TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik - Institut Software- und Multimediatechnik - Softwaretechnologie

12. An Overview of Technical Spaces

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

Technische Universität Dresden

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

Version 19-0.3, 15.11.19

1) Technical spaces
2) Model Management
3) Model Analysis
4) Mega- and Macromodels
5) Pattern Languages on M2
6) Bridging Technical Spaces

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

12.1 Technological & Technical Spaces

Technological Spaces

A ***technological space*** is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities.

- It is often associated to a given user community with shared know-how, educational support, common literature and even workshop and conference regular meetings.
  - Ex. compiler community, database community, semantic web community, automotive community
  - [Technological Spaces: an Initial Appraisal. Ivan Kurtev, Jean Bézivin, Mehmet Aksit. CoopIS, DOA'2002 Federated Conferences, Industrial Track. (2002) http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.332&rep=rep1&type=pdf]

Literature

- Christopher Brooks, Chihhong Patrick Cheng, Thomas Huining Feng, Edward A. Lee, Reinhard von Hanxleden. Model Engineering using Multimodeling. Electrical Engineering and Computer Sciences University of California at Berkeley.
  - Technical Report No. UCB/EECS-2008-39 http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-39.html
- Rick Salay, John Mylopoulos, and Steve M. Easterbrook. Using macromodels to manage collections of related models. In Pascal van Eck, Jaap Gordijn, and Roel Wieringa, editors, Advanced Information Systems Engineering, 21st International Conference, CAiSE 2009, Amsterdam, The Netherlands, June 8-12, 2009. Proceedings, volume 5565 of Lecture Notes in Computer Science, pages 141--155. Springer, 2009. [bib]
- Rick Salay, Shige Wang, and Vivien Suen. Managing related models in vehicle control software development. In Robert B. France, Jürgen Kazmeier, Ruth Breu, and Colin Atkinson, editors, Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings, volume 7590 of Lecture Notes in Computer Science, pages 383--398. Springer, 2012.
- Jean-Marie Favre and Tam Nguyen. Towards a megamodel to model software evolution through transformations. Electr. Notes Theor. Comput. Sci, 127(3):59--74, 2005.

Technical Spaces

A ***technical space*** is a metamodeling framework (in a technological space) with a metapyramid (metahierarchy), accompanied by a set of tools that operate on the models definable within the framework.

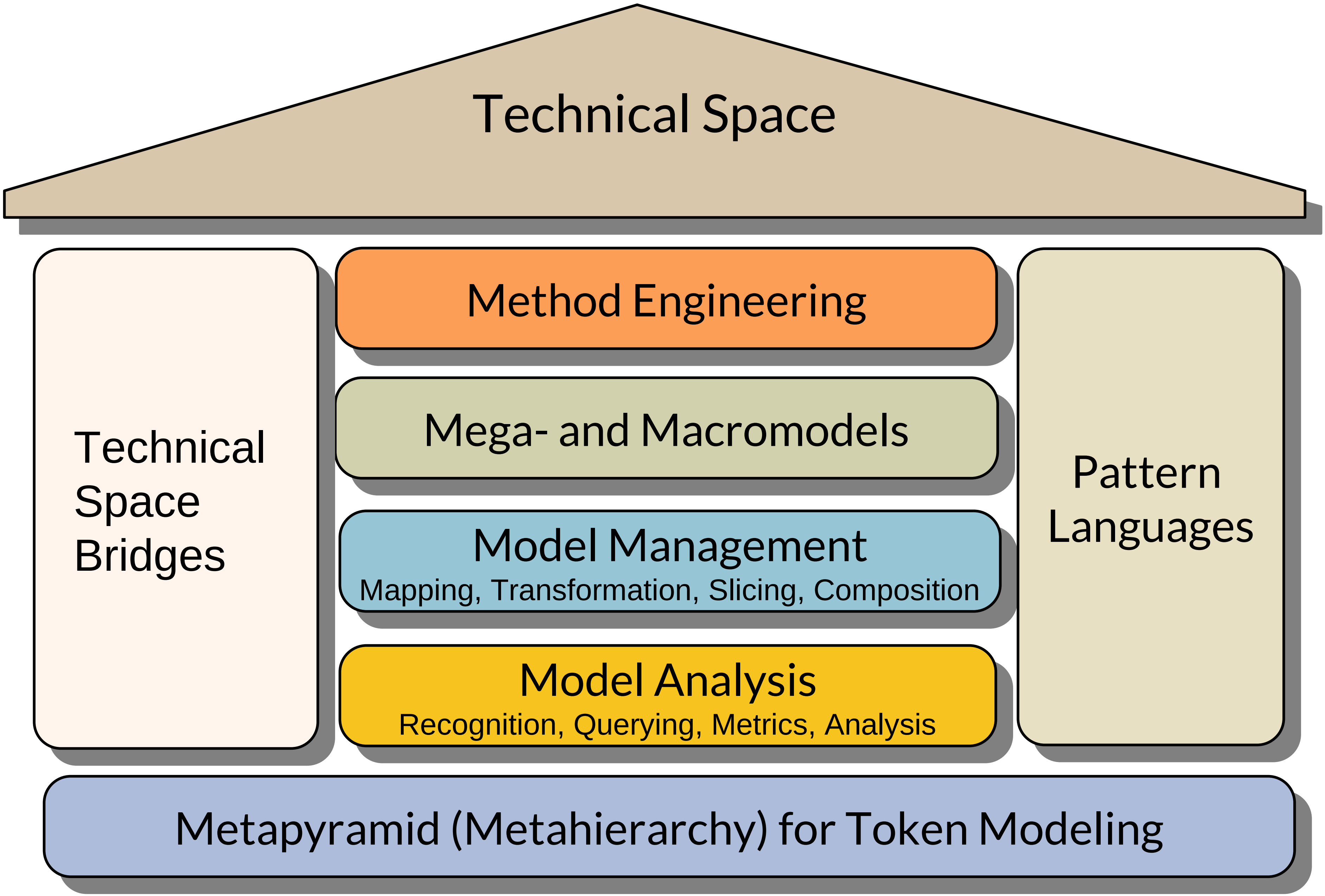
- [Model-based Technology Integration with the Technical Space Concept. Jean Bezivin and Ivan Kurtev. Metainformatics Symposium, 2005.]
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1366&rep=rep1&type=pdf
- Ingredients of a Technical Space (Technikraum):
  - A **metapyramid** (or **metahierarchy**) with data (tools, workflows, and materials on M0), Code and models (M1), languages (M2), and metalanguages (M3)
  - A **model management unit** (**model algebra** or **model composition system**)
  - A **macromodel**
- Be aware: A technological space may contain several technical spaces:
  - Compiler community: Grammarware, Tree-Ware, Graph-Ware
  - Database community: Relational database model, csv-tables, XML
  - Business software: Reports in TextWare, TableWare

The Trick of the Metapyramid

Observation:
In the metapyramid of a technical space, tools can be applied on every level.

- **Level-independence:** Tools on level M[n-1] can work on M[n]
- Tools can be *lifted* from the object to the class to the metaclass level to the metametaclass level:
- **Object-manipulating tools** on M0 work for clabjects in models on M1
  - Graph-manipulating tools on M0 for models on M1
- **Class-manipulating tools** on M1 work for clabjects in metamodels on M2
  - Model-manipulating tools on M1 work for metamodels on M2
- **Metaclass-manipulating tools** on M2 work for clabjects in metamodels on M3
  - Metamodel-manipulating tools on M2 work for metametamodels on M3

Q10: The House of a Technical Space



Packeting on all Layers

- All layers can be structed into packages

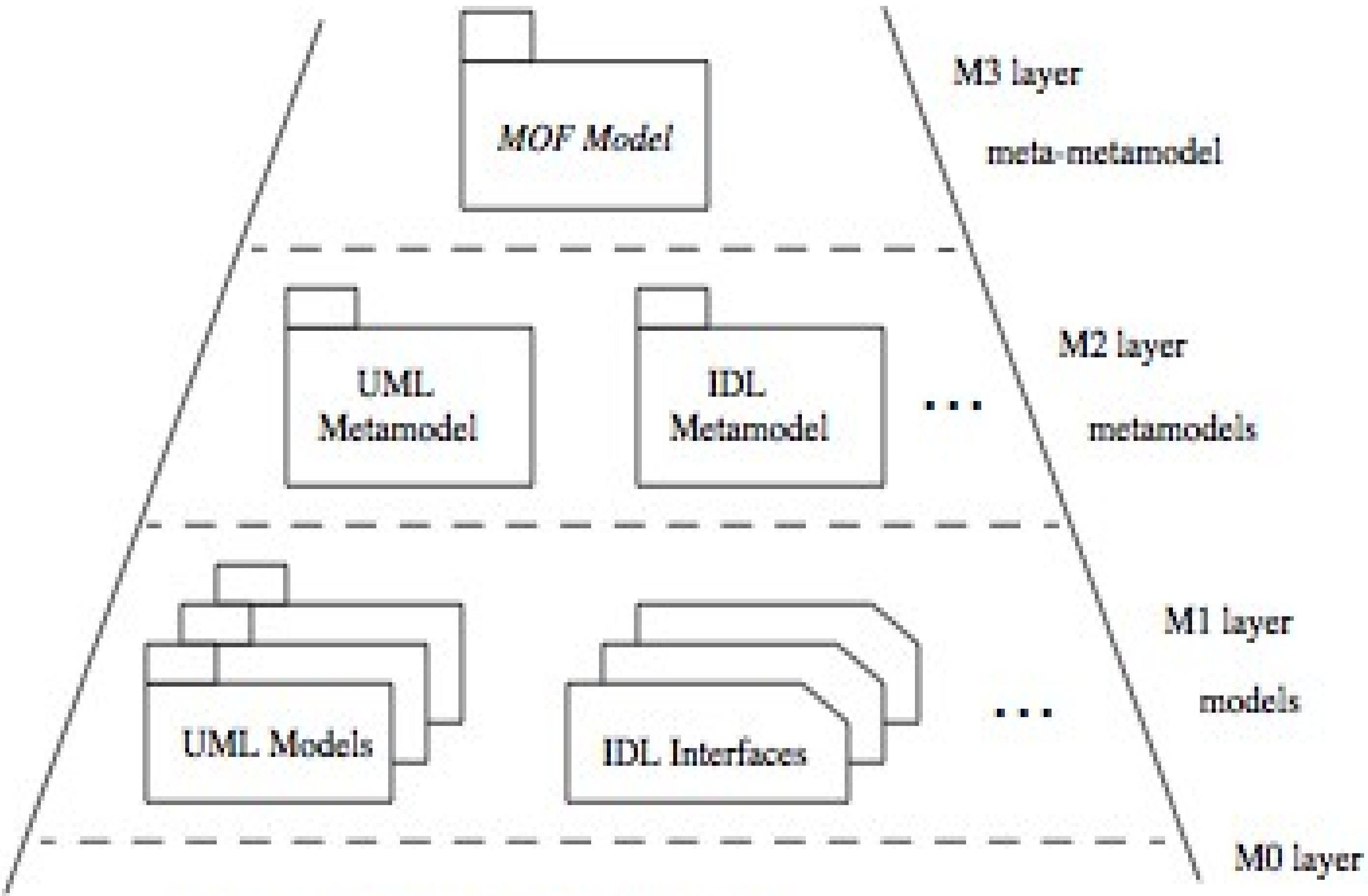


Figure 2-2 MOF Metadata Architecture

[MOF]

Q10: Overview of Technical Spaces in the Classical Metahierarchy

	Grammarware (Strings)	Text-ware	Table-ware		Treeware (trees)			Graphware/ Modelware			Role-Ware	Ontology-ware
	Strings	Text	Text-Table	Relational Algebra	NF2	XML	Link trees	MOF	Eclipse	CDIF	MetaEdit+	OWL-Ware
M3	EBNF	EBNF		CWM (common warehouse model)	NF2-language	XSD	JastAdd, Silver	MOF	Ecore, EMOF	ERD	GOPPR	RDFS OWL
M2	Grammar of a language	Grammar with line delimiters	csv-header	Relational Schema	NF2-Schema	XML Schema, e.g. xhtml	Specific RAG	UML-CD, -SC, OCL	UML, many others	CDIF-languages	UML, many others	HTML XML MOF UML DSL
M1	String, Program	Text in lines	csv Table	Relations	NF2-tree relation	XML-Documents	Link-Syntax-Trees	Classes, Programs	Classes, Programs	CDIF-Models	Classes, Programs	Facts (T-Box)
M0	Objects	Sequences of lines	Sequences of rows	Sets of tuples	trees	dynamic semantics in browser		Object nets	Hierarchical graphs	Object nets	Object nets	A-Box (RDF-Graphs)

12.2. Model Analysis in a Technical Space with Model Querying, Model Metrics, and Model Analysis

Discussing the internals of models and their model elements

The Internals of a Model

Model analysis techniques reveal the inner details of models.

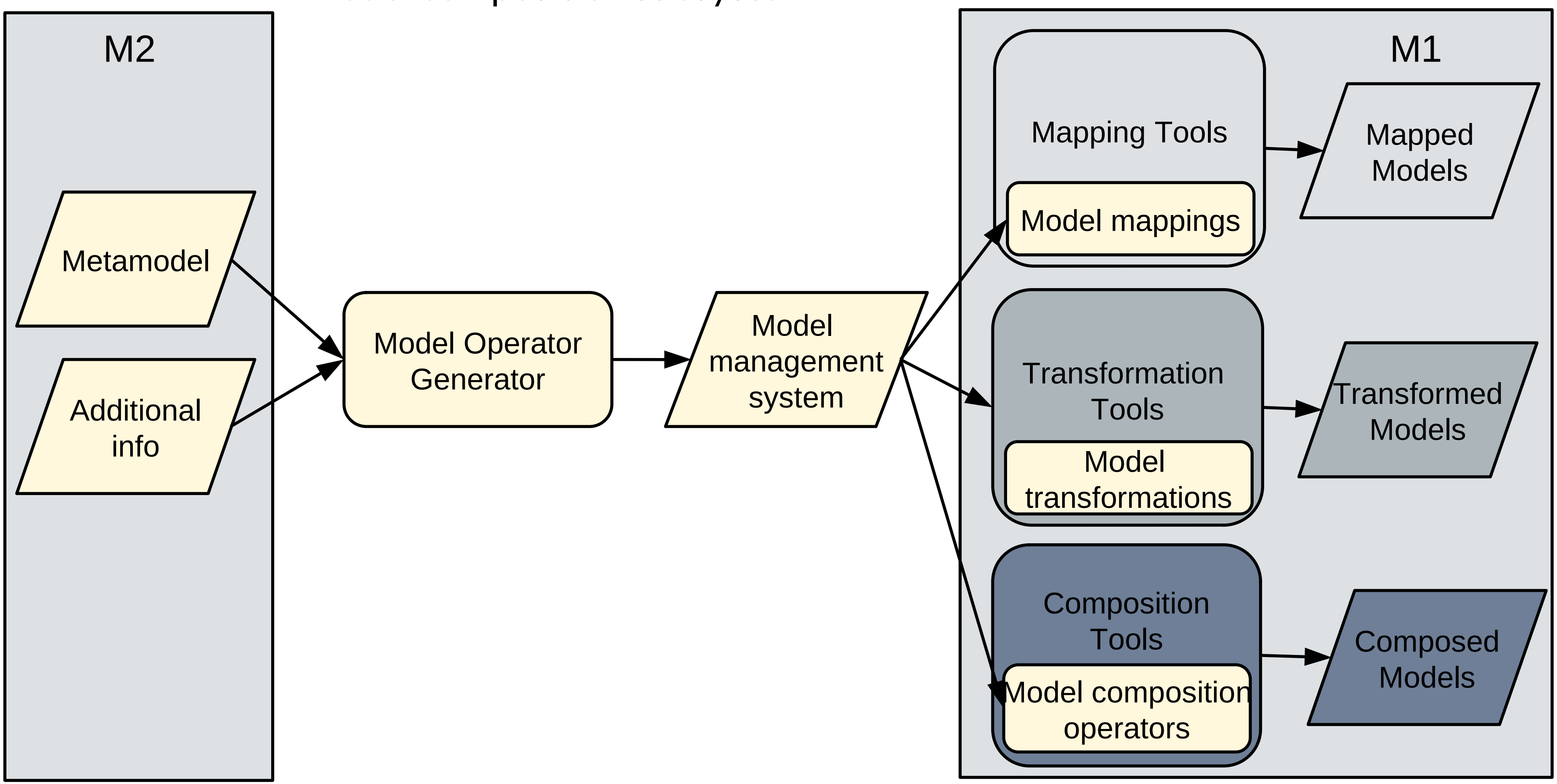
- **Model querying** searches patterns in models, described by a query or pattern match expression.
  - Searching for a method with a specific set of parameters
- **Model metrics** counts patterns in models
  - Counting the depth of the inheritance hierarchy
- **Model analysis** analyzes hidden knowledge from the models, making implicit knowledge explicit
  - Value flow analysis between variables in programs

12.3. Model Management in a Technical Space with Model Mapping, Transformation and Composition

Discussing the relationships of models and their model elements

Model Management in a Technical Space

- A ***model management system*** *manages the relationships of models*, metamodels, metametamodels of a technical space as well as the relationships of their elements
  - Model mapping subsystem
  - Model transformation subsystem
  - Model composition subsystem

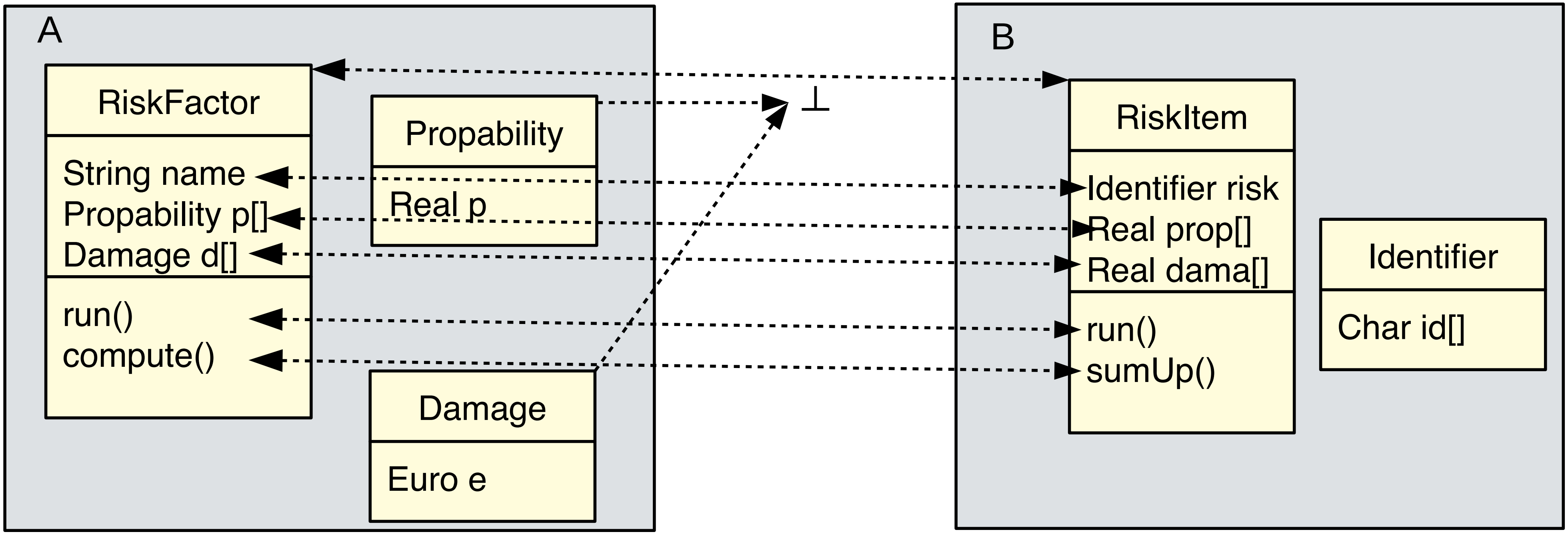


12.3.1. Model Mapping

Model Mappings

A ***model mapping*** is a mapping between the model elements of several models.

- A **trace mapping** records during a model elaboration, model restructuring or model transformation, *which model elements are copied from model A to model B, or created in B.*
- A **synchronization mapping** records *hot-links* model elements from model A to model B.

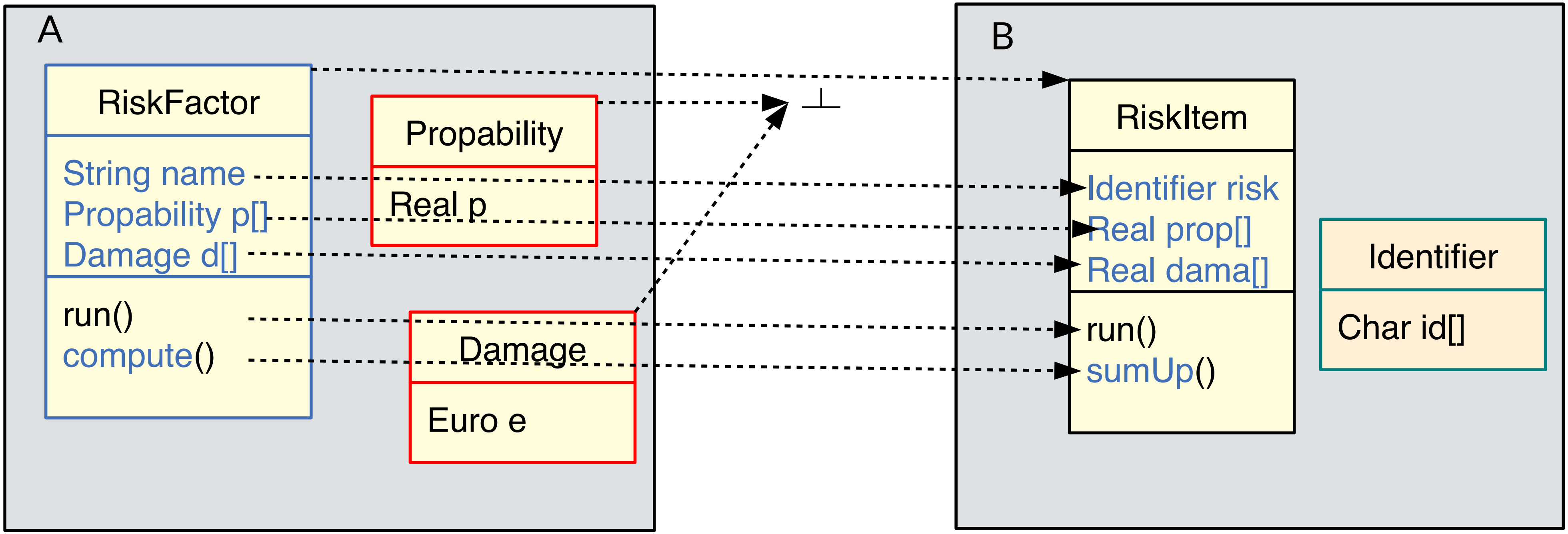


12.3.2. Model Transformation

Model Transformations

A ***model transformation*** is a program (or a specification how) to derive a model A from a model B.

- From a model mapping, two (partial) model transformations (forward and backward) may be derived.
- Deleted model elements are framed red, added elements are framed green, modified blue

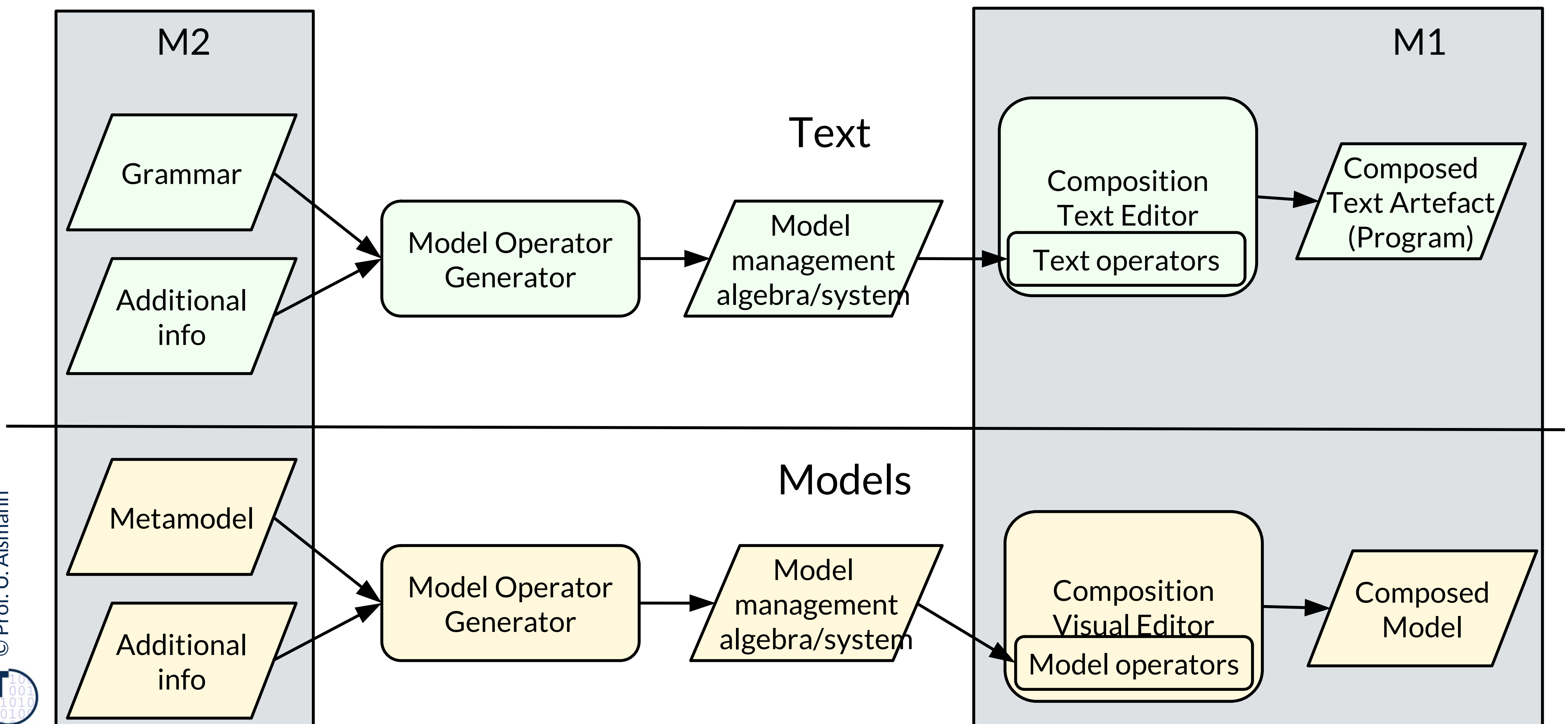


12.3.3. Model Composition with Model Algebrae and Composition Systems

Component-based Model Engineering (CBME)

Model Composition in a Technical Space

- A ***model composition system*** manages the relationships of models, metamodels, metametamodels of a technical space with a uniform model algebra
  - Operators on M1 can be generated from M2
  - Operators on M2 can be generated from M3

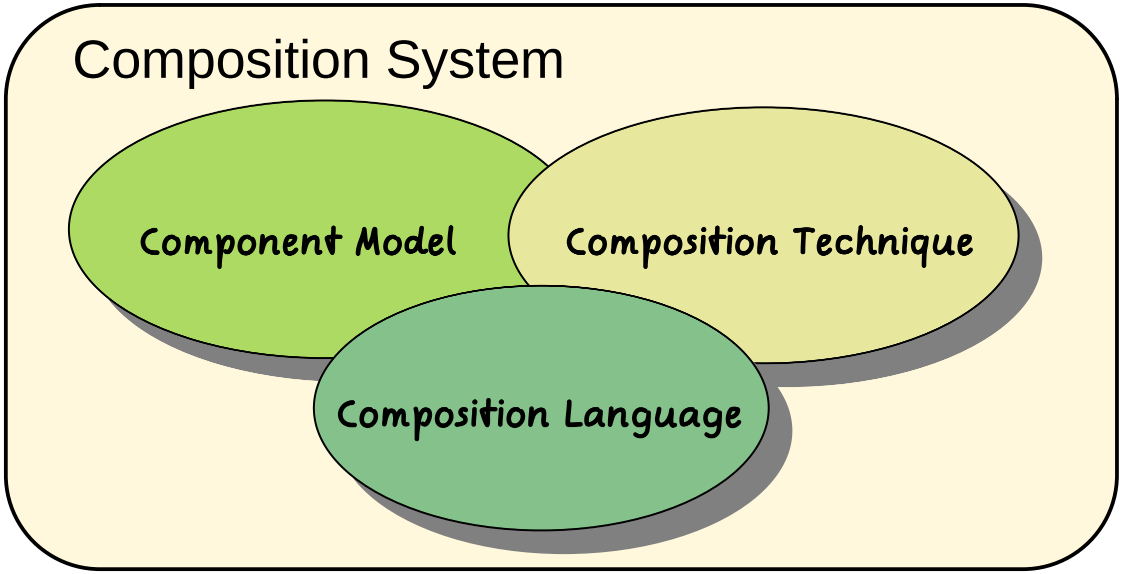


Simple Algebra for Models (on M1) and Metamodels (on M2)

- The most simple composition systems are *algebrae*, resulting in *algebraic composition.*
  - Models and metamodels can be grouped in packages (module)
  - A simple component model and composition system (see CBSE)
- Algebraic composition technique with operators on packages:
  - use (import) | merge (union) | Instance-of (element-of-reified-set)

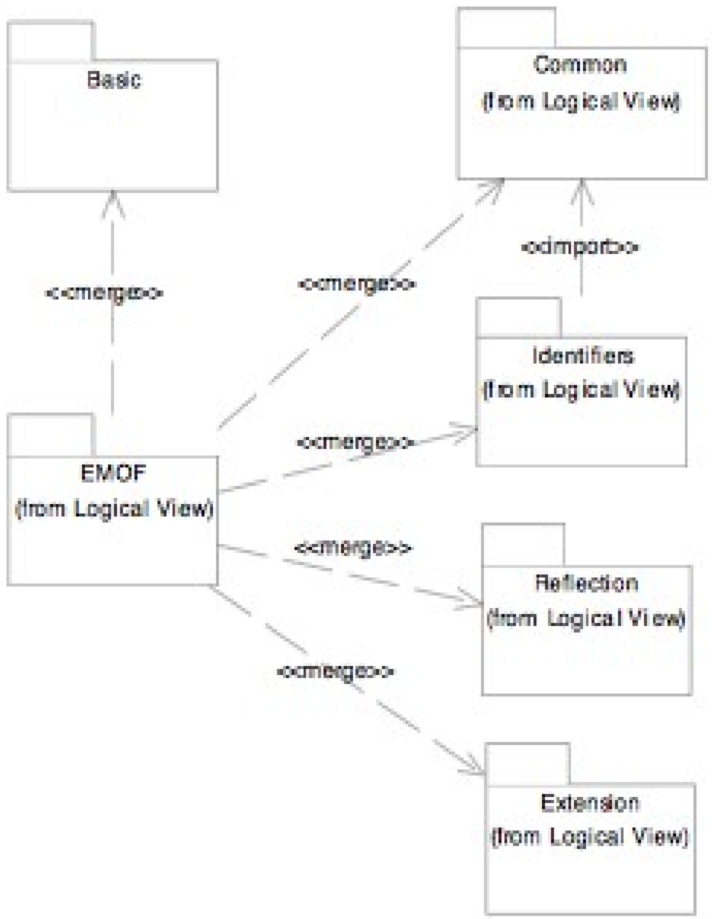
→ Metamodels are composed by unifying their views in the different packages

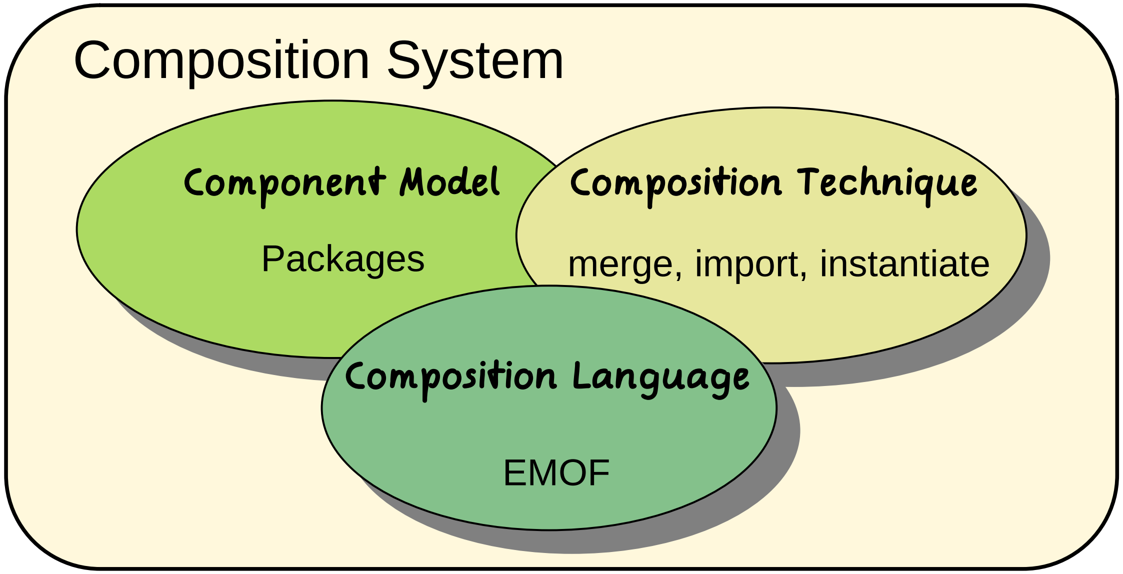
→ Metamodels can be composed from packages



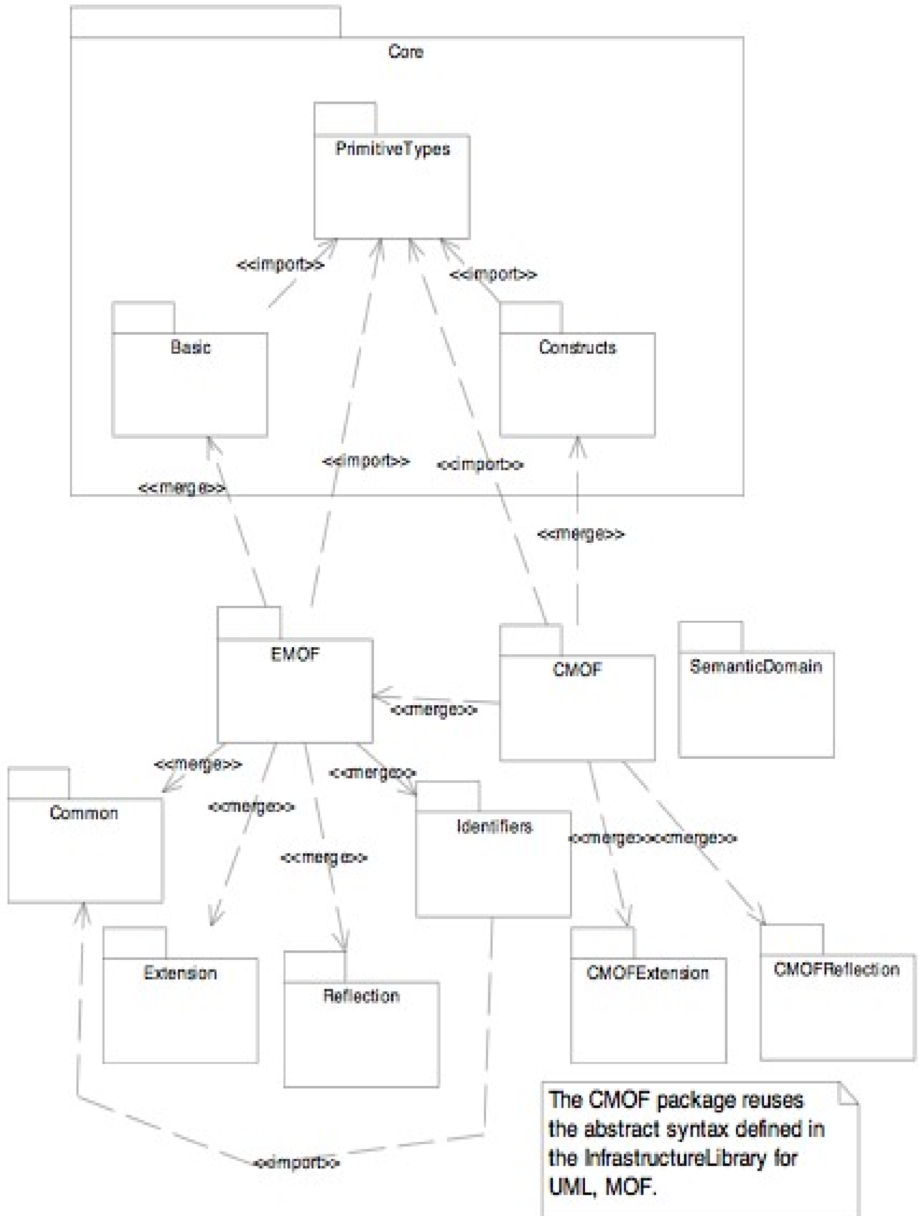
Ex.: EMOF Class Composition by EMOF Package Merge

[MOF]





Ex: CMOF Package Composition from UML Core and EMOF



[MOF]

12.3.4. Composing UML Metamodels in the MOF Technical Space

Benefit of UML-Metamodeling for MDSD Tools and Model-Driven Applications

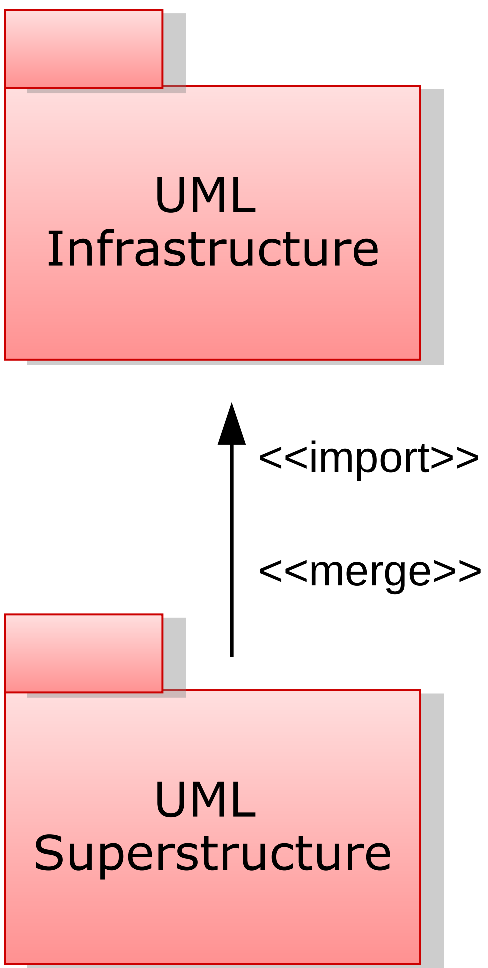
The language report of UML uses a simple metamodel algebra for the bottom-up composition of UML language.

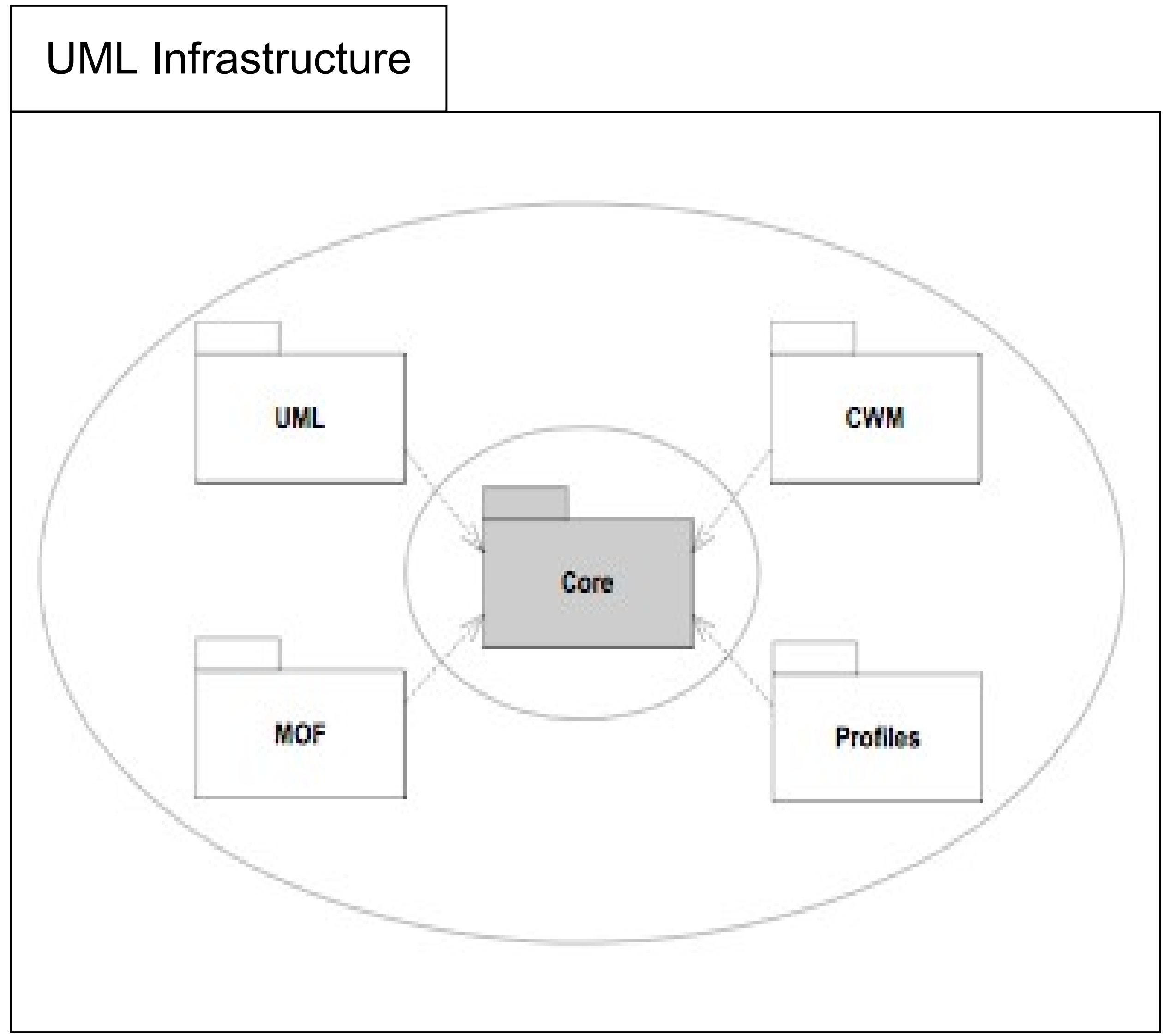
The UML-metamodel is a “logic” metamodel, because it is *composed:*

- Definition of merge operator composing metaclasses and metaclass-packages
- Defined in composable **packages**
  - With a clear **CMOF**-package architecture
  - uniform **package structure** and context-sensitive semantics for all diagrams such as Statecharts (UML-SC), Sequence Diagrams (UML-SD), etc.
- **Schemata for repositories** for uniform description of tools, materials, code, models (**metamodel-driven repositories)**
- **Exchange format** (XMI)

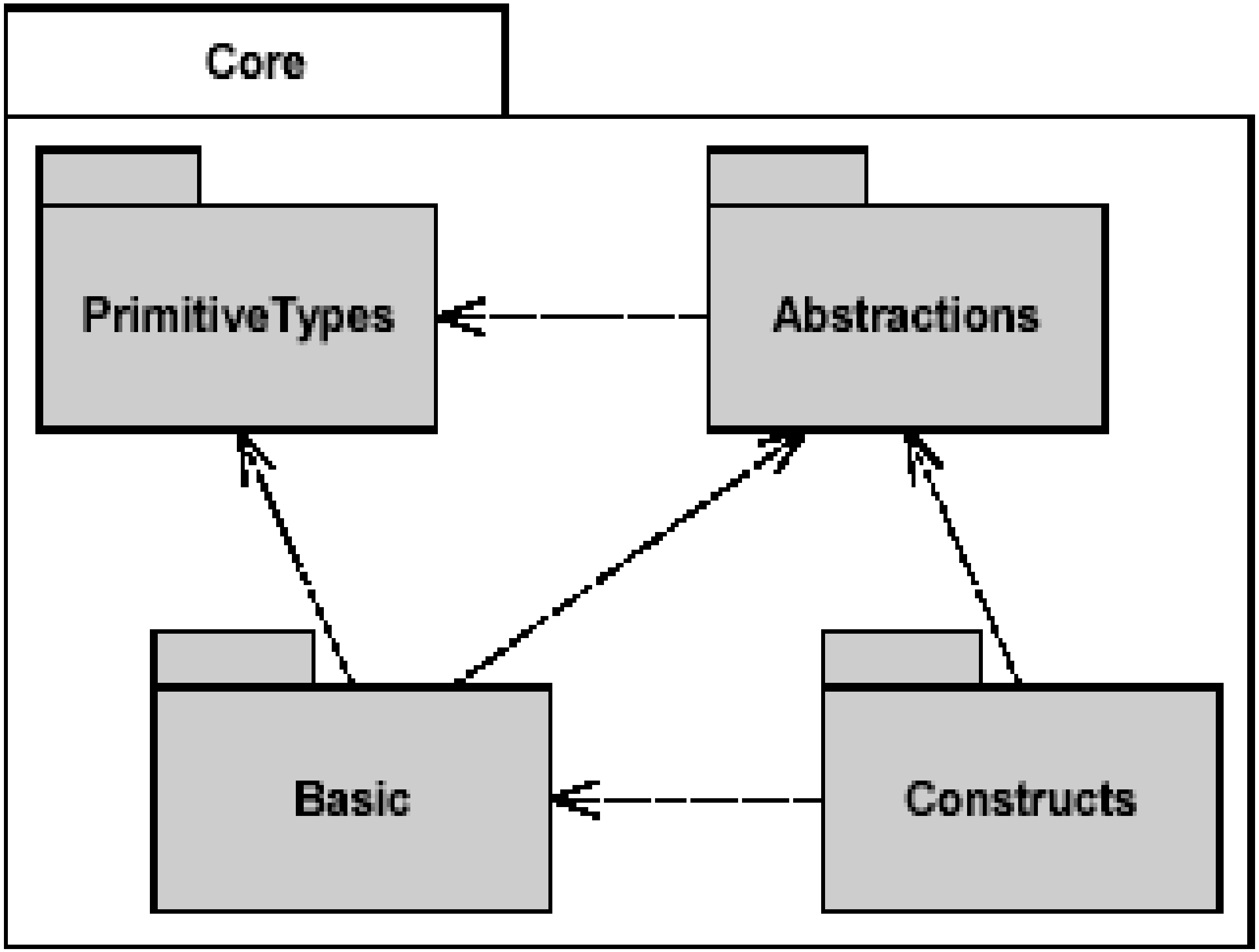
- The UML infrastructure can be used by MDSD applications

Coarse-Grain Structure of UML on M2





Core Package of the UML-Infrastructure Metamodel (M2)

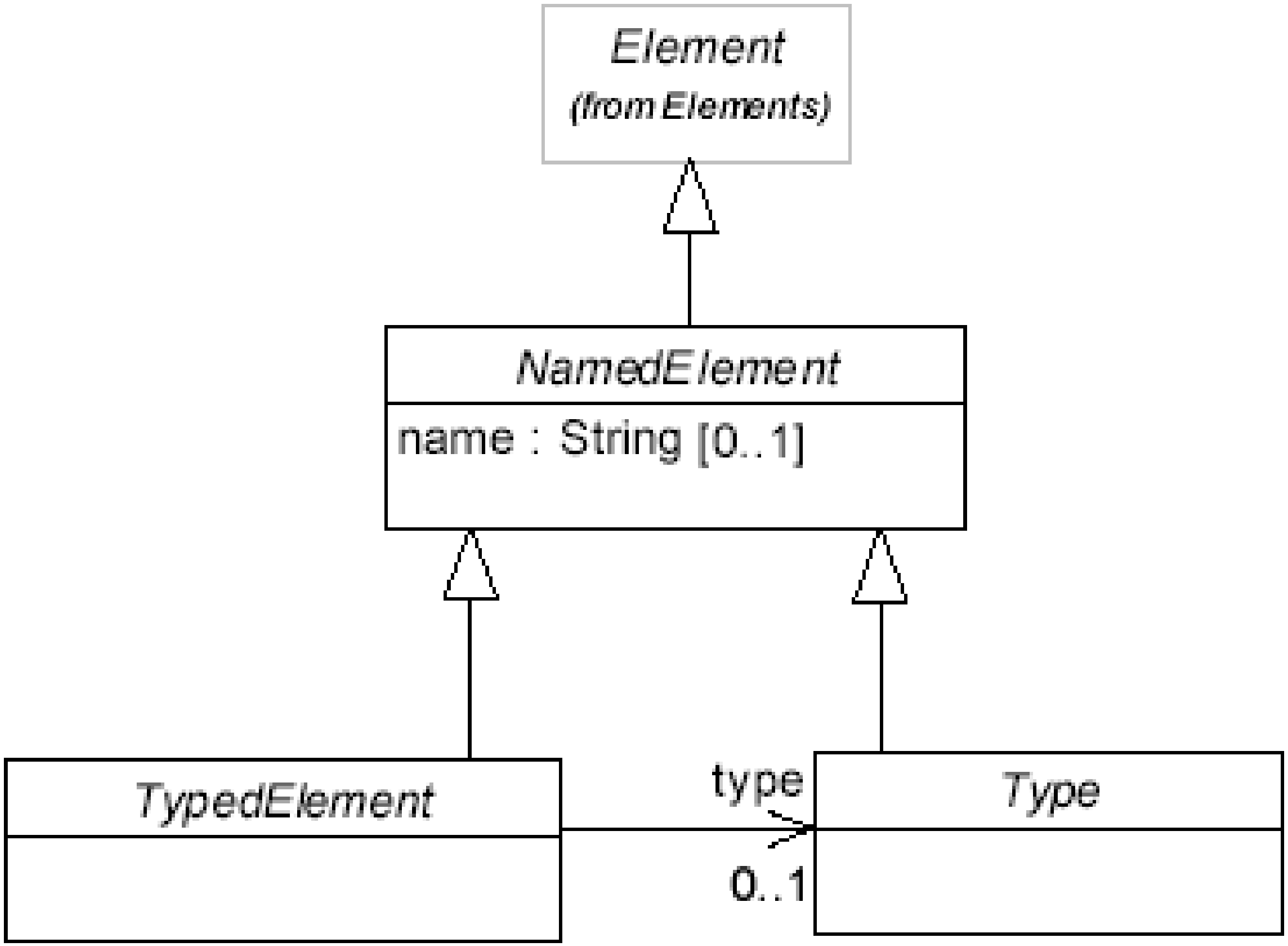


Basic: basic constructs for XMI
Constructs: Metaclasses for modeling

Abstractions: abstract metaclasses
Primitive Types: basic types

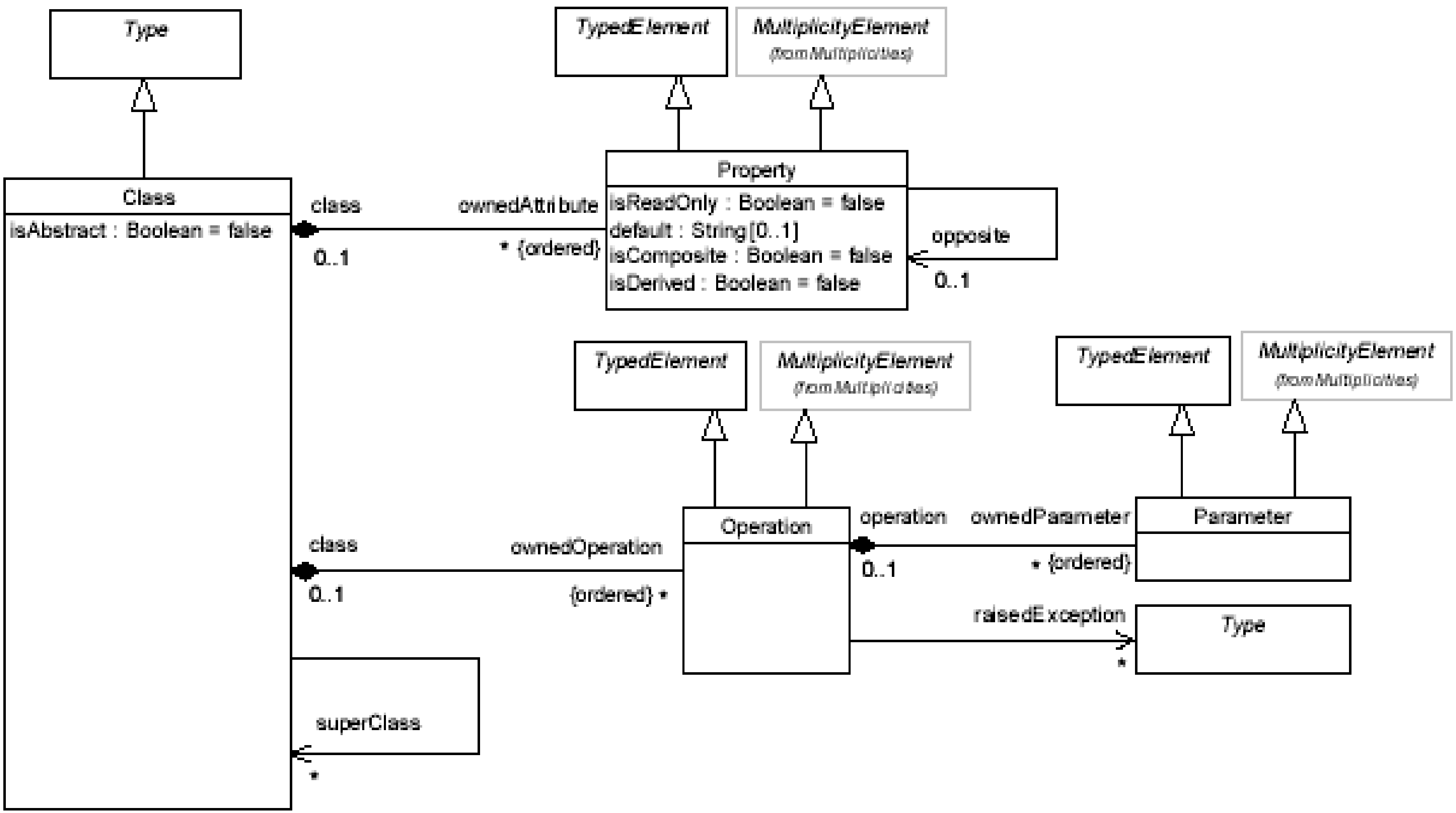
From: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

Package Basic: Uses Types from CMOF



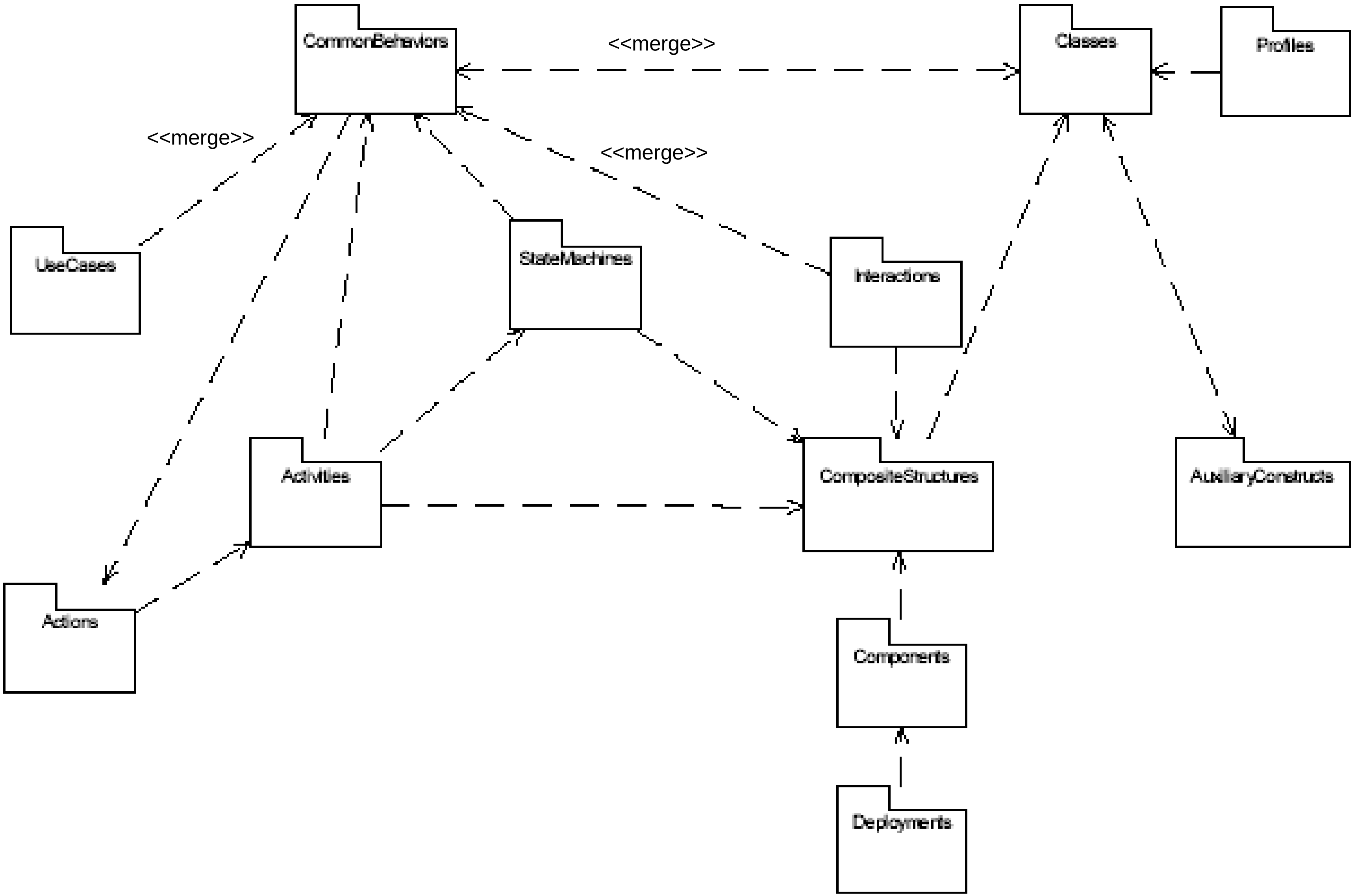
From: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

Package Basic: Classes



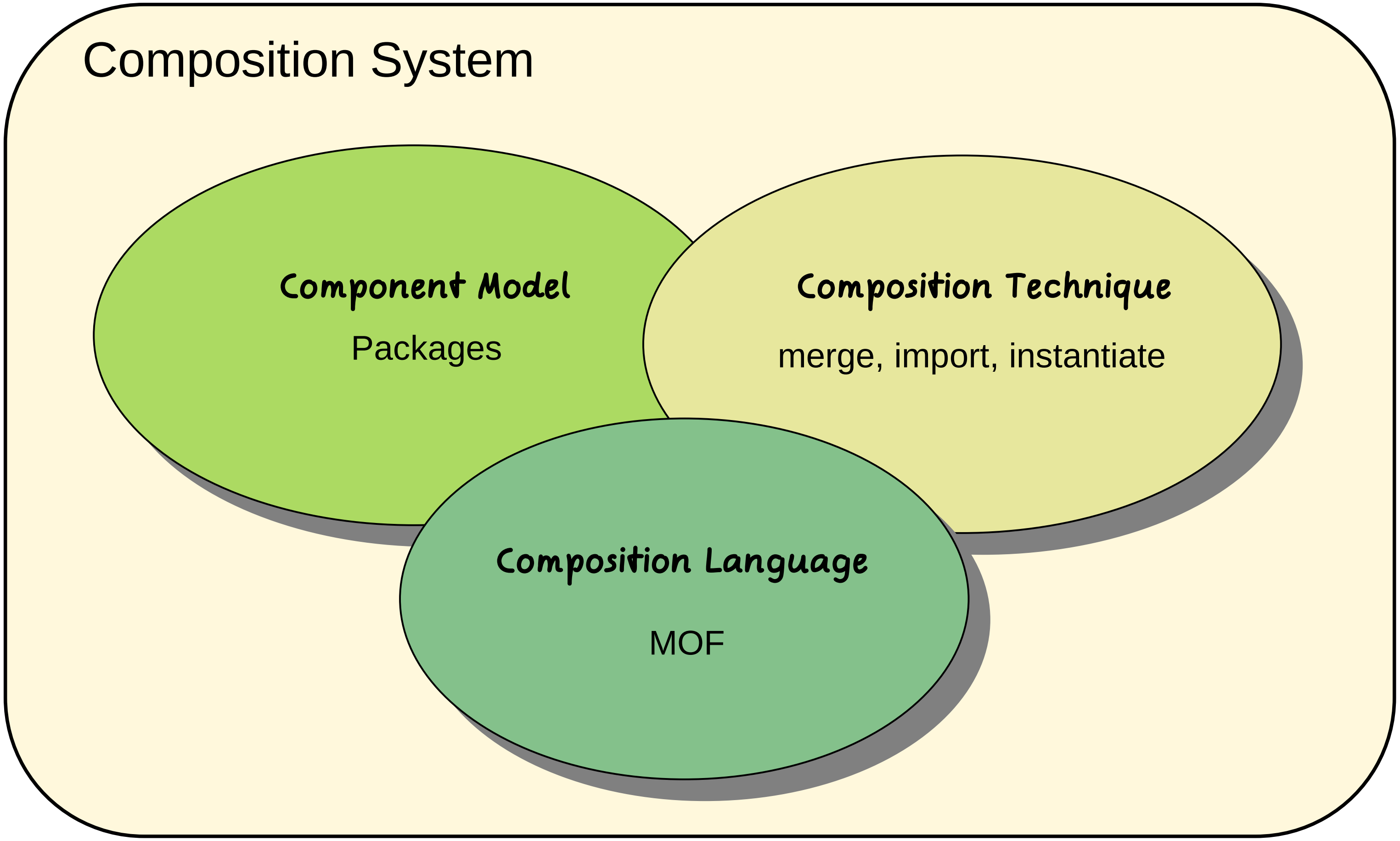
From: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

Package Composition Architecture UML 2.0 (M2)



From: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

Metamodel Composition – the Composition System of the UML Language Report



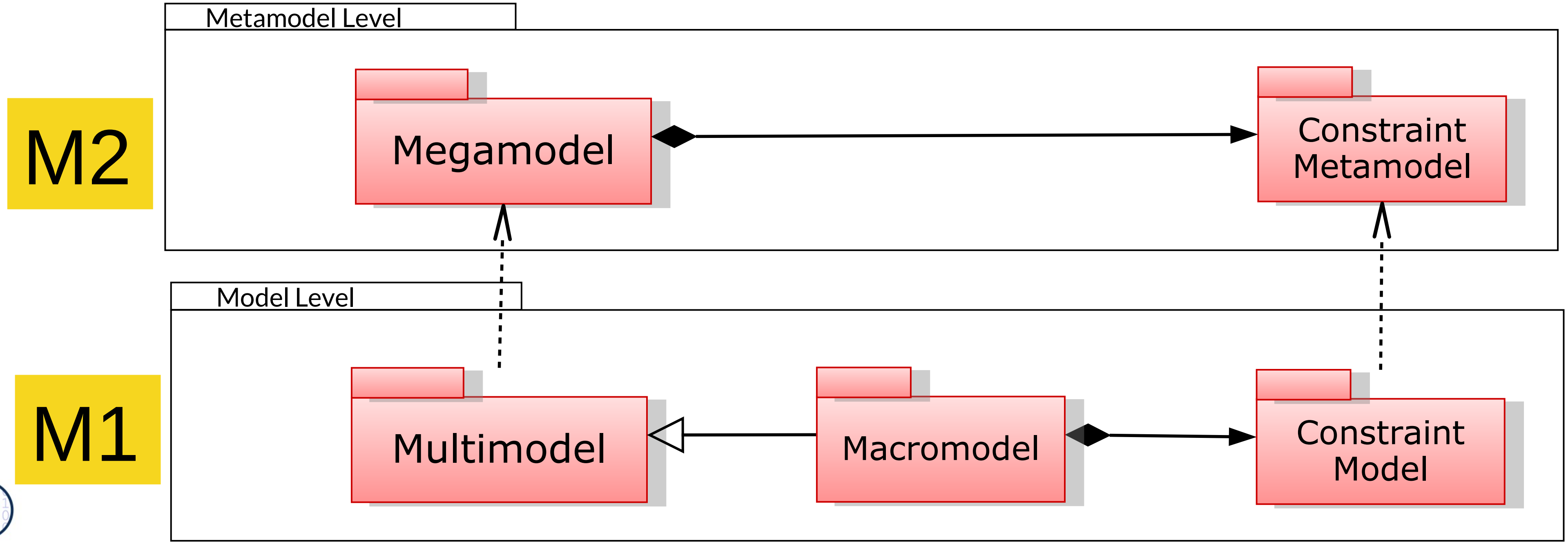
12.4 Mega- and Macromodels

In a technical space, a ***megamodel*** is an infrastructure for models and metamodels, systematically linking a set of models

Megamodels

The idea behind a ***mega-model*** is to define the set of entities and relations that are necessary to model some aspect about model-driven engineering (MDE). [Favre]

- A ***multimodel*** is a set or graph of related models.
- A ***megamodel*** is a model for a multi-model.
  - The multimodel is an instance of the megamodel (element of the of the megamodel's language)
- Usually, a technical space has one or several megamodels on M1, linking many models on M1
  - Clearifying the relationships of the M1 models by model transformations, model mappings, and model compositions
  - A megamodel uses the model management system of the technical space

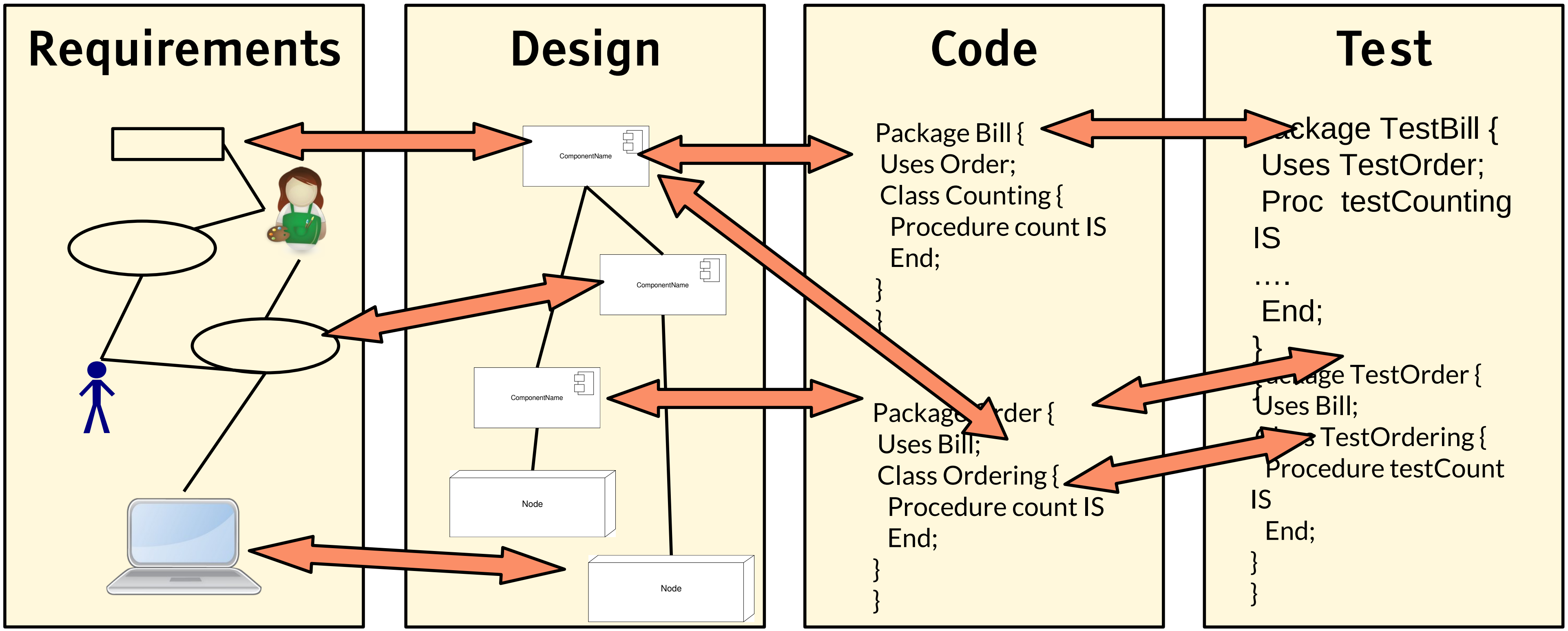


Macromodels – Multimodels with Consistency Rules

- A ***macromodel*** is a multimodel *fulfilling some consistency constraints over the models and their elements.*
  - The megamodel is adorned with a constraint metamodel
  - The graph of models obeys wellformedness constraints
  - There are **fine-grained relations** between model elements of the models, which also follow *consistency constraints*
    - **Trace mappings** between tools, materials, automata
    - **Synchronization relations** for updating

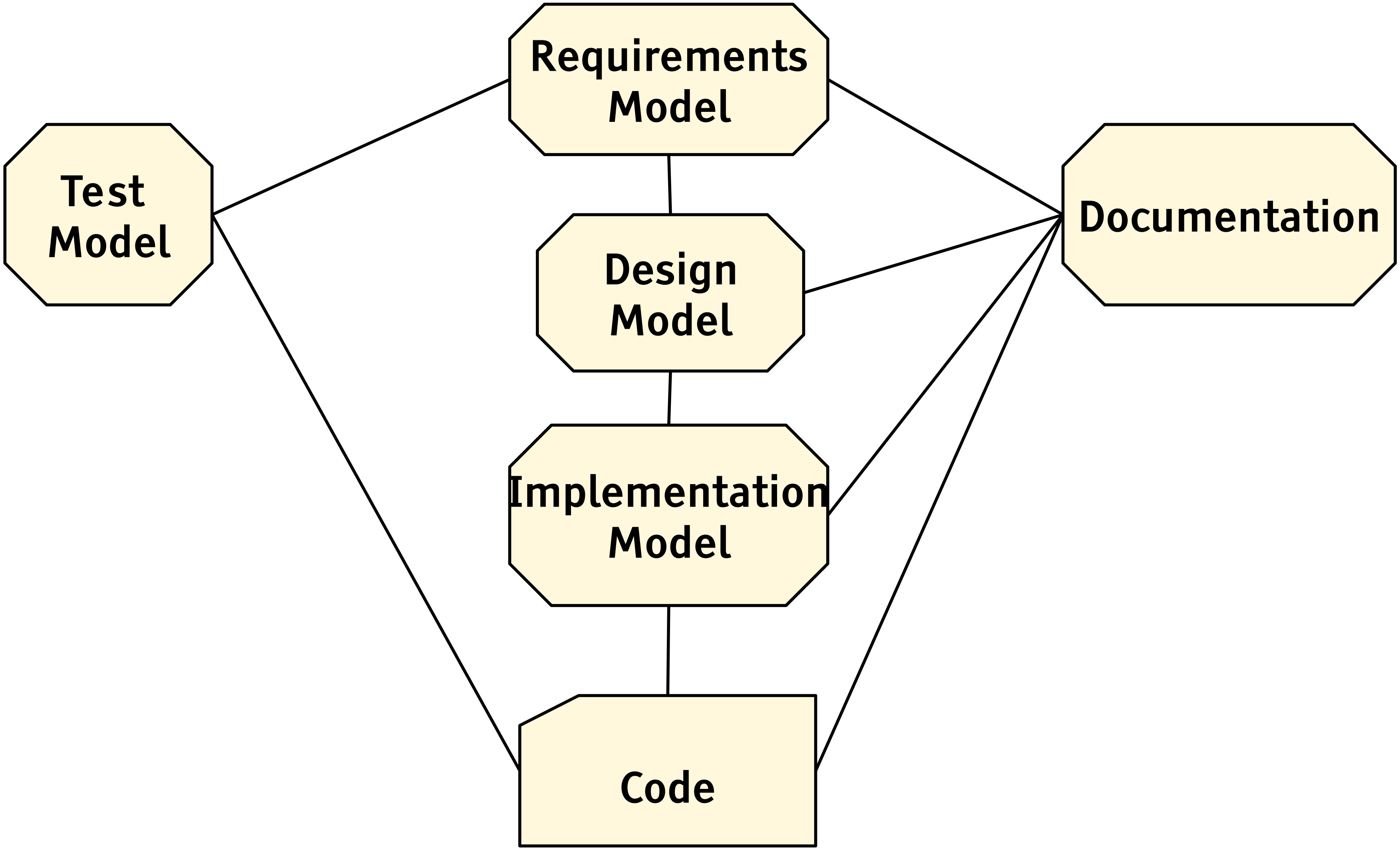
Q12: The ReDeCT Problem and its Macromodel

- The **ReDeCT problem** is the problem how requirements, design, code and tests are related (→ V model)
- Mappings between the Requirements model, Design model, Code, Test cases
- A **ReDeCT macromodel** has maintained mappings between all 4 models



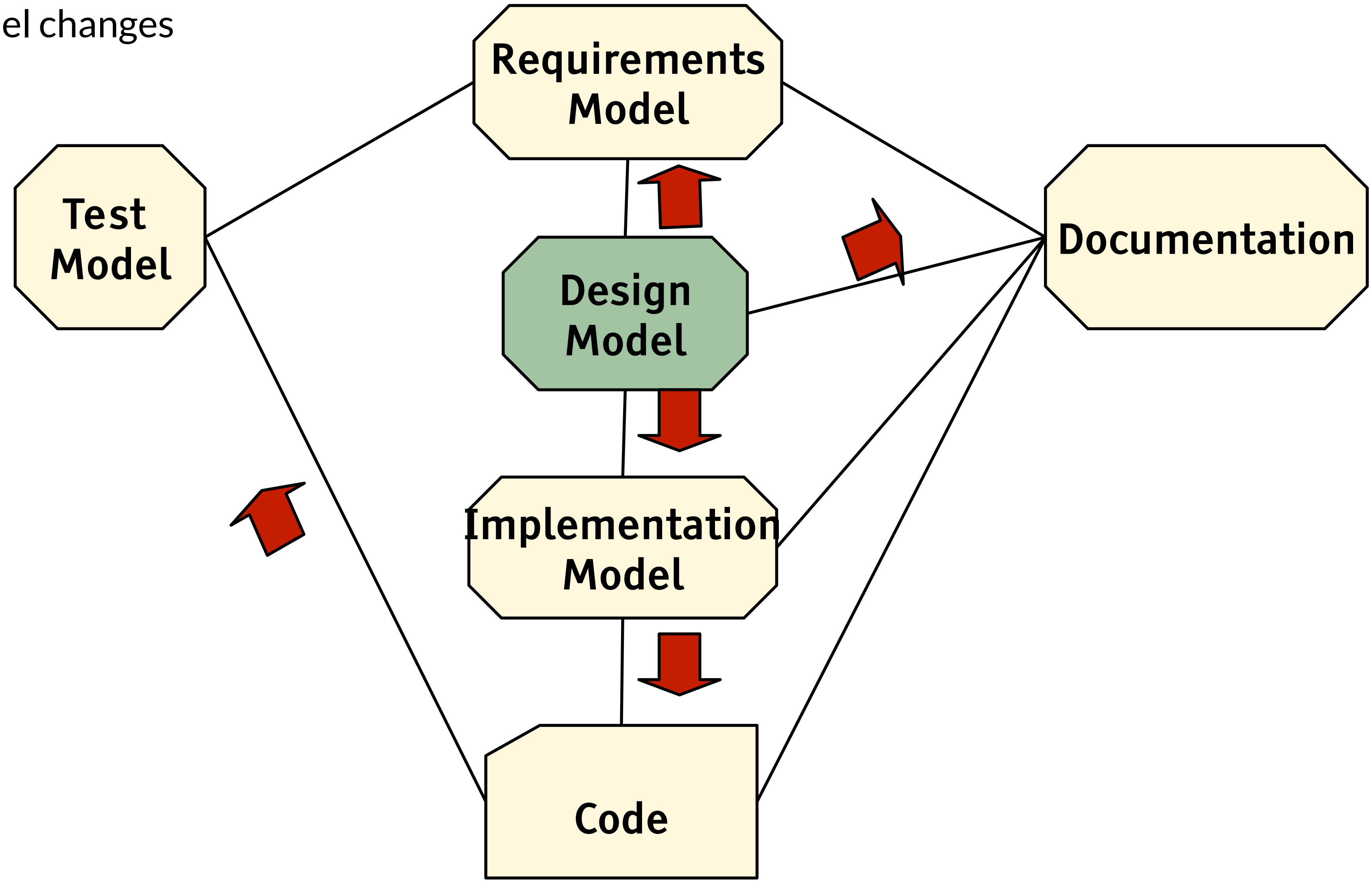
Model Synchronization in Macromodels

- ***Model synchronization*** keeps a set of connected models (the ***crowd***) *in* sync, i.e., consistent



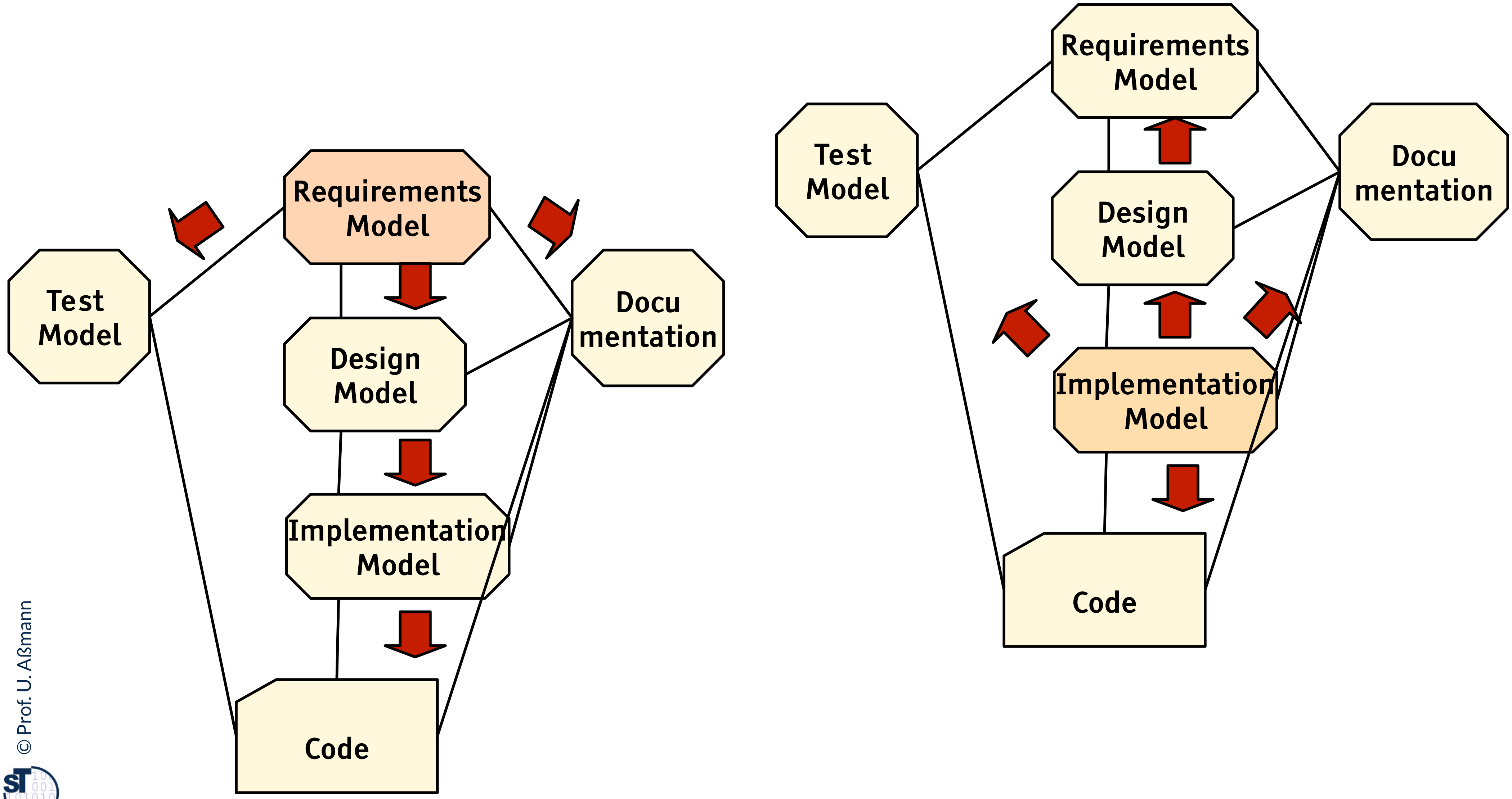
Model Synchronization in Macromodels

- In model synchronizsation, if an edit has occurred in a ***origin model***, all other connected models of a crowd (***dependent models***) are updated instantaneously, when one focus model changes



Round-Trip Engineering Changes the Model-in-Focus of the Crowd

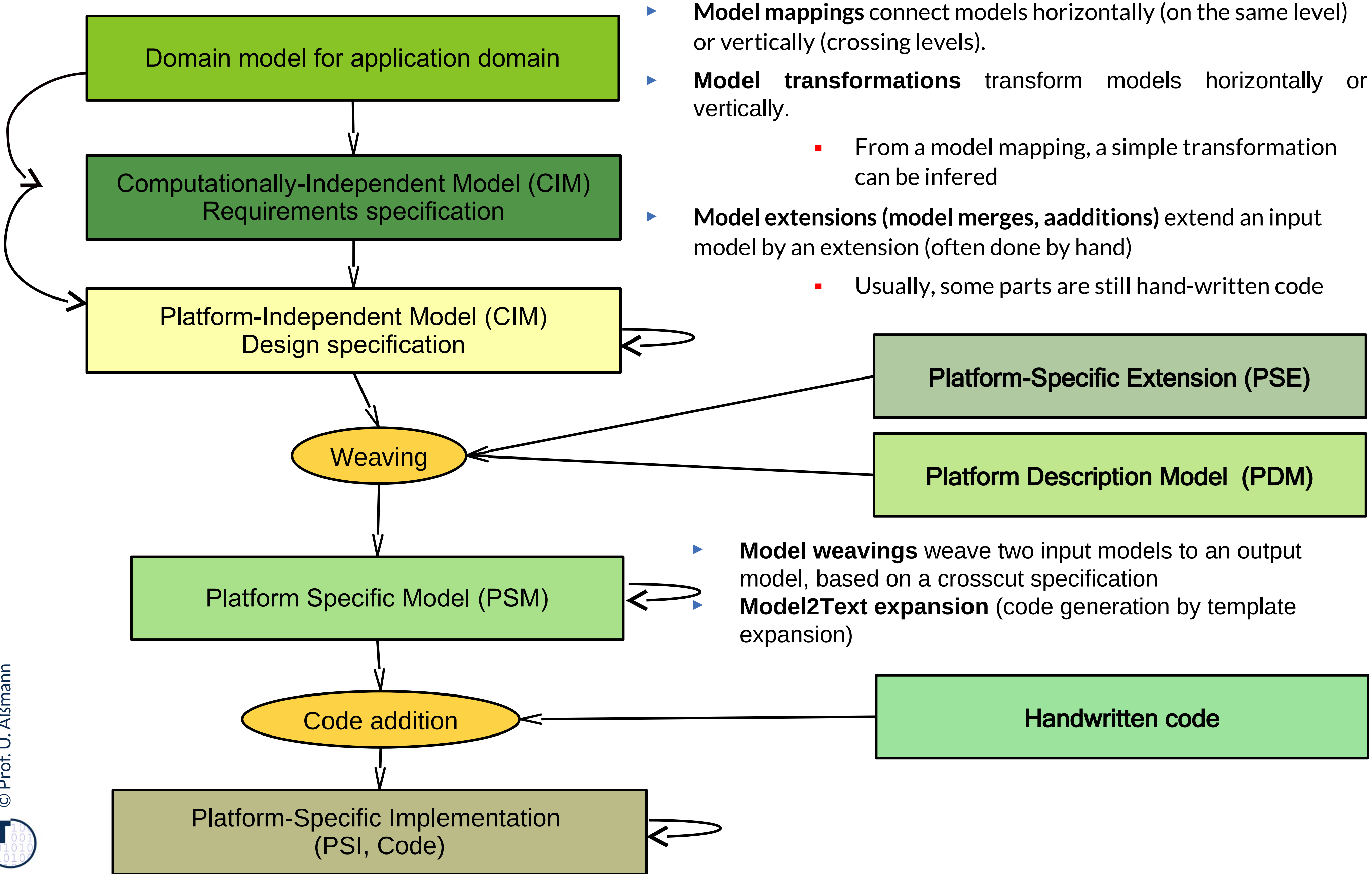
- But always performs model synchronization as a basic step



Advantages of Model Mappings in Macromodels

- **Error tracing**
  - When an error occurs during testing or runtime, we want to trace back the error to a design element or requirements element
- **Traceability**
  - We want to know which requirement (feature) influences which design, code, and test elements, so that we can demarcate modules in the solution space (product line development)
- **Synchronization in Development:**
  - Two models are called **synchronized**, if the change of one of them leads automatically to a hot-update of the other

Q9: Model Mappings and Model Weavings in the MDA Macromodel



- **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 extensions (model merges, aadditions)** extend an input model by an extension (often done by hand)
  - Usually, some parts are still hand-written code

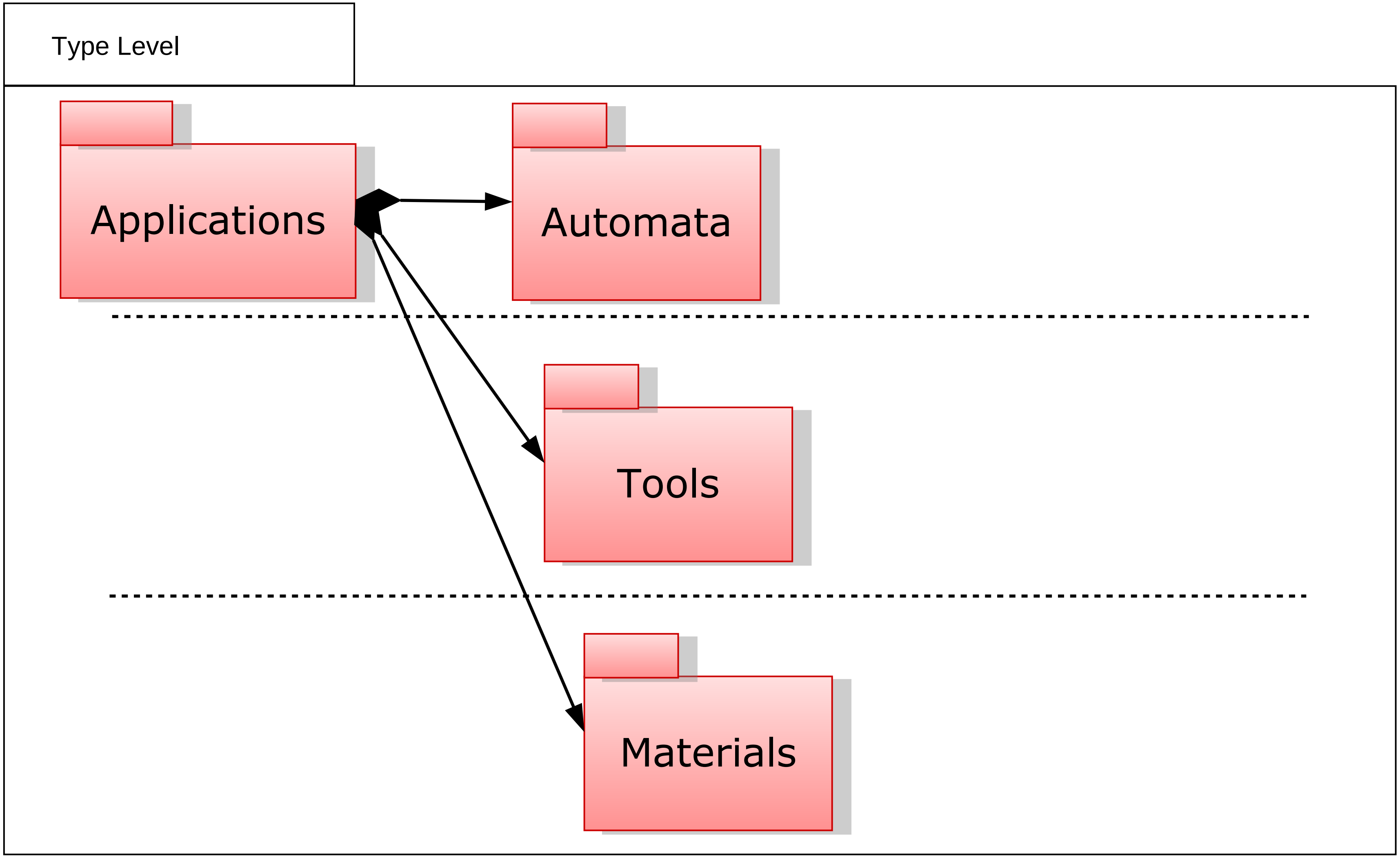
- **Model weavings** weave two input models to an output model, based on a crosscut specification
- **Model2Text expansion** (code generation by template expansion)

12.5. Pattern Languages in a Technical Space

- In a TS, several pattern languages may be used to structure the relationship of models and metamodels
- TAM can be used as Pattern Language on all levels in the metahierarchy
- However, there may be more pattern languages associated to a technical space
- Pattern languages can be expressed as stereotypes

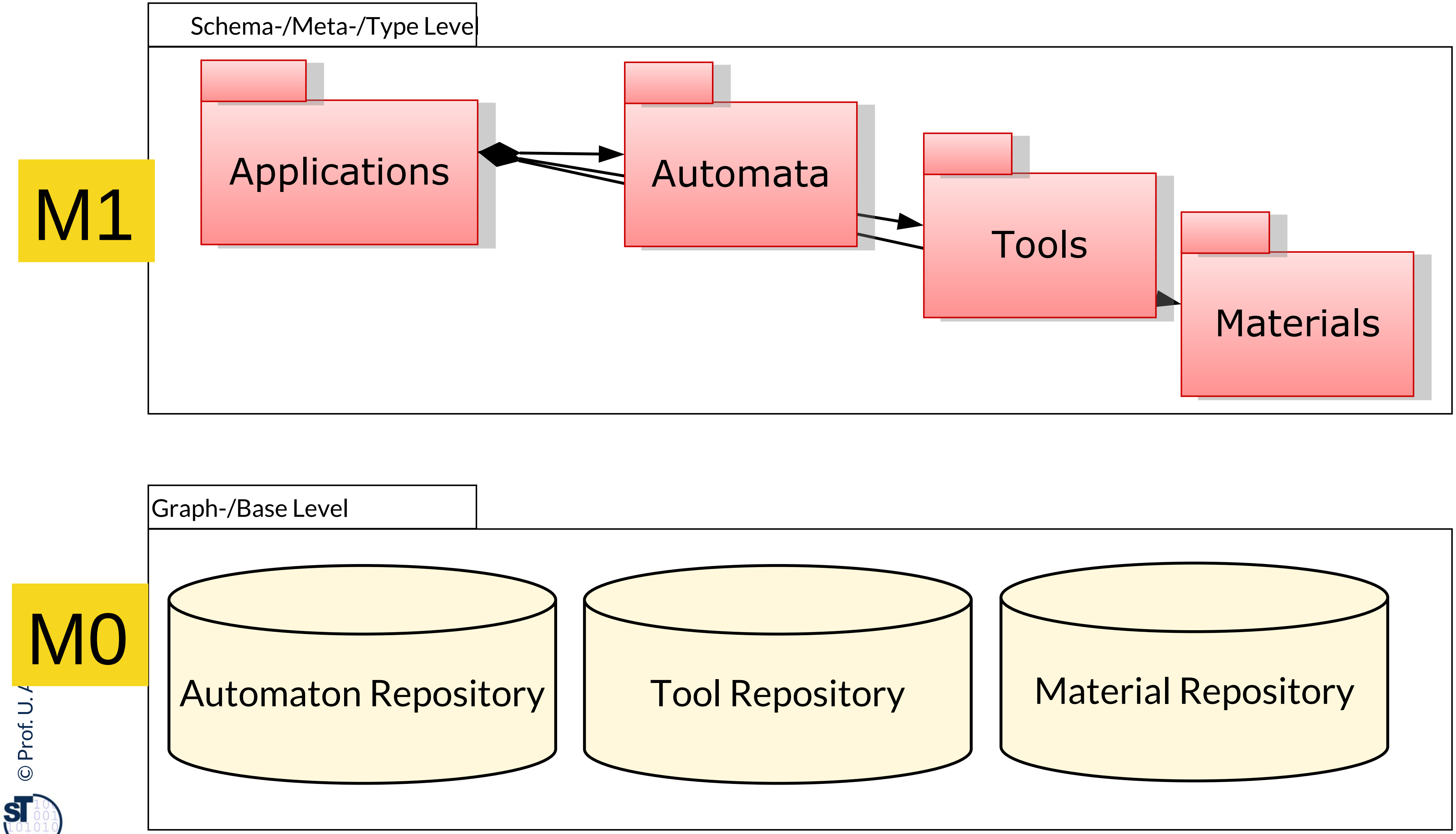
A Pattern Language Useful for all Technical Spaces TAM Structures on M1

- On M1, application class models need to define (stereotype) tools, automata, and materials.



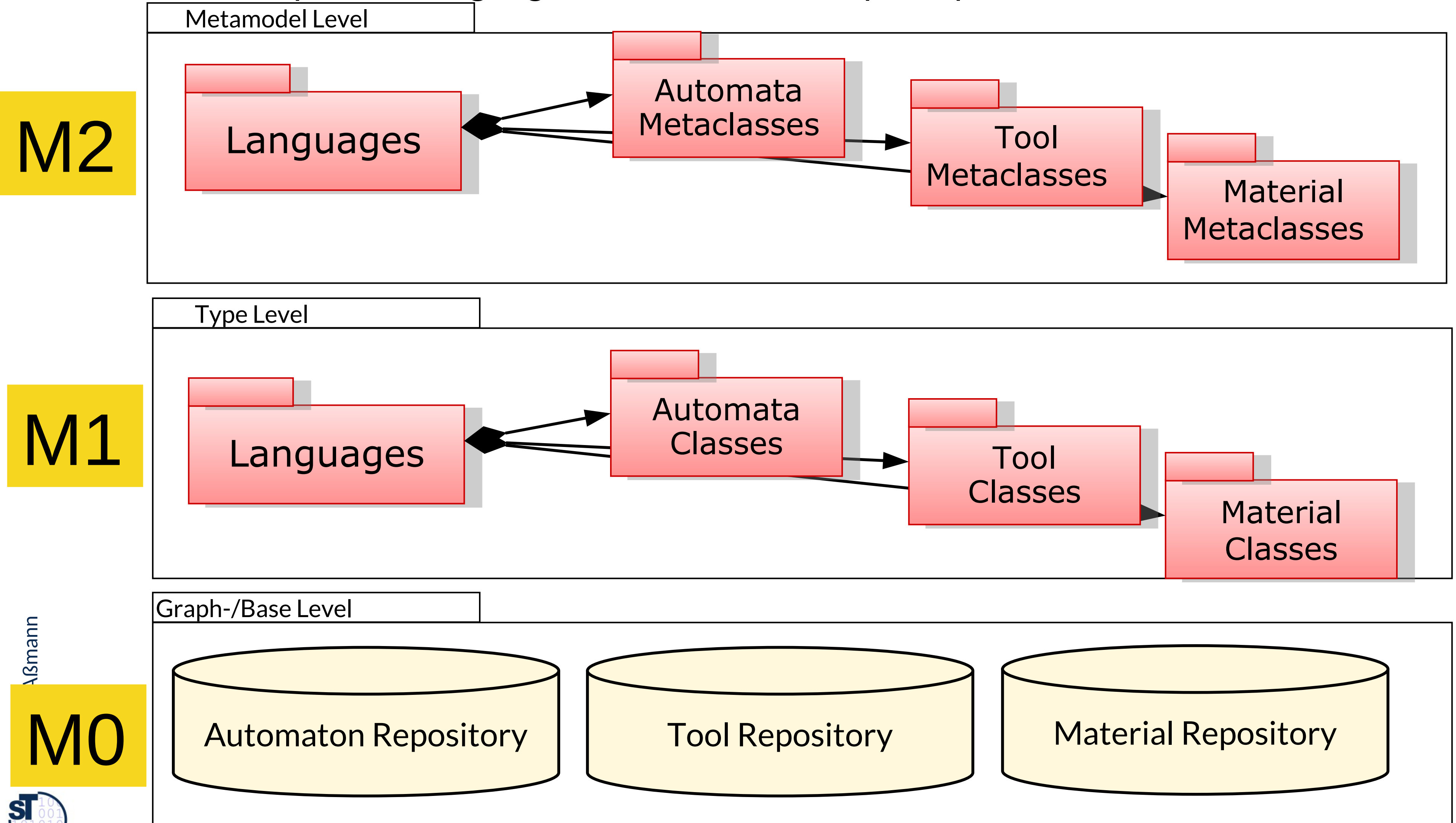
TAM Structures on M1 Provide Types for Objects in Repositories on M0

- On M1, application class models need to define (stereotype) tools, automata, and materials.



TAM Structures on M2 Provide Language Concepts for Stereotypes for Classes in M1

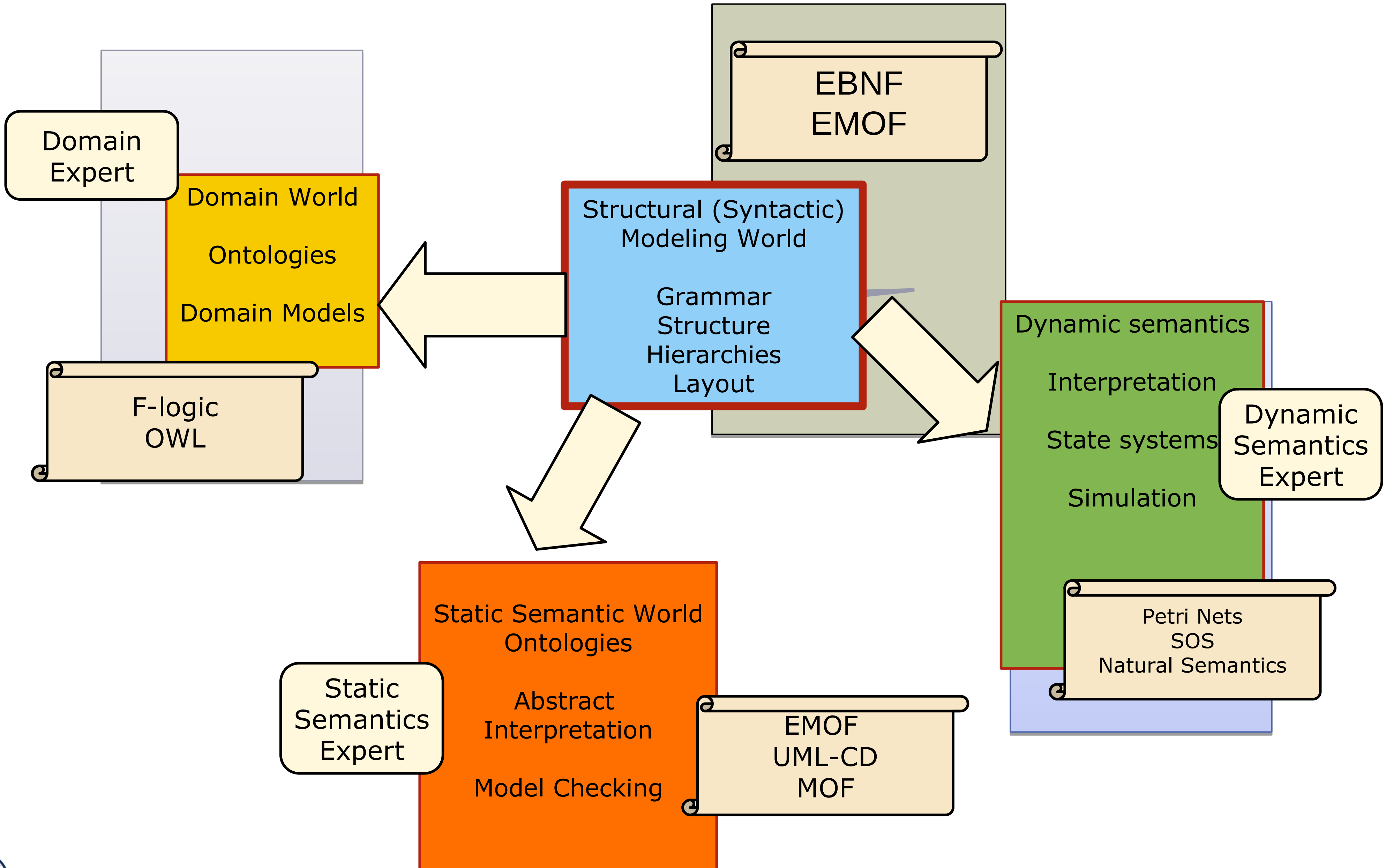
- On M2, TAM forms a DSL for stereotypes on M1
- Other pattern languages can use the same principle



12.6. Briding Technical Spaces

- While one tool/application may live in one TS, for the communication with other tools/applications, **technical space bridges** have to be built.
- Usually, a technical spaces has a subsystem for **technical space bridging.**

An Application May Need Several Technical Spaces

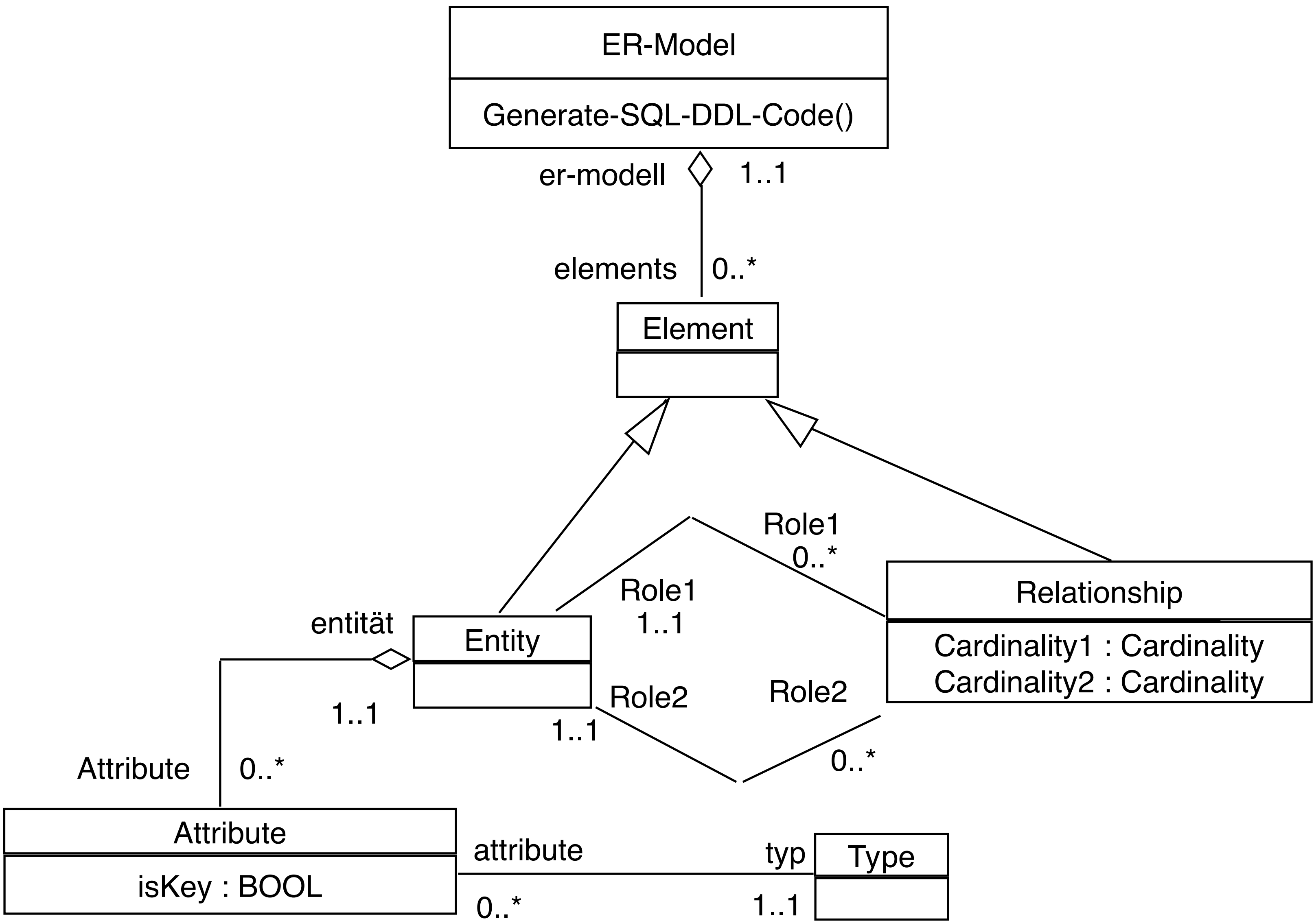


The End

12.A.1 Other Metalanguages

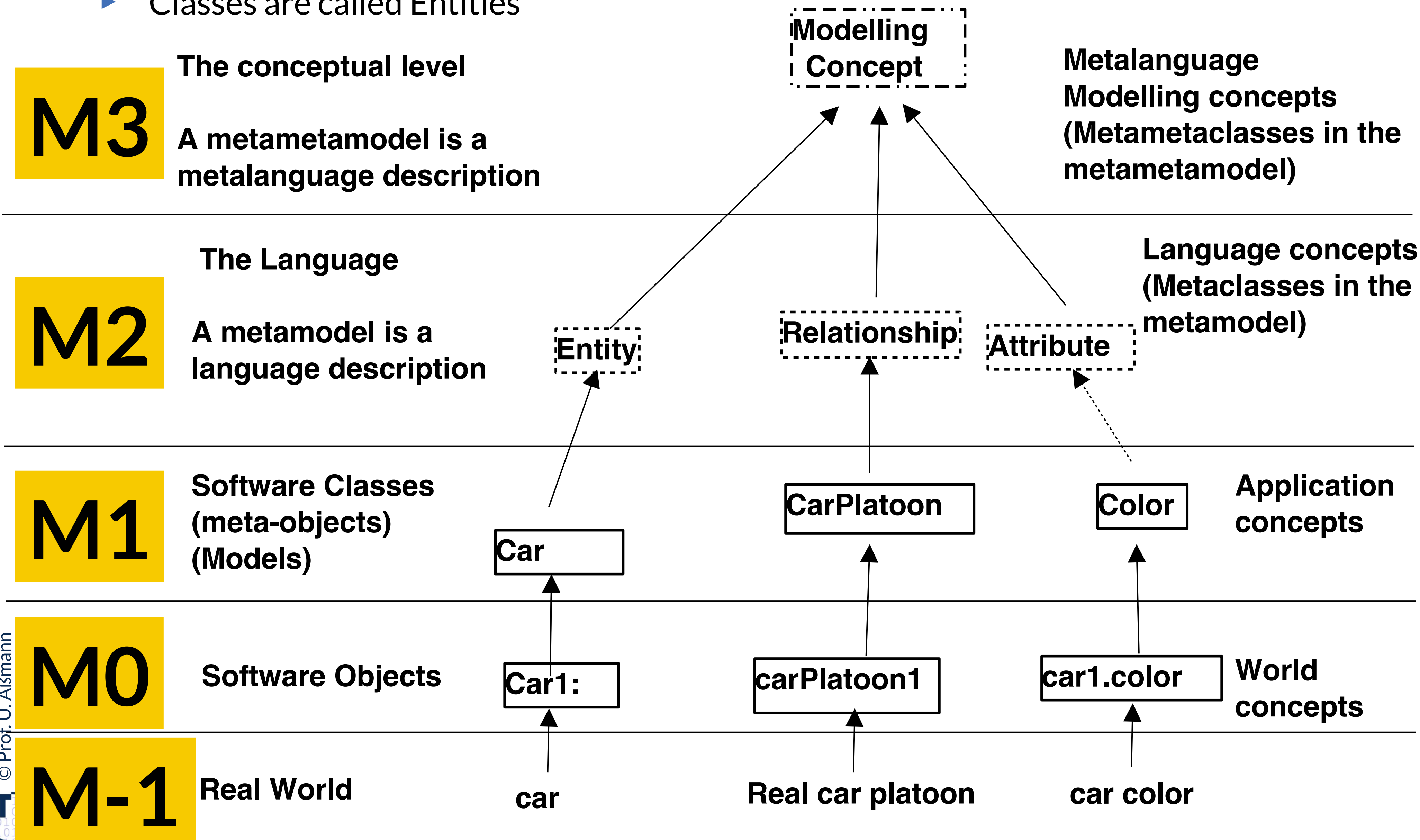
Metamodel of EntityRelationship Diagrams (ERD-ML) in MOF

- ERD is like MOF without inheritance



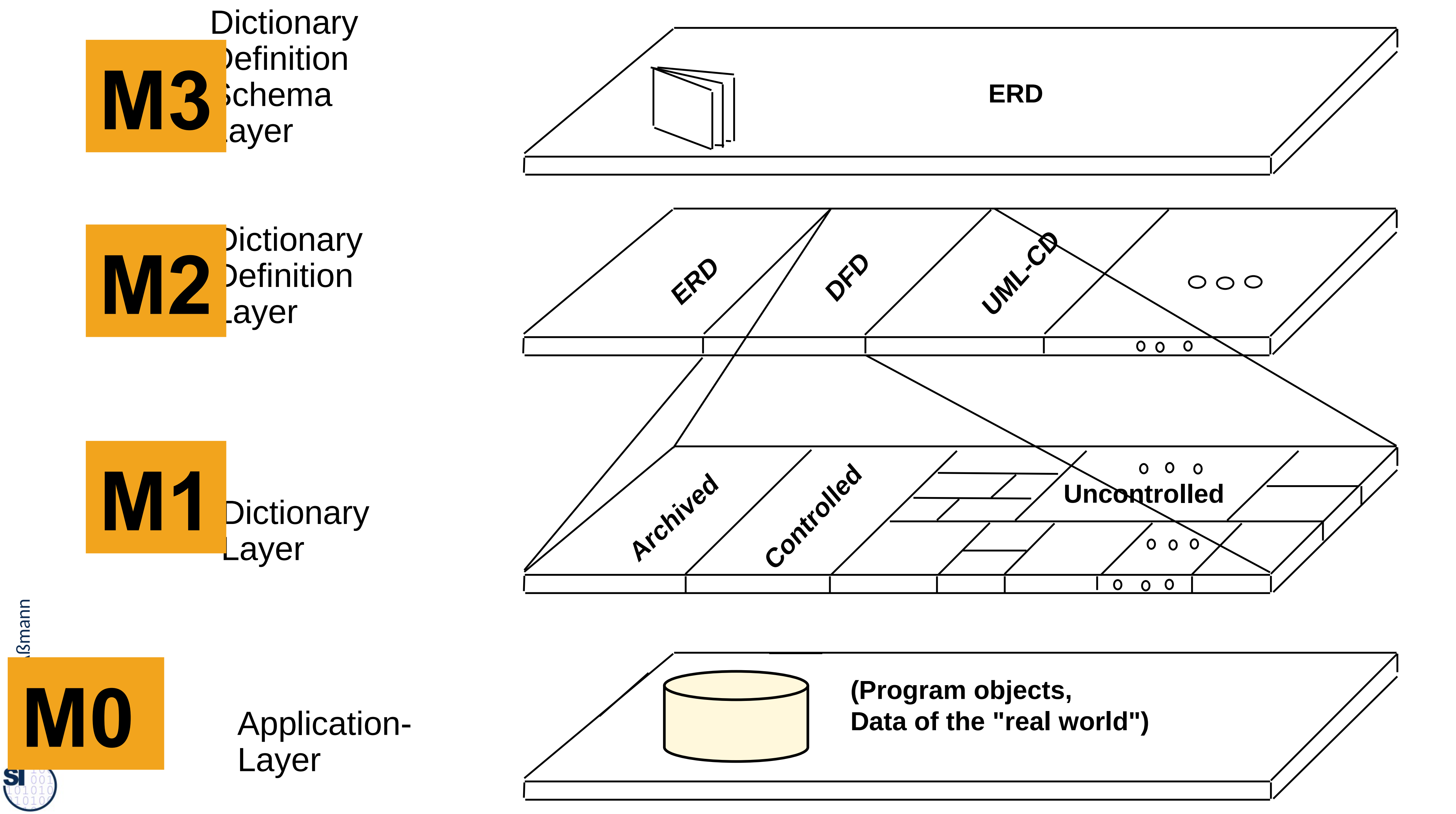
Metalevels in ERD

- Classes are called Entities



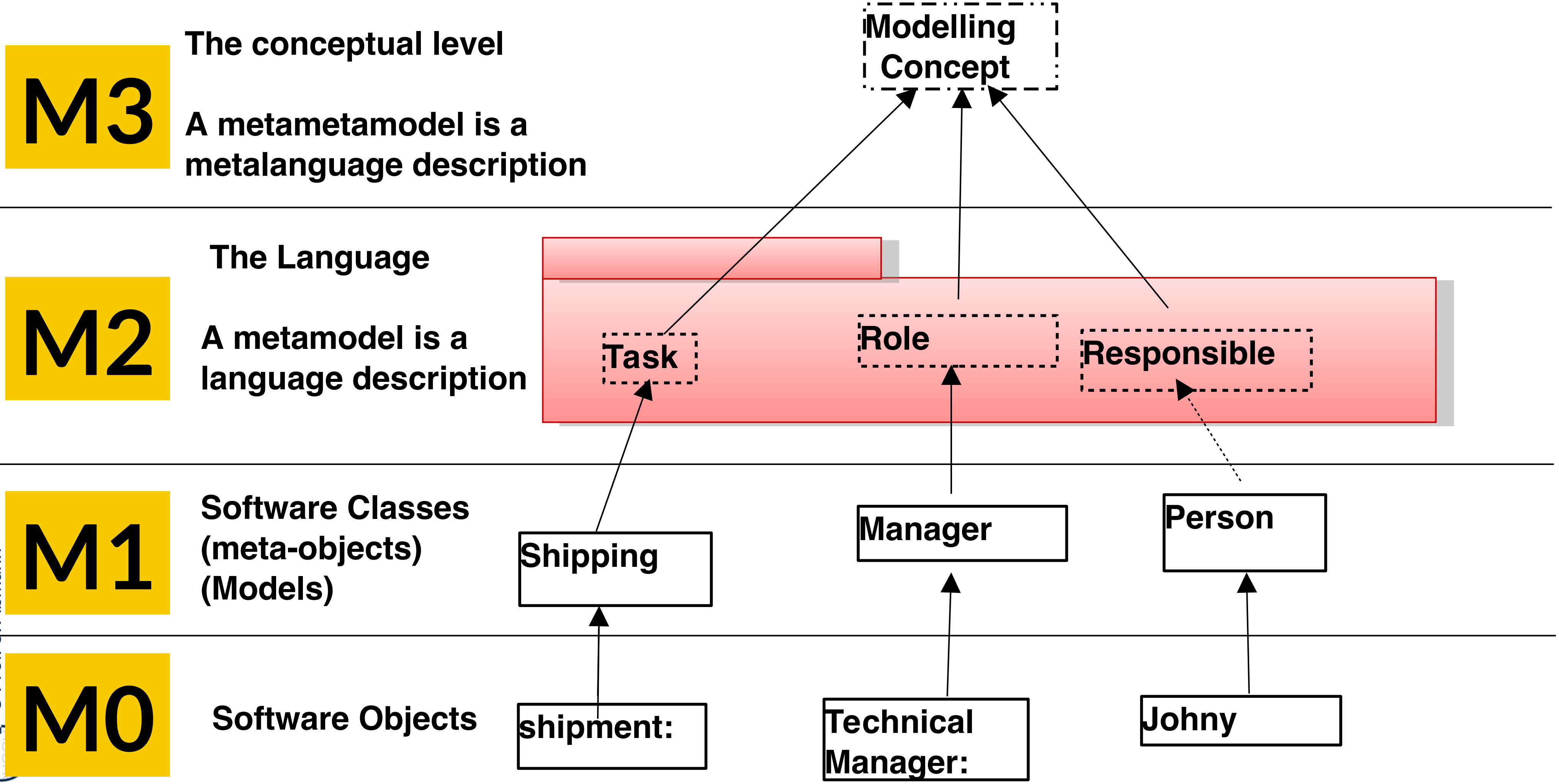
Ex.: IRDS/MOF Metahierarchy for Data Dictionaries in the Structured Analyse (SA)

- IRDS was defined in the 70s to model (persistent) data structures of applications



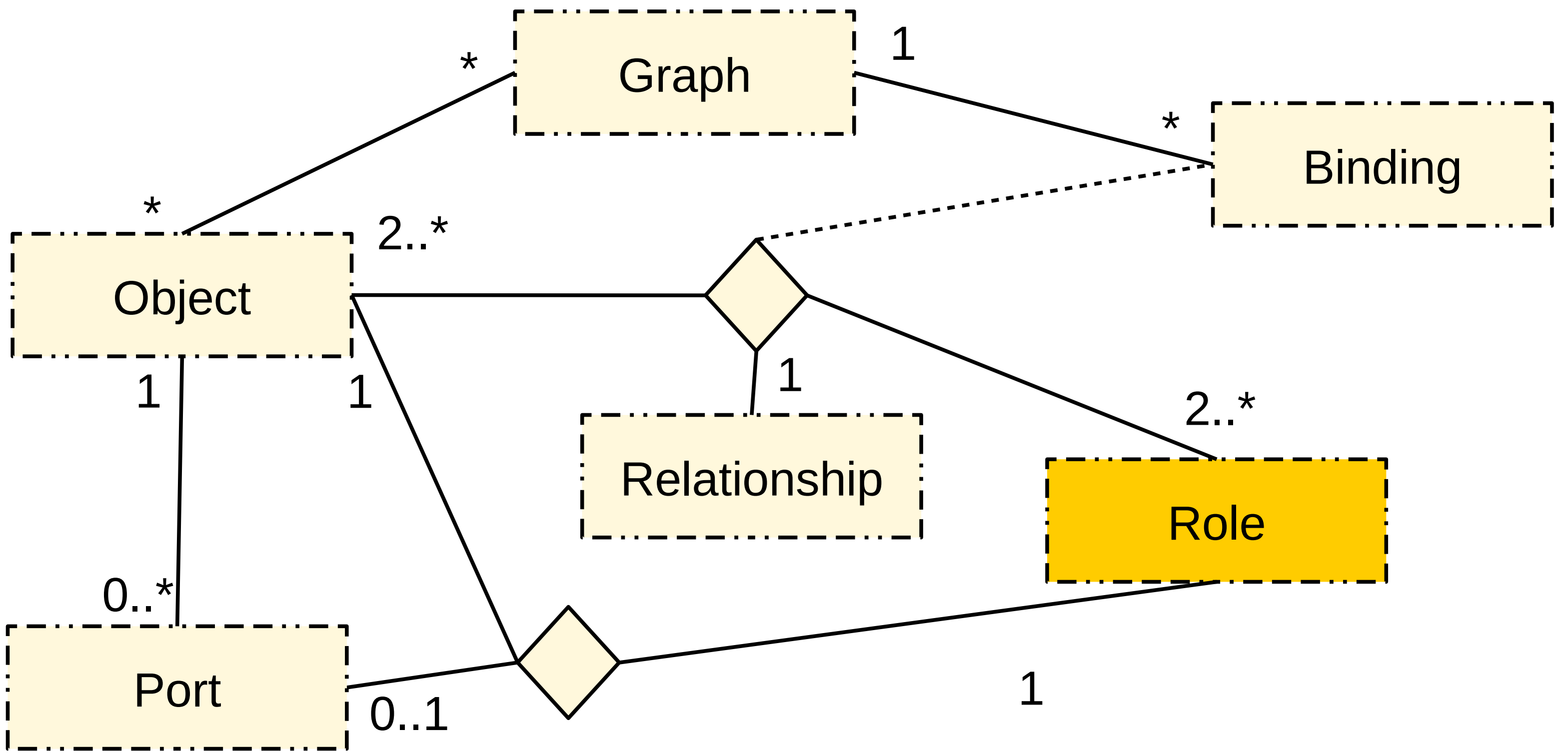
Ex.: Metahierarchy in Workflow Systems and Web Services (e.g., BPEL, BPMN, ARIS-EPK)

- It is possible to specify workflow languages with the metamodelling hierarchy
- BPEL and other workflow languages can be metamodeled
- **BPEL is metamodeled with the metalanguage XSD**



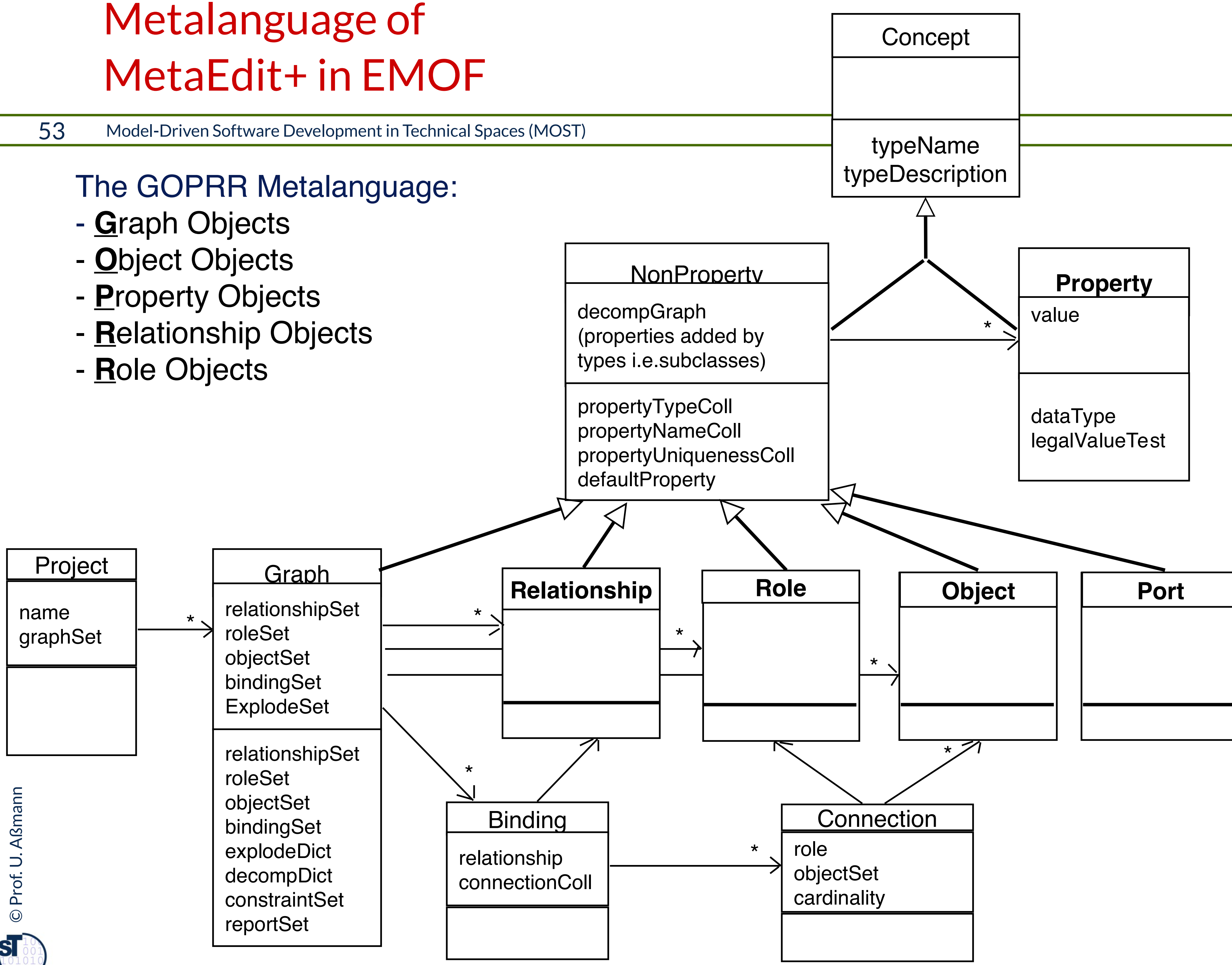
Role-Based Graph Types in MetaEdit+

- [www.metacase.com]
- The tool MetaEdit+ uses the **graph schema (metalanguage) GOPRR**:
  - Objects and their Roles; Relationships
  - Allowed Bindings between all entities:
    - a binding consists of a relationship with roles and playing objects



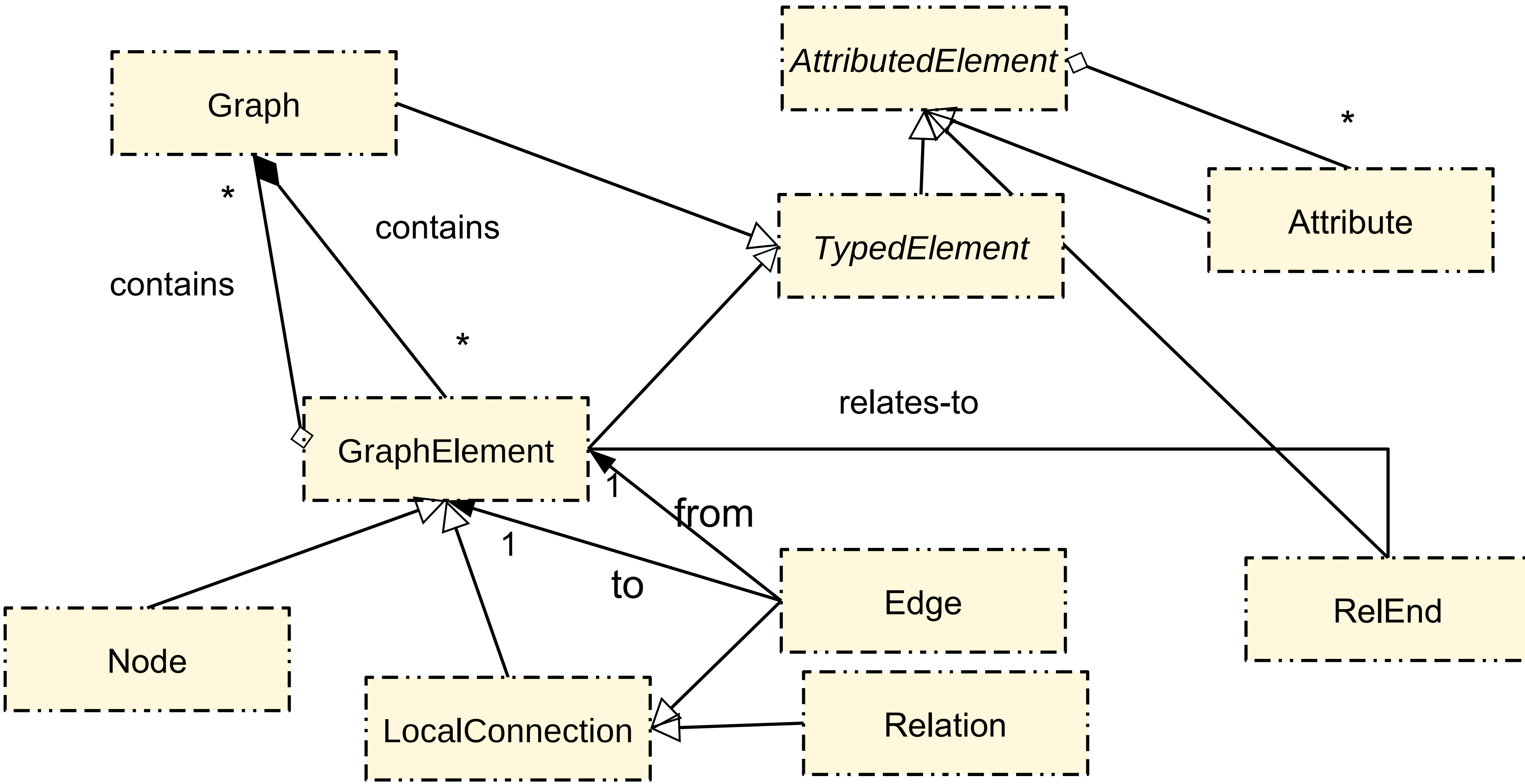
Metalanguage of MetaEdit+ in EMOF

The GOPRR Metalanguage:
- **G**raph Objects
- **O**bject Objects
- **P**roperty Objects
- **R**elationship Objects
- **R**ole Objects



GXL Graph eXchange Language – a Technical Metametamodel

- GXL is a modern graph-language (graph-exchange format)
- Contains abstractions for elements of graphs usable for generic algorithms (e.g., flexible navigation)



Richard C. Holt, Andy Schürr,Susan Elliott Sim, Andreas Winter. GXL: A graph-based standard exchange format for reengineering. Science of Computer ProgrammingVolume 60, Issue 2, April 2006, Pages 149-170

GXL-based Metamodel of Typed Attributed Graph

- GXL can be used as metalanguage (Metametamodel) on M3, to type metamodels and DSL on M2
- For example, state machines
- Alternatively, GXL can also be used as DDL on M2 (it is a lifted metamodel)