Meta-Modeling and Modeling Languages

Prof. Dr. Knut Hinkelmann
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 language.
Result of modelling is a model.
Model in Architecture

real object

model

house

architect’s drawing
(plan)
Model and Modeling Language in Architecture

real object

model

modeling language (concrete syntax)

object types:

- wall
- door
- window

house

architect’s drawing (plan)
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
- physical model
Prof. Dr. Knut Hinkelmann

Metamodel and Modeling Languages
Model and Meta-Model in Architecture

real object

house

model

architect’s drawing (plan)

modeling language (concrete syntax)

object types:
- wall
- door
- window

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

meta-model (abstract syntax)

object types:
- wall
- door
- window
Metamodel and Modeling Language

Metamodel

- The *metamodel* is a model of a model. It defines the modeling elements (concepts, relations, constraints) without specifying the layout and notation.

Modeling language

- The *modeling language* defines the notation/appearance of the modeling elements.
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.

Model:

Modeling 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 modeling language. The object „confirm order“ represents a real entity; it is an instance of the object type «task».
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
Metamodels can be defined as Class Diagrams

To model a metamodel one can use (a subset of) UML class diagrams

(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

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"
Subset of the BPMN Metamodel in UML

Source: BPMN 2.0 specification
Meta Model Hierarchy

The meta-model must again be described in some language, which has to be specified in a meta-meta-model.
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.
Domain-specific vs. General-purpose Modeling Languages

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

- General-purpose 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.

- 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, …
Degree of Domain-Specificness

- BPMN is a domain-specific modeling language for business processes
- It would be possible to make BPMN more domain-specific for business processes in a specific application area, e.g.
  - Education: specific tasks for teaching like lecture, self-study, exam with predefined roles for lecturers and students
  - Health: specific tasks for diagnosis, therapy with roles like physician and patient
Strengths and Weaknesses of Domain-specific Modeling Languages

■ Strengths

♦ Comprehensibility of models
  ● elements and relations are adequate for stakeholders
  ● domain-specific shapes

♦ Reuse of models
  ● domain-language can be standardized (e.g. BPMN, ArchiMate)

■ Weaknesses

♦ Restricted to a specific domain
  ● Only what can be expressed with the modeling 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 have two options
  1. Define a new domain-specific modelling language
     - meta model
     - modeling language
  2. Use a general-purpose modeling language
General-purpose Modeling Languages

- General-purpose modeling languages can be used to represent any kind of knowledge

- There are a wide range of general-purpose modeling languages
  - Natural language allows to express any knowledge
  - Formal languages: Typically a subset of Logic
  - Graphical Diagrams

- General-purpose graphical modeling languages have been developed in a many difference fields:
  - Artificial Intelligence: Semantic networks, Ontologies
  - Data Modeling: Entity Relationship Diagrams
  - Object-Oriented Programming: UML Class Diagrams
The Metamodel for a General-purpose Modeling Language

- The metamodel for a general-purpose modeling language has only few modeling elements
  - Class
  - Attribute
  - Association
  - 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
Modeling with a General-purpose Modeling Language

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
Customizing Modeling Languages in Visual Paradigm

- In the Visual Paradigm tool we can use stereotypes to specialize UML class diagrams.
- Stereotypes can be defined and added to any model element.
- We can define a new stereotype for a class and
  - change color
  - add an icon
- Example: stereotypes for modules and lecturer