60. Megamodels in One Technical Space

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

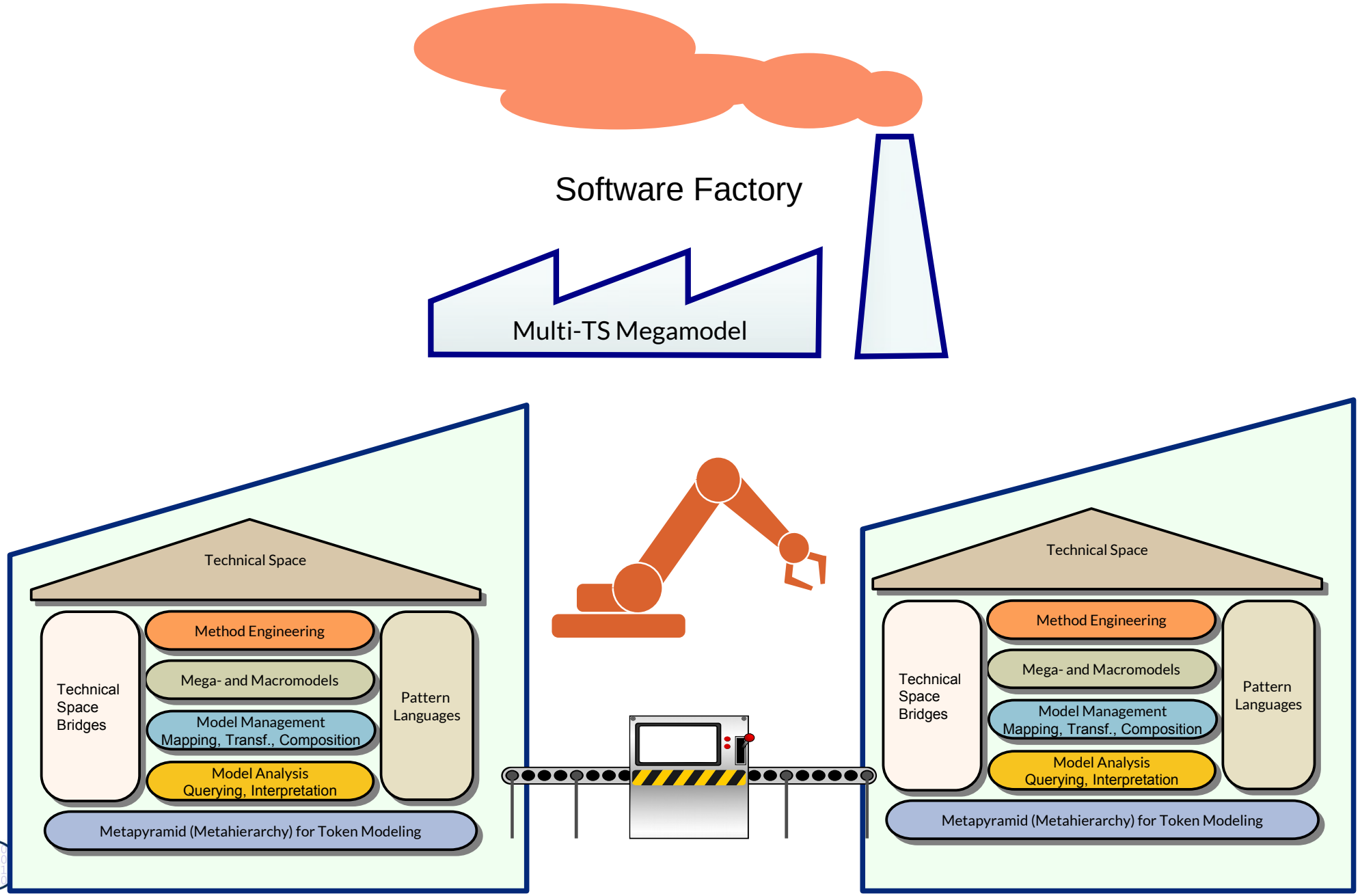
http://st.inf.tu-dresden.de/teaching/most

Version 17-0.2, 27.01.18

1) MDA
2) MDA Toolkits
3) RoSIMA

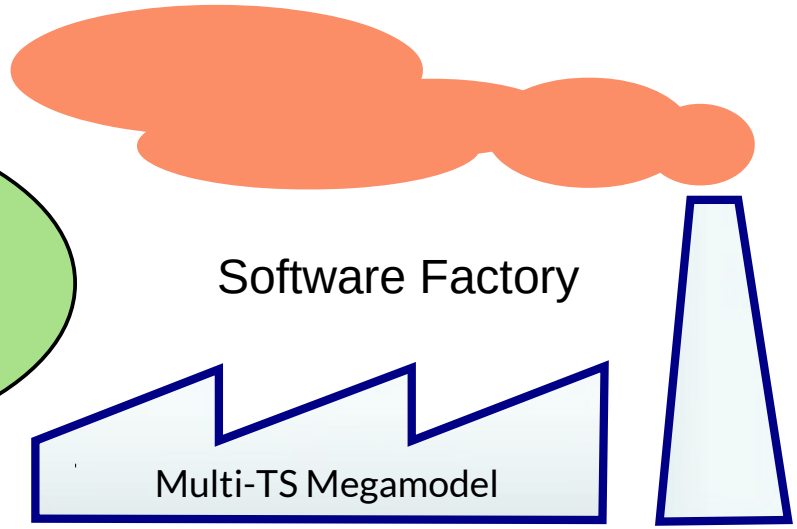
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

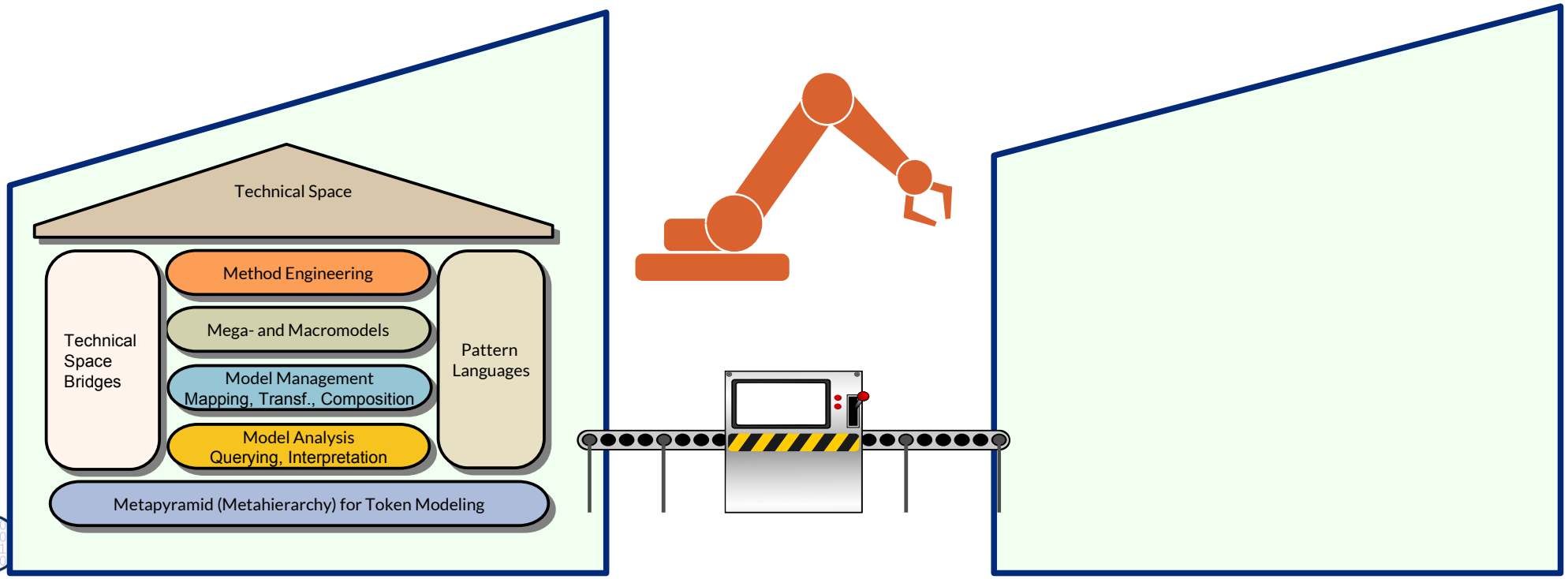
Q11: A Software Factory's Heart: the Multi-TS Megamodel



Here: Only 1 Technical Space

In this chapter:
1-TS Megamodel
MDA, RoSI-MA





Literature

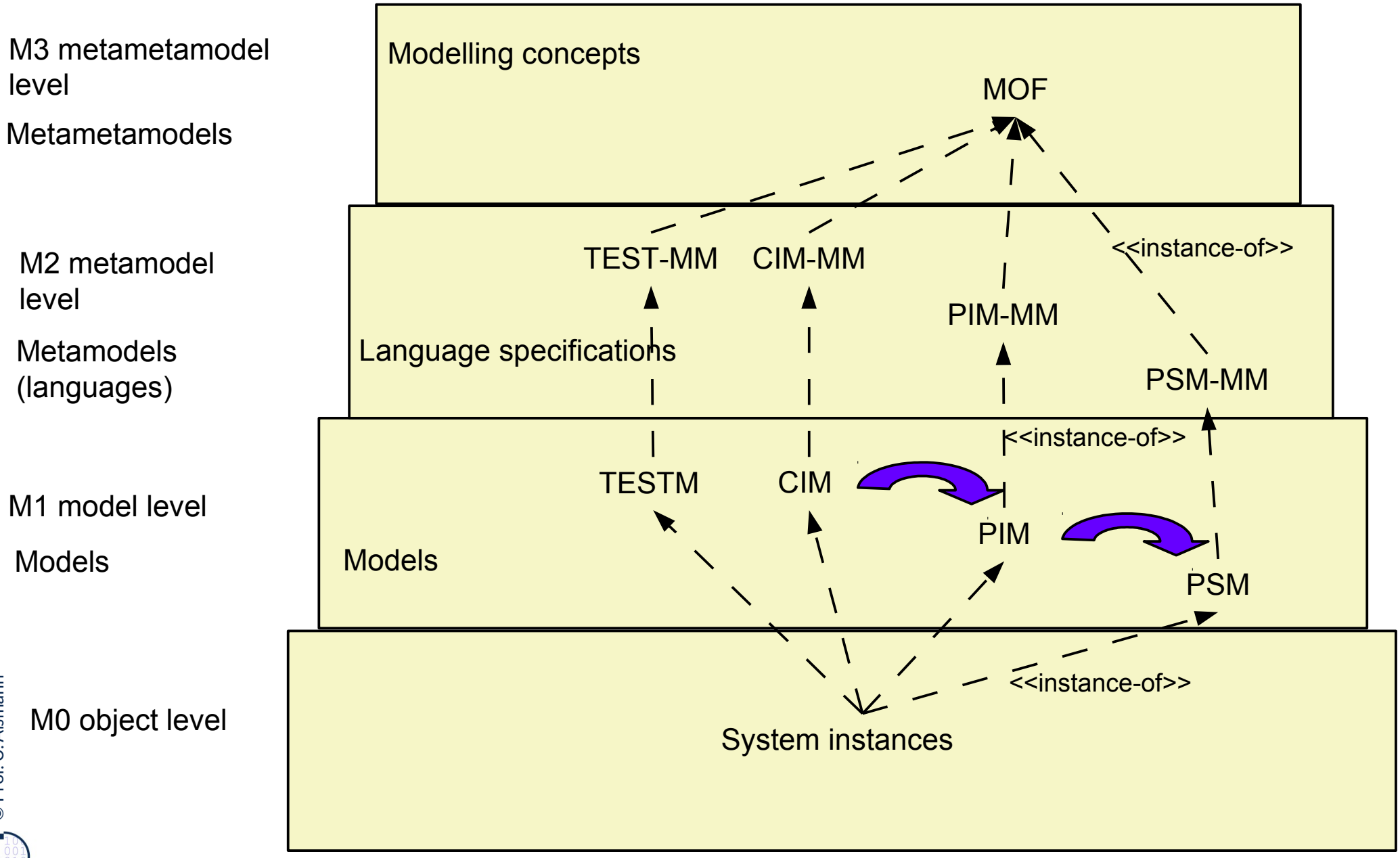
- Alan Brown. An introduction to Model Driven Architecture. Part I: MDA and today's systems
  - http://www.ibm.com/developerworks/rational/library/3100.html
- Quelle: Petrasch, R., Meimberg, O.: Model Driven Architecture - eine praxisorientierte Einführung in die MDA. Dpunkt-Verlag. 2006
- Frédéric Jouault and Ivan Kurtev. On the Architectural Alignment of ATL and QVT. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06). ACM Press, Dijon, France, chapter Model transformation (MT 2006), pages 1188—1195.
  - http://atlanmod.emn.fr/bibliography/SAC06a
- Tutorial über ATL "Families2Persones"
- http://www.eclipse.org/m2m/atl/doc/ATLUseCase_Families2Persons.ppt
- ATL Zoo von Beispielen
  - http://www.eclipse.org/m2m/atl/atlTransformations
- Kevin Lano. Catalogue of Model Transformations
  - http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf
- Implementation in ATL
  - http://www.eclipse.org/m2m/atl/atlTransformations/EquivalenceAttributesAssociations/EquivalenceAttributesAssociations.pdf

Model-Driven Software Development (MDSD) in 1 Technical Space

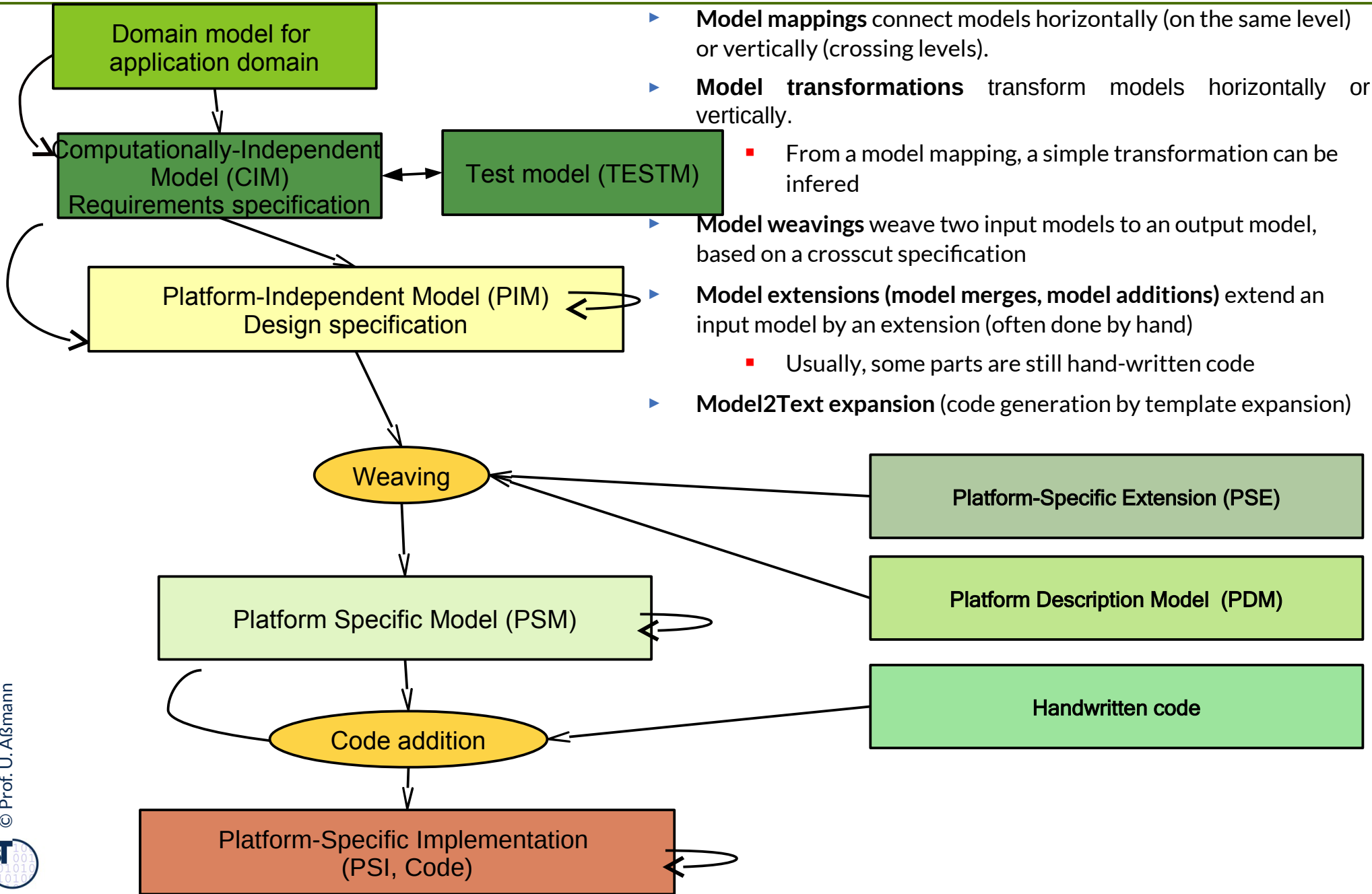
- MDSD in 1-TS falls into several main development methods with a megamodels:
  - Engineering with metamodels in ReDeCT-like megamodels (integrated software life-cycle management tools):
    - for integrated requirements, documentation, and testing along the life-cycle
  - Engineering with DSL (domain-specific modeling, DSM) (Meta-CASE toolkits)
    - For simplifying the specification of domain-specific software
  - Model-Driven Architecture (MDA) (MDA toolkits):
    - For platform-specific variation
- **Model mappings** correlate models defining trace relations between model elements
  - From them, model transformations can easily be derived
- **Model transformations**
  - **Horizontal model transformations** transform a model within a single language
  - **Vertical model transformations** transform a model from a higher-level language to a lower-lewel language (**lowering**)
  - **Broadband model transformations** transform a model from a higher-level set into a lower-level set of a broadband (wide-spectrum) language
- **Model weavings** extend models by other models

61.1 Model-Driven Architecture (Modellgetriebene Architektur, MDA)

The MDA Embedded in the MOF Metapyramid



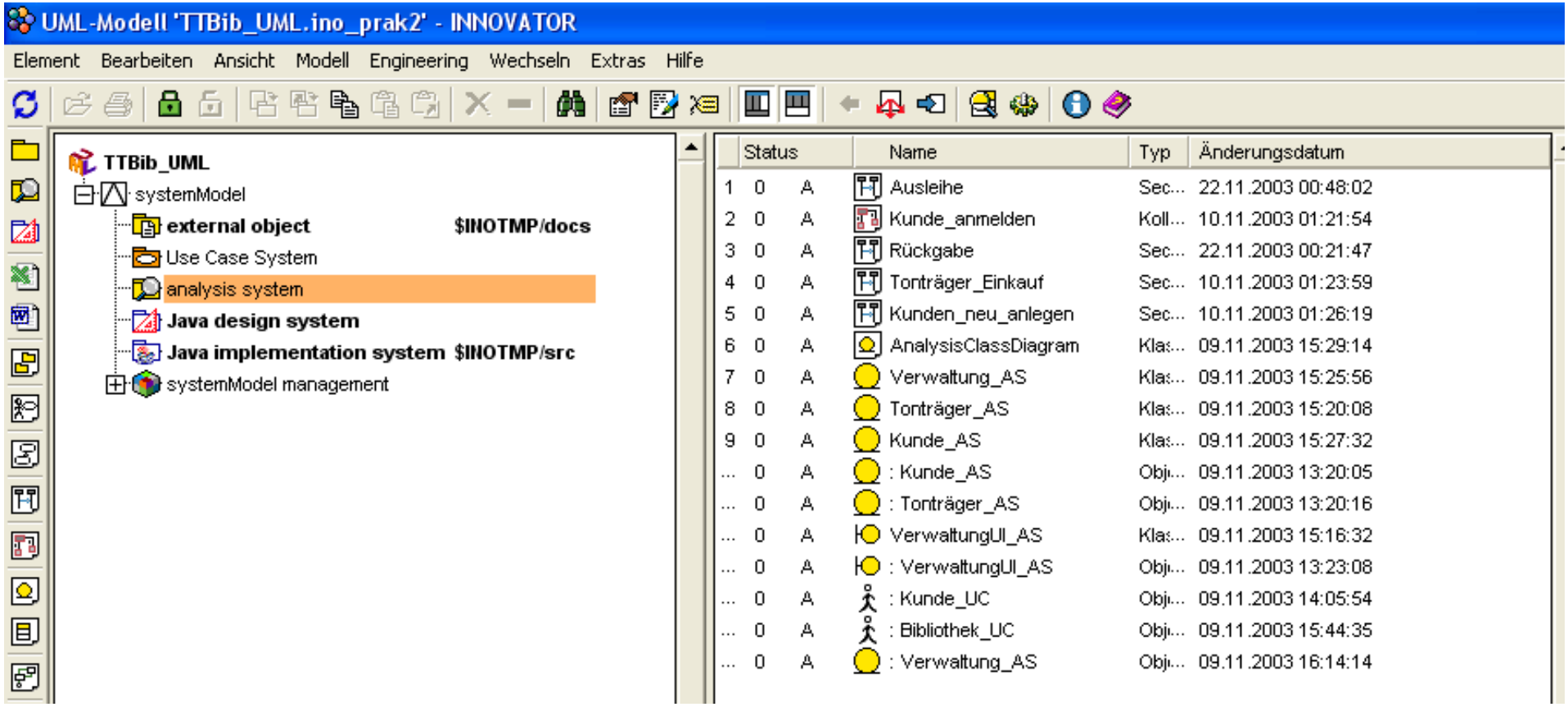
Model Mappings and Model Weavings



- **Model mappings** connect models horizontally (on the same level) or vertically (crossing levels).
- **Model transformations** transform models horizontally or vertically.
  - From a model mapping, a simple transformation can be infered
- **Model weavings** weave two input models to an output model, based on a crosscut specification
- **Model extensions (model merges, model additions)** extend an input model by an extension (often done by hand)
  - Usually, some parts are still hand-written code
- **Model2Text expansion** (code generation by template expansion)

PIM and PSM and Model Mapping in MID INNOVATOR

- Innovator can specify transformations between its models



PIM und PSM gemäß der MDA

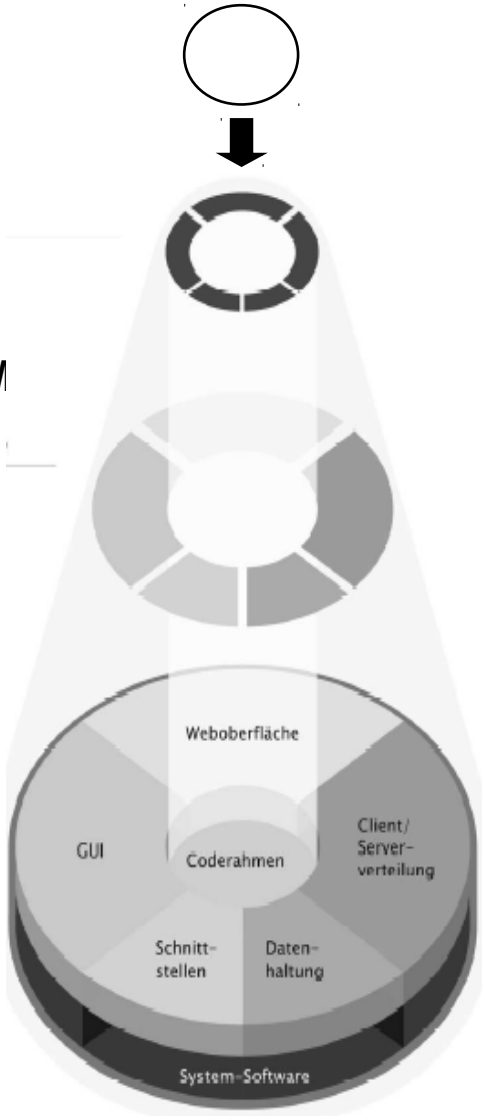
Für die unterschiedlichen Abstraktionsebenen **PIM** und **PSM** stehen verschiedene Beschreibungsmittel zur Verfügung:

Fachkonzept auch CIM
(Computation independent model)

Plattformunabhängiges Modell
(UML, OCL, XMI)

Plattformspezifisches Modell PSM
Basiskomponenten (JB)
Steuerungskomponenten
Infrastrukturkomponenten (EJB, CCM, COM+, .NET)
Anwendungskomponenten

Plattformspezifische Implementierung (PSI)
in Programmiersprache

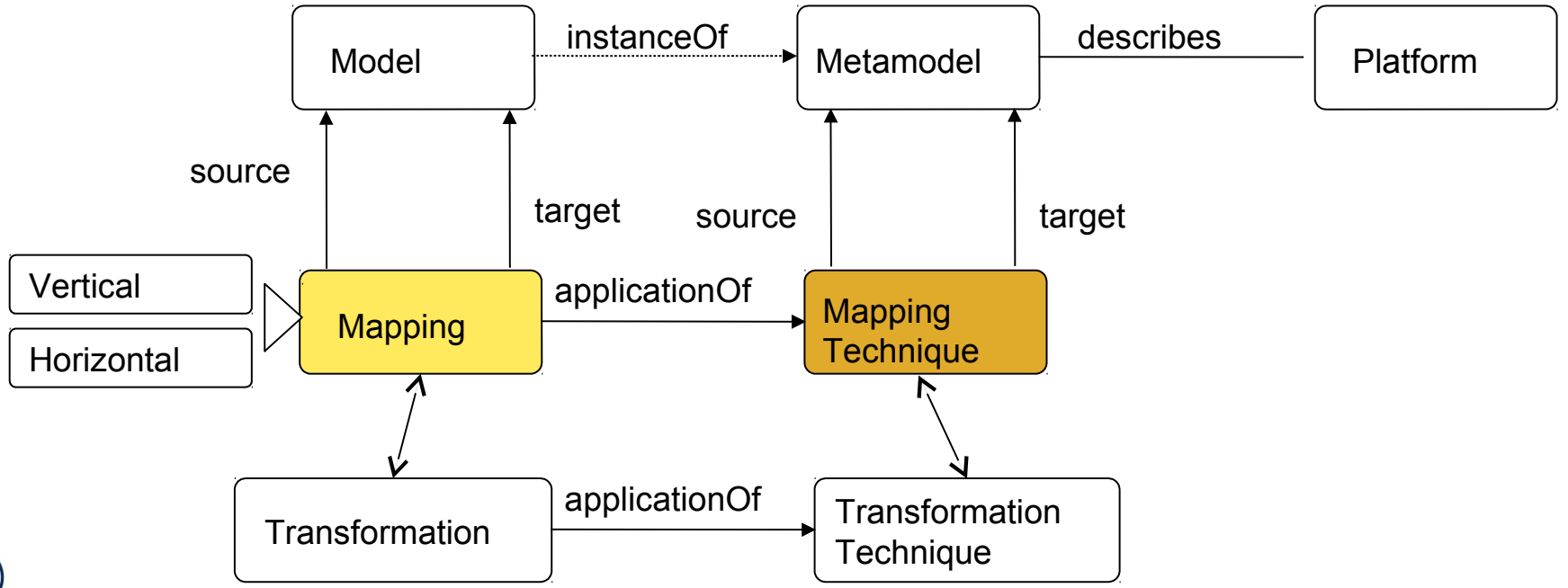


Ein **PSM** berücksichtigt die jewei-lige Basistechnologie, auf der ein **PIM** zum Einsatz kommen kann (CORBA-Broker, .NET-Spezifika oder das Web-Service-Protokoll SOAP). Auch **PSM**s können mit der UML modelliert werden. In jedem Fall werden aus den **PSM**s die **Code-gerüste** erzeugt, die die Komponenten-Entwickler dann weiter bearbeiten.

Quelle: Warum JANUS MDA und MDA JANUS ist; Whitepaper der Firma otris Software AG Dortmund; URL: www.otris.de

What are Model Mappings?

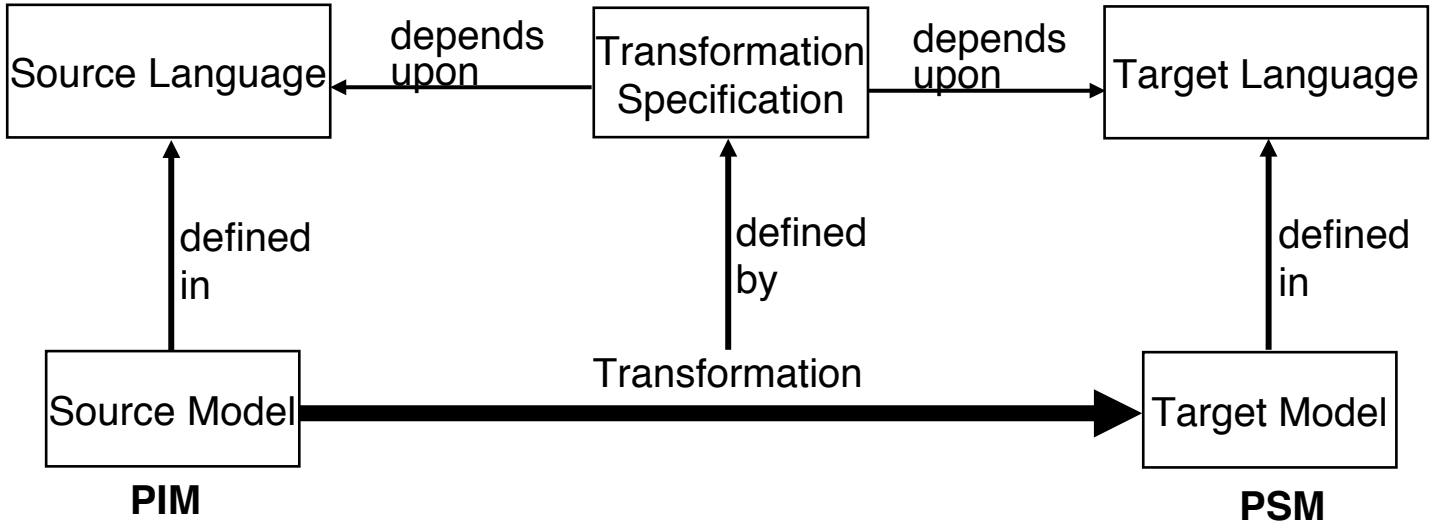
- Remember Model: “A model is a representation of a part of a function of a system, its structure, or behavior”
- A model mapping can be generated from a model analysis
- The mappings are automatic or semi-automatic: step-wise refinement of the model by transformation
  - From a model mapping, transformations can be generated



MDA Transformation Process

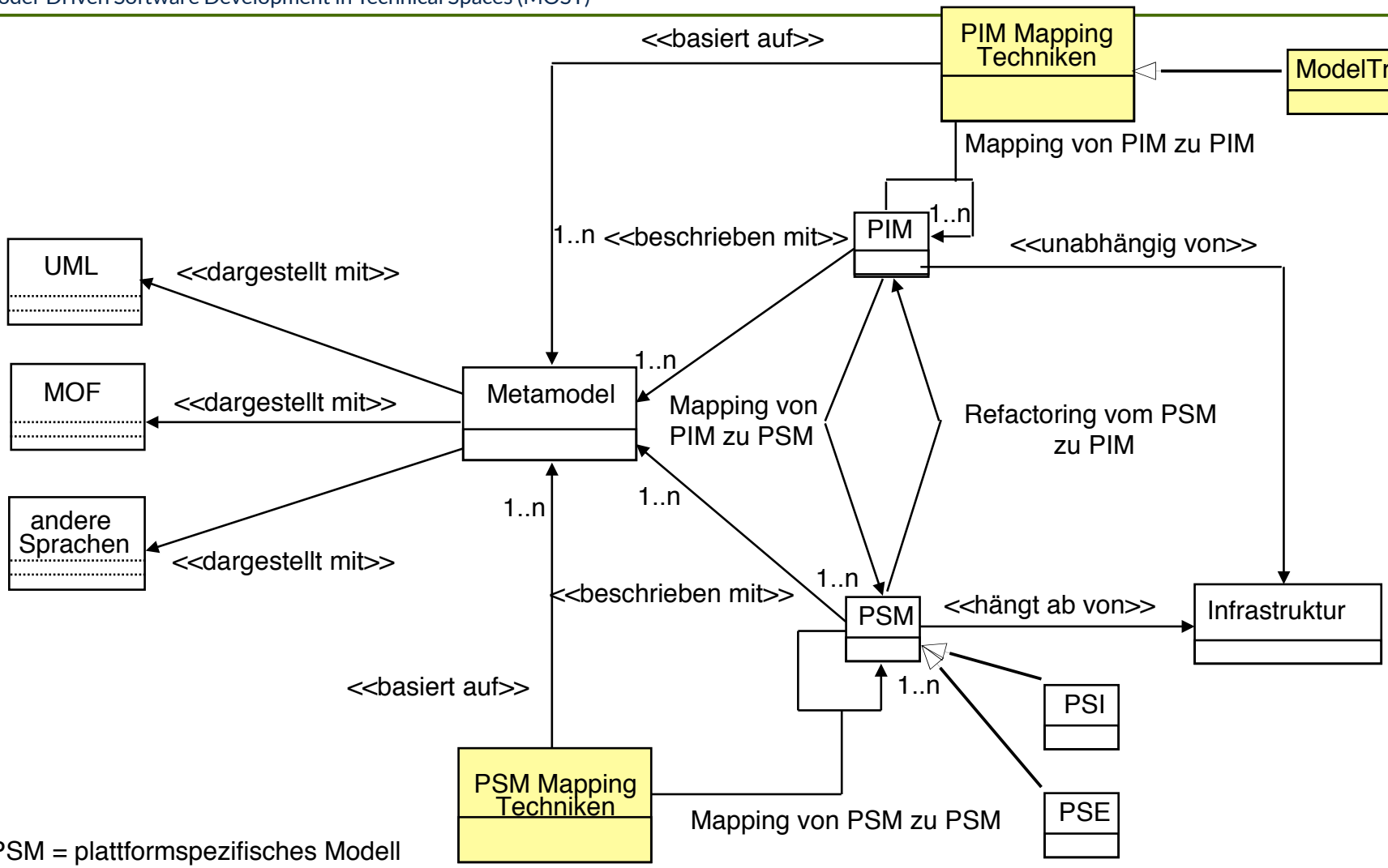
Aus plattformunabhängigem (*independent)* Metamodell **PIM** sind mittels Regeln, Techniken plattformspezifische (*specific*) Modelle **PSM** zu entwerfen, zu generieren, oder abzuleiten, um neue Anwendungen für eine bestimmte (Komponenten-)Plattform zu erhalten.

Ein weiteres Ziel von MDA ist die Integration solcher Technologien wie CORBA, J2EE, .Net und XML als *Plattform*.



Quelle: Kleppe, A., Warmer, J., Bast, W.: MDA Explained - Practice and Promise of the Model Driven Architecture; Addison Wesley 2003 (Draft 25.10.02)

A Metamodel of the MDA Megamodel



PSM = plattformspezifisches Modell
PSE = plattformspezifische Erweiterung
PSI = plattformspezifische Implementation

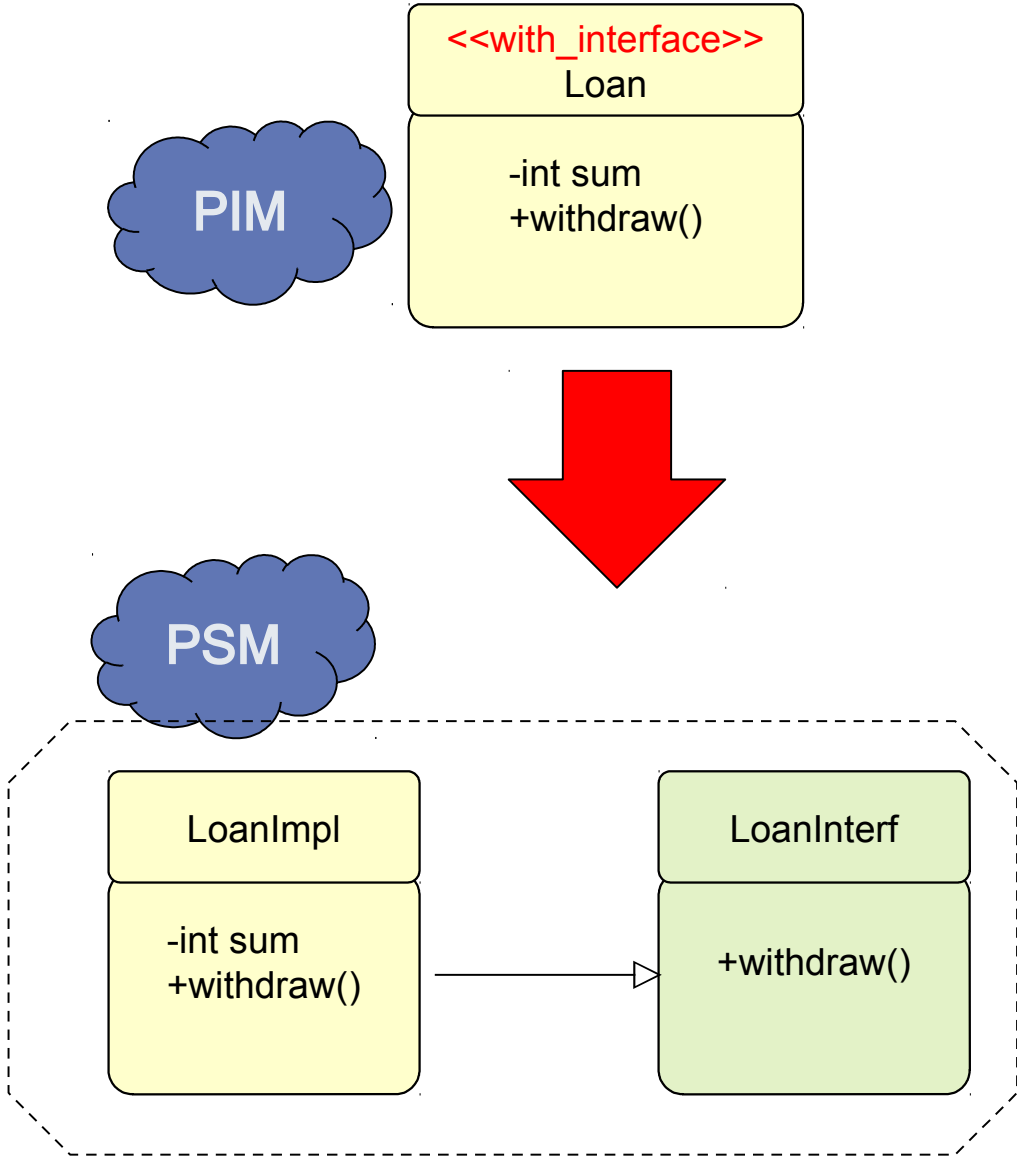
Transformationen bezeichnet man auch als **Abbildungen** (**mappings***)*. Mapping von PIM zu PIM schafft neue „Business Viewpoints“, von PSM zu PIM Abstraktionen aus plattformabhängigen Implementierungen und zwischen PSM weiteren Verfeinerungen oder Zielplattformen.

Model Management in Megamodels

- In the MDA megamodel, graph-based models must be maintained with:
  - Model algebrae
    - Lookup and query of model elements
    - Diff, comm, union, compose of models
    - Model composition with port-graph algebrae (invasive composition)
  - Version management
  - Konfiguration management

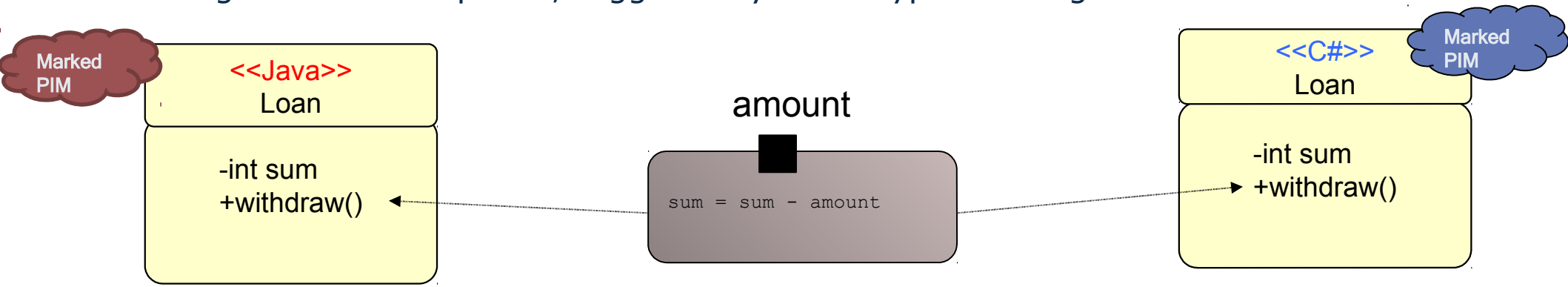
Morphic Mappings on Marked PIMs

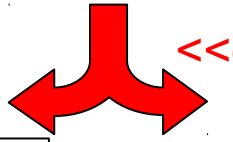
- **1:1 or 1:n mappings** are important for **marked PIMs:**
  - *Stereotypes* introduce an exclusively-owns relationship from 1 element of the PIM to n elements in the PSM
  - Supported by many MDA tools, such as AndroMDA
- The stereotype creates a mapping between a PIM class and a set of PSM classes
  - The stereotype tells the MDA system how to transform the PIM class to the PSM
  - The stereotypes partition the PSM: The border of a partition is demarcated by the PIM stereotype tag



Example of a Marked PIM and the Induced Model Transformations

- Tags (stereotypes) are mapped to different class implementations in a PSM
- Here: mapping of a class and activity diagram to different languages, using different code generation templates, triggered by stereotype marking





PSI Java

```
// Java implementation as a decorator
class Loan extends Account {
  // decorator backlink
  Account upper;

  private int sum;
  public void withdraw(
   int amount) {
   sum -= amount;
}
```

PSI C#

```
// C# implementation: a partial class
partial class Loan : Account {
  private int sum;
  public void withdraw(
   int amount) {
   sum -= amount;
}
```

Cartridges are Transformation Libraries for Marked PIMs

- **Cartridges** define both the model mapping from a PIM to a PSM *and* the model transformation
  - manual marking of the PIM
  - selective transformation of the marked PIM classes
  - automatic transformation using the mapping and transformations from the cartridge
  - no manual specifications of mappings and transformations necessary

60.2 MDA Toolkits

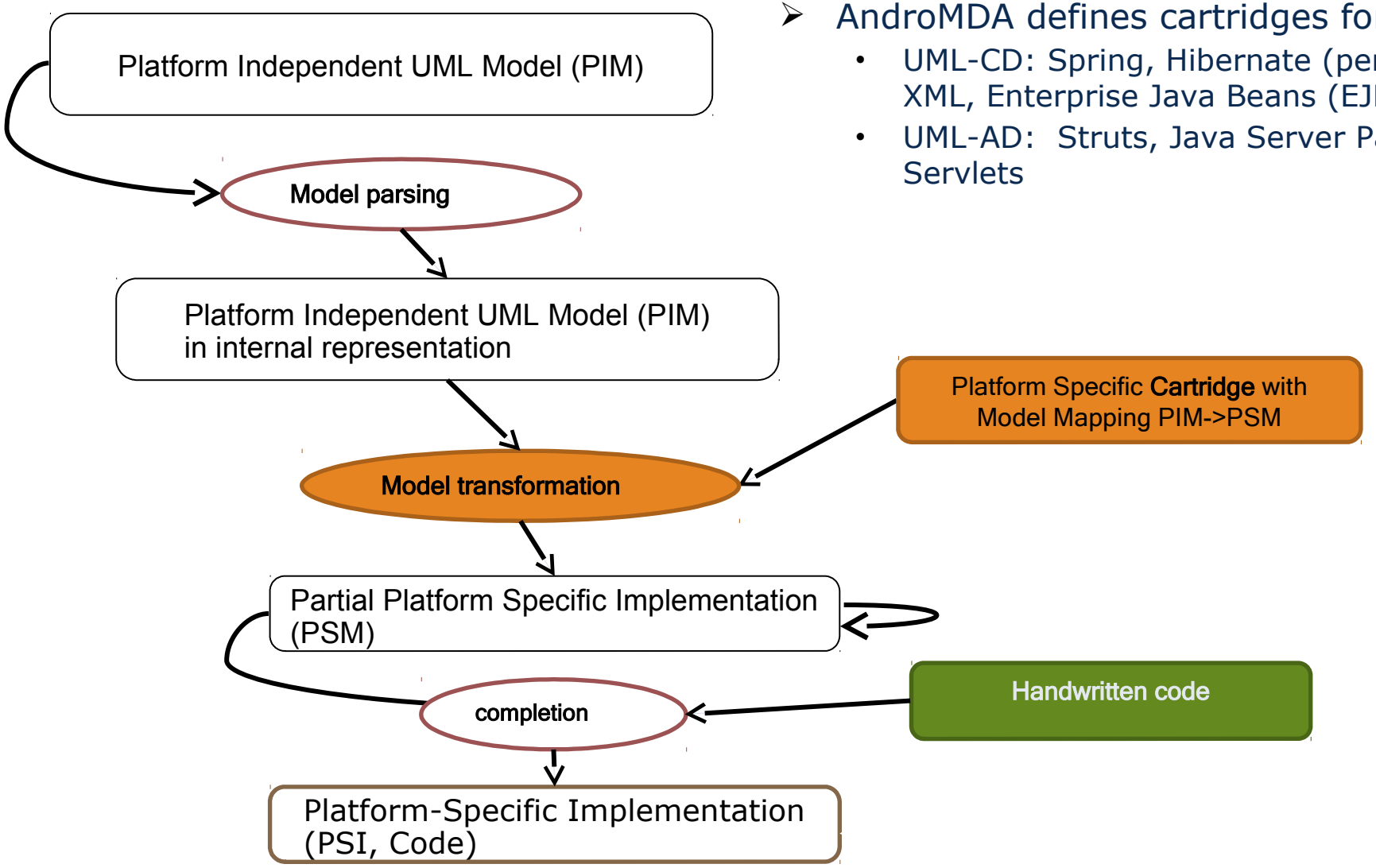
Important Features of MDA Toolkits

- **Model-to-Model Mapping** bzw. **Model-to-Model Transformation** (e.g., PIM to PSM) with cartridges
- User definition of cartridges with query and transformation languages
  - e.g., with QVT, ATL, Graph writing or XML Rewriting
- **Forward- und Reverse-Engineering**
  - Code generation (Model-to-Code Transformation, PSM to PSI)
    - Mapping to a programming language (e.g., with JMI)
- **Roundtrip-Engineering** auf der Code-Ebene zur Unterstützung des Single-Source-Prinzips
- **Model-driven Testing:** generation of test cases ad test data based on models

[Petrasch, R., Meimberg, O.: Model Driven Architecture - eine praxisorientierte Einführung in die MDA; dpunkt-verlag 2006]

60.2.1 AndroMDA, a Leading MDA Toolkit

- AndroMDA defines model mappings in platform-specific **cartridges.**
  - A cartridge contains a mapping from UML to e.g., Java, C# or C++ and a model transformation



- AndroMDA defines cartridges for
  - UML-CD: Spring, Hibernate (persistency), XML, Enterprise Java Beans (EJB)
  - UML-AD: Struts, Java Server Pages(JSP), Servlets

60.2.2 MDA Toolkit ArcStyler

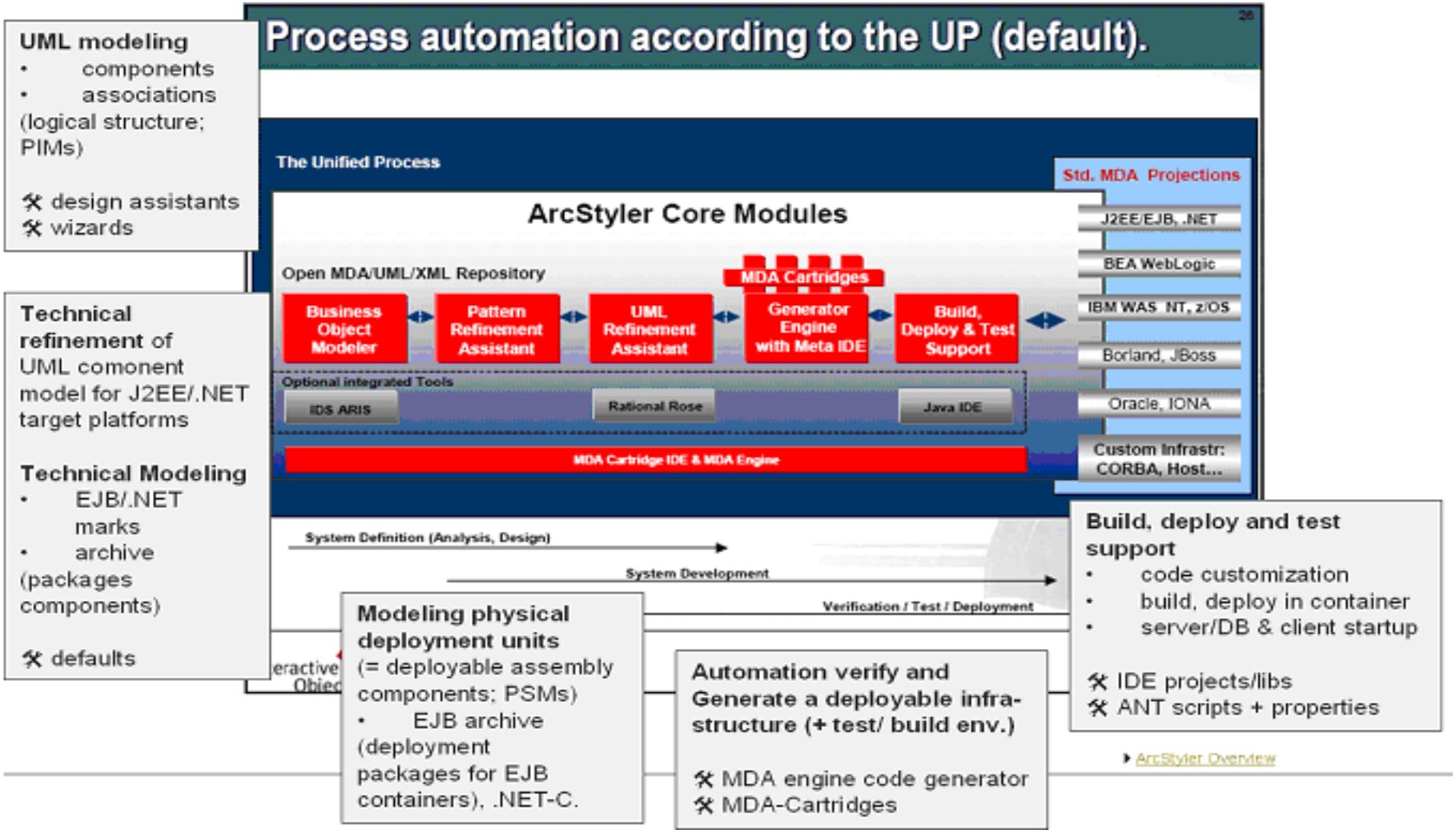
ArcStyler is a toolkit working with several UML-editors such as MagicDraw or Rational Rose

- Cartridges for model mappings and transformations
- **Object Modeler** for requirements modeling; based on CRC-Cards
- **Pattern Refinement Assistent** transforms the domain model interactively into a PIM UML-model (with MagicDraw or Rational Rose)
  - With annotation of design decisions
- **Refinement of the PIM**
  - Horizontal refinement on PIM level
  - Vertical transformation to PSM or PSI (code generation)
- **Code completion (Codevervollständigung)** and optimization for an application platform
- **Component generation** for user interface
- Generation for build tools
- Generation for database persistency

http://www.software-kompetenz.de/servlet/is/27460/?print=true

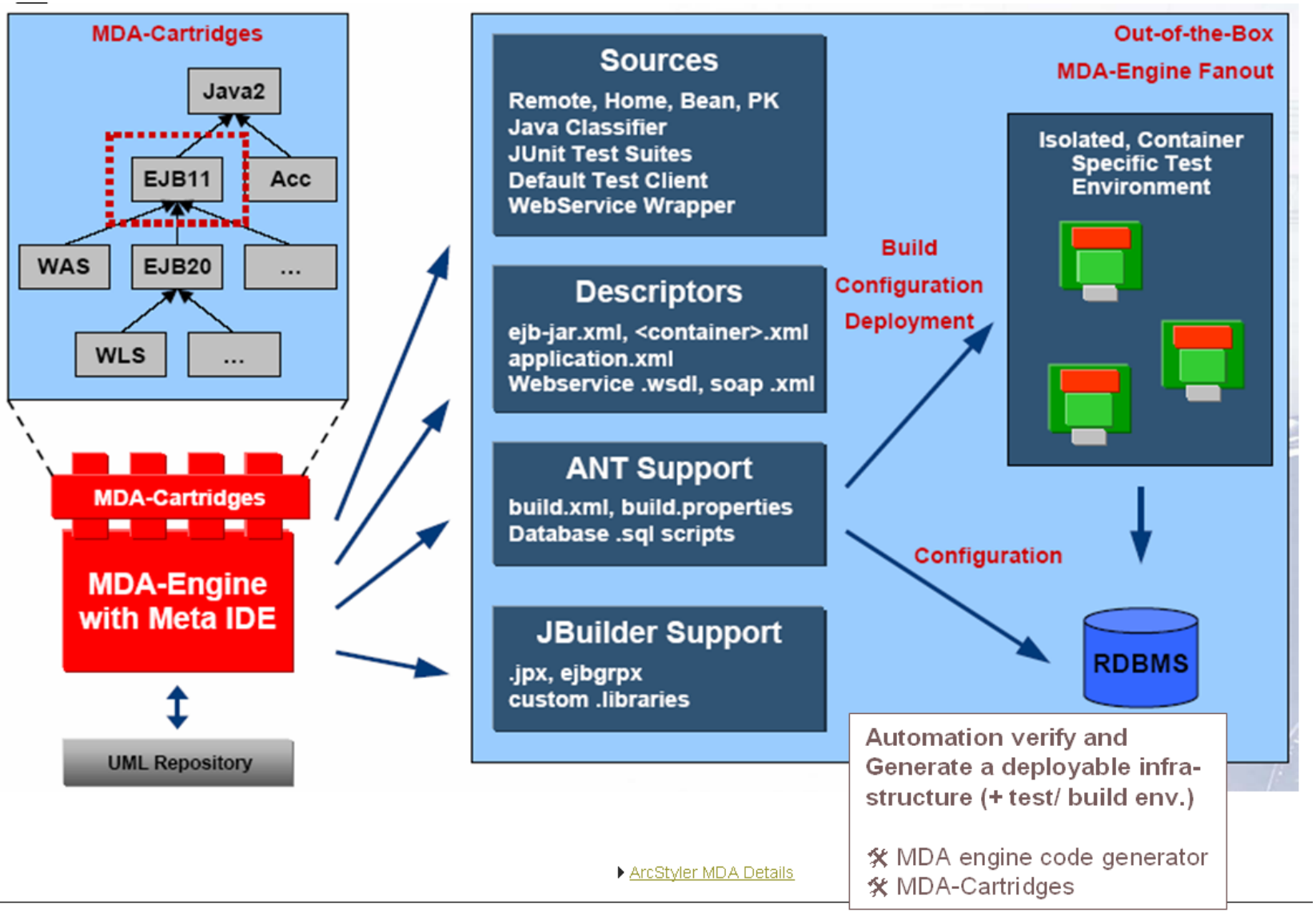
Versteegen, G.: Wege aus der Plattformabhängigkeit - Hoffnungsträger Model Driven Architecture; Computerwoche 29(2002) Nr. 5 vom 1. Febr. 2002

Process of ArcStyler



http://www.interactive-objects.com/products/arcstyler/supportdocumentation.html
http://arcstyler.software.informer.com/

Cartridges and Generated Artifacts



Quelle: Butze, D.: Entwicklung eines Praktikums für die werkzeugestützte Softwareentwicklung nach der Model-Driven-Architecture; Großer Beleg an der Fakultät Informatik der TU Dresden 2004

Some MDA Tools

	Integrated into	URL
AndroMDA	Eclipse	http://www.andromda.org/
XText, Xpand	Eclipse	http://www.eclipse.org/Xtext/
IBM Rational Suite Software Architect	Eclipse	
BITplan smart Generator	Eclipse	http://www.bitplan.com/

Quelle: Petrasch, R., Meimberg, O.: Model Driven Architecture - eine praxisorientierte Einführung in die MDA; dpunkt-verlag 2006

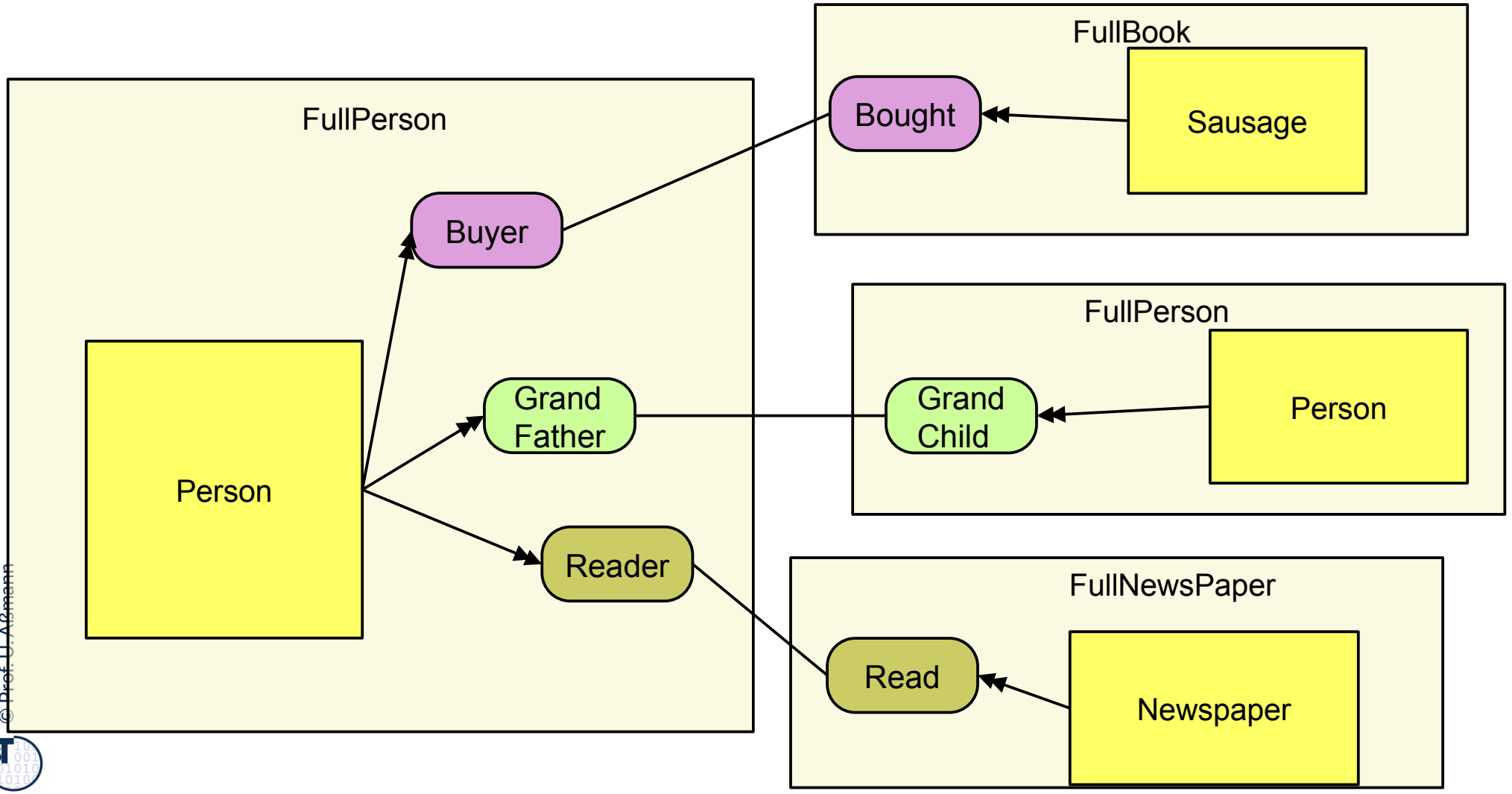
60.3 The Megamodel of RoSI: RoSIMa

- The Megamodel of RoSI and its traceability of model elements is extremely simple, because the role-based models and metamodels are factorizing objects

The Steimann Factorization of Natural and Role Types

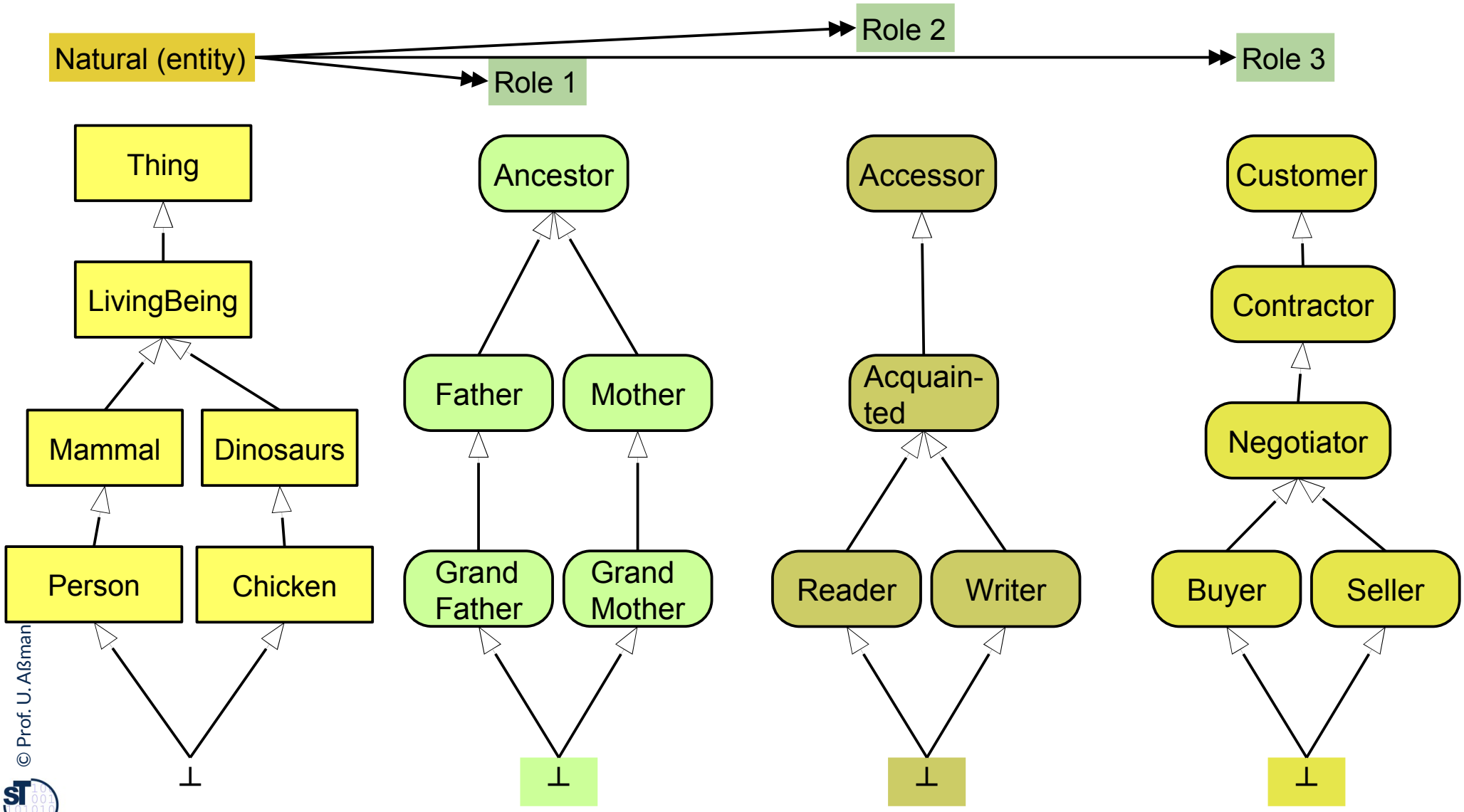
Splitting a full type into its *natural* and *role-type* components

- FullType = Natural x (role-type, role-type, ...)
- FullPerson = Person x (Reader, Father, Customer, ..)



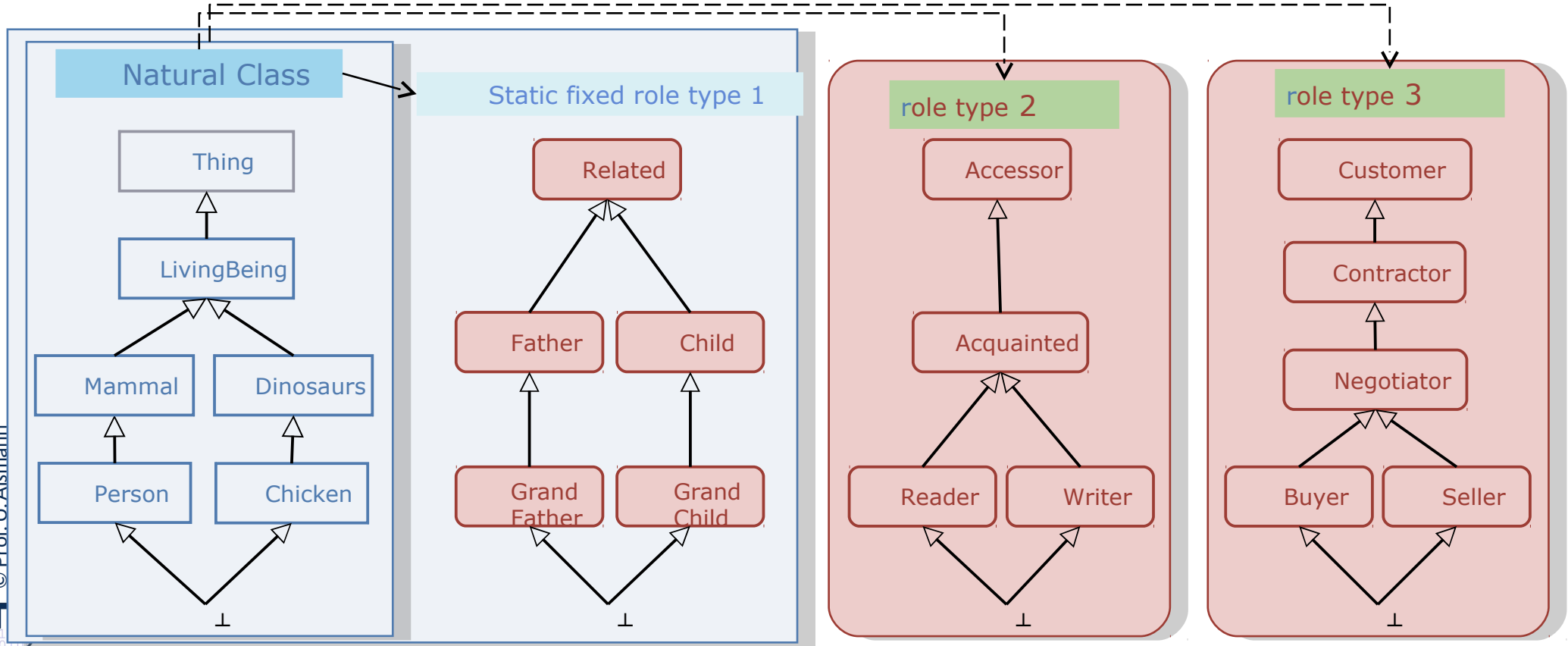
Full Type is from Inheritance Product Lattice

Q: What is a reading buying grandfather person? (A: tuple type)



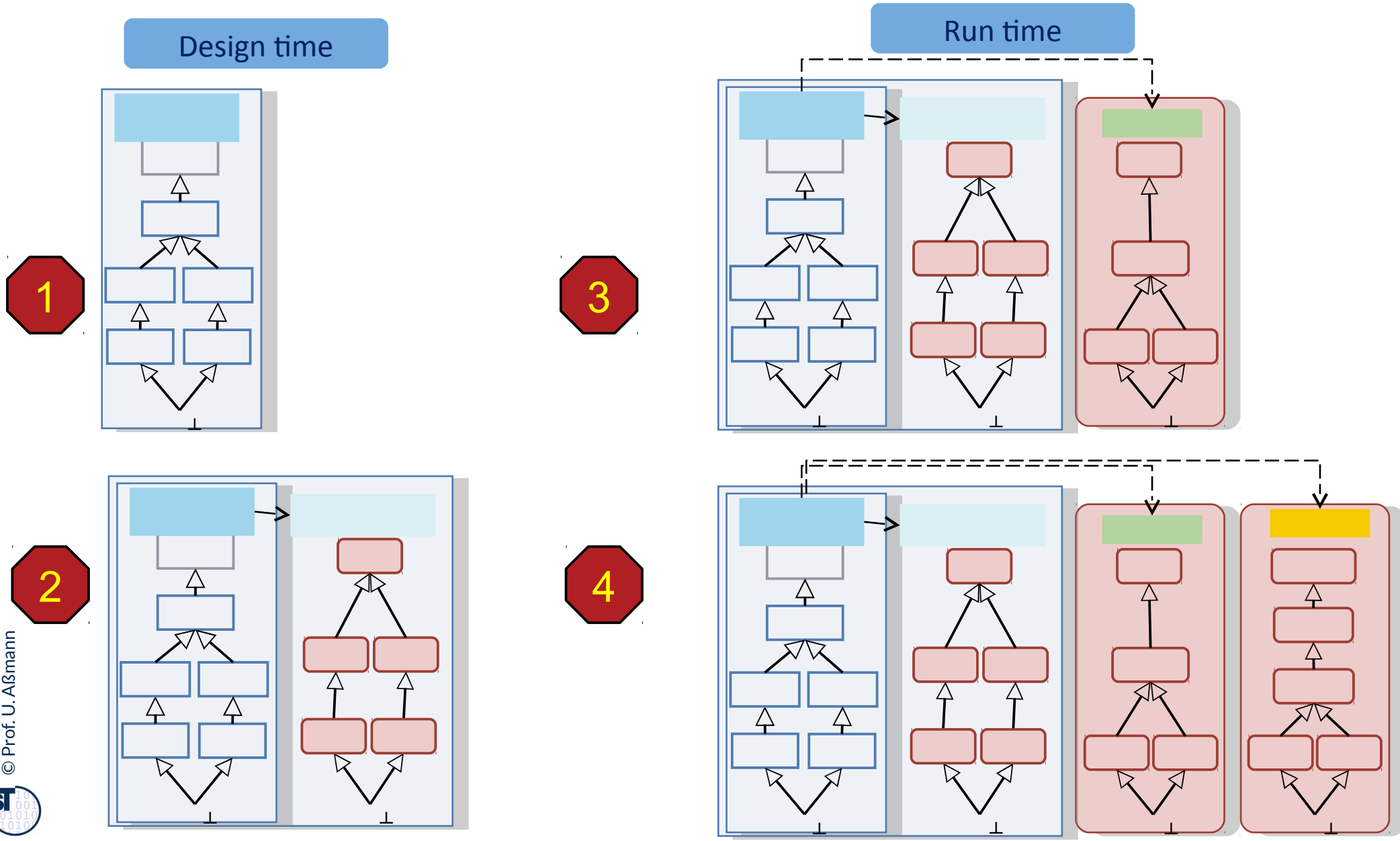
Scalable Bindung Time of Contexts with the Factorization

- **Scalable Binding**: Roles can also be bound statically, if mixins are used as implementation (fixing the context)
- Consequences for object life time, cohesion, allocation, adaptation, reconfiguration



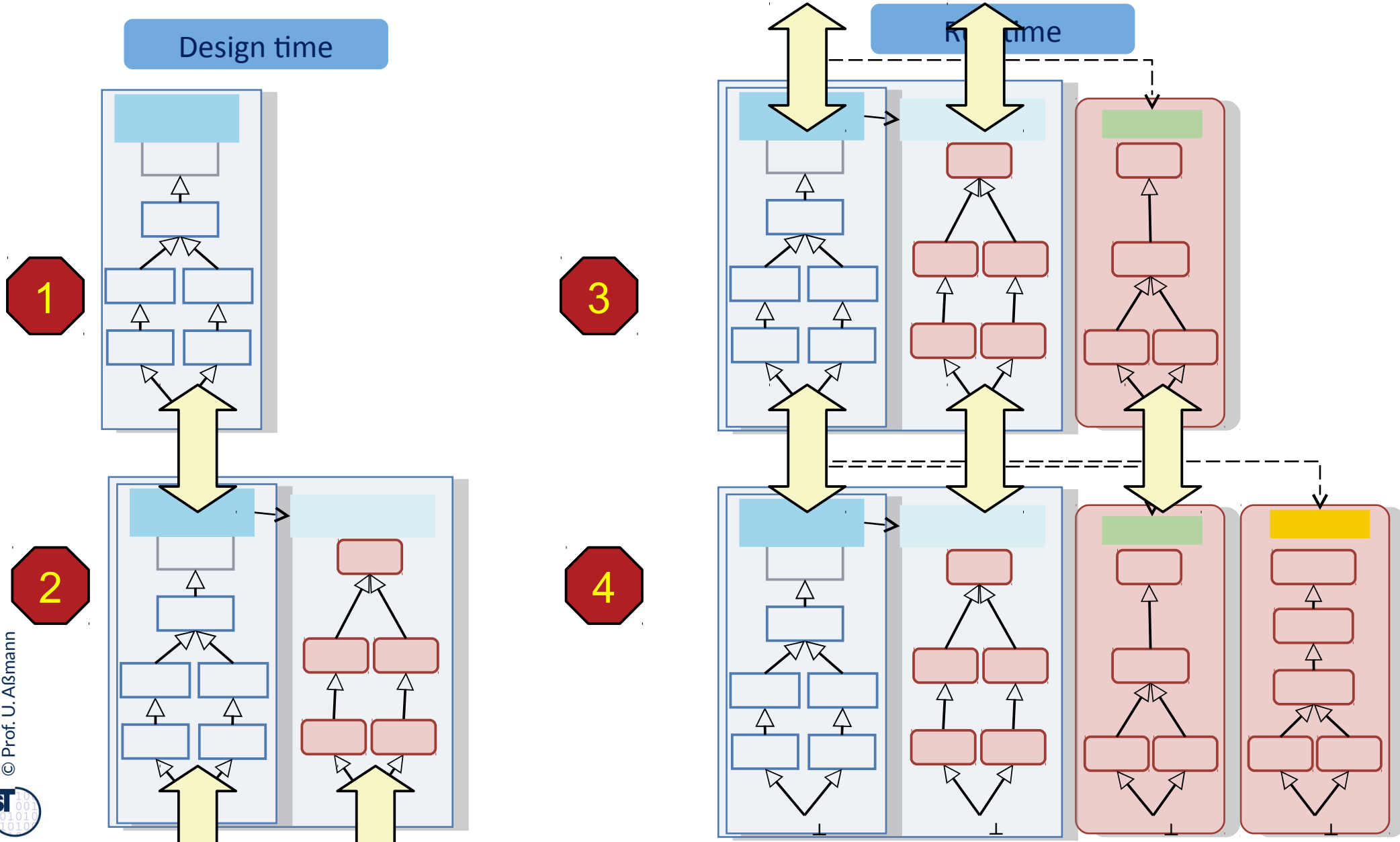
RoSI Megamodel (RoSIMa): Refinement by Role Allocation

- Refinement by allocation of roles – static roles at design time, dynamic roles at runtime



RoSIMa: Traceability in Refinement by Role Allocation

- Refinement by allocation of roles – static roles at design time, dynamic roles at runtime



RoSI Megamodel (RoSIMa): Cross-Layer Role-Based Refinement in the Software Life Cycle

- Refinement by allocation of roles provides **simple traceability** because Natural objects STAY the same
- Platform properties are „technical“ roles of the objects
  - Technical plattforms are static contexts
  - Dynamic contexts (place, time, service quality)

Causal Mapping of contexts and fludity From requirements level to runtime

Domain Model

Requirements

Design

PSM

Implementation

Run time context 1

Run time context 2

Run time context 3

Natural	Fixed Role 1	Fixed Role 2	Fixed Role 3	Fixed Role 4	Dynamic role 1	Dynamic role 2	Dynamic role 3
Person							
Person	Customer						
Person	Customer	Customer Design					
Person	Customer	Customer Design	Platform-specific Behavior				
Person	Customer	Customer Design	Platform-specific Behavior	Full static behavior			
Person	Customer	Customer Design	Platform-specific Behavior	Full static behavior	Behavior in Context 1		
Person	Customer	Customer Design	Platform-specific Behavior	Full static behavior	Behavior in Context 1	Behavior in Context 2	
Person	Customer	Customer Design	Platform-specific Behavior	Full static behavior	Behavior in Context 1	Behavior in Context 2	Behavior in Context 3

End

- Why do the models of MDA form a megamodel?
- Which trace link types are important for MDA?
- Why is a role-bsed model better for traceability?
- How does RoSIMa achieve global traceability from requirements to run time?
- How will megamodel look like that provides Link-tree-based models and Role-based factorization of objects?
  - How does a trace link look like?
  - Where are the trace links stored?
  - Why can XML be used as simple exchange format in these megamodels?