11. Metamodellierung und Technikräume

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de
Version 13-0.3, 24.10.13

1) Metamodellierung
   - Meta-Hierarchie
   - Meta-Object-Facility (MOF)
2) Metasprachen
3) Modell- und Metamodell-Komposition
4) Technikräume
5) Megamodelle

Obligatorische Literatur

- Ed Seidewitz. What models mean. IEEE Software, 20:26-32, September 2003.
  - http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1231147&tag=1
- Jean Bézivin. Model Driven Engineering: An Emerging Technical Space. In R. Lämmel, J. Saraiva, and J. Visser (Eds.): GTTSE 2005, LNCS 4143, pp. 36 – 64, 2006. Springer.
- Uwe Aßmann, Steffen Zschaler, and Gerd Wagner. Ontologies, meta-models, and the model-driven paradigm. In Coral Calero, Francisco Ruiz, and Mario Piattini, editors, Ontologies for Software Engineering and Technology. Springer, 2006.
  - http://www.springer.com/computer/swe/book/978-3-540-34517-6?cm_mmc=Google-_-Book%20Search-_-Springer-_-0
- Steffen Staab, Tobias Walter, Gerd Gröner, and Fernando Silva Parreiras. Model driven engineering with ontology technologies. In Uwe Aßmann, Andreas Bartho, and Christian Wende, editors, Reasoning Web, volume 6325, Lecture Notes in Computer Science, pages 62-98. Springer, 2010.
  - http://www.uni-koblenz.de/~staab/Research/Publications/2010/reasoningweb2010.pdf

Andere Literatur

- Kurtev, I., Bezivin, J., Aksit, M.: Technological Spaces: An Initial Appraisal. In: International Symposium on Distributed Objects and Applications, DOA Federated Conferences, Industrial track, Irvine. (2002)
- Model-based Technology Integration with the Technical Space Concept. Jean Bezivin and Ivan Kurtev. Metainformatics Symposium, 2005.
- Gaševic, Dragan, Djuric, Dragan, Devedžic, Vladan. Model Driven Engineering and Ontology Development, 2nd ed., 2009, ISBN 978-3-642-00281-6
  - http://www.springer.com/computer/swe/book/978-3-642-00281-6?cm_mmc=Google-_-Book%20Search-_-Springer-_-0
- [MOF] Metaobject Facility. OMG. 1.4 and 2.0. www.omg.org
- [Nill] C. Nill. Analysis and Design Modeling Using Metaphorical Modeling Entities. A Modeling Language for the Tools and Materials Approach. Diplomarbeit Technische Universität Dresden, 2006.
- [Atkinson/Kühne] Colin Atkinson and Thomas Kühne. Model-driven development: A metamodeling foundation. IEEE Software, 20(5):36-41, 2003.
- [Favre] Jean-Marie Favre. Foundations of model (driven) (reverse) engineering: Models. Technical report, ADELE Team, Laboratoire LSR-IMAG Université Joseph Fourier, Grenoble, France, 2004. vol. 1-3.
- [Kendall] D. T. Chang and E. Kendall. Metamodels for RDF Schema and OWL. Proceedings of the First International Workshop on the Model-Driven Semantic Web (MDSW 2004), Monterey, USA, September 21, 2004.

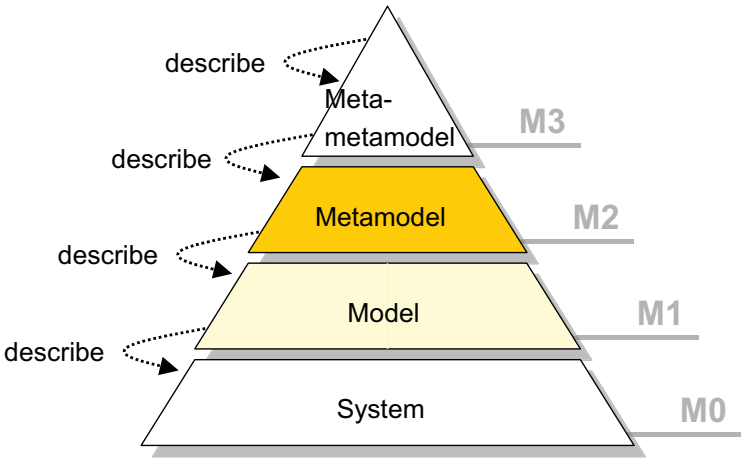
11.1 Metamodellierung

Modelle in der Softwaretechnik

- **Prozessmodelle**
  - **Phasenmodelle** definieren Tätigkeiten und ihre Verknüpfung beim Ablauf großer Software-Entwicklungsvorhaben.
  - **Vorgehensmodelle** unterscheiden Tätigkeiten (Aktivitäten) und von ihnen erzeugte Ergebnisse (Dokumente, Produkte) sowie ihr Zusammenwirken.
- **Domänenmodelle** beschreiben den mittels der Methoden modellierten Problembereich (Analyse) aus der realen Welt des Anwenders.
- **Systemmodelle**
  - **Architekturmodelle** Verteilung von Software-(Modell-)Bestandteilen nach bestimmten Anordnungen, Mustern(Pattern) bzw. Referenzmodellen (Client-Server, Web-Verteilung, ECMA-Referenzmodell, UI)
  - Software-**Strukturmodelle** veranschaulichen den Aufbau der konkreten Software-Struktur im Lösungsbereich (Entwurf) (meist UML-CD).
  - **Datenmodelle** illustrieren die Struktur von Daten (z.B. Relationales Modell)
- **Metamodelle** bieten Typen für Modellelement an. Sie sind Modelle, deren Instanzen selbst Modelle sind. Sie beschreiben die *Struktur* von Prozess-, Domänen- und Systemmodellen

Motivation

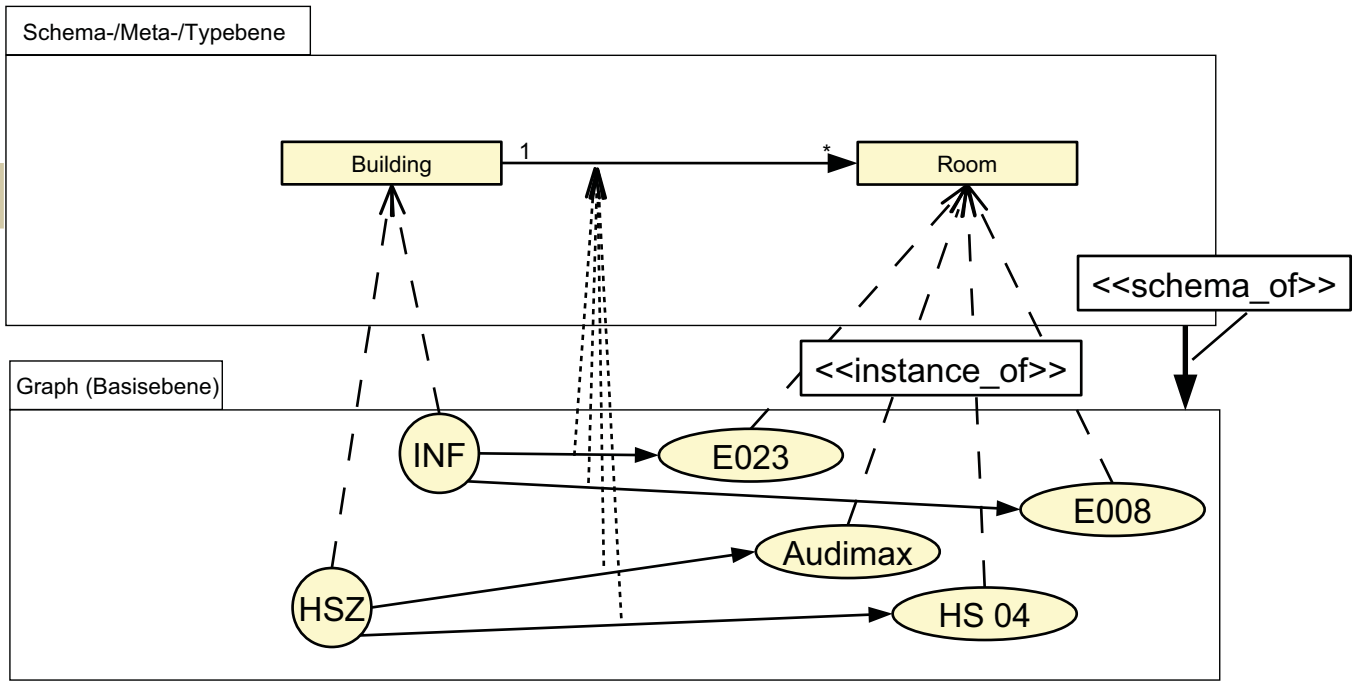
- Models are widely used in engineering disciplines
- Need for **tool support** that enables model-editing
- Domain experts want **domain specific languages** (DSL)
  → domain specific models
- do not build model editors from scratch each time
  → **reuse** functionality
  → use meta-information



[F. Klar, TU Darmstadt]

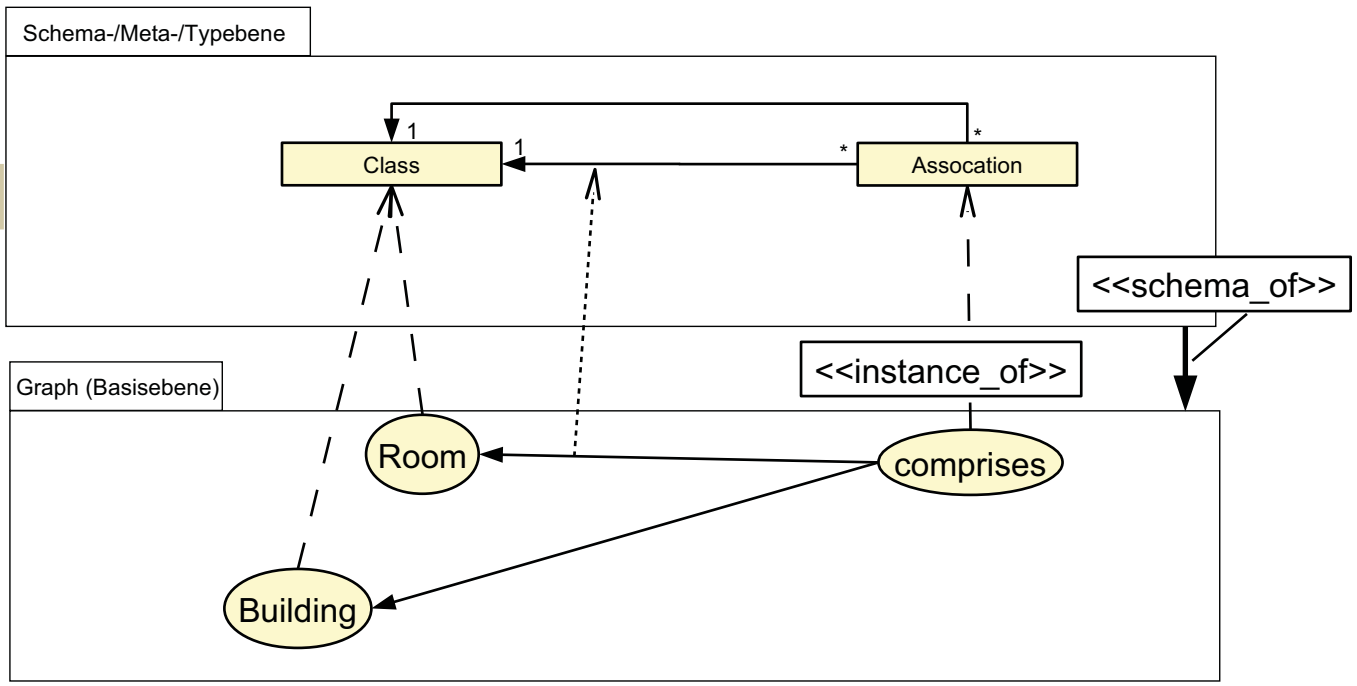
Typisierte Objekte (Modelle und Metamodelle)

- Objekte können typisiert sein
- Unterscheide **Schema- bzw. Meta- bzw. Typebene** von **Instanzebene**



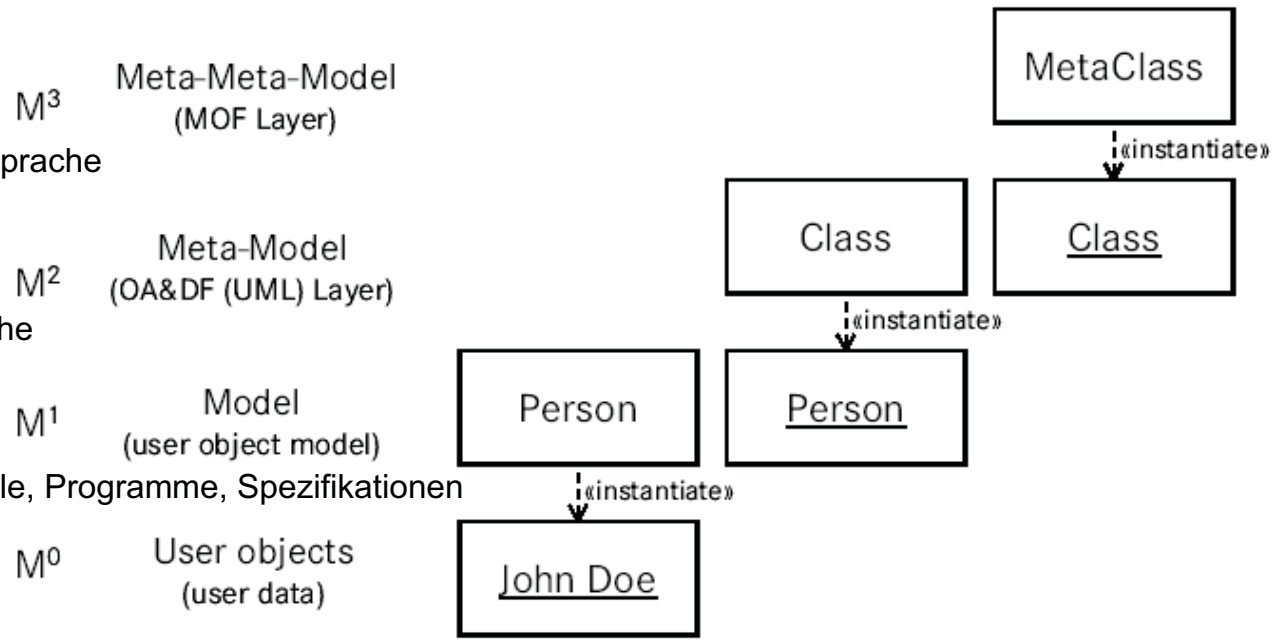
Typisierte Objekte (Modelle und Metamodelle)

- Objekte können typisiert sein
- Unterscheide **Schema- bzw. Meta- bzw. Typebene** von **Instanzebene**



Prof. U. Aßmann, Softwareentwicklungswerkzeuge (SEW)

OMG's 4-Schichten Metamodel Architektur (MOF-Metahierarchie (urspr. IRDS Metahierarchie)



Quelle: Jeckle, M.: XML basierter Metadatenaustausch; 6. Fachgruppentreffen „*Objektorientierte Softwareentwicklung*" der Gesellschaft für Informatik am 27.1.99 in München

R. Dolk. Model management and structured modeling: the role of an information resource dictionary system. Commununications of the ACM(CACM), 31:704-718, June 1988.

Das Metametamodel (Metasprache, Meta language)

- Ein **Metametamodel (Metasprache)** ist ein Graphschema einer Sprache
  - Es bietet Typen für die Konzepte einer Sprache an, also allgemeine Modellierungskonzepte
  - Es umfasst die kontext-sensitive Syntax der Sprache (Konzepte und deren Relationen untereinander)
  - Es definiert Struktur, kein Verhalten!
- Ein Metametamodell ist normalerweise schlank (minimalistisch)
- Ein einheitliches Metametamodell ist nicht in Sicht... (tower of babel)

Tower of Babel Problem

Jan-Pieter Breughel (wikipedia)

Metametamodels - Overview

- Meta Object Facility – MOF
  - Complete MOF – CMOF
    - UML core
  - Essential MOF – EMOF
    - **Ecore** (Eclipse)
- GOPRR – Graph Object Property Role Relation
- GXL – Graph eXchange Language

Overview of Metalanguage MOF (CMOF: Complete MOF)

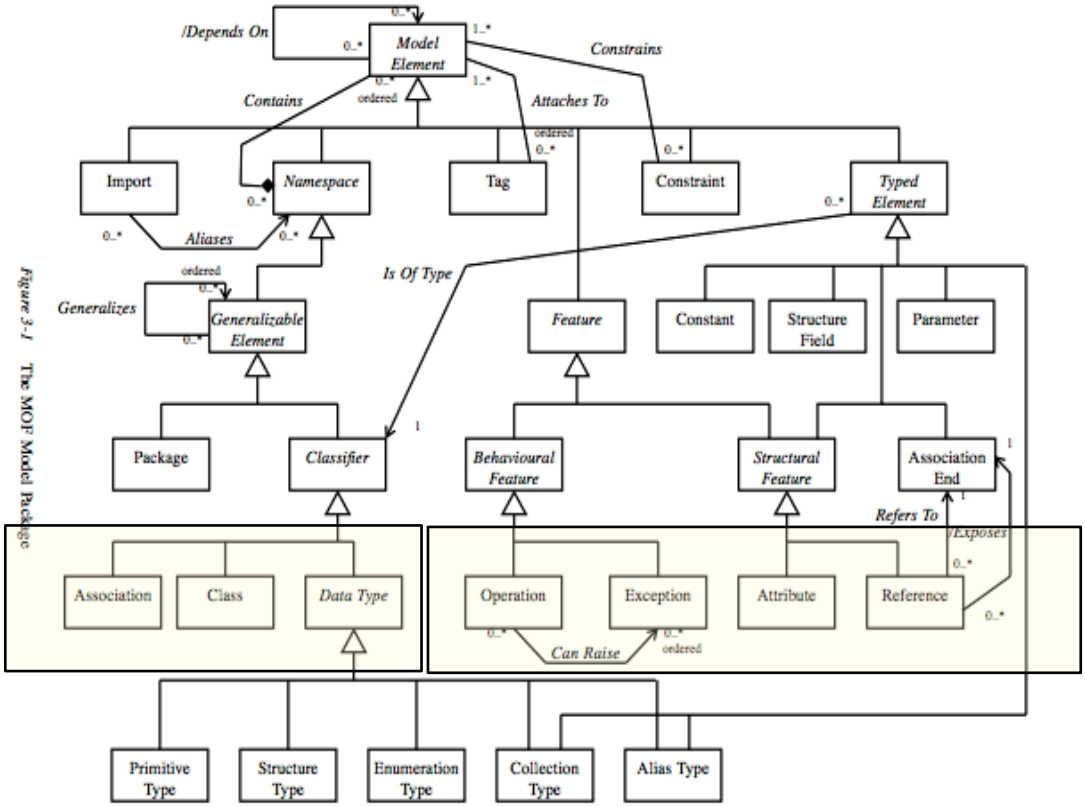
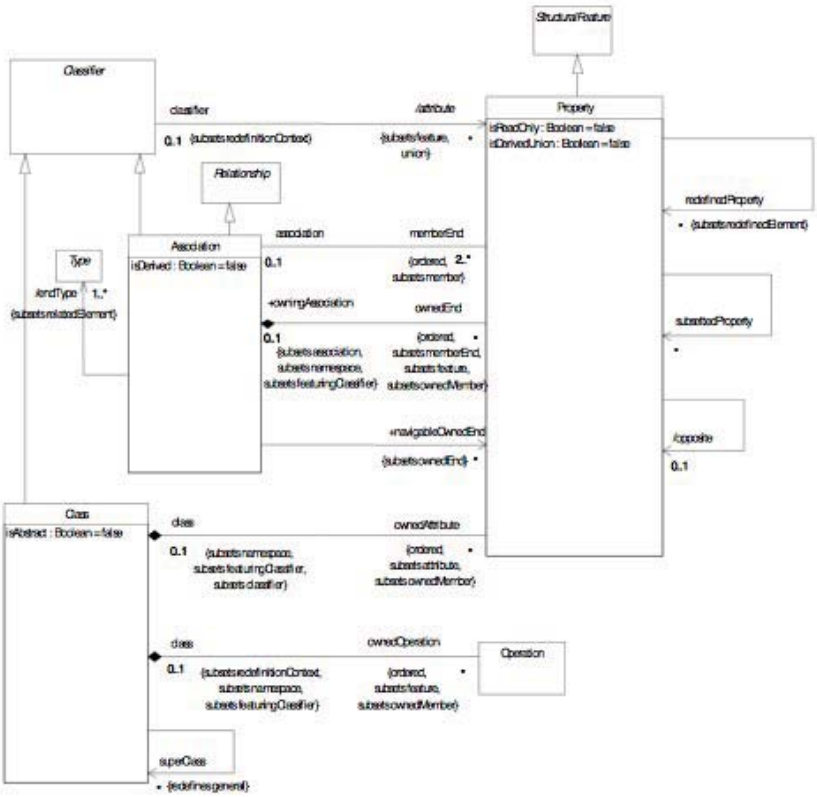


Figure 3-1 The MOF Model Package

[MOF]

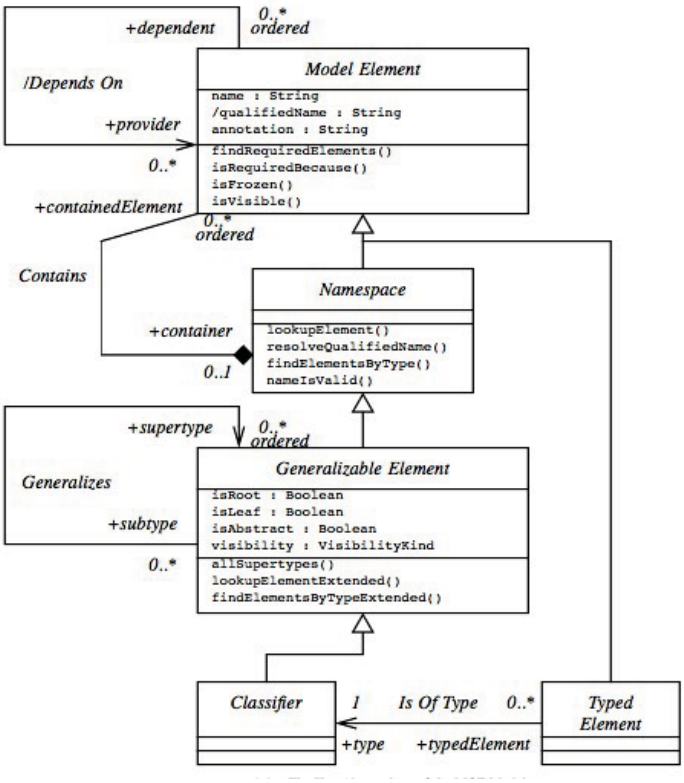
UML Core

- UML core ist Teil von MOF, und auch Teil von UML-CD



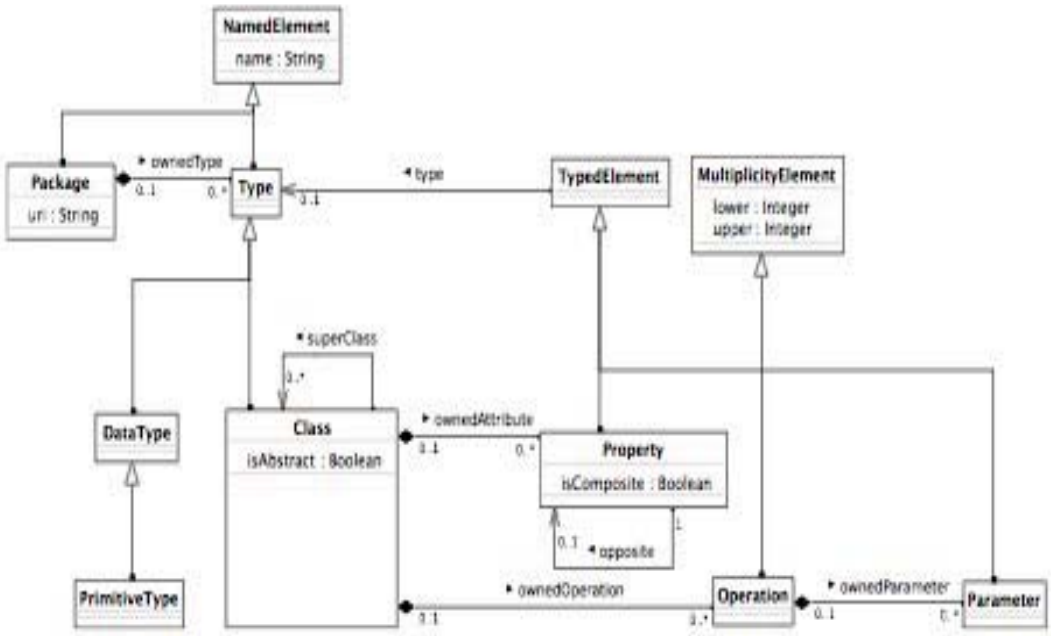
[MOF]

MOF Central Types



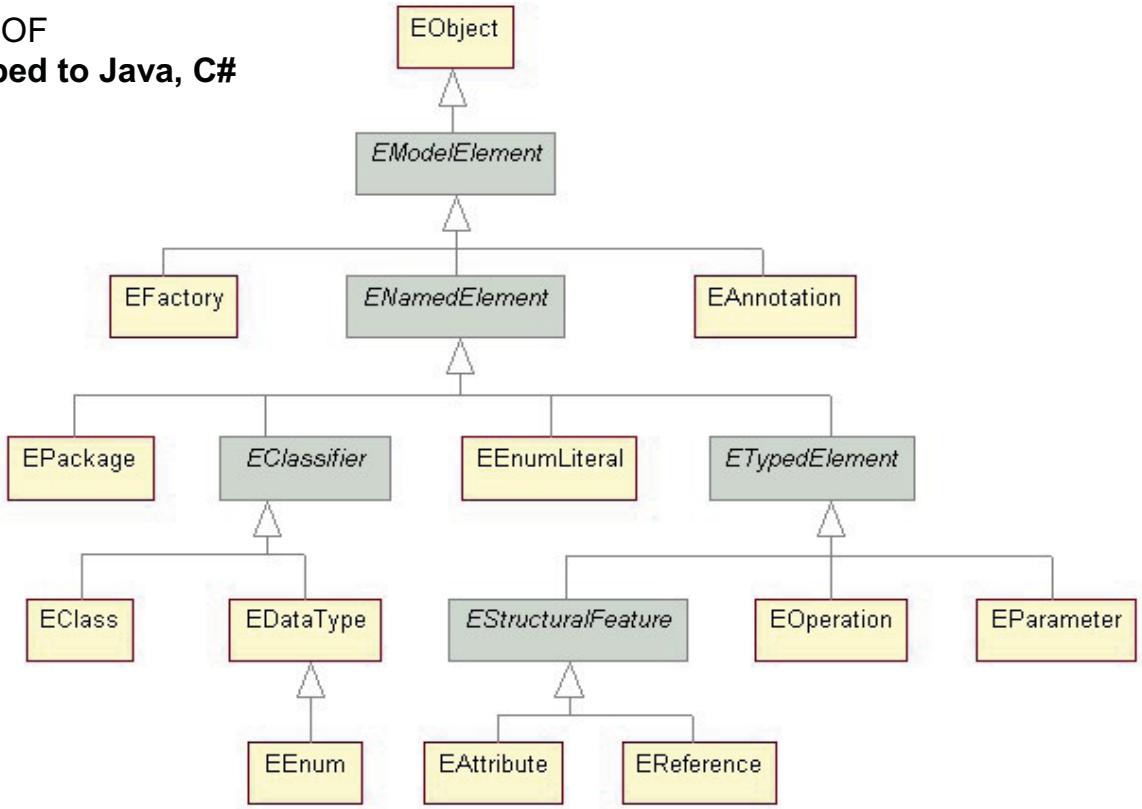
[MOF]

Central MOF Metaclasses with Associations

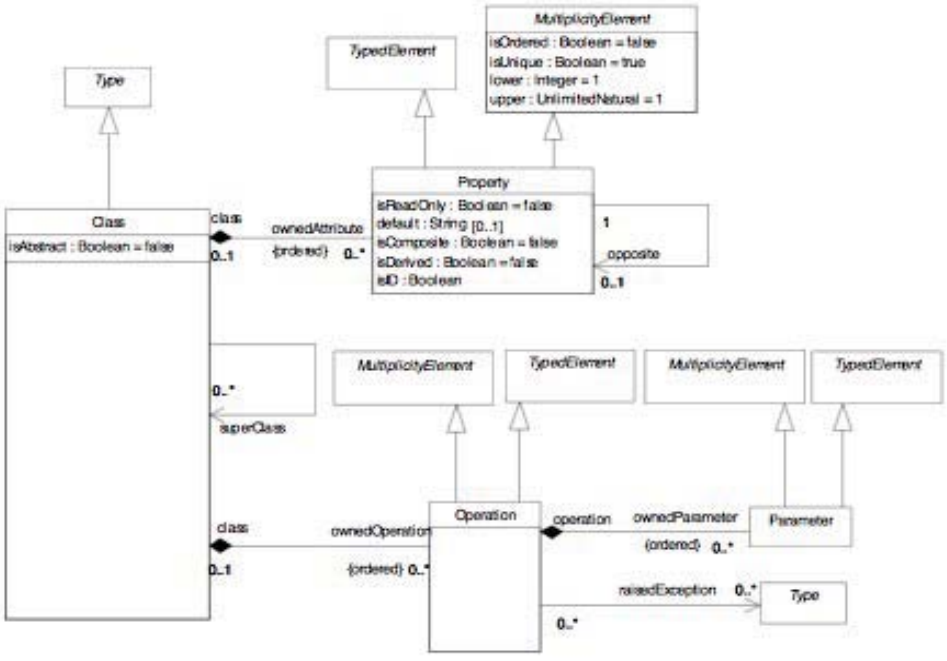


EMOF (Essential MOF)

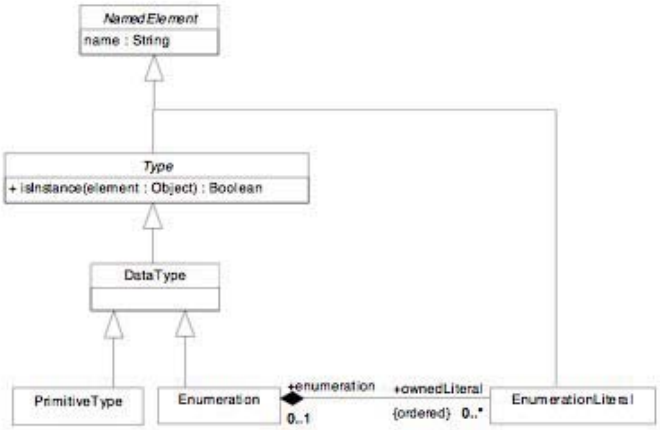
Subset of CMOF
Can be mapped to Java, C#

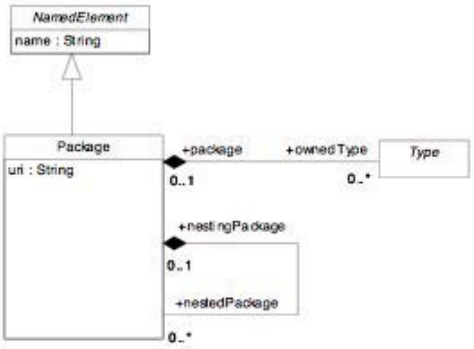


EMOF Classes in Detail

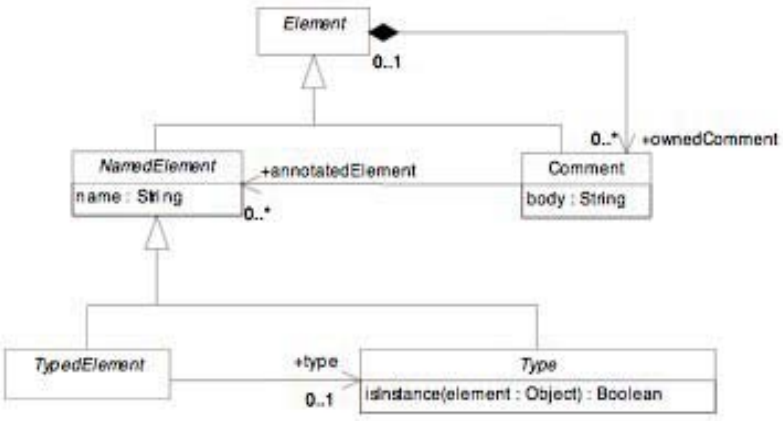


EMOF Data Types and Packages



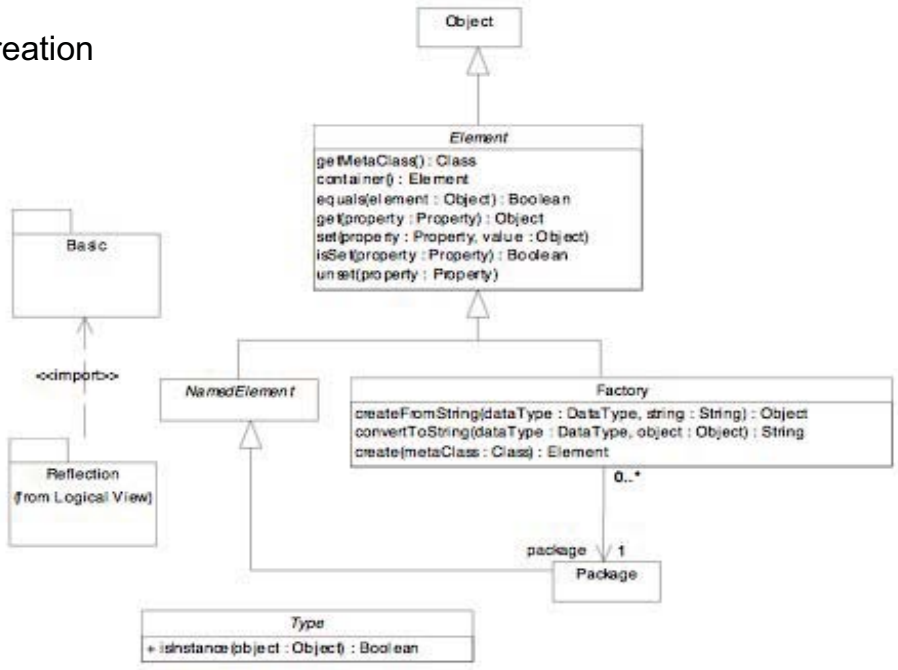


EMOF Types

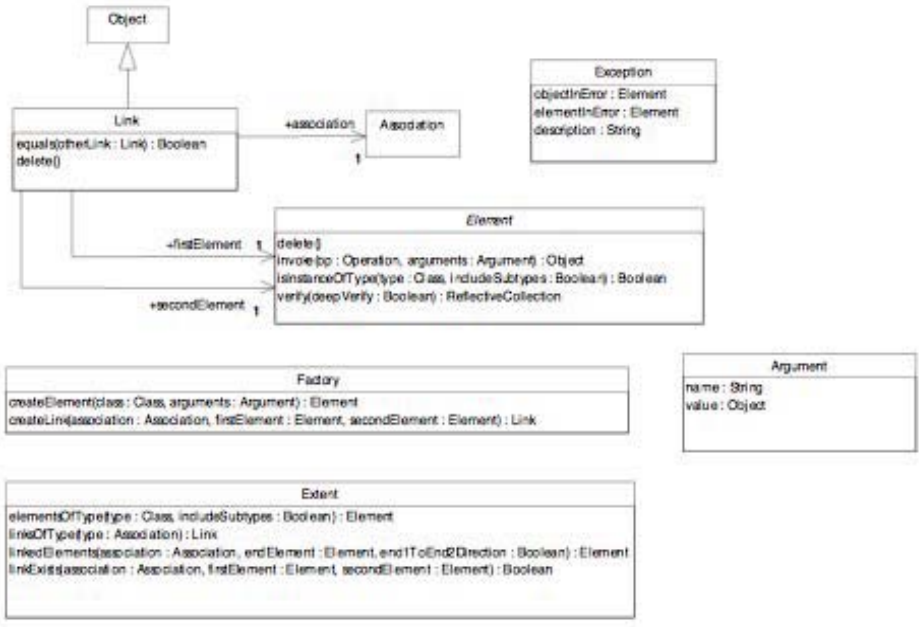


EMOF Reflection

offers access to the metamodel
(getMetaClass())
provides a Factory, for creation
of a Class from String

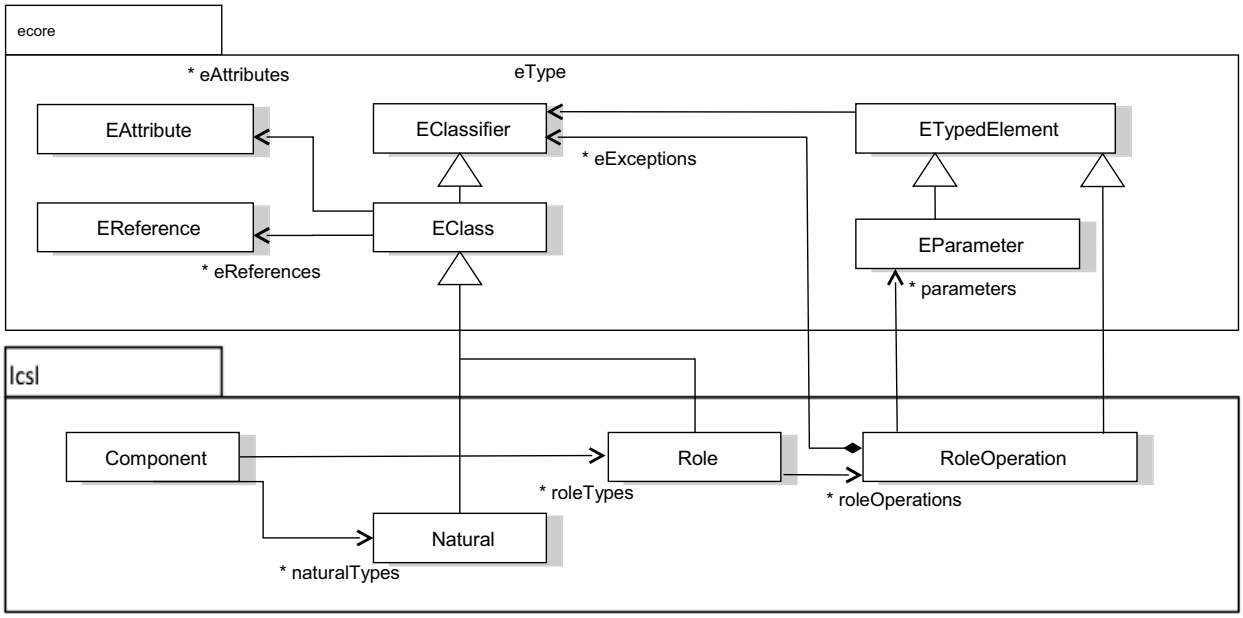


CMOF Reflection



Ex.: EMOF and its Implementation Eclipse ecore

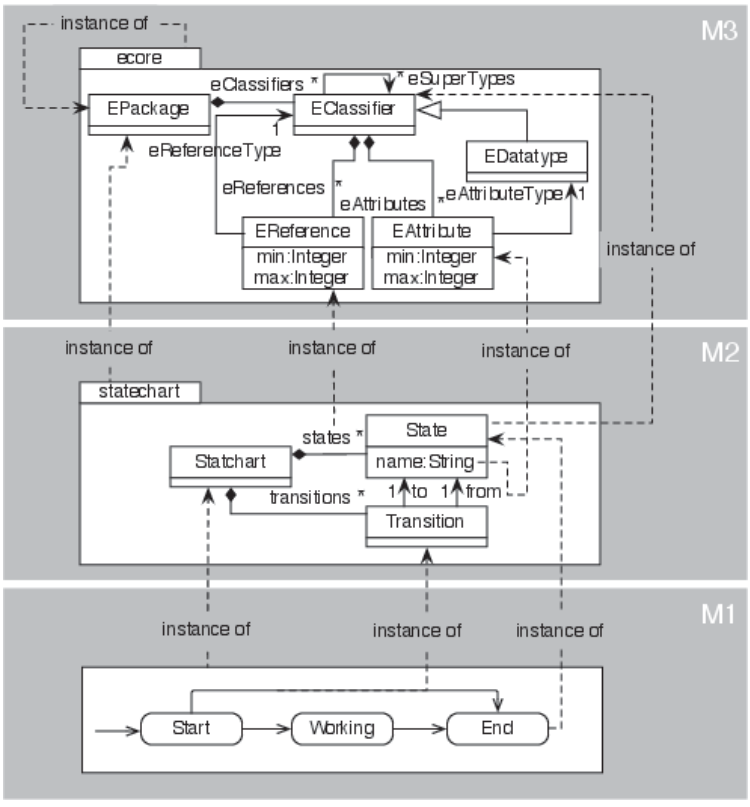
lcsl is a domain-specific language for component-based modeling (C. Wende)



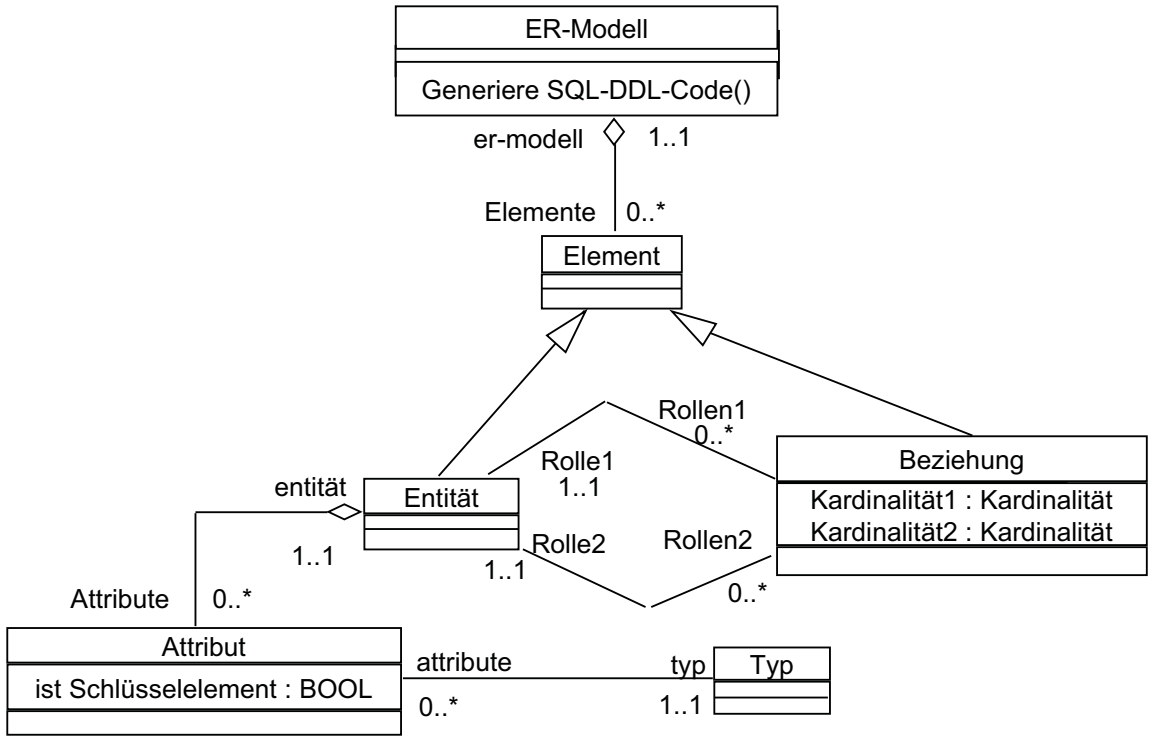
EMOF/Ecore based Metamodel of Statecharts

Ecore is the Eclipse implementation of EMOF, provided by the Eclipse Modeling Framework (EMF).

Here:
a metamodel of statecharts (M2),
a set of states and their transitions (M2), and the Ecore Metalanguage.

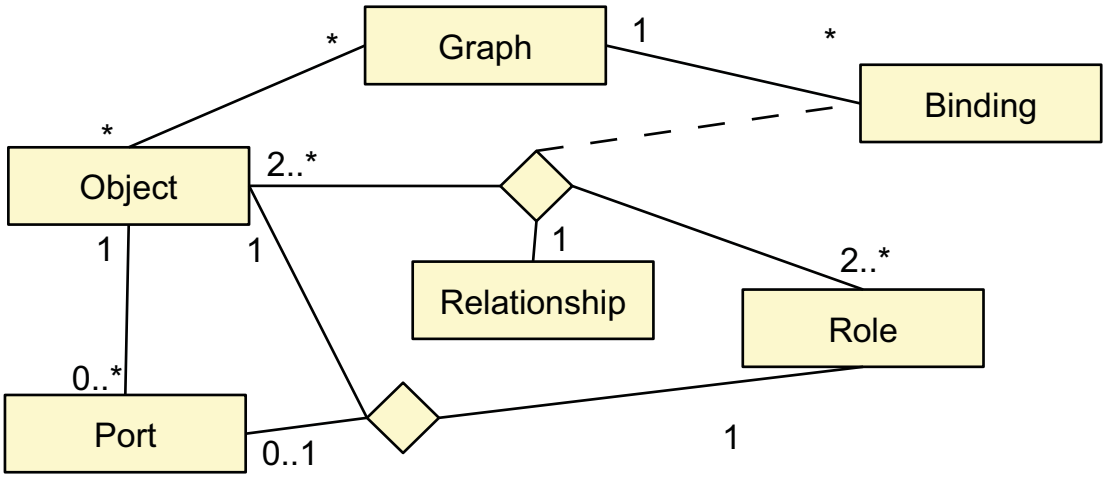


Meta-Modell von EntityRelationship-Diagrammen (MOF ohne Vererbung)



Graph Types in MetaEdit+

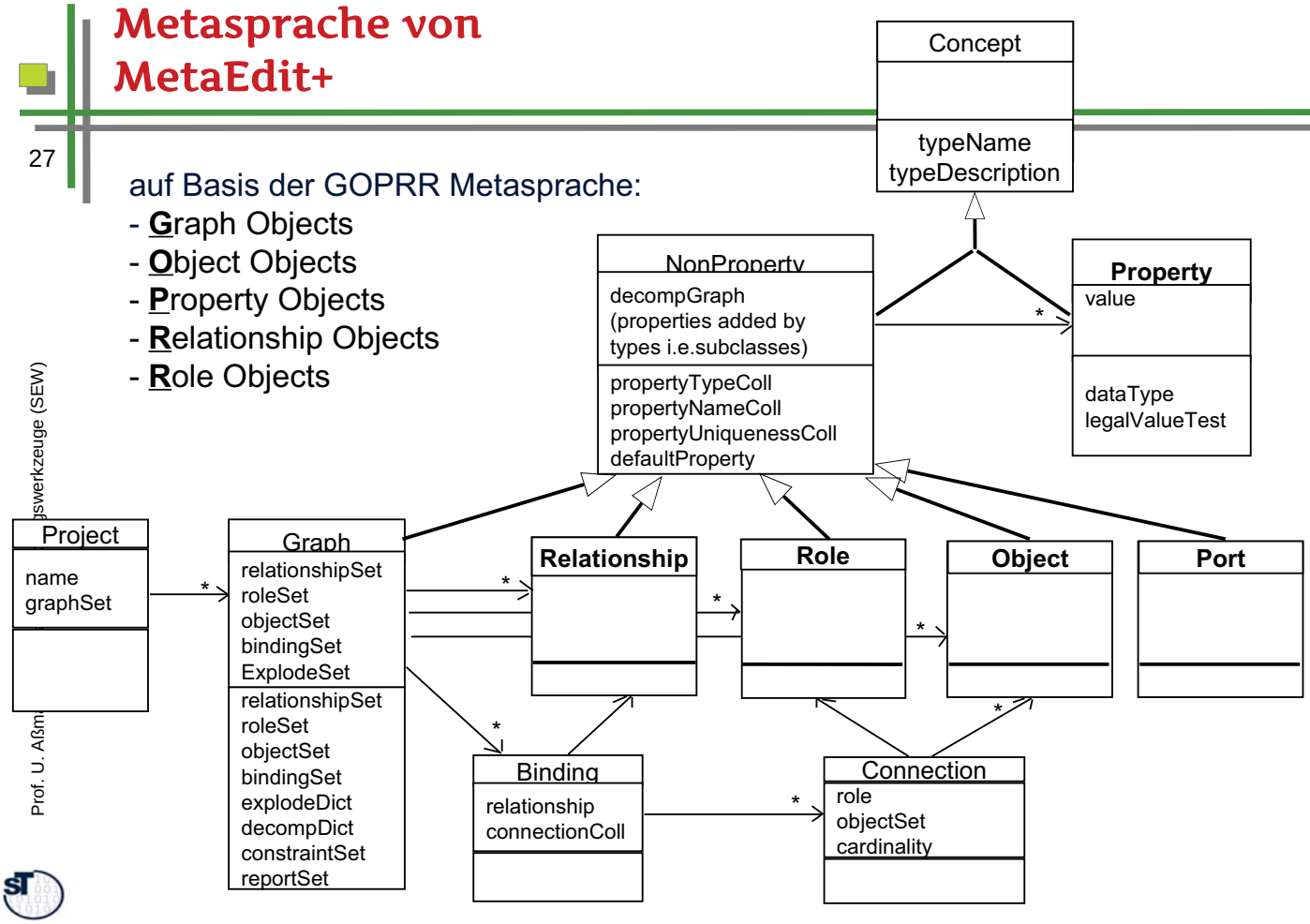
- [www.metacase.com]
- The tool MetaEdit+ uses the **graph schema (metalanguage) GOPRR**:
  - Objects
  - Roles
  - Relationships
  - Allowed Bindings between all entities:
    - a binding consists of a relationship with roles and playing objects



Metasprache von MetaEdit+

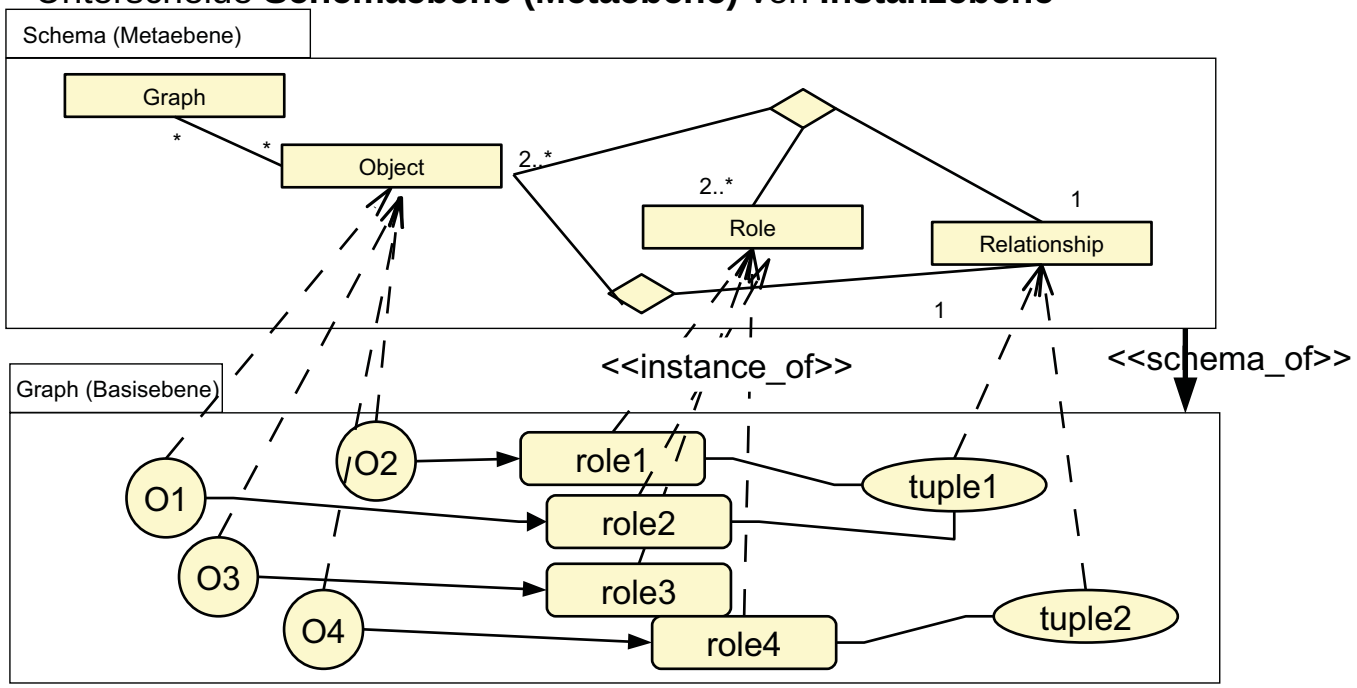
auf Basis der GOPRR Metasprache:

- **G**raph Objects
- **O**bject Objects
- **P**roperty Objects
- **R**elationship Objects
- **R**ole Objects



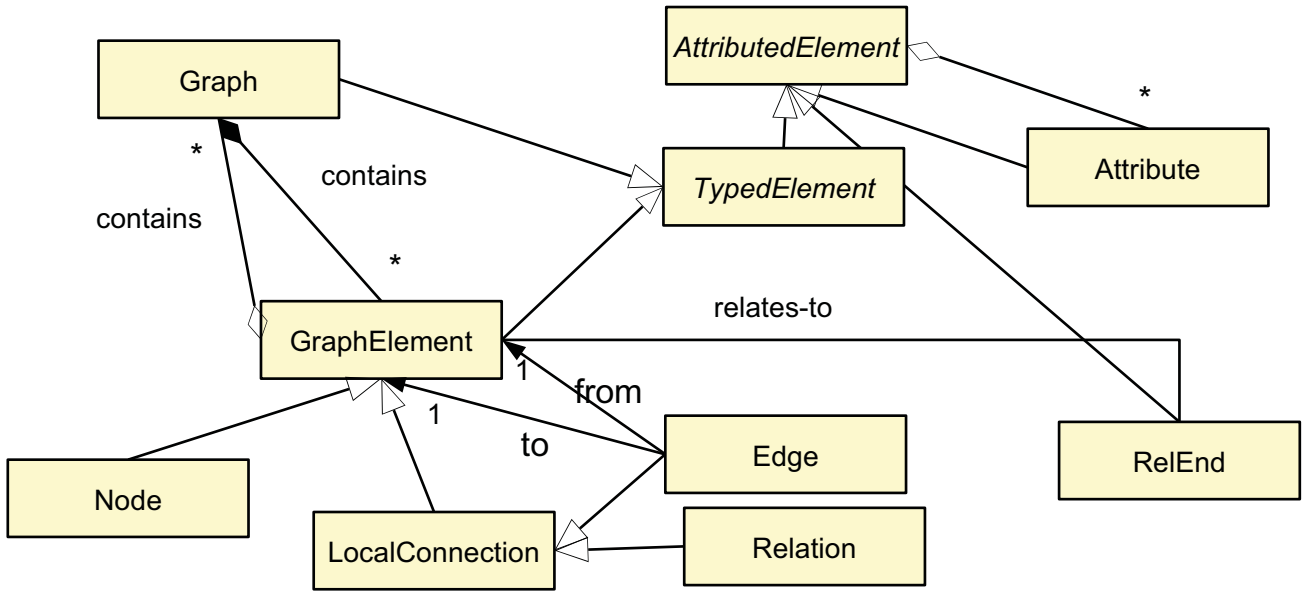
Typisierte Graphen (Modelle und Metamodelle)

- Graphen können typisiert sein, aber die Schemata können unterschiedlich aussehen (→ Metamodellierung)
- Unterscheide **Schemaebene (Metaebene)** von **Instanzebene**



GXL Graph eXchange Language – Ein technisches Metamodell

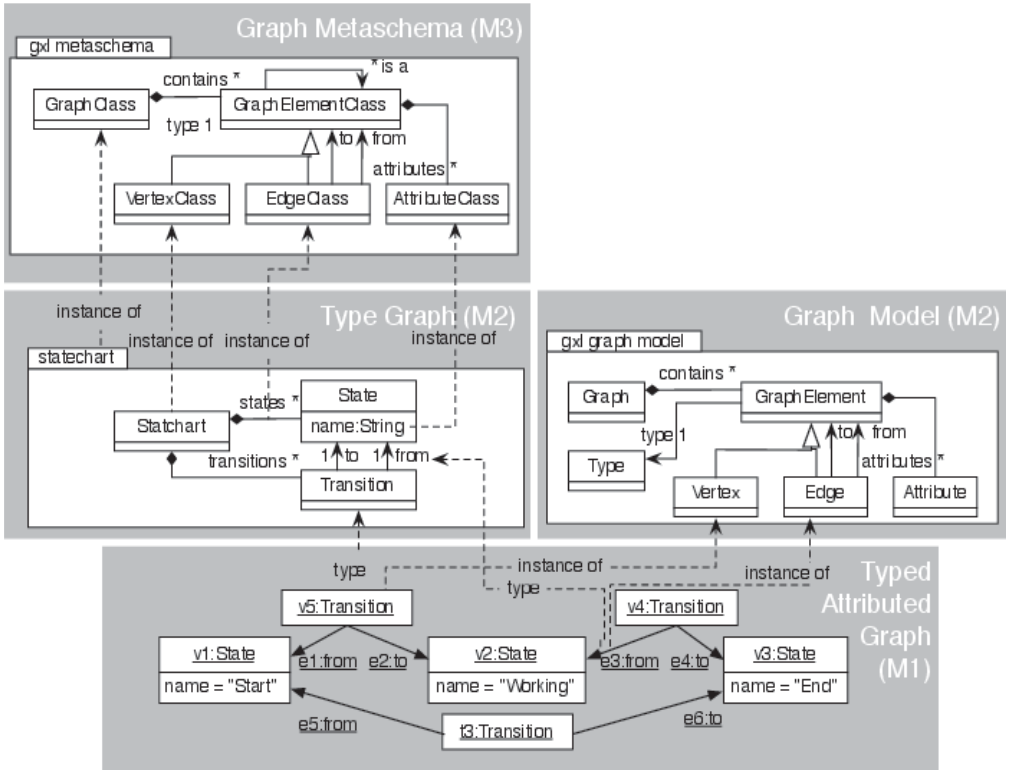
- GXL ist eine moderne Graph-Sprache (Graph-Austauschformat)
- Enthält Abstraktionen für Elemente von Graphen, die für generische Algorithmen genutzt werden können (flexible Navigation)



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

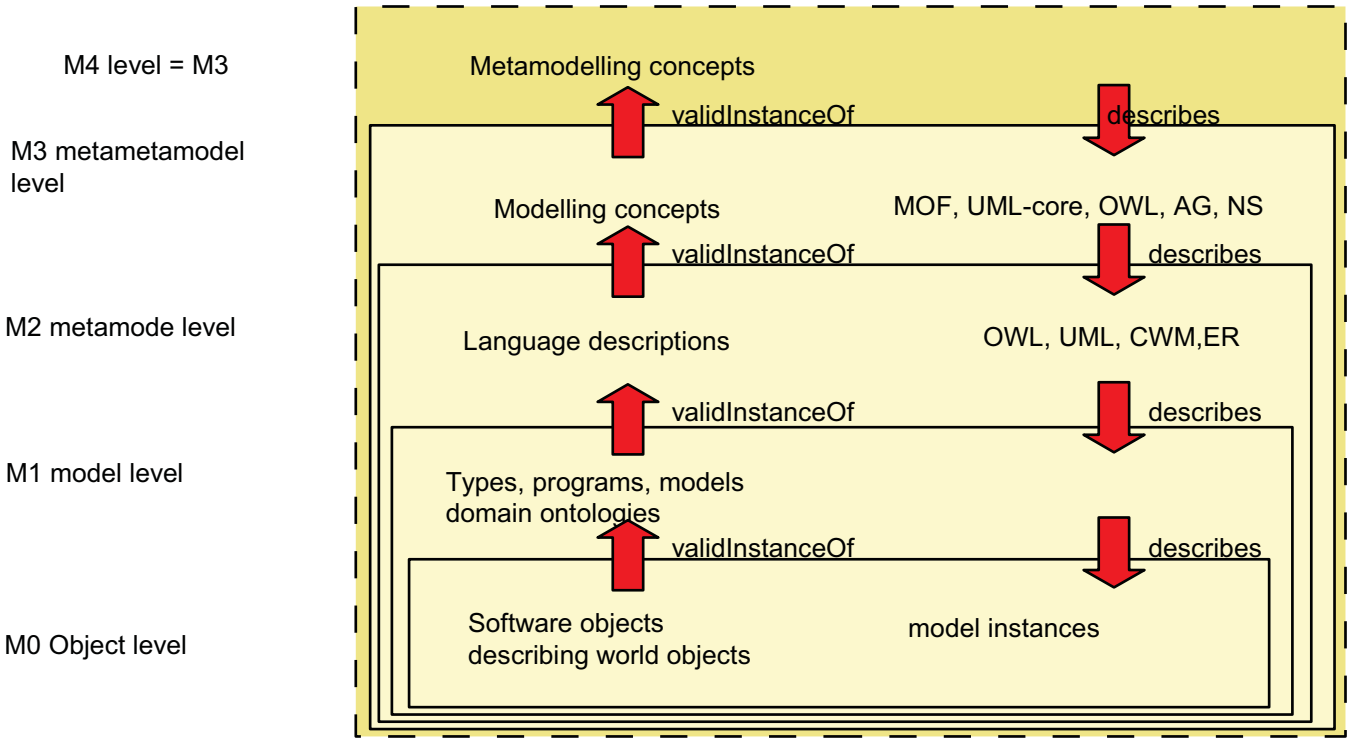
GXL-based Metamodel of Typed Attributed Graph

- Auf M2 können auch domänenspezifische Sprachen definiert werden mit Anwendungssemantik (hier statecharts)
- GXL kann als Metasprache (Metametamodell) genutzt werden, sowohl für Graphen als auch für Statecharts und andere domänenspezifische Sprachen

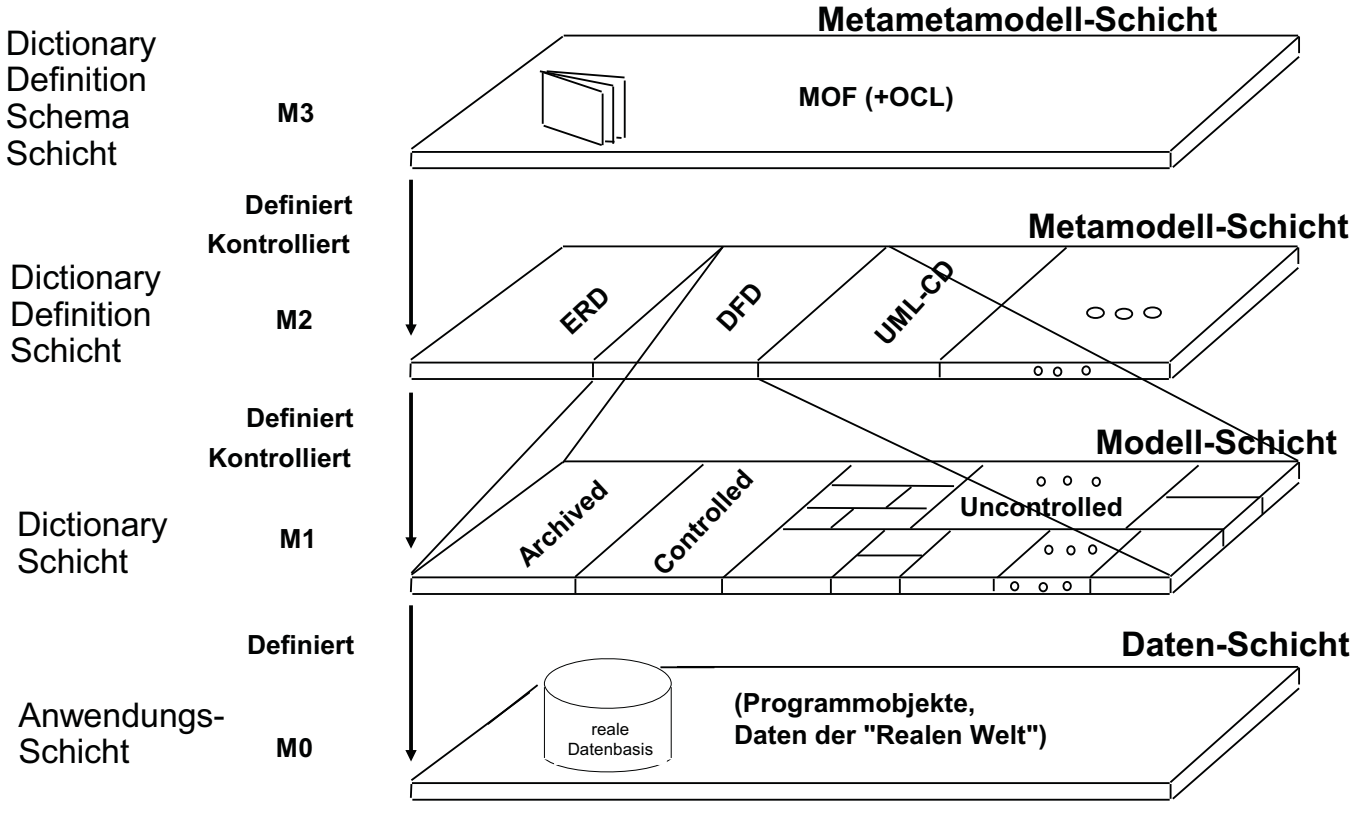


The IRDS/MOF Metamodelling Hierarchy

- aka *metapyramid*



Bsp.: IRDS/MOF Metahierarchie für Data Dictionaries in der Strukturierten Analyse (SA)



Paketierung

- Alle Schichten können in Pakete (Module) eingeteilt sein

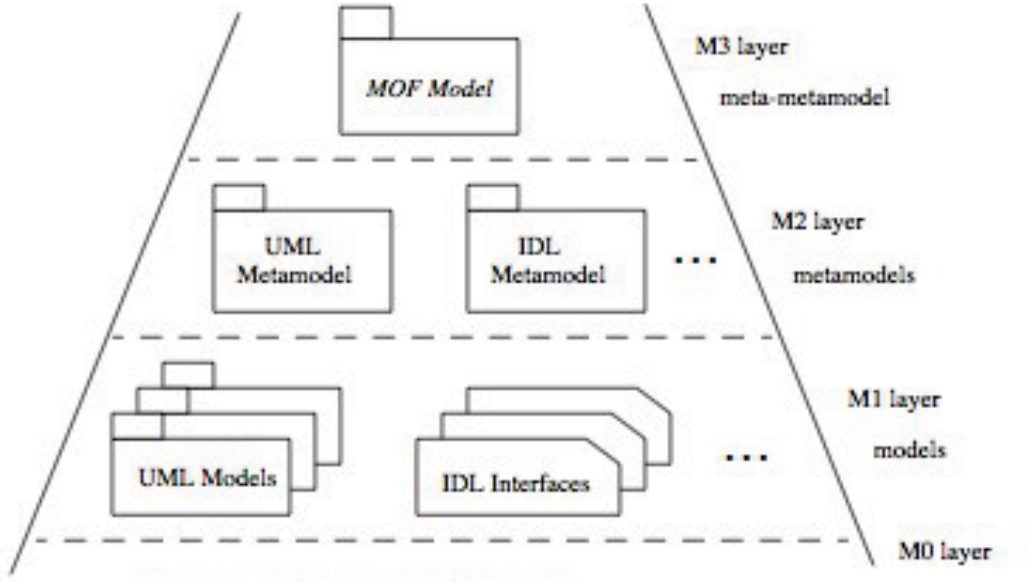


Figure 2-2 MOF Metadata Architecture

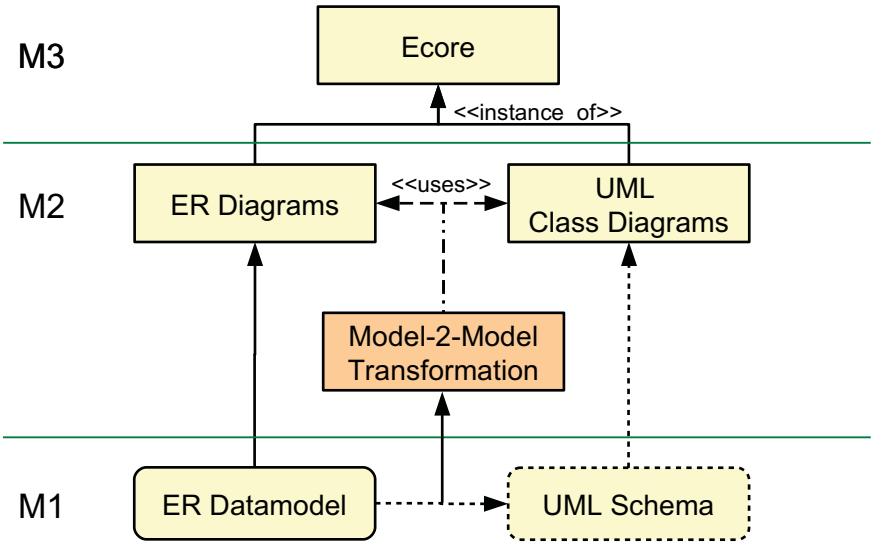
[MOF]

Nutzen der Metamodell-Architektur

- Mittels **Meta-Metamodellen (Metasprachen)** lassen sich
  - beliebige Metamodelle konkreter *Modellierungssprachen* definieren und
  - Metamodelle für *domänenspezifischen Sprachen (DSL)* definieren.
- Auf Basis von Metaebenen können verschiedene Beschreibungssprachen ineinander **überführt** werden *(Model-driven Architecture; MDA)*
  - Hierarchische Anordnung der einzelnen Modellebenen ermöglicht schrittweise Verfeinerung der semantischen Konzepte
  - **Transformationsbrücken** (z.B. Transformation eines ER-Diagrams in ein UML-Diagram, wenn beide Diagram-Sprachen als Instanz von Ecore erstellt wurden)
- Metamodelle bieten:
  - prägnante, präzise Definition von Softwareobjekten und -dokumenten
  - Vertiefung semantischer Beziehungen und Regeln (Konsistenzprüfung)
  - automatisierte Implementation von Werkzeugen für zu unterstützende Methoden
  - Fähigkeit der Selbstbeschreibung und Überprüfbarkeit mit eigenen Mitteln

Excurse: Modeltransformations

- Modeltransformations defined in Layer M_{i+1} specify how to transform models on M_i
  - Source and target metamodel
- **Benefit**: Transformation can be reused for all models, which are instances of the source meta-model



Metamodelle für CASE

Metamodelle für CASE basieren auf textuellen oder graphischen **Beschreibungen einer Methode** oder einer **Notation**, aus deren Interpretation, Compilier und CASE-Werkzeuge **generiert, konfiguriert** oder **parametrisiert** werden können.

- Die auf Metamodellen beruhenden SEU werden oftmals auch als **Meta-CASE** bezeichnet.
  - Die Sprache, in der die Metamodelle erstellt werden, wird Metasprache genannt (Auf Ebene M3)
  - Sie beinhalten im Allgemeinen eine Technologie zum Entwickeln und zum Erzeugen von CASE.
  - unterstützen eine oder mehrere Entwicklungsmethoden
  - unterstützen automatisch (Generierung,funktionaler Aspekt) oder halbautomatisch (Modellierung, statischer Aspekt) die Entwicklung von CASE-Tools

Quellen: http://www.uni-koblenz.de/FB4/Institutes/IST/AGEbert/MainResearch/MetaTechnology/Kogge
http://www.cs.usask.ca/grads/vsk719/academic/856/project/node8.html
http://www.cs.ualberta.ca/~softeng/Theses/zhu.shtml
http://www.metacase.com/de/index.html

Metamodeling – Benefits

F. Klar, TU Darmstadt

Metamodeling - Summary

- Discussed metametamodels are
  - Complete MOF and Essential MOF (EMOF/CMOF)
    - Ecore as Eclipse's implementation of EMOF
    - UML core – a subset of CMOF
  - GXL – Graph eXchange Language
  - GOPRR – Graph, Object, Property, Role, Relation
- Meta-Hierarchy
  - M3 to M0
- Example metamodels (Statecharts, ER, Class Diagrams

Excurse: Metaprogramming

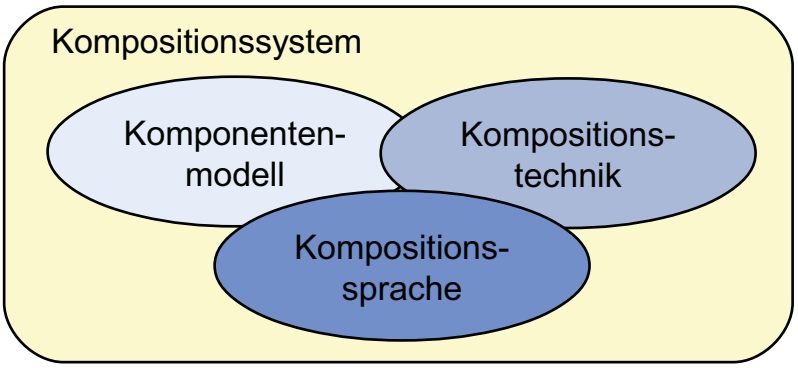
- **Metaprogramme (Reflektive Programme)** liefern Code, besitzen also ein Metamodell oder Grammatik als Typsystem
- **Metaprogramm-Prozeduren** (Semantische Macros, Programmable Macros [Weise/Crew]) sind durch das Metamodell bzw. die Grammatik typisiert:
  - Ihre Pararametertypen sind Metaklassen oder Nicht-Terminale
  - Sie haben als Rückgabewert eine Metaklasse oder Nicht-Terminal

11.3 Modell- und Metamodell-Komposition

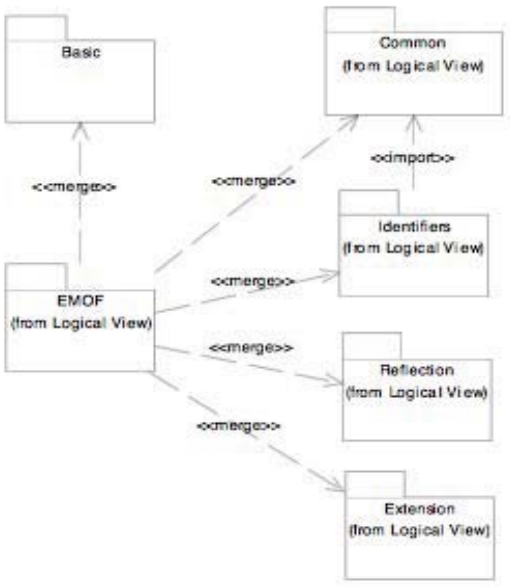
Einfaches Kompositionssystem für Modelle

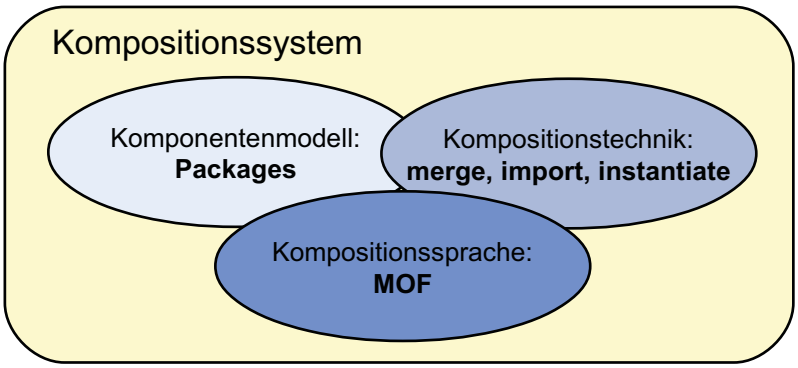
- Modelle und Metamodelle können in **Pakete** eingeteilt werden.
- Pakete sind Module, ein einfaches **Komponentenmodell** (siehe CBSE)
- Kompositionstechnik mit Kompositionsoperatoren auf Paketen (sehr einfache **Modellalgebra**):
  - use (import)
  - merge (union)
  - instance-of

→ heute werden Metamodelle aus Paketen komponiert



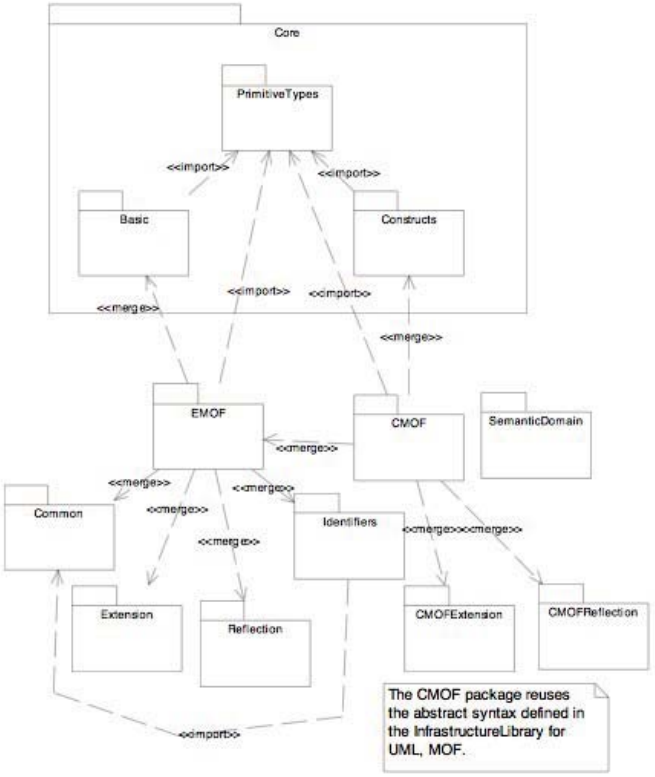
EMOF Classes Composition





[MOF]

CMOF Package Composition from UML Core and EMOF



[MOF]

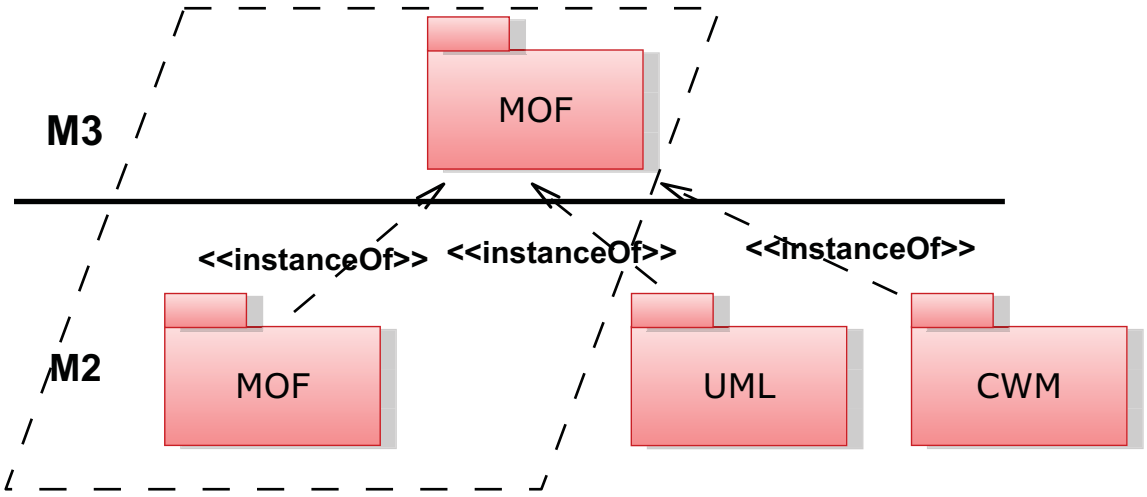
Promotion von Metamodellen

Ein Metamodell einer Datenstruktursprache aus M2 wird **angehoben (lifted, promoted)**, wenn es als Metasprache auf M3 genutzt wird

- MOF ist eigentlich eine einfache DDL (Datendefinitionssprache, Strukturspache) für Graphen
  - Man kann es auf M2 nutzen, um mit Paket-Merge neue Sprachen zu definieren, z.B. wie bei UML
  - Man kann es auf M3 nutzen, um Metamodelle als Instanzen zu bilden

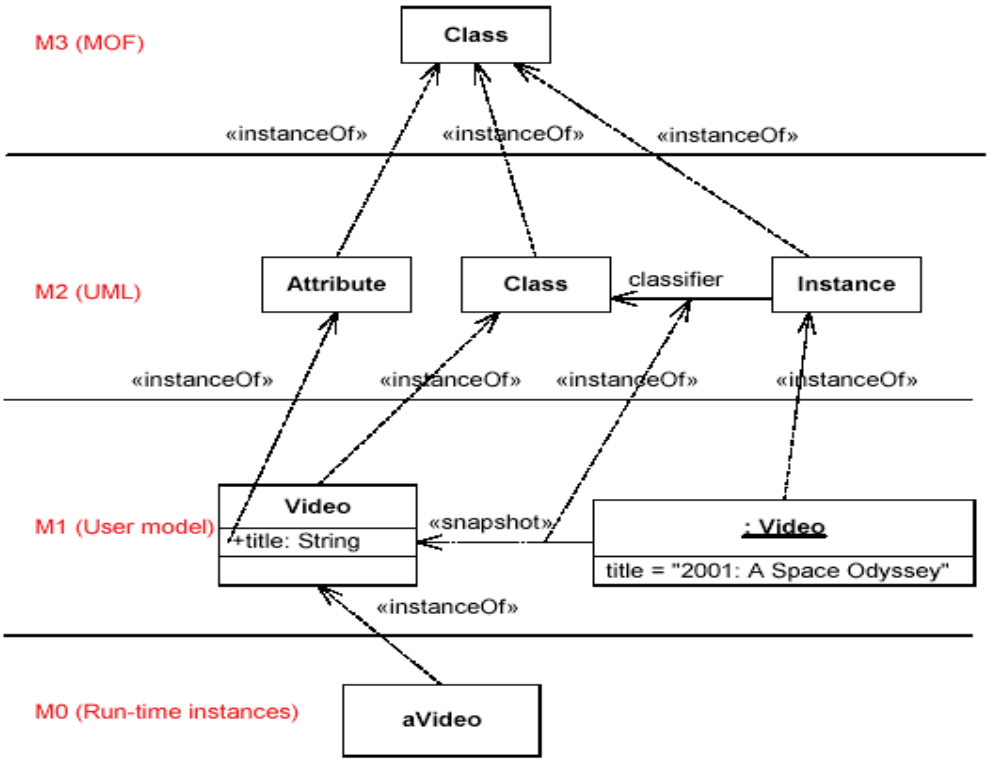
Von MOF abgeleitete Metamodelle

- MOF ist **selbstbeschreibend**, d.h. die Struktur von MOF ist in MOF spezifiziert
- MOF ist *angehoben (lifted)*, weil auf M2 und M3 verwendbar



11.4 Repetition: MOF und die UML-Metahierarchie

Bsp: MOF-Metamodell-Hierarchie für UML



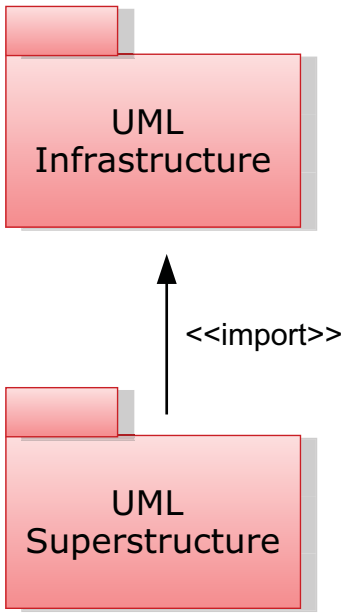
Quelle: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

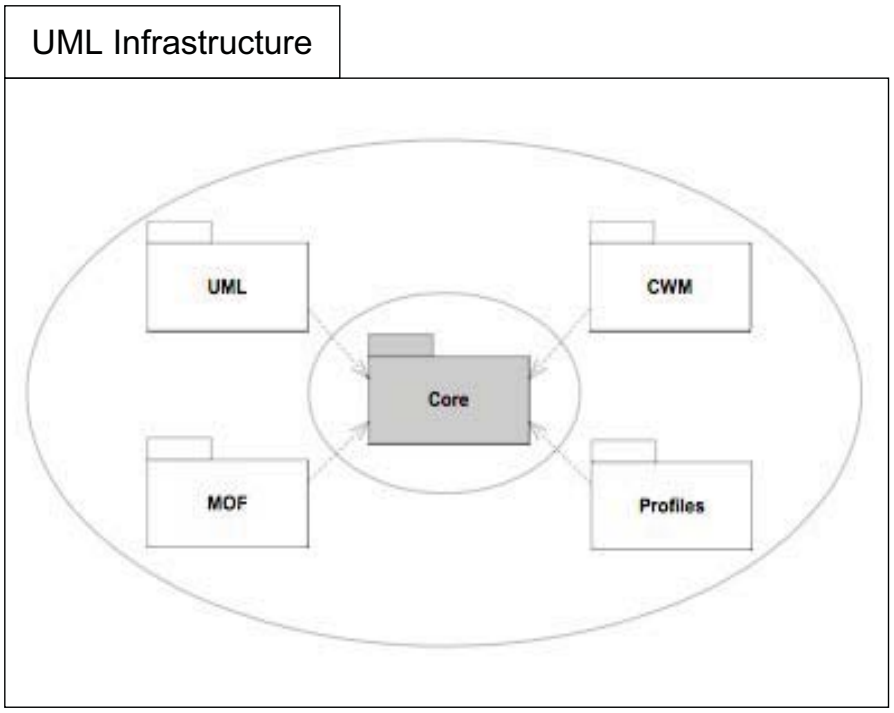
Bedeutung der UML-Metamodellierung für CASE

Das UML-Metamodell ist ein logisches (kein physikalisches oder Implementations-) Metamodell:

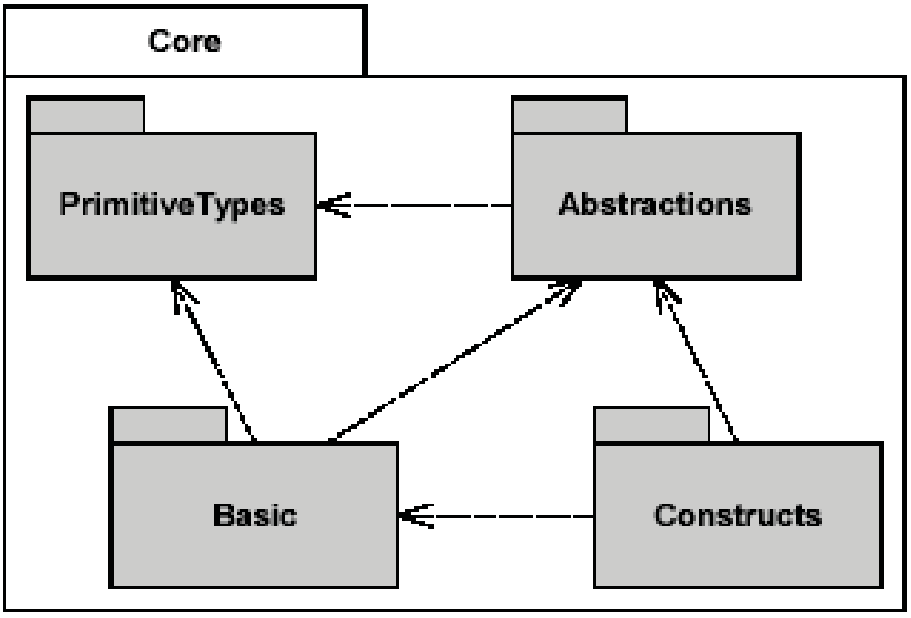
- aufgebaut aus **Paketen**, die komponiert werden können
- aufgebaut auf die **CMOF**-Paketstruktur
- einheitliche **Struktur** (kontextsensitive Semantik) für alle darzustellenden Diagramminstanzen, wie Statecharts (SC), Sequence Diagrams (SD), etc.
- **Schema für Repositories** zur einheitlichen Datenbeschreibung
- **Austauschformat** (XMI) für CASE-Werkzeugdaten
- Nutzung für Non-Standard-Applikationen möglich, wie multimediale und Echtzeit-Applikationen

Struktur von UML auf M2





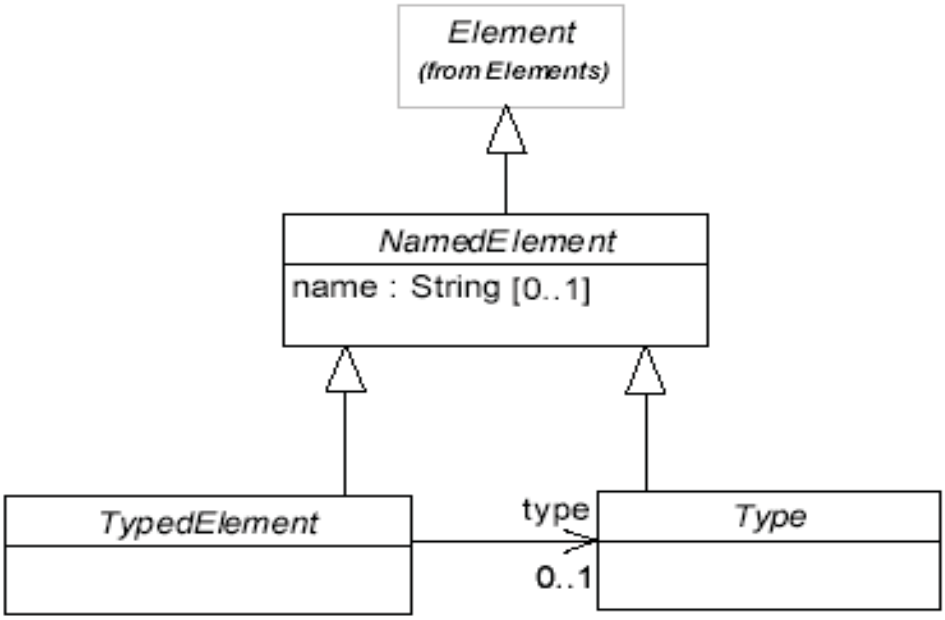
Core Package des UML-Metamodells (M2)



Basic: Grundkonstukte für XMI
Constructs: Metaklassen für ooModellierung
Abstractions: abstrakte Metaklassen
Primitive Types: vordefiniert im Metamodell

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

Package Basic: Types from CMOF

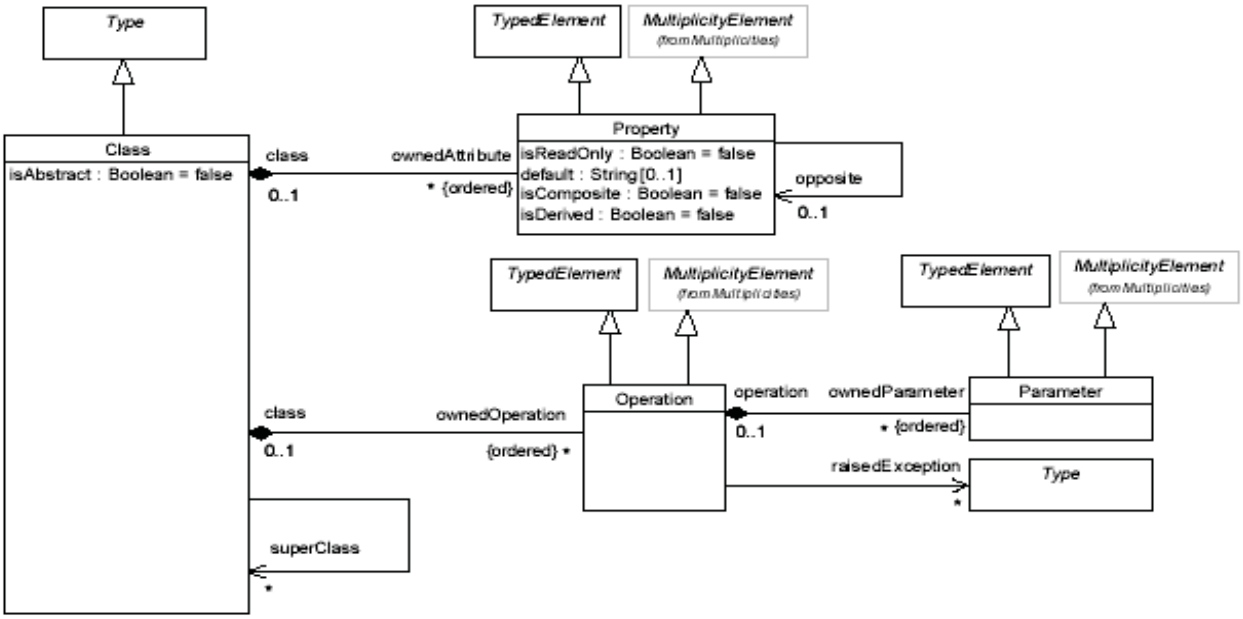


Die abstrakten Metaklassen dienen zum Benennen und zur Typdefinition von Elementen

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

Package Basic: Classes

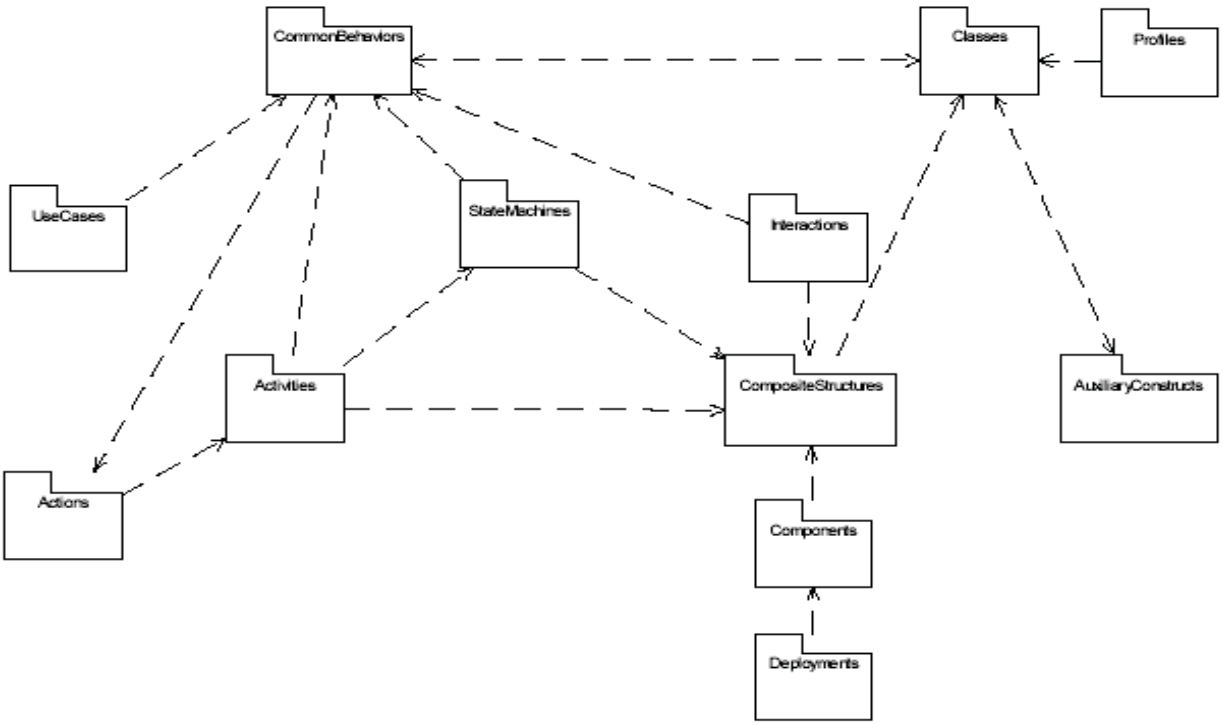
Definieren einer Klasse als Typ und Festlegung der weiteren Elemente zur klassenbasierten Modellierung



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

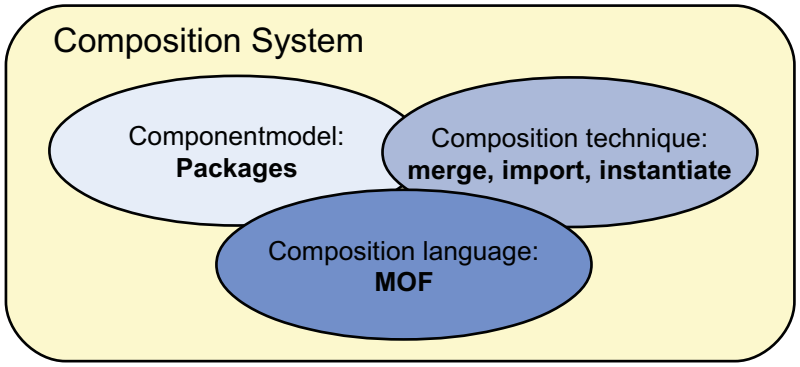
Package Composition UML 2.0 (M2)

Enthält alle Pakete zur systemstrukturierten und verhaltensorientierten Modellierung



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

((Meta-)Meta-)Model Composition – Summary



11.5 Technological & technical 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
  - [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]

A **technical space** is a model management framework 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]

Technikraum

Ein **Technikraum** (technical space) ist eine Plattform (Raum) zum Management von Modellen, durch eine Metasprache (auf M3) geprägt

- Ein Technikraum stellt Daten (auf M0), Code und Modelle (auf M1) sowie Sprachen (auf M2) zur Verfügung
  - Code und Modelle können mit den Operatoren einer **Modell-Algebra** manipuliert werden
  - Diese Operatoren bilden elementare Werkzeuge und können in komplexe Werkzeuge eingebettet werden
- SEU und MetaCASE unterstützen nur einen Technikraum
- Achtung, ein Technologieraum kann mehrere Technikräume enthalten:
  - Compiler community: Grammarware, Tree-Ware, Graph-Ware

Werkzeuge nur dann kombinierbar, wenn sie im gleichen Technikraum leben

Technikräume über der Metahierarchie

	Grammarware (Strings)		Tableware		Treeware (Bäume)		Graphware/Modelware				Ontology-ware
	Strings	Text	Text-Tabelle	Relationale Algebra	XML	NF2	MOF	Eclipse	CDIF	MetaEdit+	OWL-Ware
M3	EBNF	EBNF		CWM (common warehouse model)	XSD	NF2-Sprache	MOF	Ecore	ERD	GOPPR	RDFS OWL
M2	Grammatik einer Sprache	Grammatik mit Zeilentrennern	csv-header	Relationales Schema	XML Schema-beschreibung, z.B. xhtml	NF2-Schema	UML-CD, -SC, OCL	UML, many others	CDIF-Sprachen	UML, many others	HTML XML MOF UML DSL
M1	String, Programm	Text in Zeilen	csv Datei	Relationen	XML-Dokumente	NF2-Baumrelation	Klassen, Programme	Klassen, Programme	CDIF-Modelle	Klassen, Programme	Fakten (T-Box)
M0	Objekte				dynamische Semantik im Browser		Objektnetze				A-Box (RDF-Graphen)

Modelmanagement im Technikraum

- Eine **Modelmanagement-Umgebung** verwaltet Modelle eines Technologieraums mit einer einheitlichen einsortigen Modell-Algebra
  - Operatoren und Werkzeuge auf M1 können aus M2 generiert werden

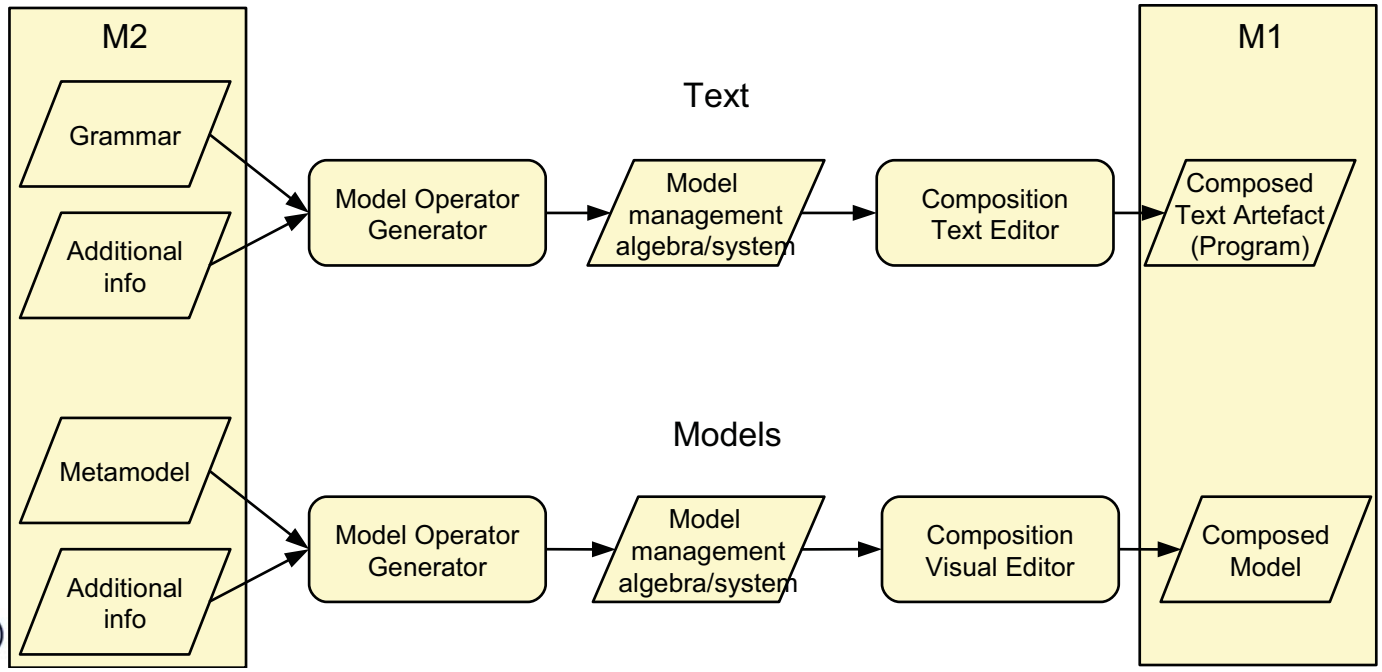


Abbildung von MOF-basierten Metamodellen auf andere Technikräume

MOF Mappings relate an M2-level metamodel specification to other M2 and M1-level artifacts, as depicted in Figure 2-6.

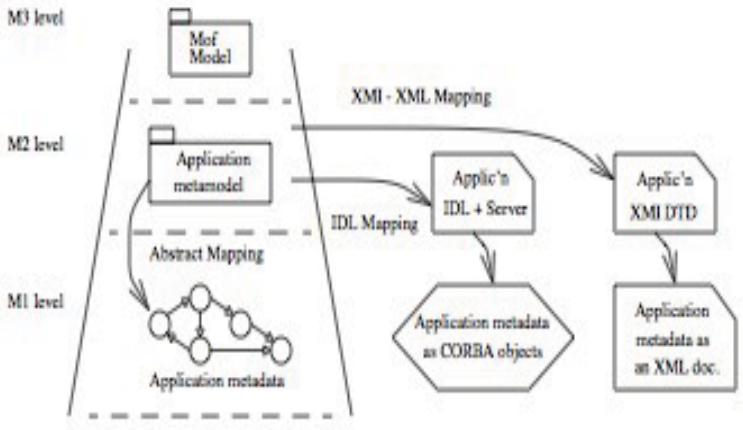


Figure 2-6 The function of MOF Technology Mappings

Multi-Technikraum-Werkzeuge

Ein **Multi-Technikraum-Werkzeug** ist ein Software-Werkzeug, das mehrere Technikräume zugleich nutzt.

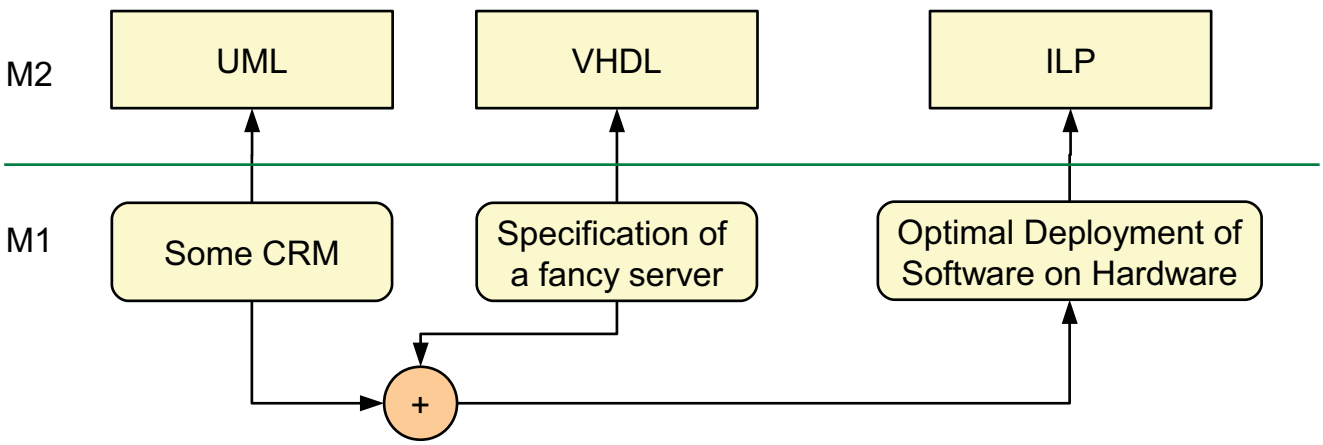
- Heute setzen alle Werkzeuge auf einem Technikraum auf. Es werden aber viele Technikräume zugleich benutzt, um eine große Software zu konstruieren (XML, Java, C, csv, ...)
- Die Werkzeuge der Zukunft werden mehrere Technikräume zugleich beherrschen
- Technikraumbrücken müssen gebaut werden

Model Engineering ist das Brücken von Technikräumen und kooperative Arbeiten in mehreren zugleich.

- Bezivin's Model Engineering Metapher: Die Welt besteht aus verschiedenen Dörfern, die durch Straßen verbunden sind. Jede Sorte von Ingenieur verwaltet ein bis mehrere Dörfer (“model villages”, Technologieräume). Straßen und Brücken zwischen diesen Technologieräumen zu bauen, ist unsere Aufgabe
- Das Ziel der Vorlesung ist, Model Engineering begreiflich zu machen.

Example

- To automate the optimization of software systems you need
  - A language to describe software systems (e.g., UML)
  - A language to describe hardware (e.g., VHDL)
  - A language to express the optimization problem (e.g., ILP)

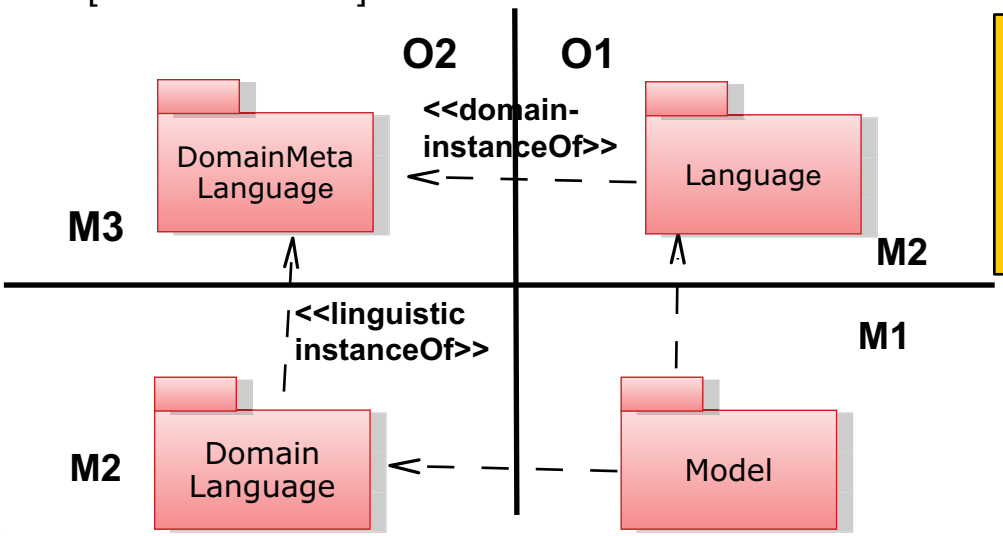


11.6 Megamodelle

Ein Megamodell ist eine Infrastruktur für Metamodelle und Modelle

2-D Metamodellierung

- Die Metahierarchie ist nicht die einzige Meta-Struktur.
- Man kann instance-of auch 2-dimensional anordnen. Dann ist jedes Modellelement instanz dreier Metaklassen, von der Sprache, der domänenspez. Sprache und der domänenspez. Metasprache
- [Atkinson/Kühne]



Eine stereotype instance-of-Struktur heißt **Megamodell** [Favre]

The End