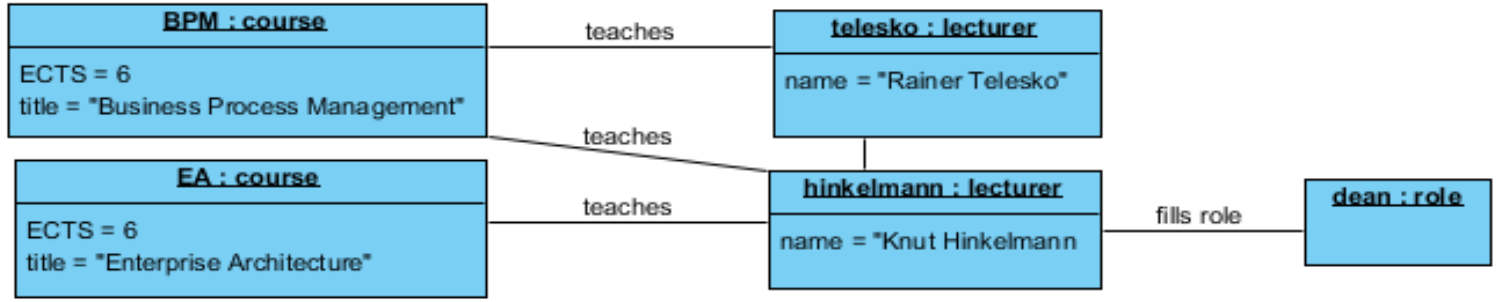
The Semantic Network modeled in UML

The metamodel for this generic modeling language corresponds to subsets of UML Class Diagrams and UML Object Diagrams

Classes



Objects



The classes specify a (new) domain-specific metamodel – In this case for modeling modules of a study program

Disadvantage: No specific modeling shapes

Modeling of Enterprise Architectures

- EA Frameworks provide a structure for the EA description
- The stakeholders and their concerns as well as the goals of the enterprise determine what should be in the EA description
- Based an that the metamodels are defined/select:
  - If available choose domain-specific metamodels/modeling languages
  - If there are no domain-specific modeling languages (in your tool) for some elements, use a generic modeling language (e.g. UML class diagrams)