MDA and Application Development– a Vision

Agenda

1. UML – What is it and what is it good for ?
2. MDA – What is it and what is it good for ?
3. MDA – Sample Process
4. QVT – A View into Transformation

MDA and Application Development

Agenda

1. UML – What is it and what is it good for ?
2. MDA – What is it and what is it good for ?
3. MDA – Sample Process
4. QVT – A View into Transformation

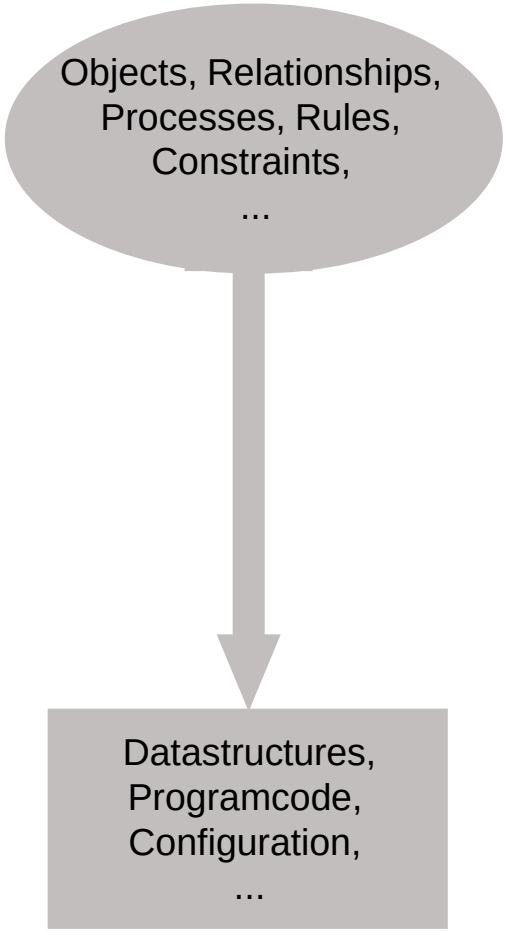
UML - Motivation

UML – what it is

- UML is a language, it is not a method
- UML is a specification of the OMG → so it may be/is a standard
- UML is unified → it contains many concepts
  to describe many aspects of the real world in models
- UML is a language based on MOF → it is fully MOF compliant
- UML and OCL belong together → defining constraints is part of modeling

UML - Motivation

UML – what it is good for

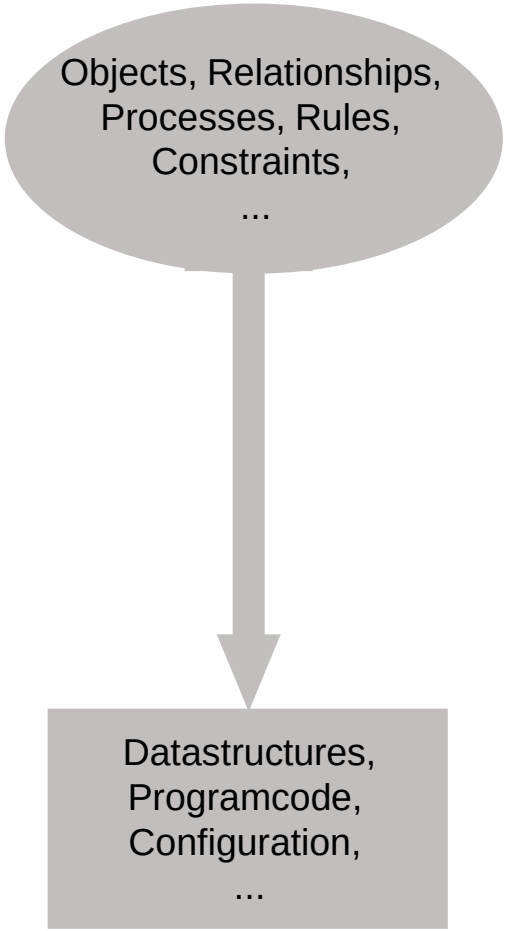


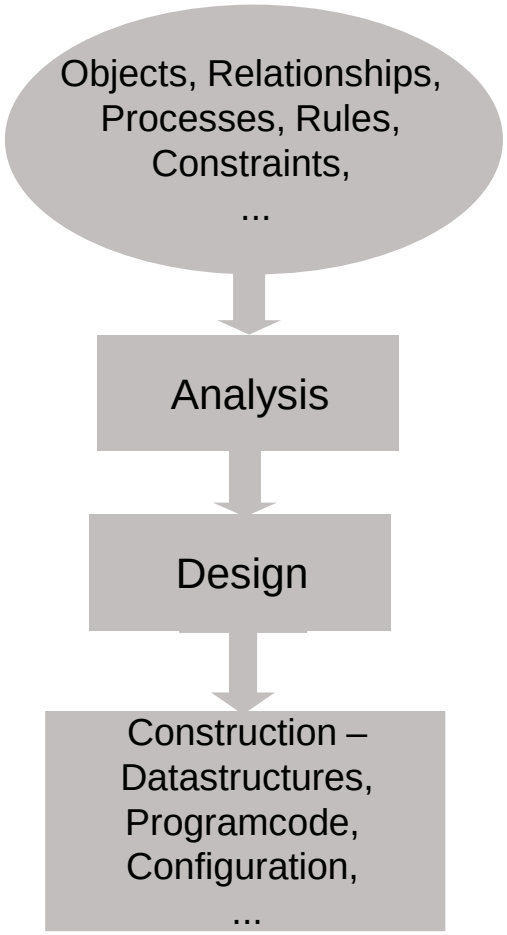
The Real World
Natural Languages,
Domain Languages,
Specific Modeling
Languages

The Computer World
Implementation
Languages,
Databases,
Application Server, …

UML - Motivation

UML – what it is good for



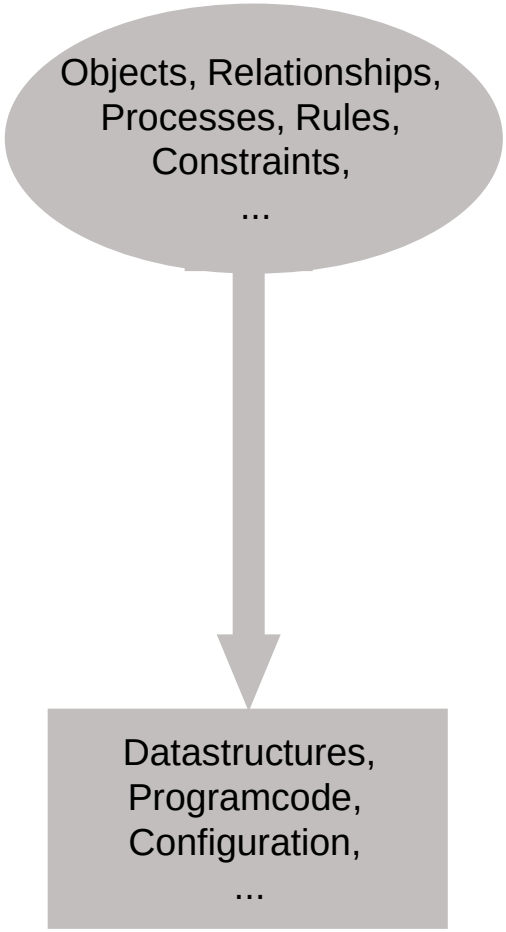


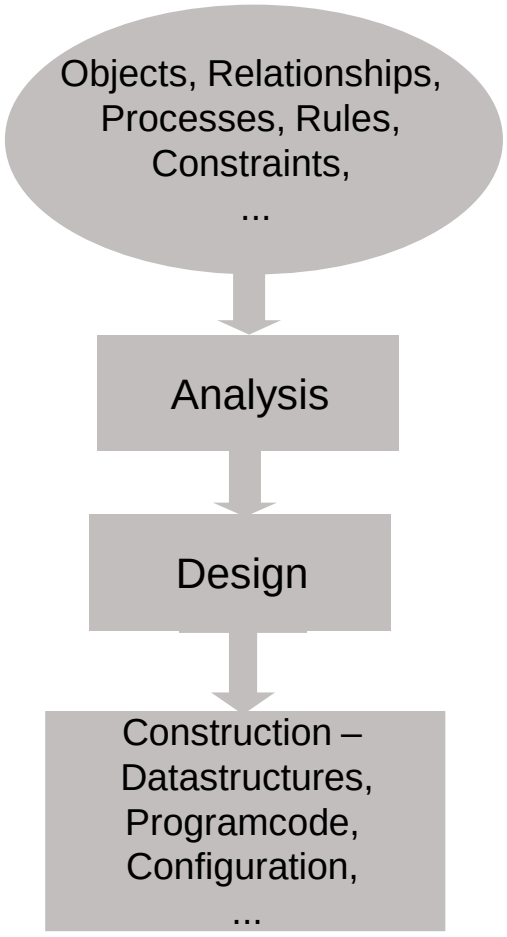
The Real World
Natural Languages,
Domain Languages,
Specific Modeling
Languages

The Computer World
Implementation
Languages,
Databases,
Application Server, …

UML - Motivation

UML – what it is good for





The Real World
Natural Languages,
Domain Languages,
Specific Modeling
Languages

UML (/ BPMN)

UML

The Computer World
Implementation
Languages,
Databases,
Application Server, …

MDA and Application Development

Agenda

1. UML – What is it and what is it good for ?
2. MDA – What is it and what is it good for ?
3. MDA – Sample Process
4. QVT – A View into Transformation

MDA - Motivation

MDA – what it is

- MDA is - like UML and OCL - a specification of the OMG
- MDA is not a language and not a method → it is a concept
- MDA is an approach for model-driven development of architectures
- MDA is not primarily an approach for developing software
- MDA is based on other OMG concepts like UML2, OCL, QVT, MOF

MDA - Motivation

One of the basic ideas – thinking in abstraction layers

CIM - Computation Independet Modeling

PIM - Platform Independet Modeling

PSM - Platform Specific Modeling

IM - Implementation Modeling

(C - and of course coding, instead this is not a MDA abstraction layer)

MDA - Motivation

Another basic idea – thinking in model and metamodel layers

M3 - Meta-Metamodel, the concepts and elements for modeling metamodels, f.e. the UML2 Class Diagram and OCL

M2 - Metamodel, a formal model for a modeling language, f.e. the specification of UML2

M1 - Model, f.e. some UML model of an aspect of the real world

M0 - Instances in the real world, f.e. objects, actions, states, communication

MDA - Motivation

Yet the „most“ basic idea – doing the application development with models

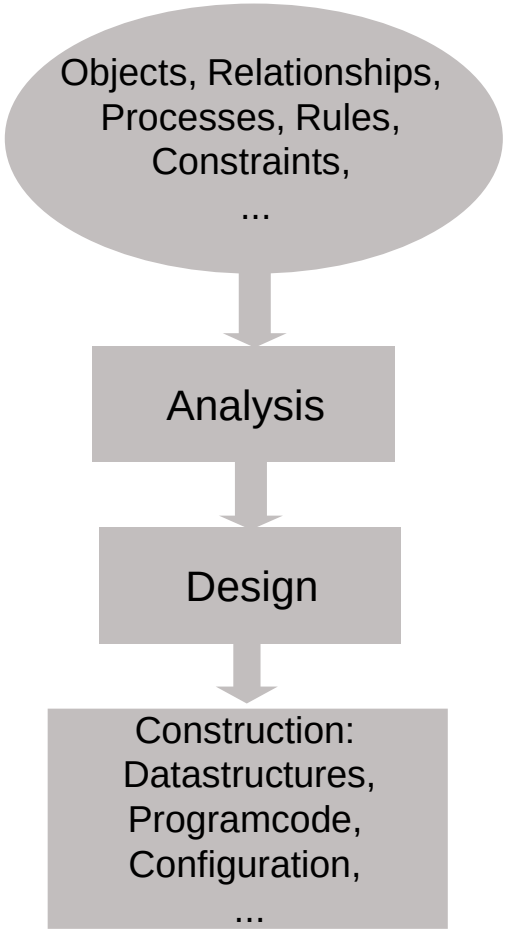
- A (necessary) pre-condition for MDA:
  formal models, based on UML-Metamodels

- Example: all UML-modeltypes are based on UML-Metamodels

  $\Rightarrow$ these are formal models

MDA - Motivation

MDA – what it is good for



CIM: natural languages,
specific modeling languages,
BPMN

PIM: UML

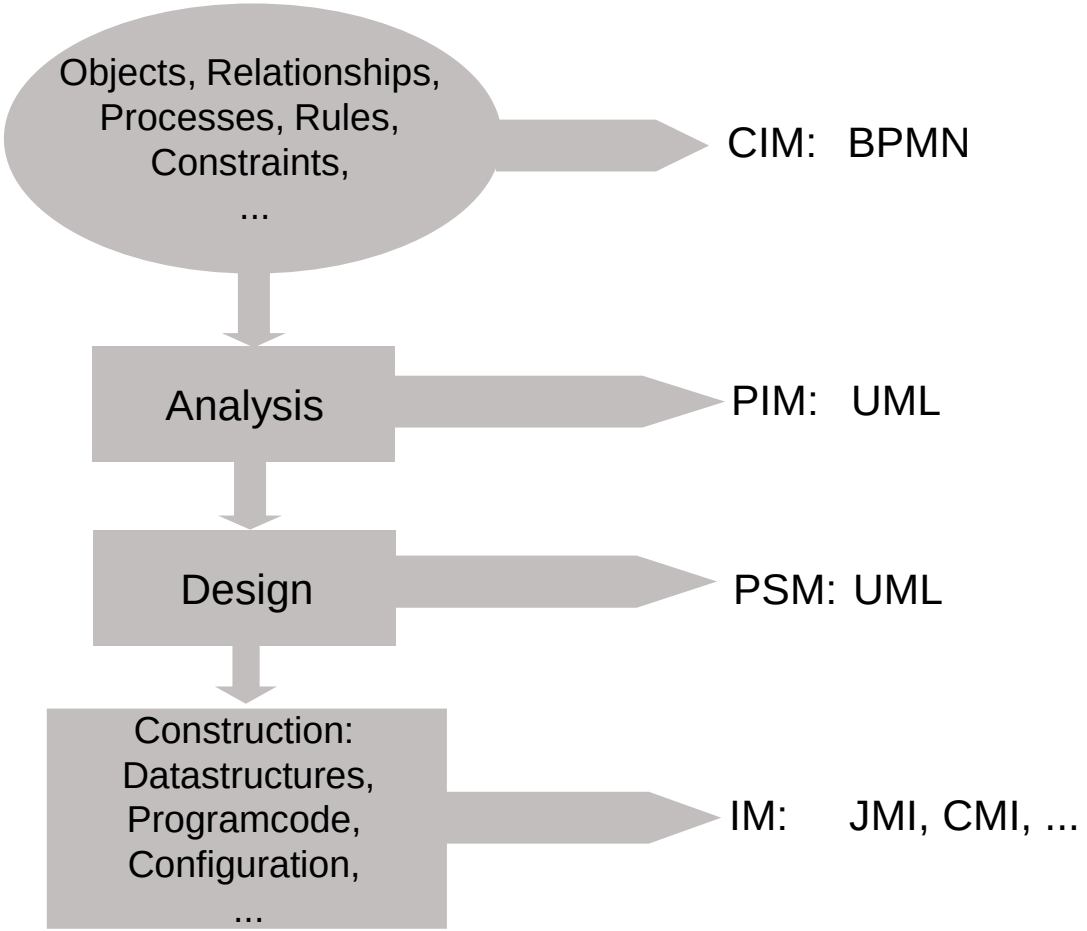
PSM: UML

IM: implementation models
(JMI, CMI, EJB, ...)

MDA - Motivation

MDA – what it is good for

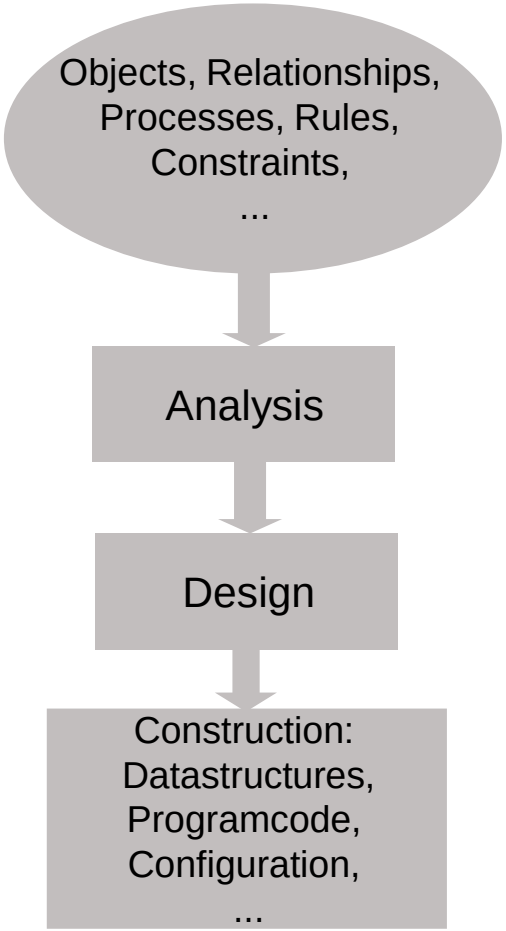
1. Modeling

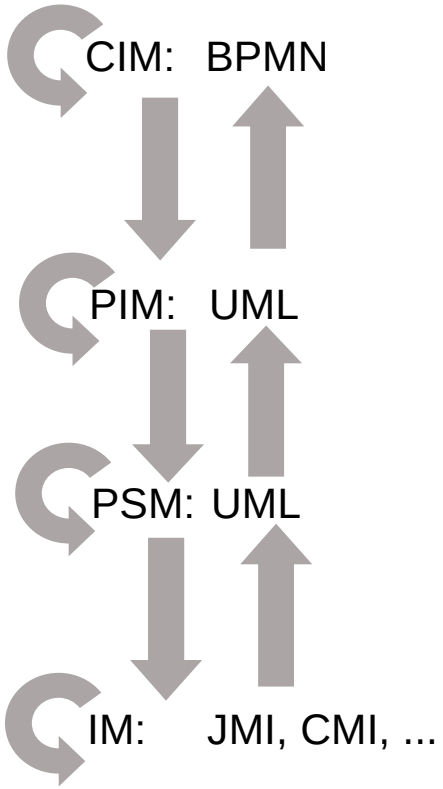


MDA - Motivation

MDA – what it is good for

2. Transformation





MDA and Application Development

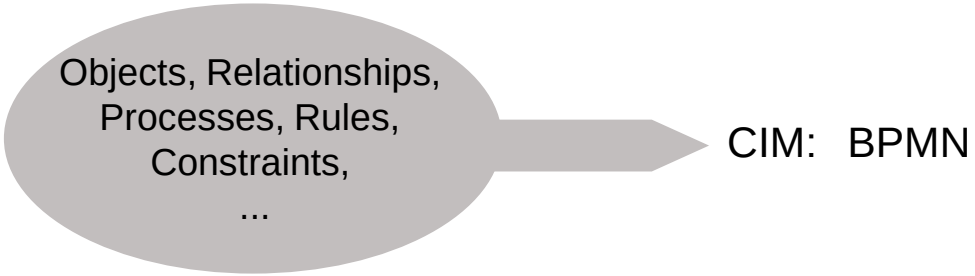
Agenda

1. UML – What is it and what is it good for ?
2. MDA – What is it and what is it good for ?
3. MDA – Sample Process
4. QVT – A view into transformation

MDA - Process

MDA – let‘s drive models from the real world to applications

Modeling



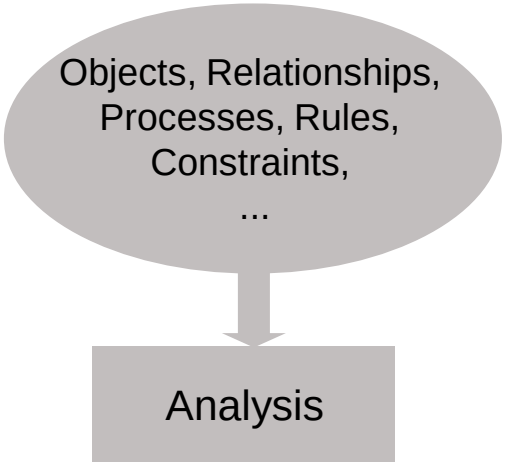
1. Step: Requirement Analysis

   Description of the real world‘s aspects, circumstances and conditions using natural language or some domain language. Modeling with domain relevant models, f.e. BPMN

MDA - Process

MDA – let‘s drive models from the real world to applications

Transformation



CIM: BPMN

PIM: UML

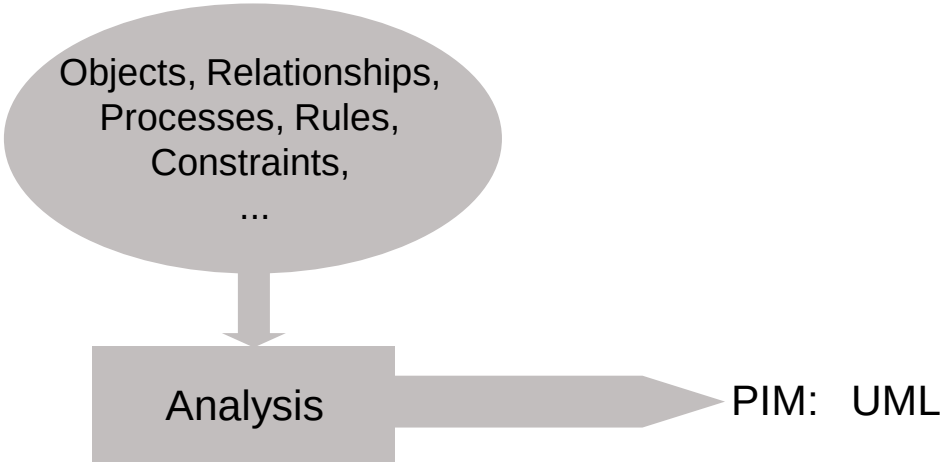
2. Step: Transformation

   Transfer of CIM models into the next layer,
   f.e. transfer of BPMN-models into UML
   Activity-Diagrams

MDA - Process

MDA – let‘s drive models from the real world to applications

Modeling



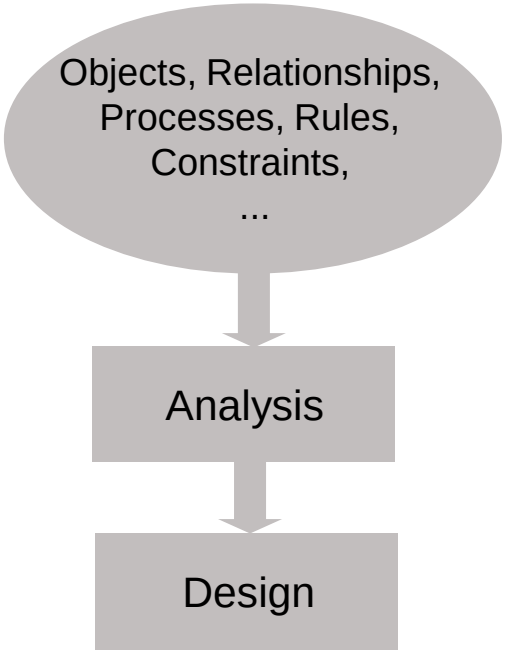
3. Step: Analysis – domain specific system definition

   modeling of domain circumstances in formal models;
   distinction between structure and behavior

MDA - Process

MDA – let‘s drive models from the real world to applications

Transformation



4. Step: Transformation

   Transfer of formal domain models into platform-specific models

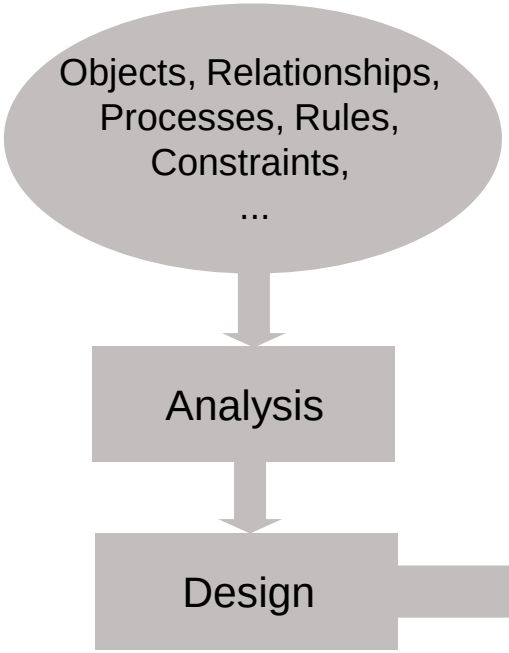
PIM: UML

PSM: UML

MDA - Process

MDA – let‘s drive models from the real world to applications

Modeling



5. Step: IT-specific modeling – IT concept

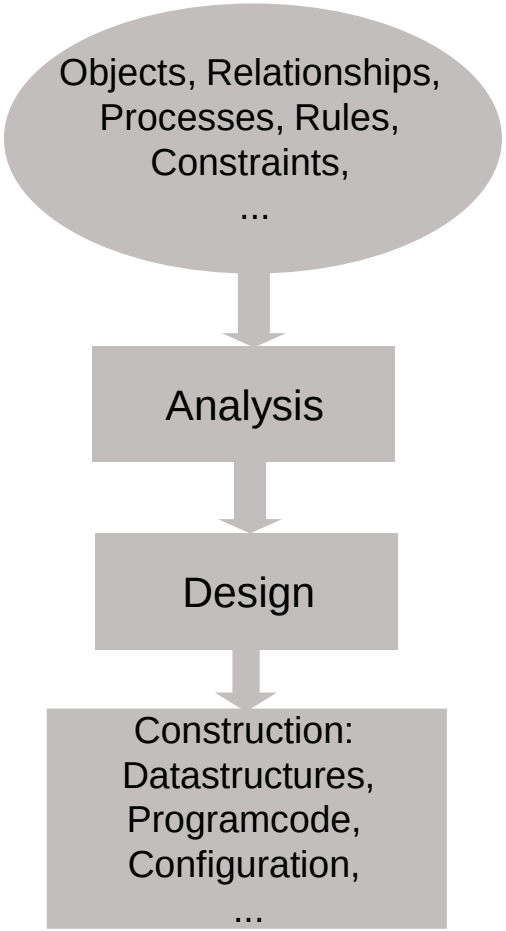
   Specification of the application concerning to DP, platform-specific aspects, f.e. J2EE or .NET. Refinement and concretizing of the architectures, i.e. UML class diagrams into ERM.

PSM: UML

MDA - Process

MDA – let‘s drive models from the real world to applications

Transformation



6. Step: Transformation into implementation model

   Transfer of platform specific models into implementation models (code, data structures, DB schemes, ...)

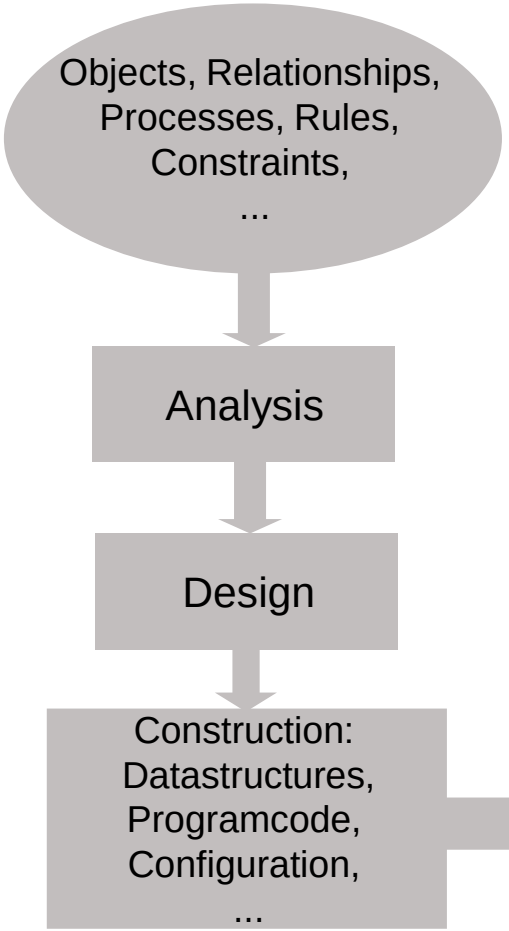
PSM: UML

IM: JMI, CMI, ...

MDA - Process

MDA – let‘s drive models from the real world to applications

Modeling



7. Step: Implementation modeling

   Modeling in the IM layer means working on the technical concepts with models and languages of the concrete DP environment; coding, building, constructing, configuring, …

IM: JMI, CMI, ...

MDA and Application Development

Agenda

1. UML – What is it and what is it good for ?
2. MDA – What is it and what is it good for ?
3. MDA – Sample Process
4. QVT – A View into Transformation

Query View Transformation

QVT – what it is

QVT is - like UML and OCL - a specification of the OMG

QVT describes a concept and languages for Model-to-Model transformation

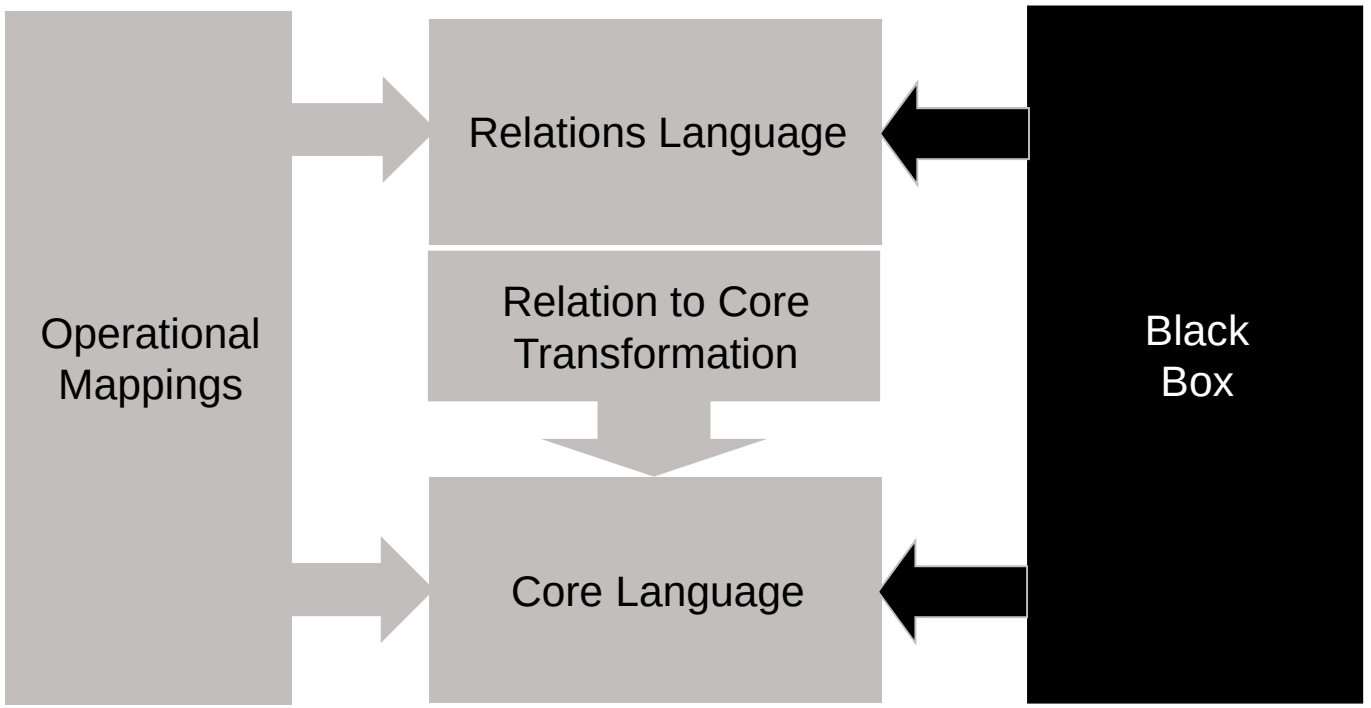
QVT languages are seperated into a descriptive one – Relations Language –
and an imperative one – Operational Mappings

QVT languages are based on a Core Language,
that is something like a „virtual mapping machine“

QVT is developed from OCL
so transformation ***mappings*** are more or less OCL-expressions

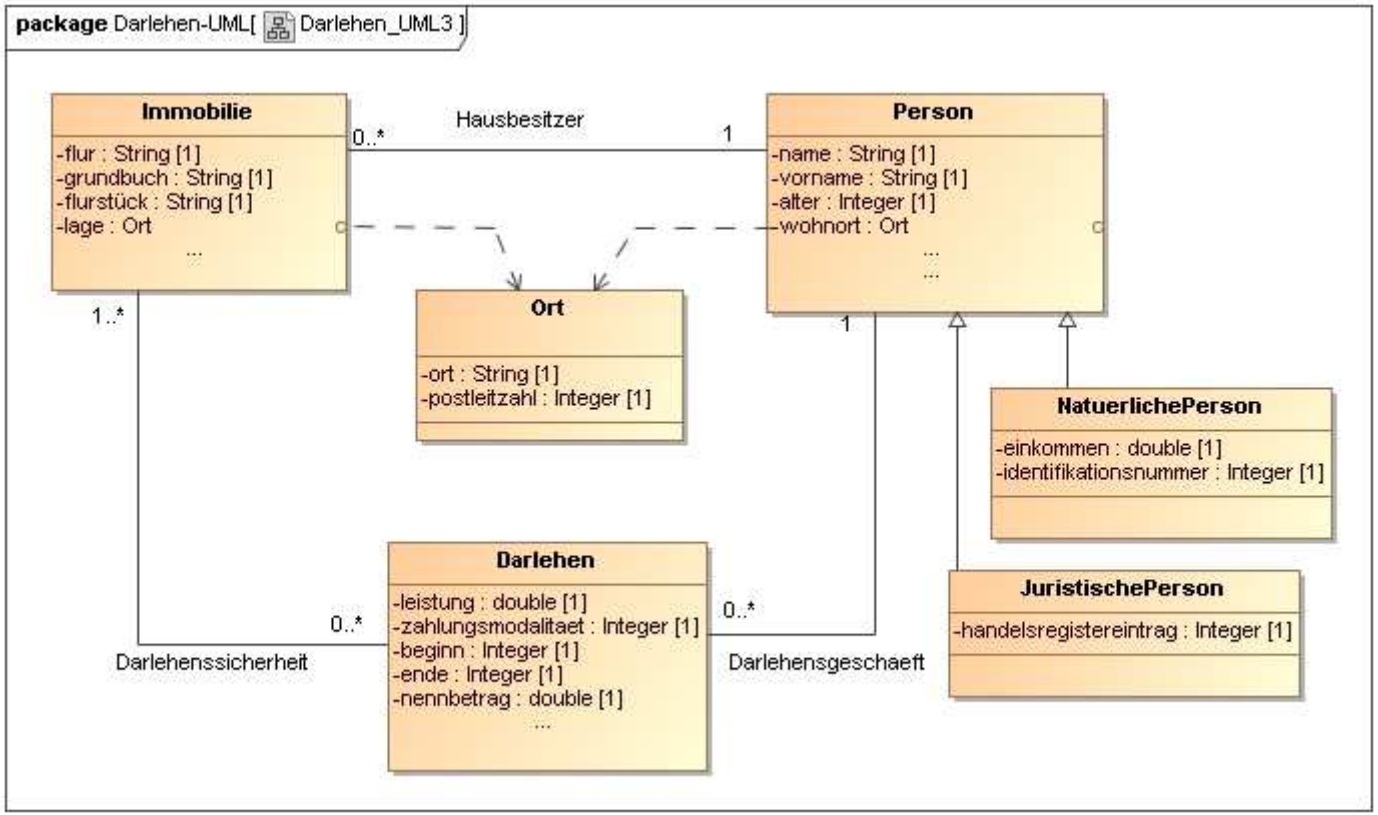
Query View Transformation

QVT - Architektur



Query View Transformation

A simple sample model



Query View Transformation

A transformation with Relations Language

```
/*
 * transform a UMl package into a ERM schema
 */
transformation HelloWorld ( source : UML, target : ERM )
{
   top relation SourceToTarget
   {
      packageName : String;

      checkonly domain source sourcePackage : UML::Package
      {
         name = packageName
      };

      enforce   domain source targetSchema  : ERM::Schema
      {
         name = packageName
      };
   }
}
```

Query View Transformation

A transformation with Operational Mapping

```
/*
 * transform a UM package into a ERM schema
 */
transformation HelloWorld ( in source : UML, out target : ERM );

main()
{
  source.objects()[Package]->map createHello();
}

mapping Package::createHello () : Schema
{
   name := "HelloWorld";          -- Erzeugung eines Schemas namens 'HelloWorld,

   end { log ("Hello, World!"); } –- log-Ausgabe auf der Konsole
}
```

Query View Transformation