

11. Metamodellierung

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und
Multimediatechnik
<http://st.inf.tu-dresden.de>
Version 11-1.1, 20.10.11

- 1) Metamodellierung
- 2) Metasprachen und
Metamodelle
- 3) Modell- und Metamodell-
Komposition
- 4) MOF und die UML-
Metahierarchie
- 5) Technikräume
- 6) Megamodelle



SEW, © Prof. Uwe Aßmann

1

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>



Prof. U. Aßmann, SEW

2

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.
- ▶ Gašević, 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
- ▶ [Niil] C. Niil. 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.

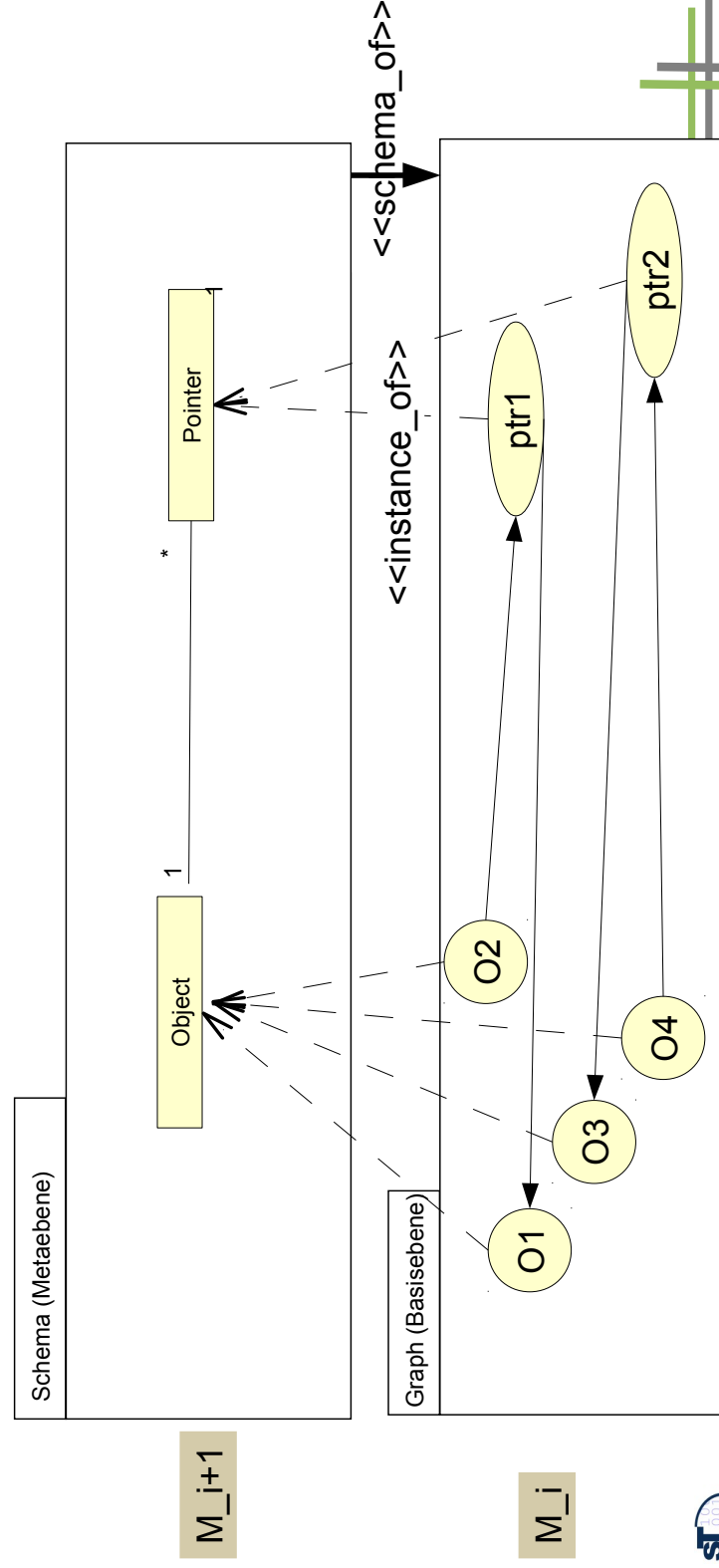
11.1 Metamodellierung

▶ 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** sind Modelle, deren Instanzen selbst Modelle sind. Sie beschreiben die Struktur von Prozess-, Domänen- und Systemmodellen

Typisierte Objekte (Modelle und Metamodelle)

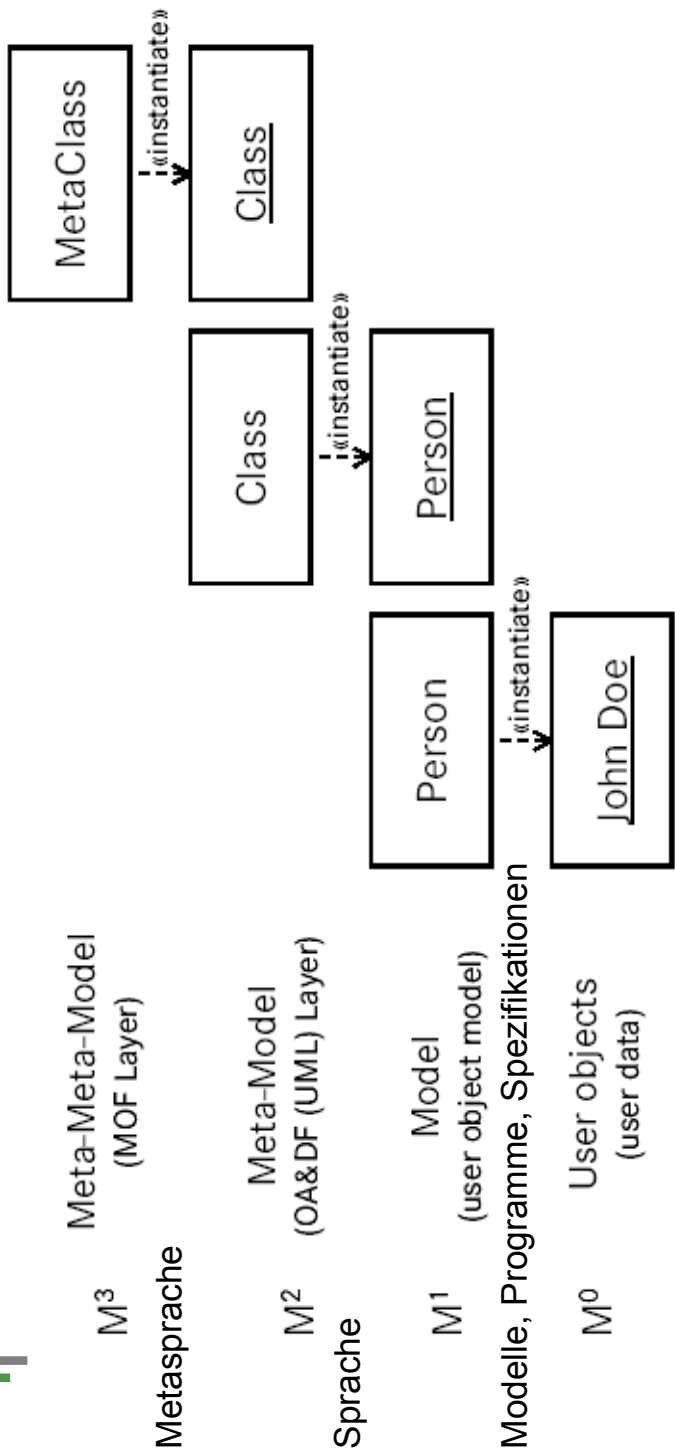
- ▶ Objekte können typisiert sein
- ▶ Unterscheide **Schemaebene (Meta-, Typeebene)** von **Instanzebene**



M_{i+1}

M_i

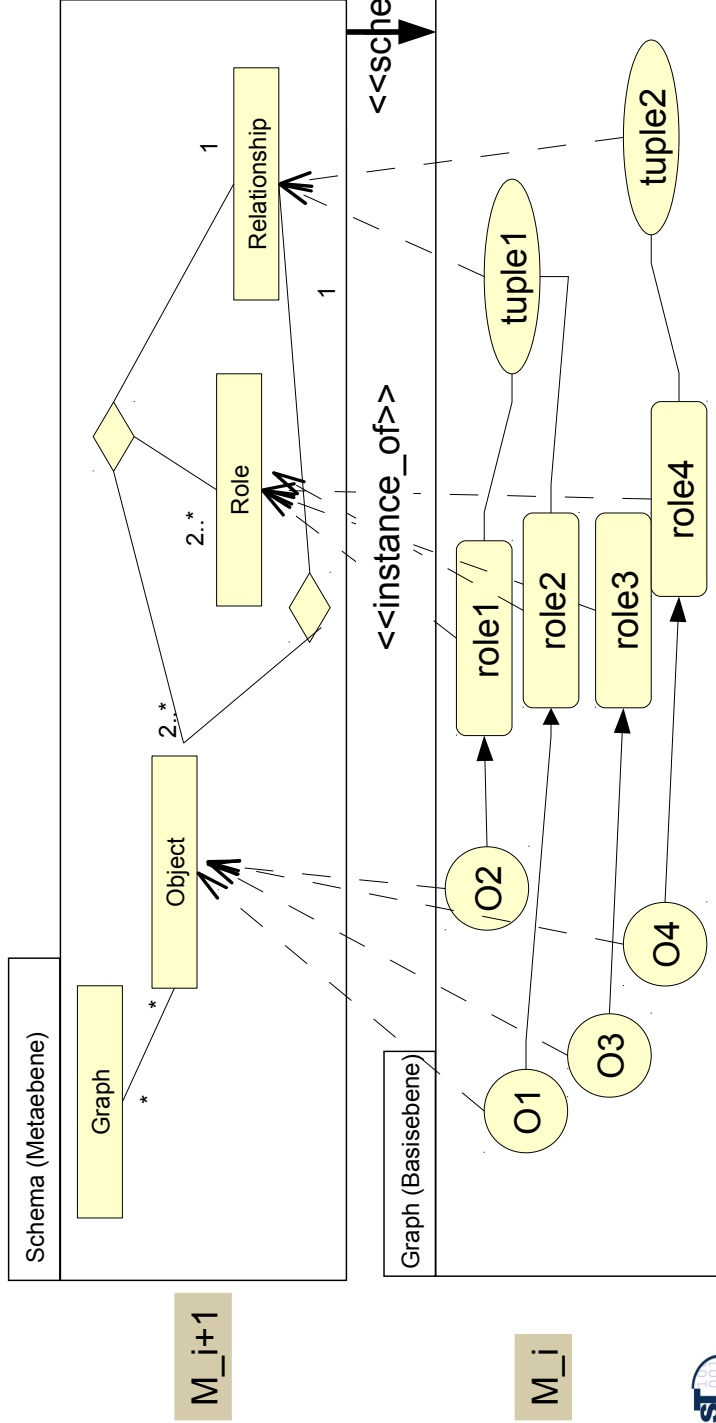
OMG's 4-Schichten Metamodel Architektur (MOF-Metahierarchie (urspr. IRDS Metahierarchie))



Quelle: Jeckle, M.: XML basierter Metadaten austausch; 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. Communications of the ACM(CACM), 31:704-718, June 1988.

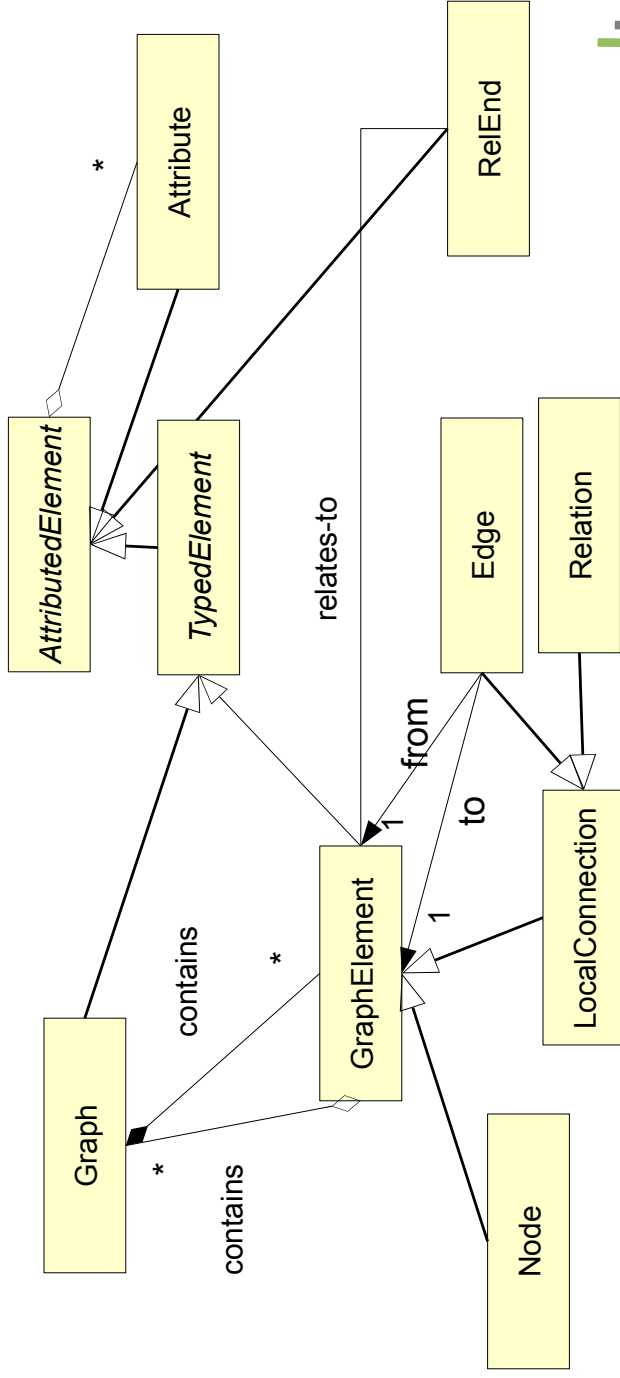
Typisierte Graphen (Modelle und Metamodelle)

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



GXL Graph eXchange Language

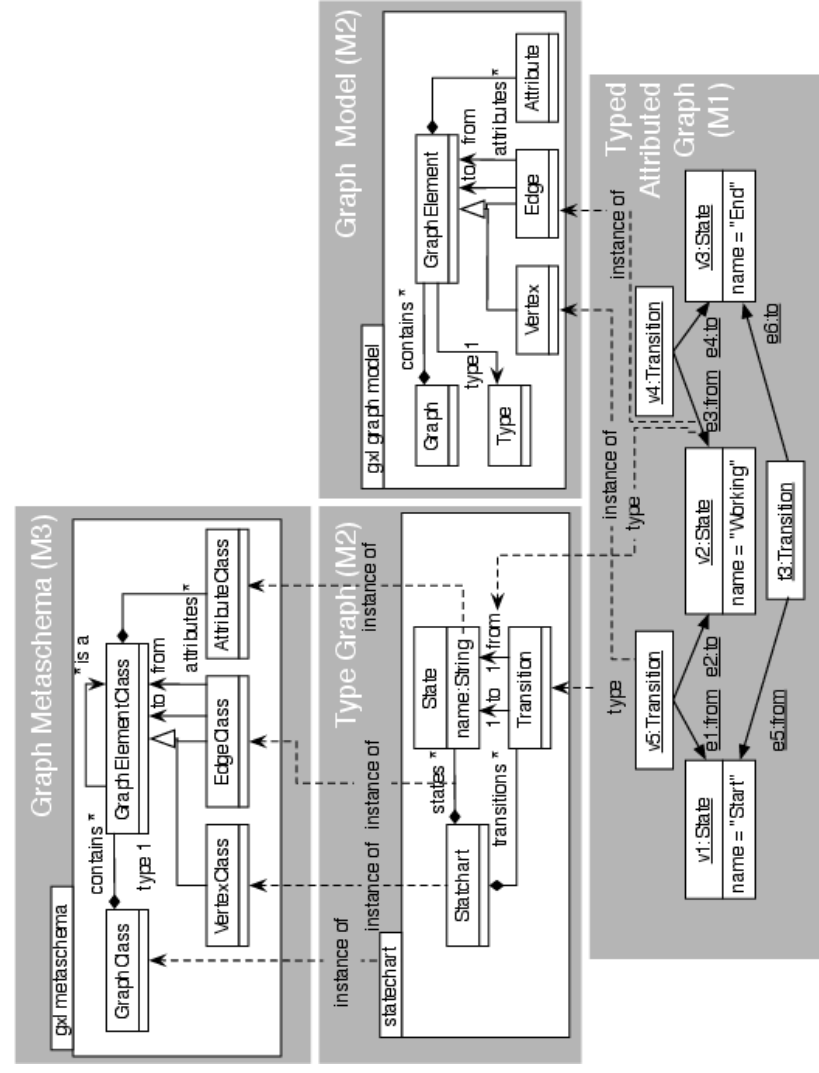
- ▶ GXL ist eine moderne Graph-Sprache (Graph-Austauschformat)
- ▶ Enthält Abstraktionen für Elemente von Graphen, die für generische Algorithmen genutzt werden können (flexible Navigation)

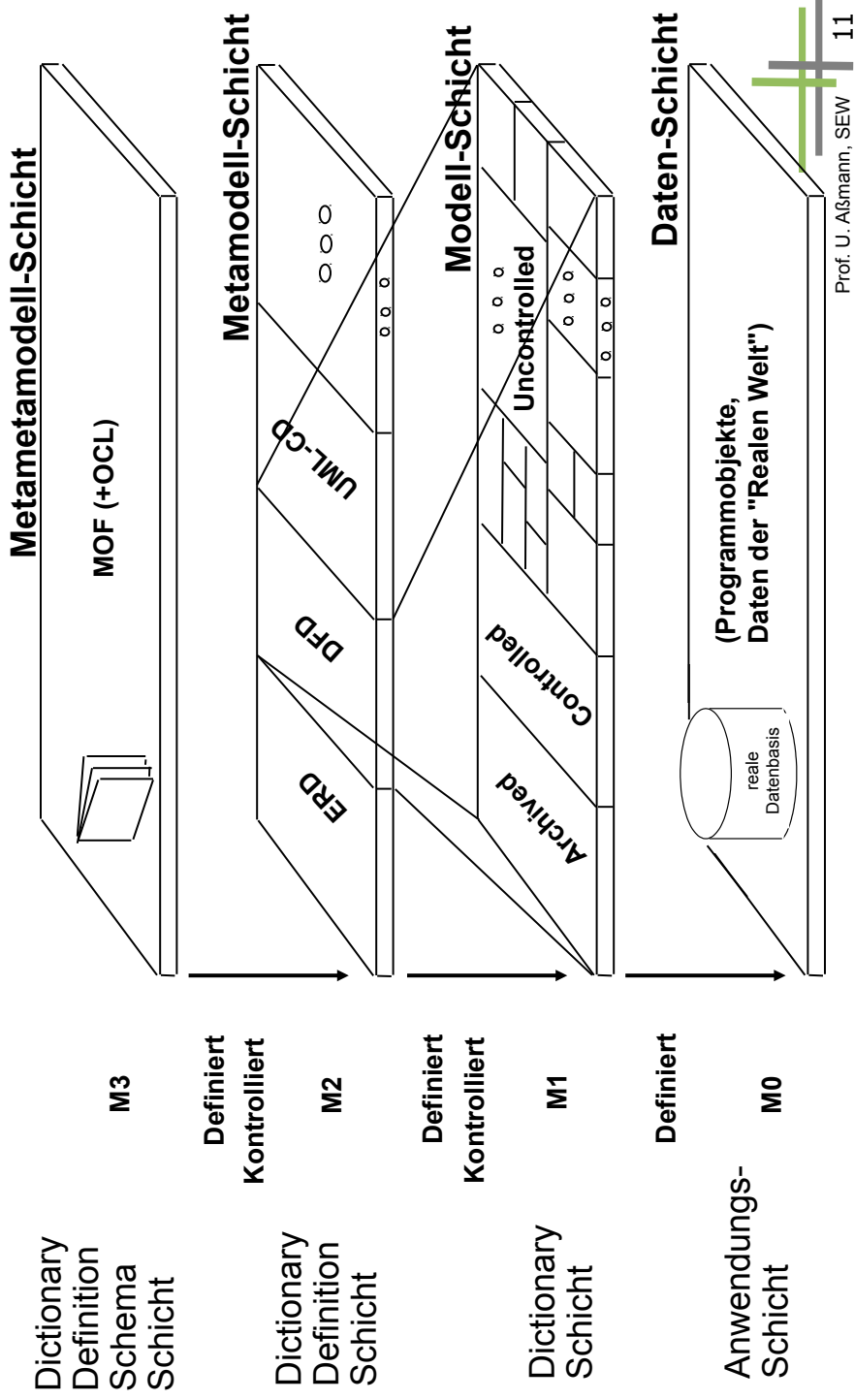


Richard C. Holt, Andy Schürr, Susan Elliott Sim, Andreas Winter. GXL: A graph-based standard exchange format for reengineering. Science of Computer Programming Volume 60, Issue 2, April 2006, Pages 149-170

GXL-based Metamodel of Typed Attributed Graph

- ▶ GXL kann als Metasprache (Metametamodell) genutzt werden





Paketierung

- ▶ Alle Schichten können in Pakete (Module) eingeteilt sein

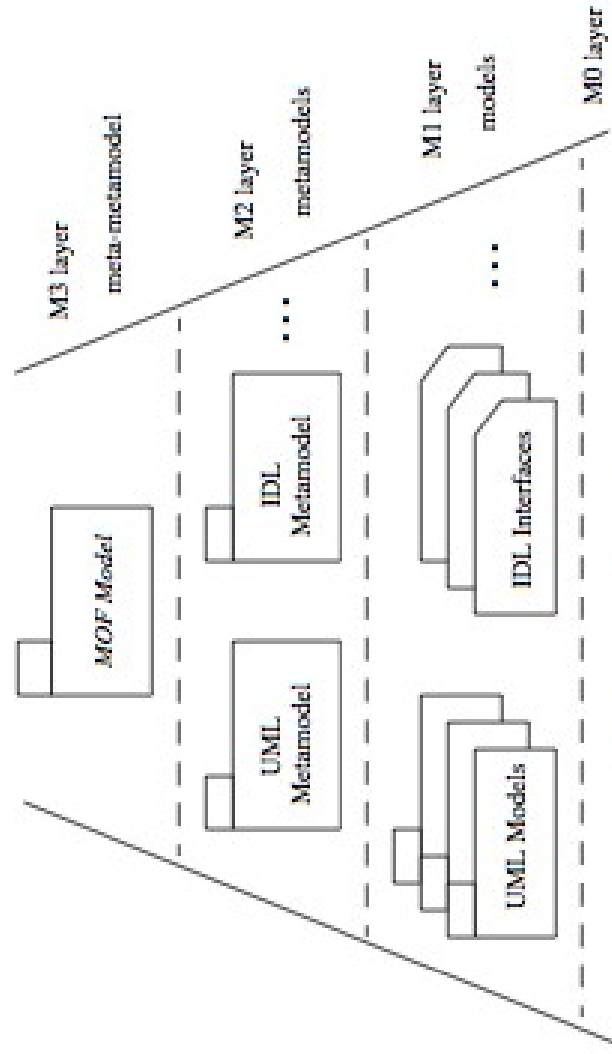
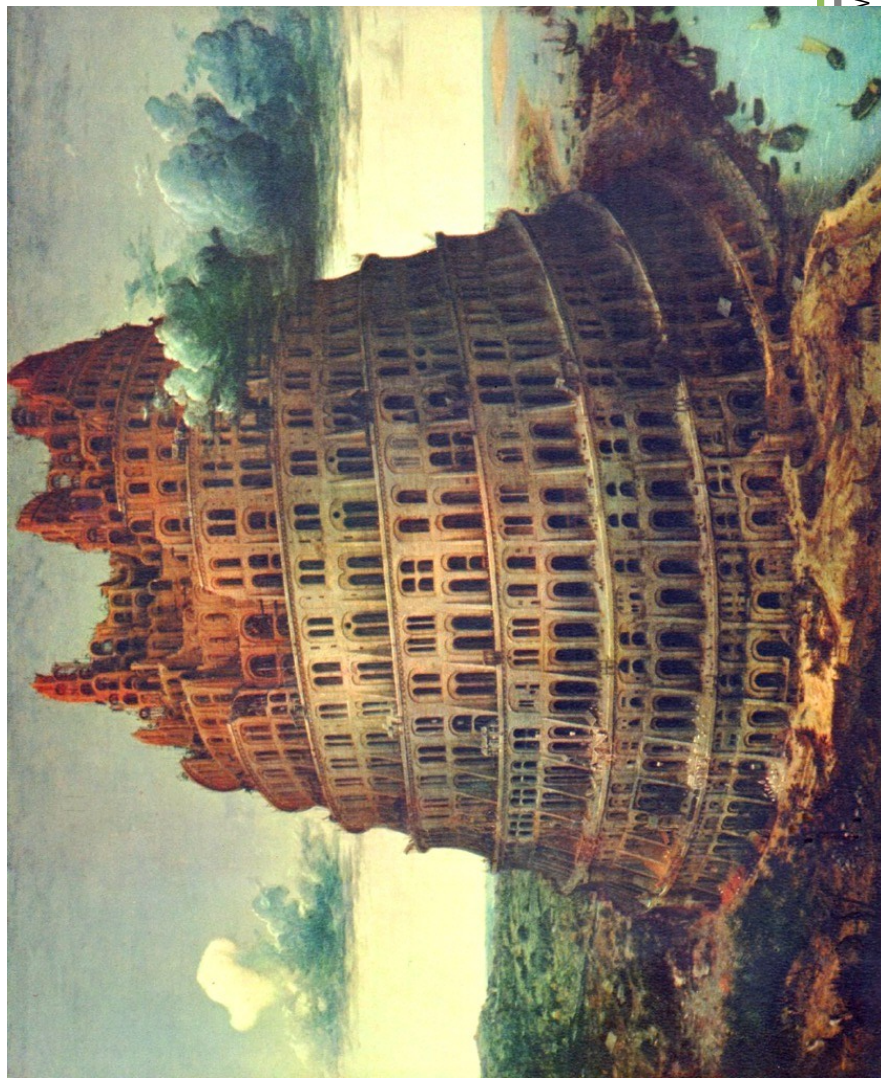


Figure 2-2 MOF Metadata Architecture

Das Metametamodel (Metasprache, Meta language)

- ▶ Ein **Metametamodel** ist ein Graphschema einer Sprache
 - Es fasst die kontext-sensitive Syntax der Sprache (Konzepte (Metametaklassen), Relationen)
 - Es definiert Struktur, kein Verhalten
- Ein Metametamodel ist normalerweise schlank (minimalistisch)
- Ein einheitliches Metametamodel ist nicht in Sicht... (tower of babel)

Tower of Babel Problem



Jan-Pieter
Breughel
(wikipedia)

Overview of Metalanguage MOF (CMOF: Complete MOF)

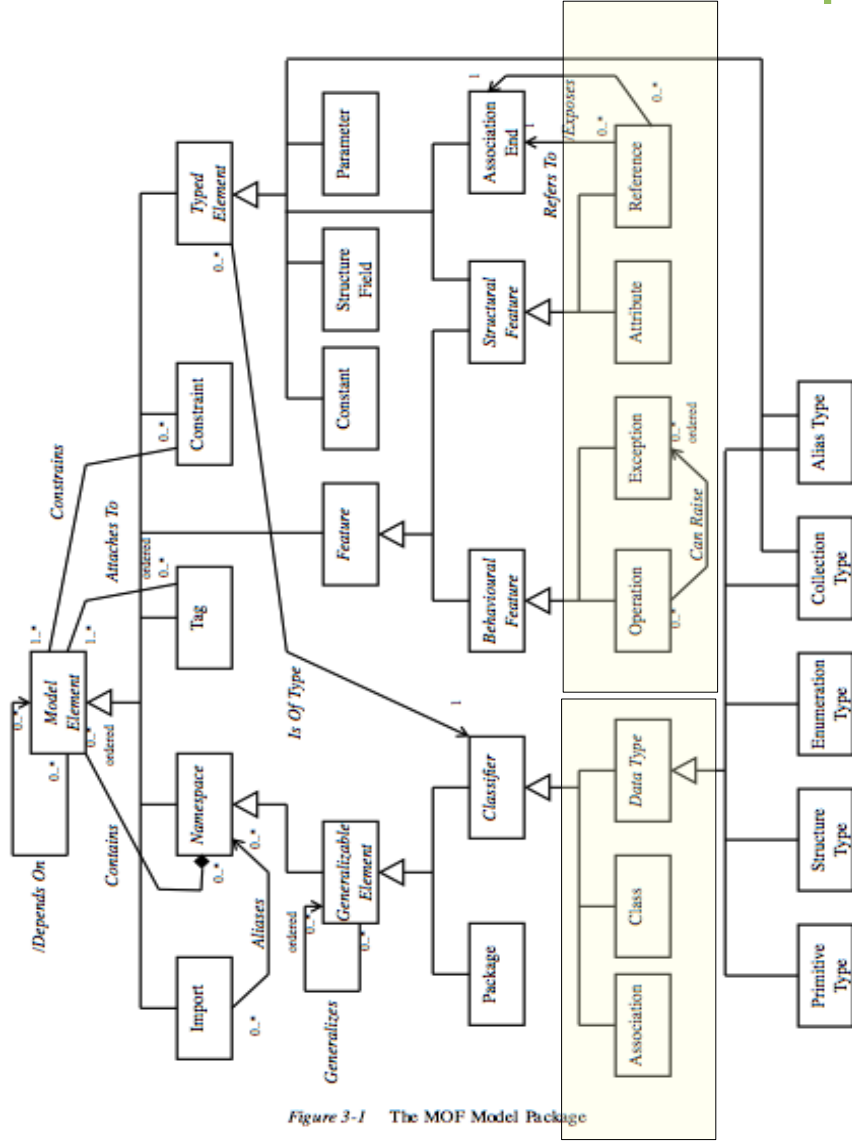
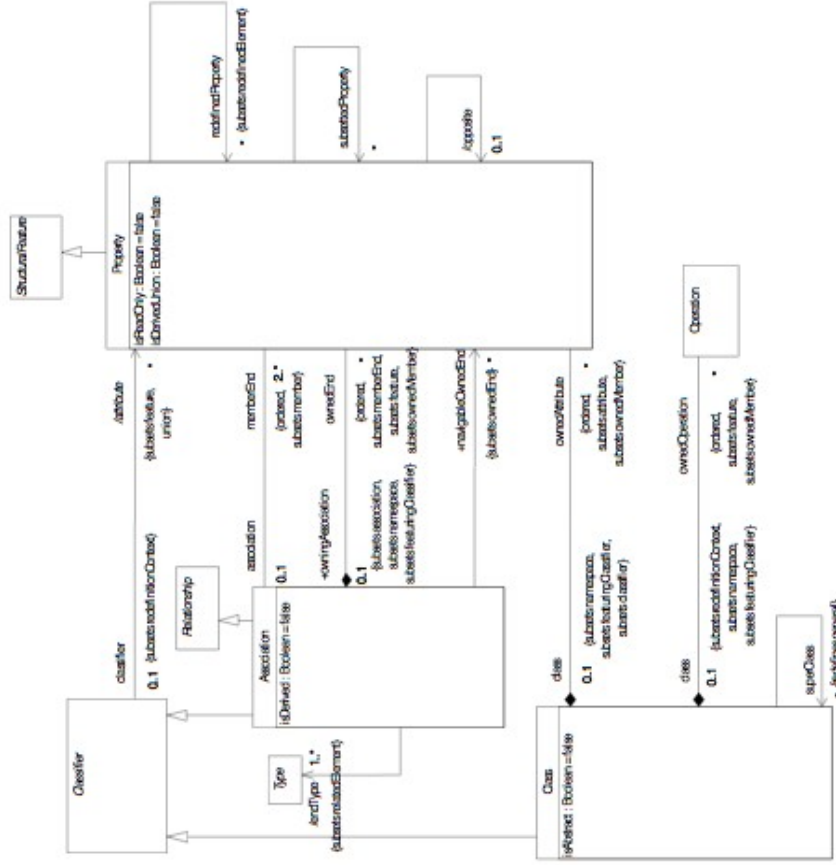


Figure 3-1 The MOF Model Package

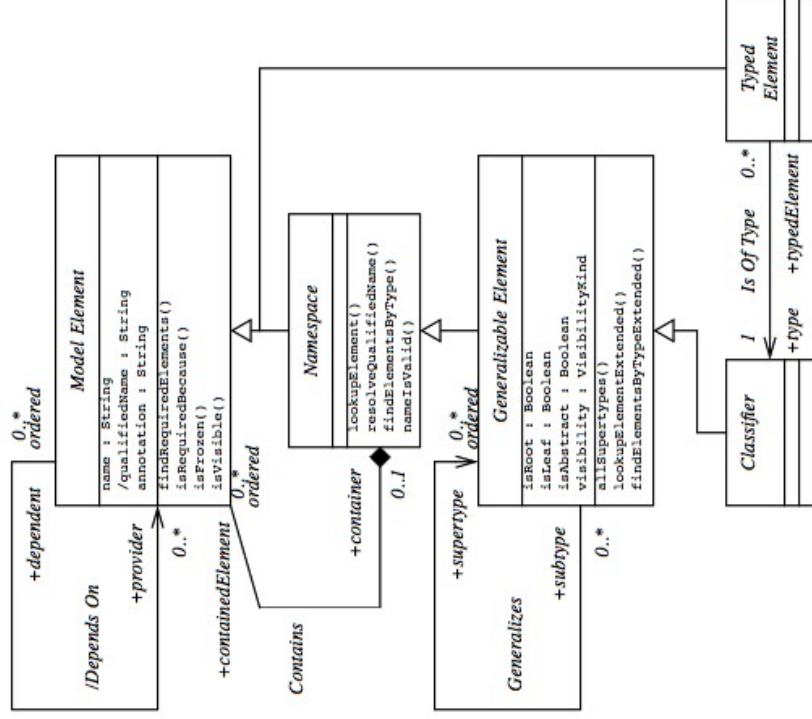


UML Core

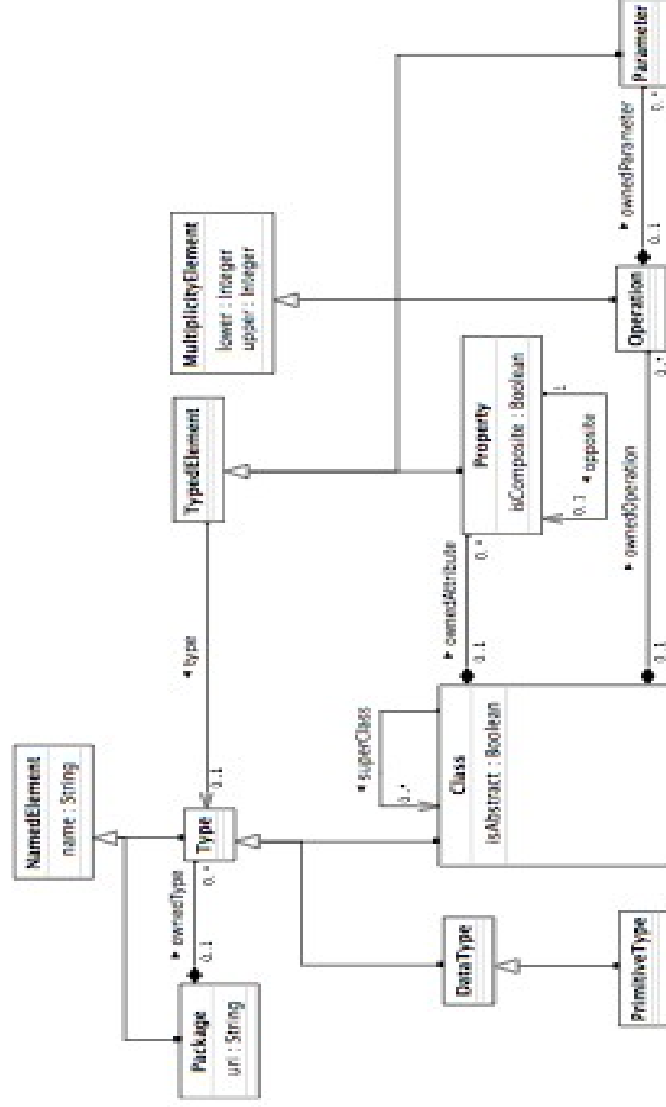
- UML core ist Teil von MOF, und auch Teil von UML-CD



MOF Central Types

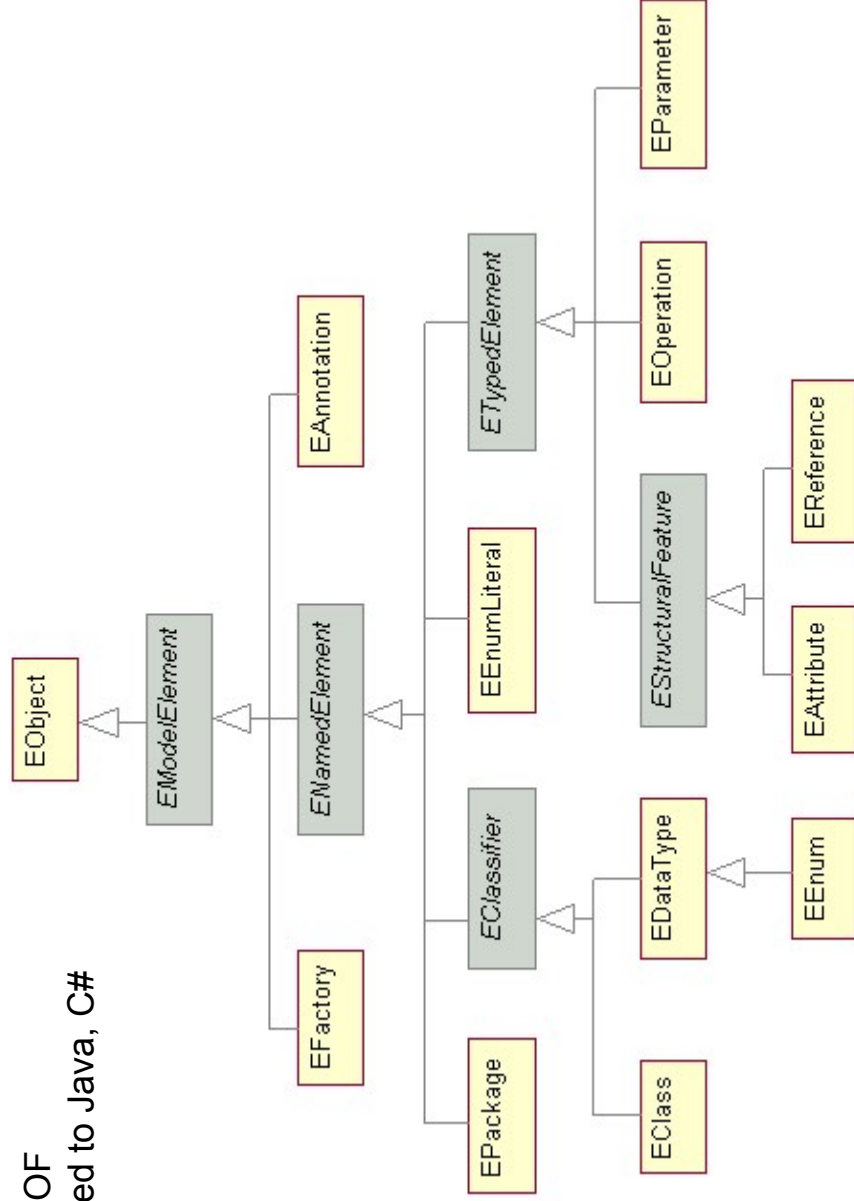


Central MOF Metaclasses with Associations



EMOF (Essential MOF)

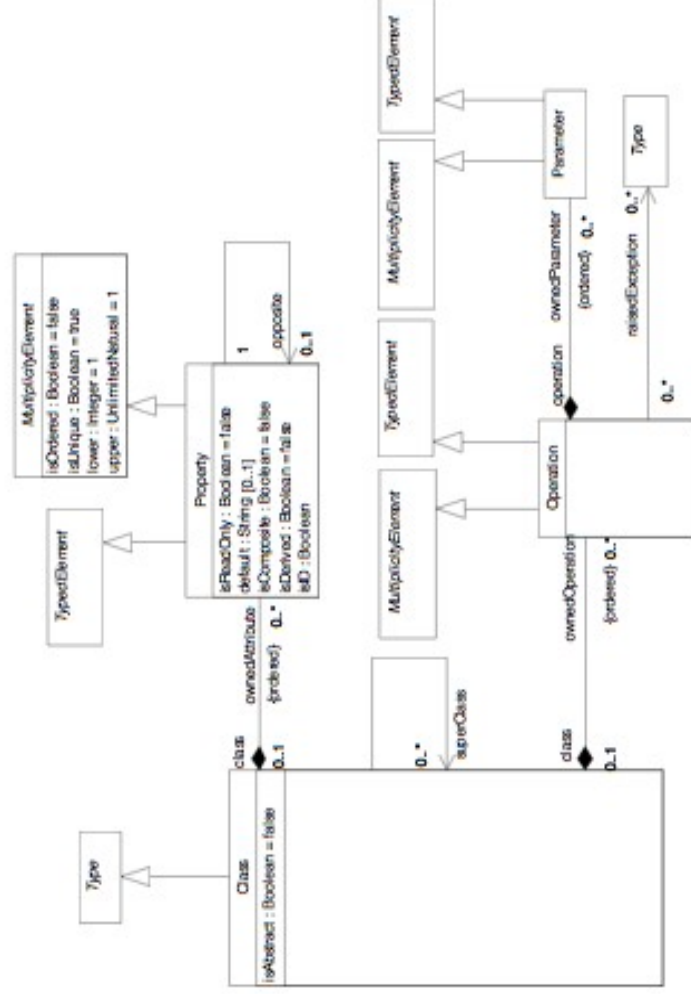
Subset of CMOF
Can be mapped to Java, C#



[MOF]



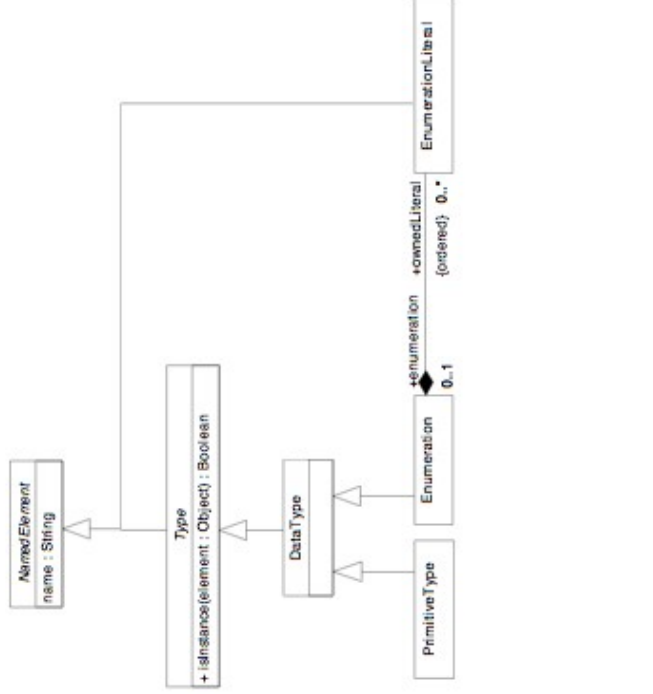
EMOF Classes in Detail



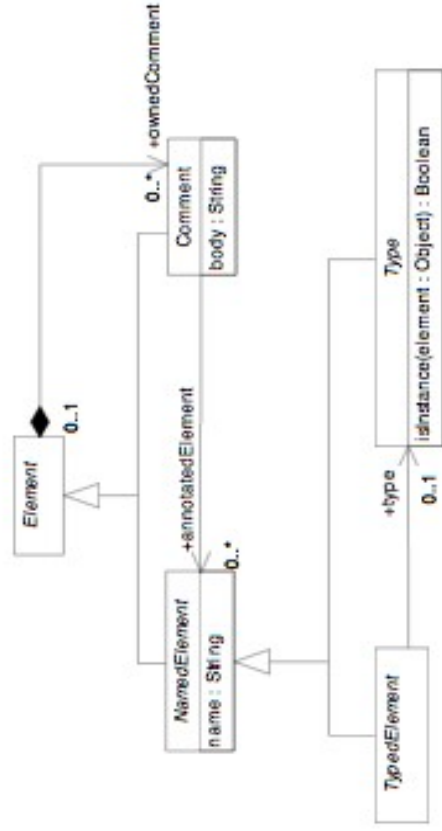
[MOF]



EMOF Data Types and Packages

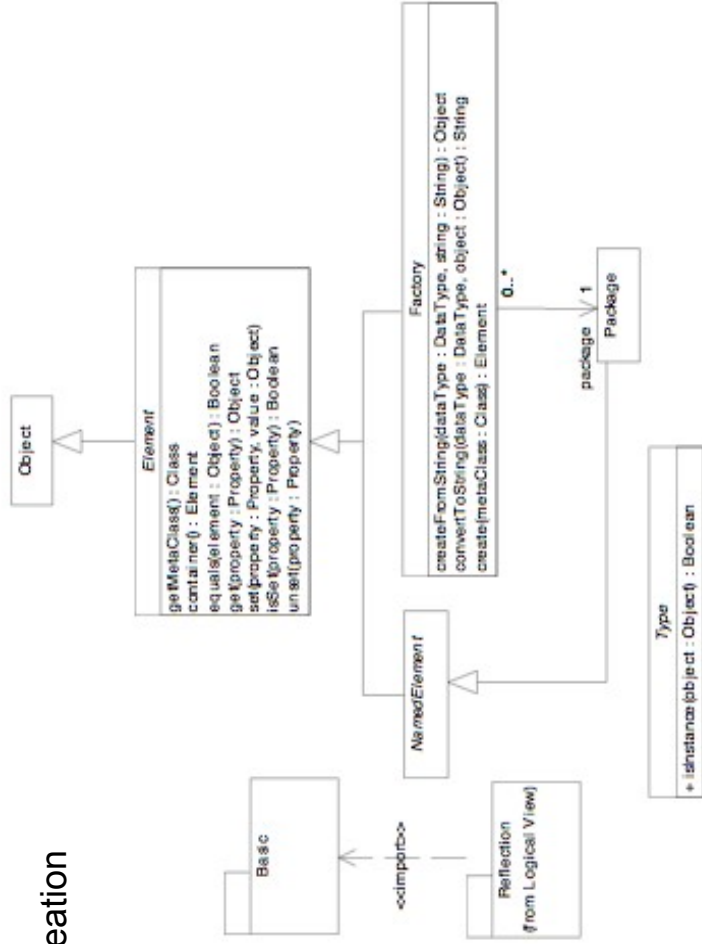


EMOF Types

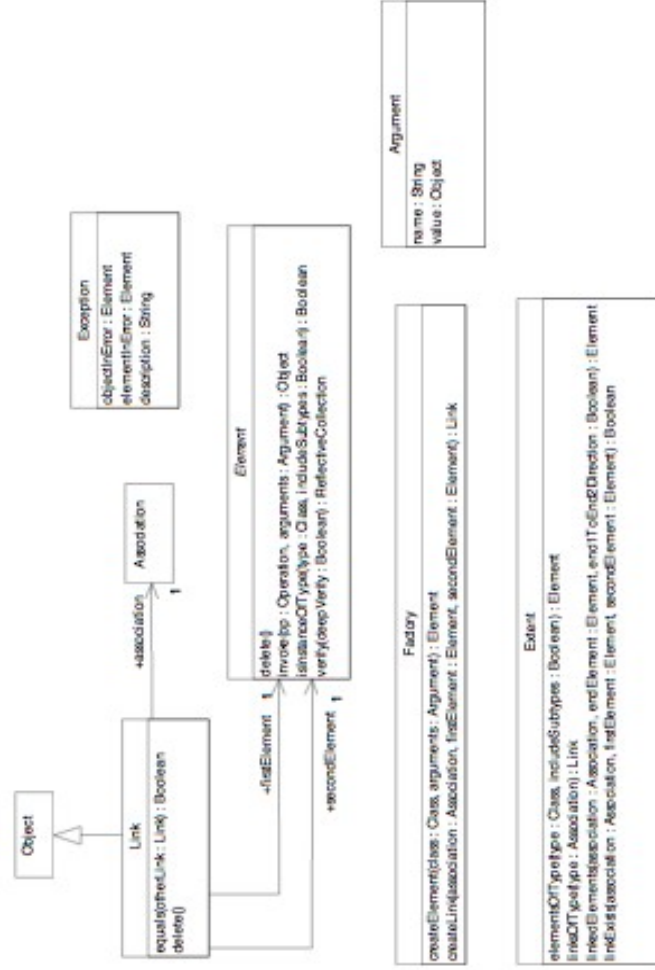


EMOF Reflection

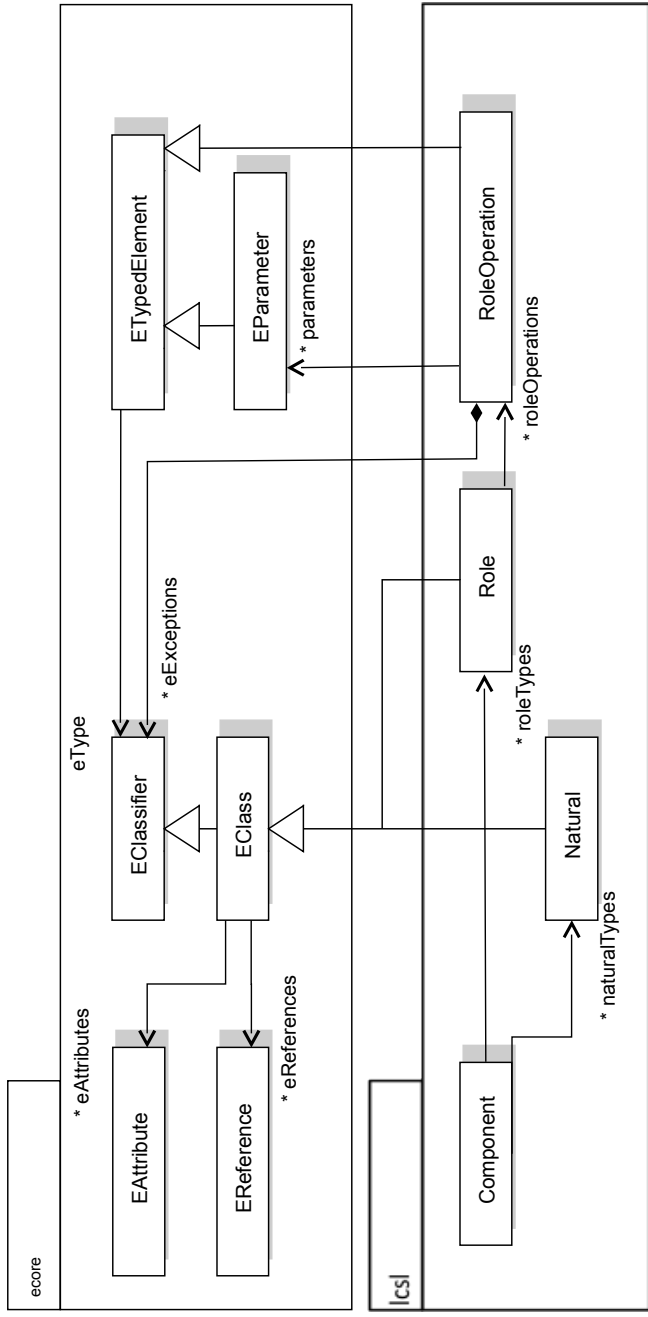
offers access to the metamodel
(getMetaClass())
provides a Factory, for creation
of Class from String



CMOF Reflection



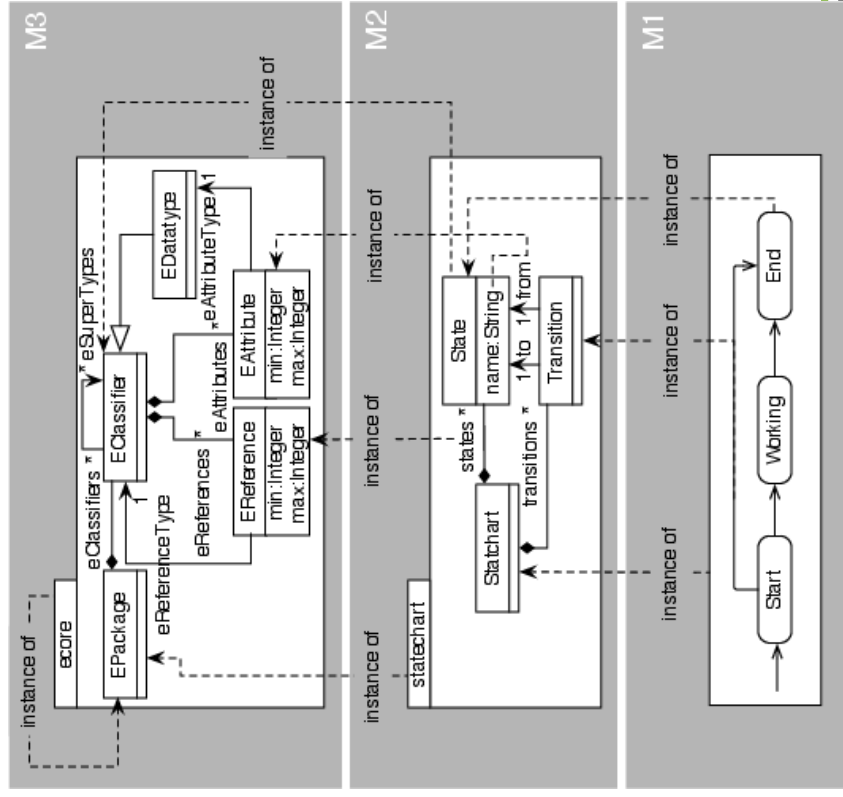
lcs1 is a domain-specific language for component-based modeling (C. Wende)



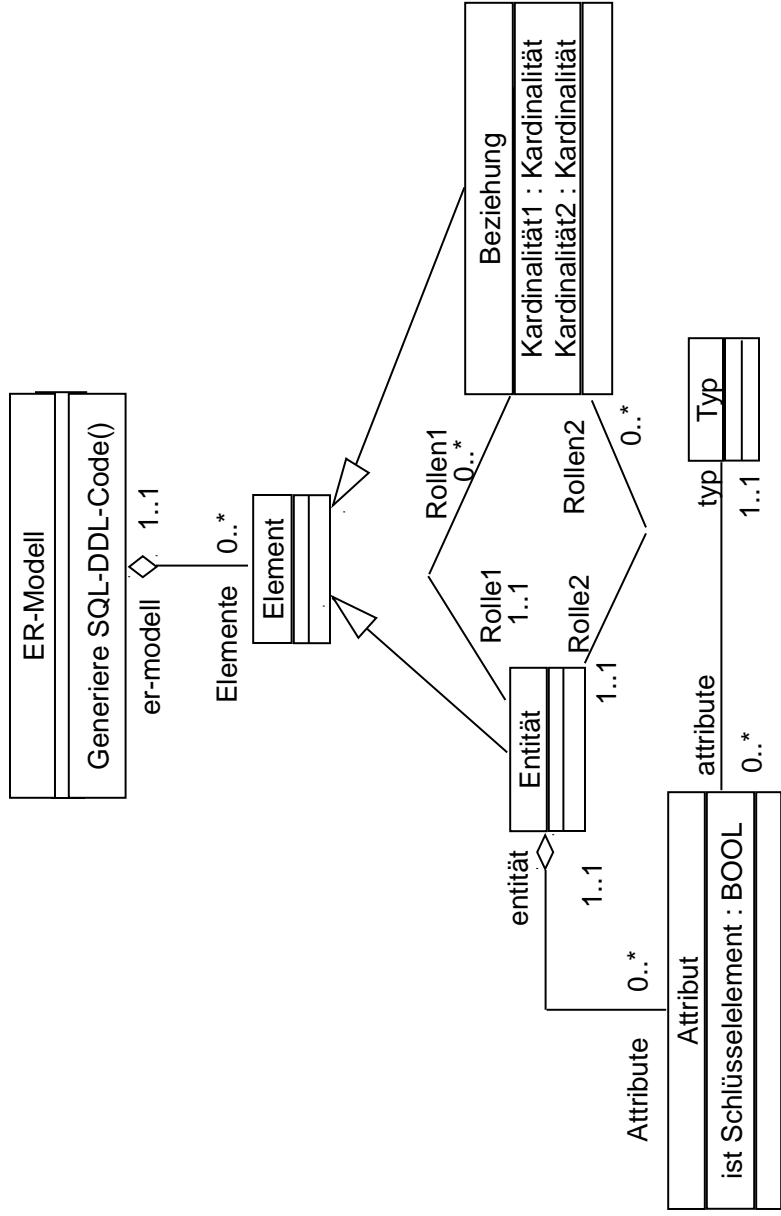
EMOF/Ecore based Metamodel of Statecharts

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.

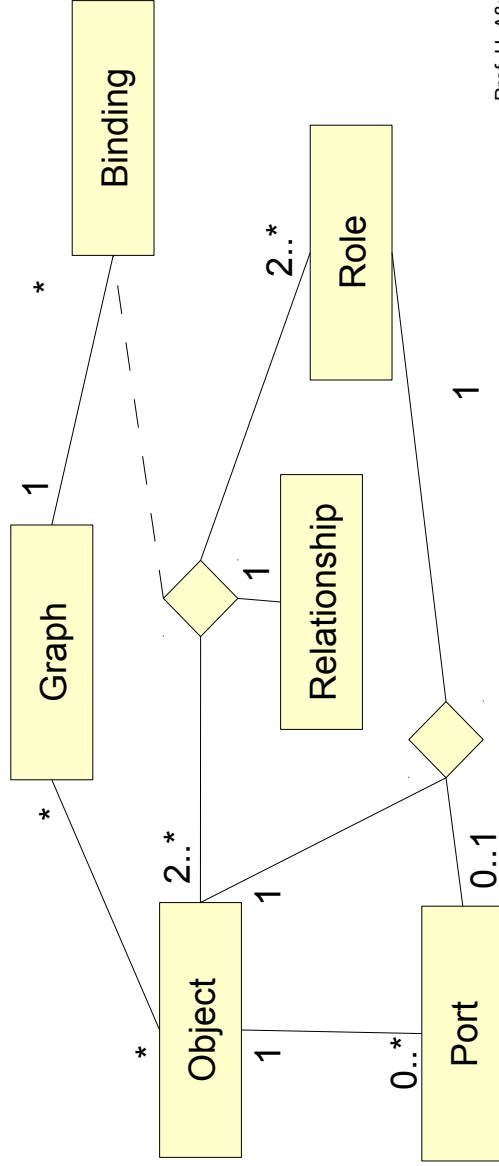


Meta-Modell von EntityRelationship-Diagrammen (MOF ohne Vererbung)



Graph Types in MetaEdit+

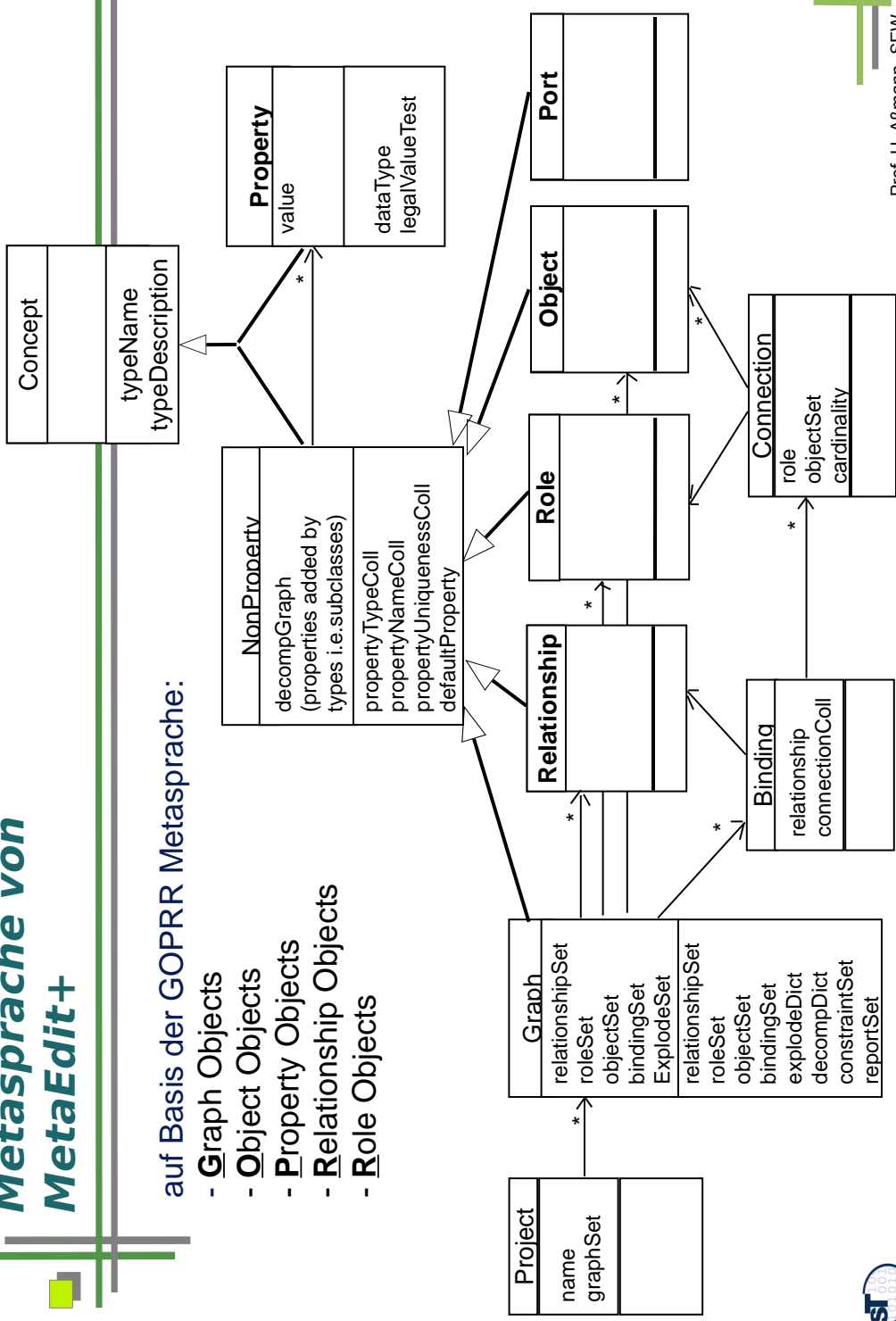
- ▶ [www.metacase.com]
- ▶ The tool MetaEdit+ uses a **graph schema (and metalanguage) GOPPR**:
 - Objects
 - Roles
 - Relationships
 - Allowed Bindings between all entities:
 - a binding consists of a relationship with roles and playing objects



Metasprache von MetaEdit+

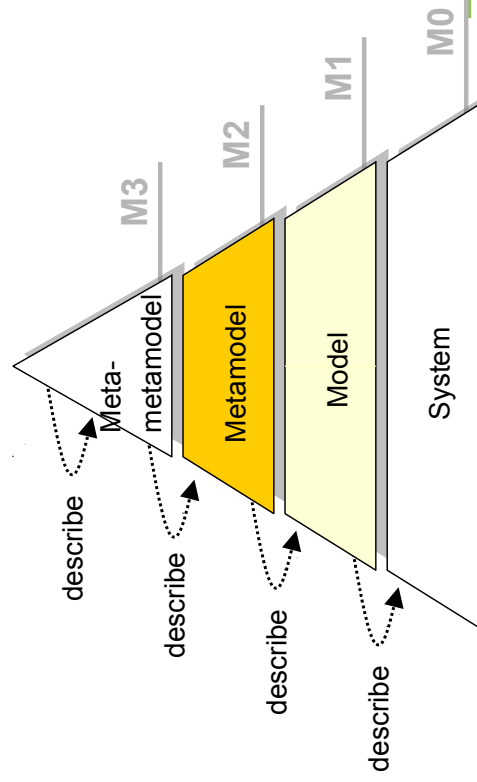
auf Basis der GOPRR Metasprache:

- Graph Objects
- Object Objects
- Property Objects
- Relationship Objects
- Role Objects



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



Metamodeling - Goals



Constraints

- Constraints for detailed definition of language
- Definition of erroneous states
- Rules to comply with special design guidelines



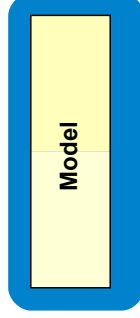
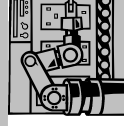
- (Meta-)Modeling of language constructs
- Definition of language structure
- Domain specific semantics

Metamodel

- Transformationen to repair erroneous models
- Conversion of incompatible models into design compliant models
- Automatic adaption to design guidelines

Abstract Syntax

Transformation



Nutzen der Metamodell-Architektur für domänenspez. Sprachen

- ▶ Mittels **Meta-Metamodellen (Metasprachen)** lassen sich definieren
 - beliebige Metamodelle konkreter Modellierungssprachen definieren.
 - Metamodelle von domänenspezifischen Sprachen (DSL)
- ▶ Auf Basis von Metaebenen können verschiedene Beschreibungssprachen ineinander **überführt** werden
 - Hierarchische Anordnung der einzelnen Modellebenen ermöglicht schrittweise Verfeinerung der semantischen Konzepte
 - Transformationsbrücken (z.B. mit EMFText, oder XMI)
- ▶ 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

Metamodelle für CASE basieren auf textuellen oder graphischen **Beschreibungen einer Methode** oder einer **Notation**, aus deren Interpretation, Compiler 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>

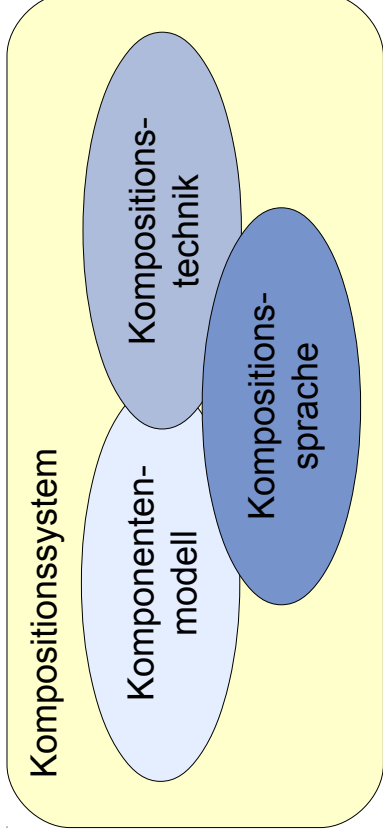
11.3 Modell- und Metamodell-Komposition

Einfaches Kompositionssystem für Modelle

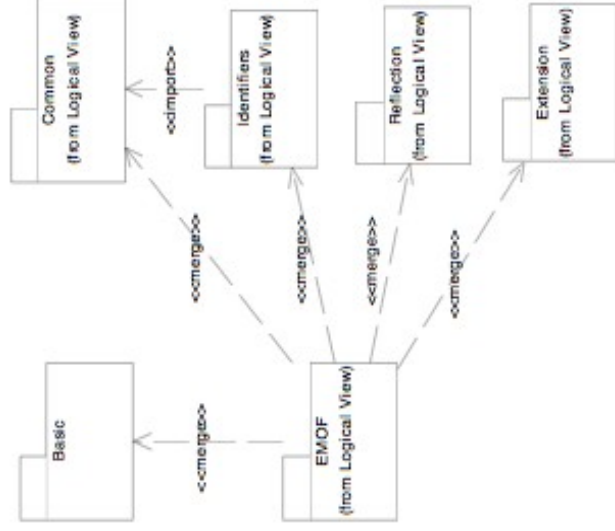
- ▶ Modelle und Metamodelle können in **Pakete** eingeteilt werden.
- ▶ Pakete sind Module, ein einfaches **Komponentenmodell** (siehe CBSE)
- ▶ Kompositionstechnik mit Kompositionsooperatoren auf Paketen (sehr einfache **Modellalgebra**):

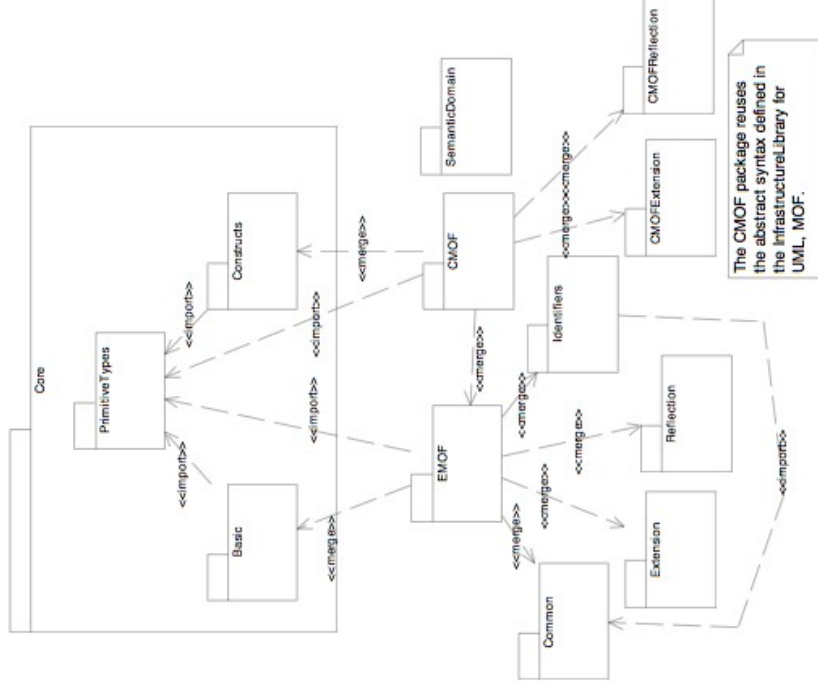
- use (import)
- merge (union)
- instance-of

→ heute werden Metamodelle aus Paketen komponiert



EMOF Classes Composition





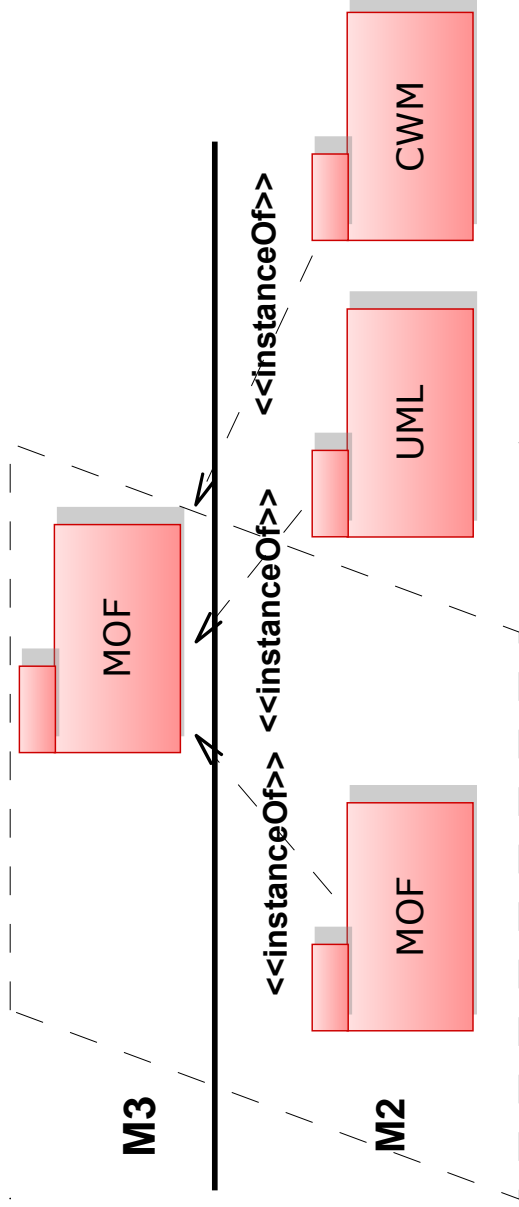
Promotion von Metamodellen

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

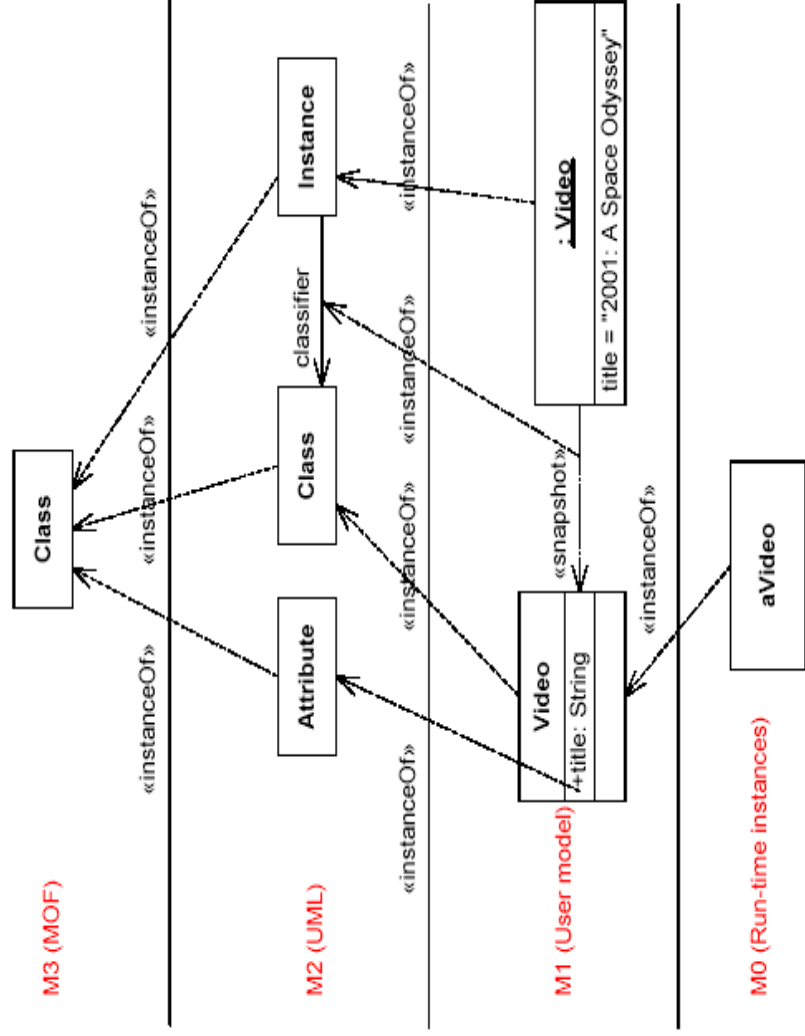
- ▶ MOF ist eigentlich eine einfache DDL (Datendefinitionssprache, Struktursprache) für Graphen
 - Man kann es auf M2 nutzen, um mit Paket-Merge neue Sprachen zu definieren, z.B. wie bei UML
 - Man kann es auf M3 nutzen, um Metamodelle als Instanzen zu bilden

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 MOF und die UML-Metahierarchie

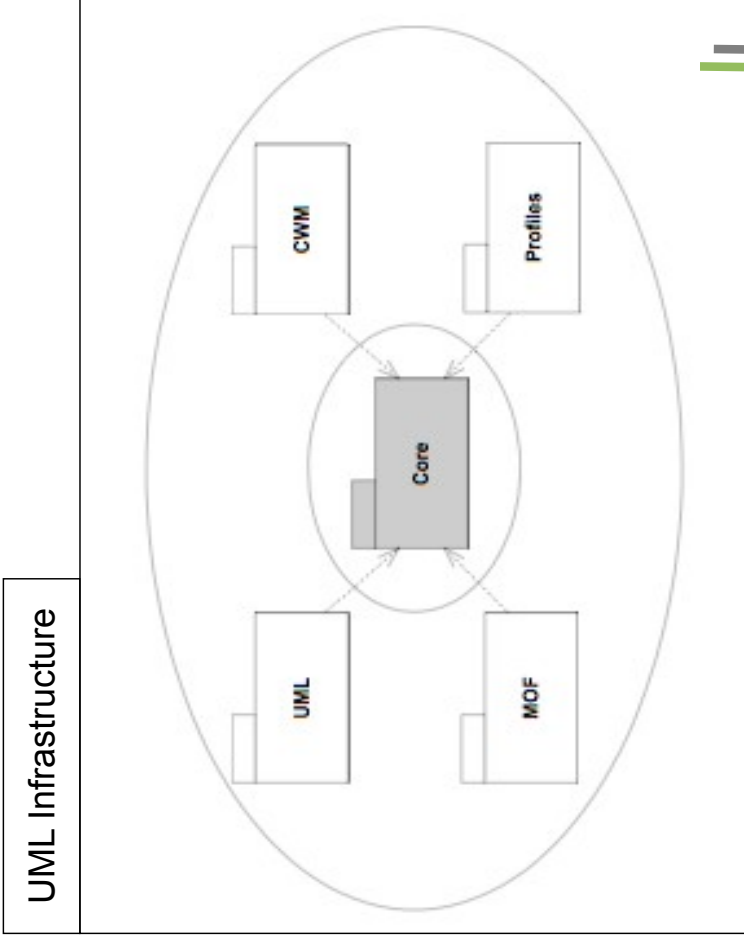
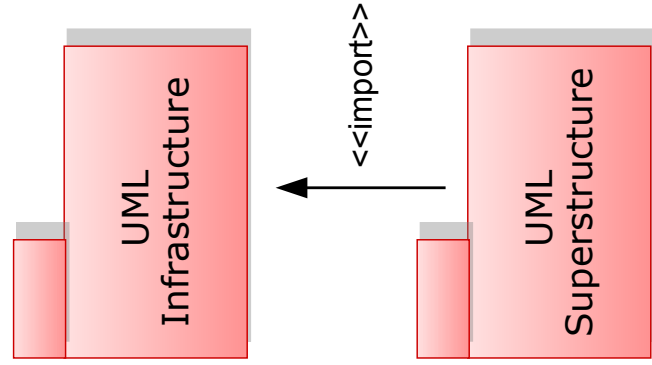


Bedeutung der UML-Metamodellierung für CASE

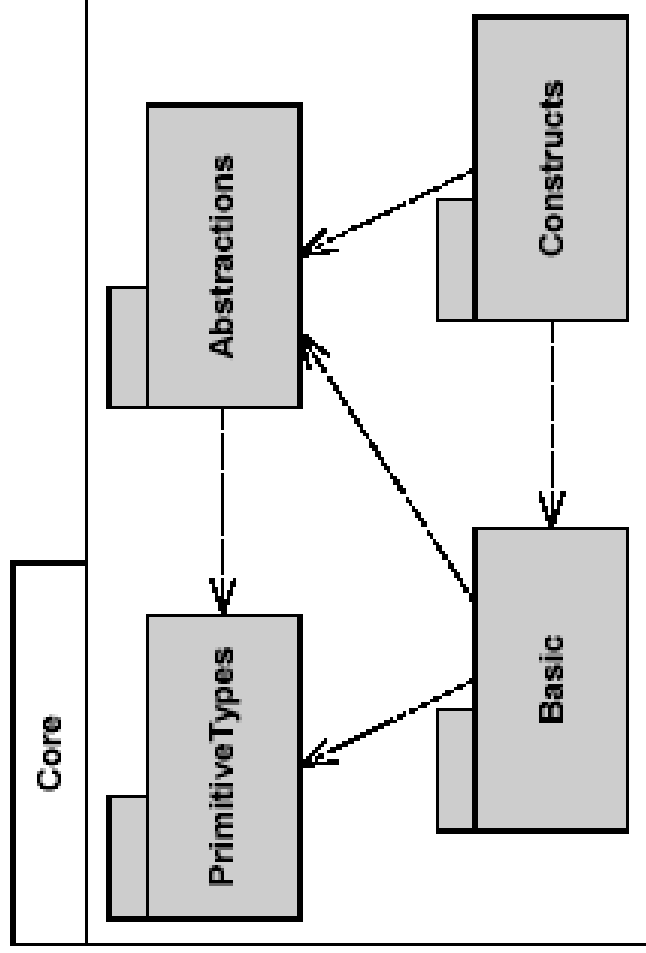
Das UML-Metamodell ist ein logisches (kein physikalisches oder Implementations-) Metamodell, mit

- ▶ aufgebaut aus **Paketen**, die komponentiert werden können
- ▶ aufgebaut auf die **CMOF**-Paketstruktur
- ▶ einheitliche **Struktur** (kontextsensitive Semantik) für alle darzustellenden Diagramminstanzen, wie Statecharts (SC), Message Sequence Charts (MSC), 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)



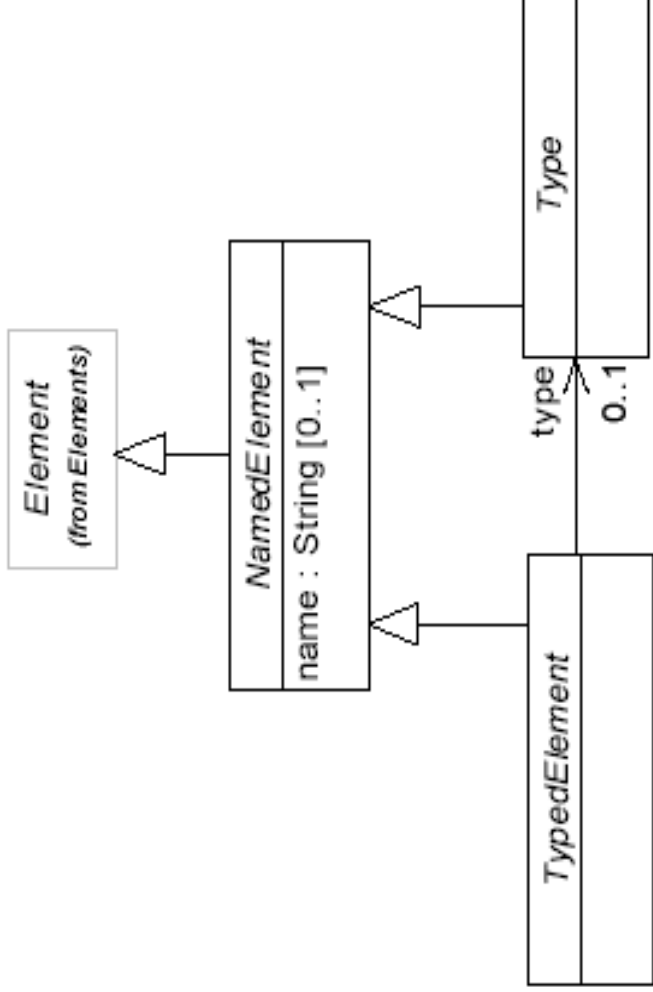
Basic: Grundkonstrukte für XMI

Abstractions: abstrakte Metaklassen

Constructs: Metaklassen für ooModellierung

Primitive Types: vordefiniert im Metamodell

Package Basic: Types from CMOF



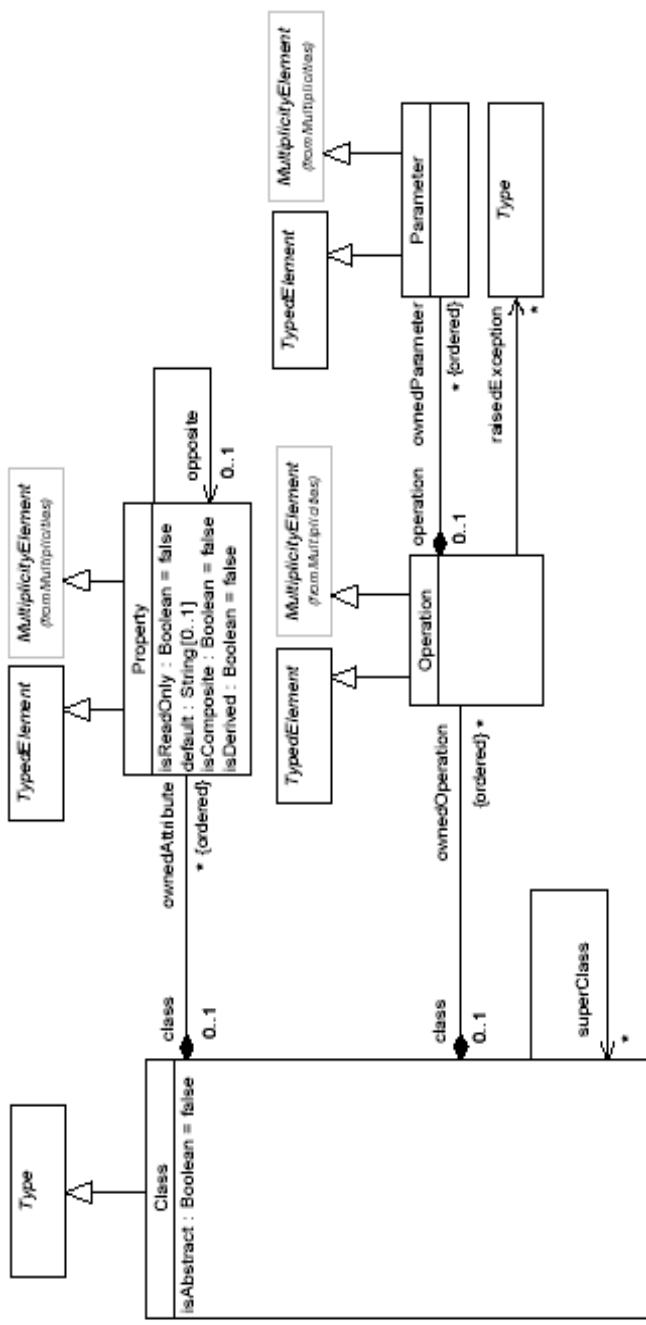
Die abstrakten Metaklassen dienen zum Benennen und zur Typdefinition von Elementen

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

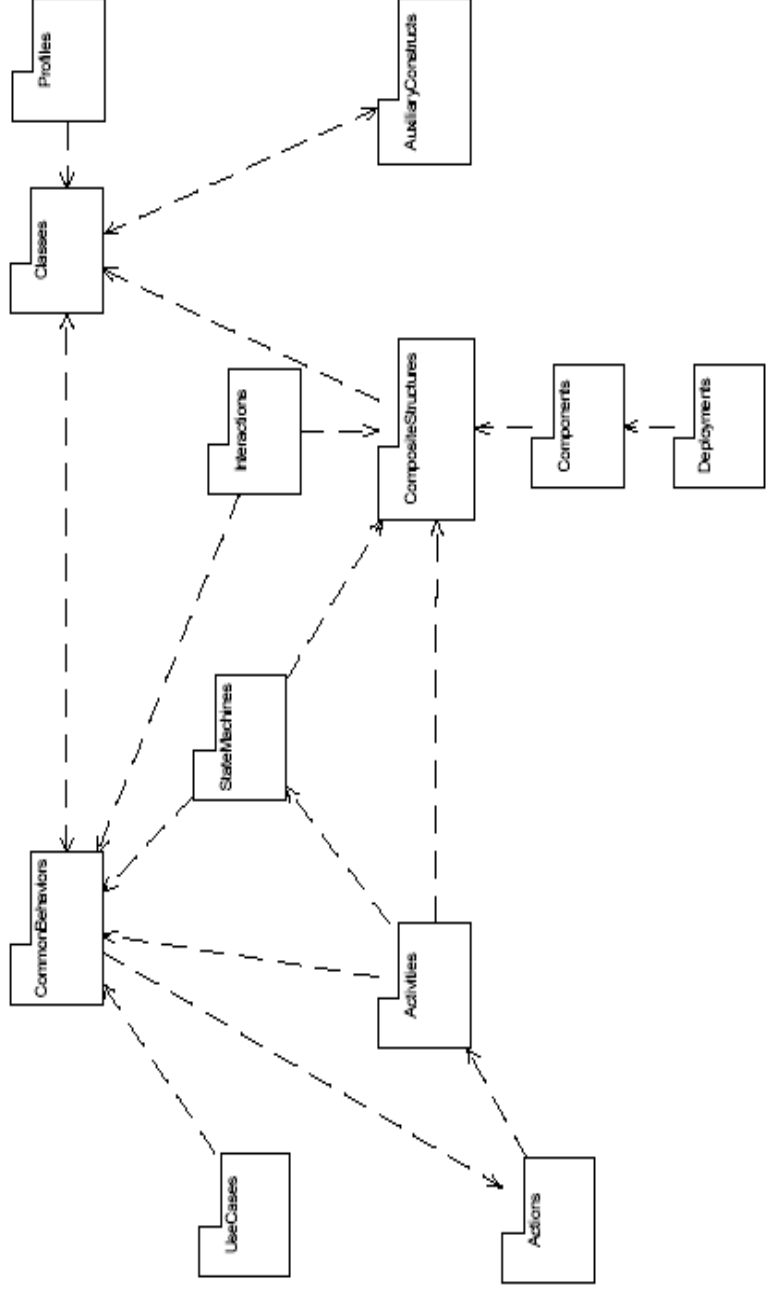


Quelle: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15



Package Composition UML 2.0 (M2)

Enthält alle Pakete zur systemstrukturierten und verhaltensorientierten Modellierung



11.5 Technikräume

..and 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
 - [Technological Spaces: an Initial Appraisal. Ivan Kurtev, Jean Bézin, 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 Bezin 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

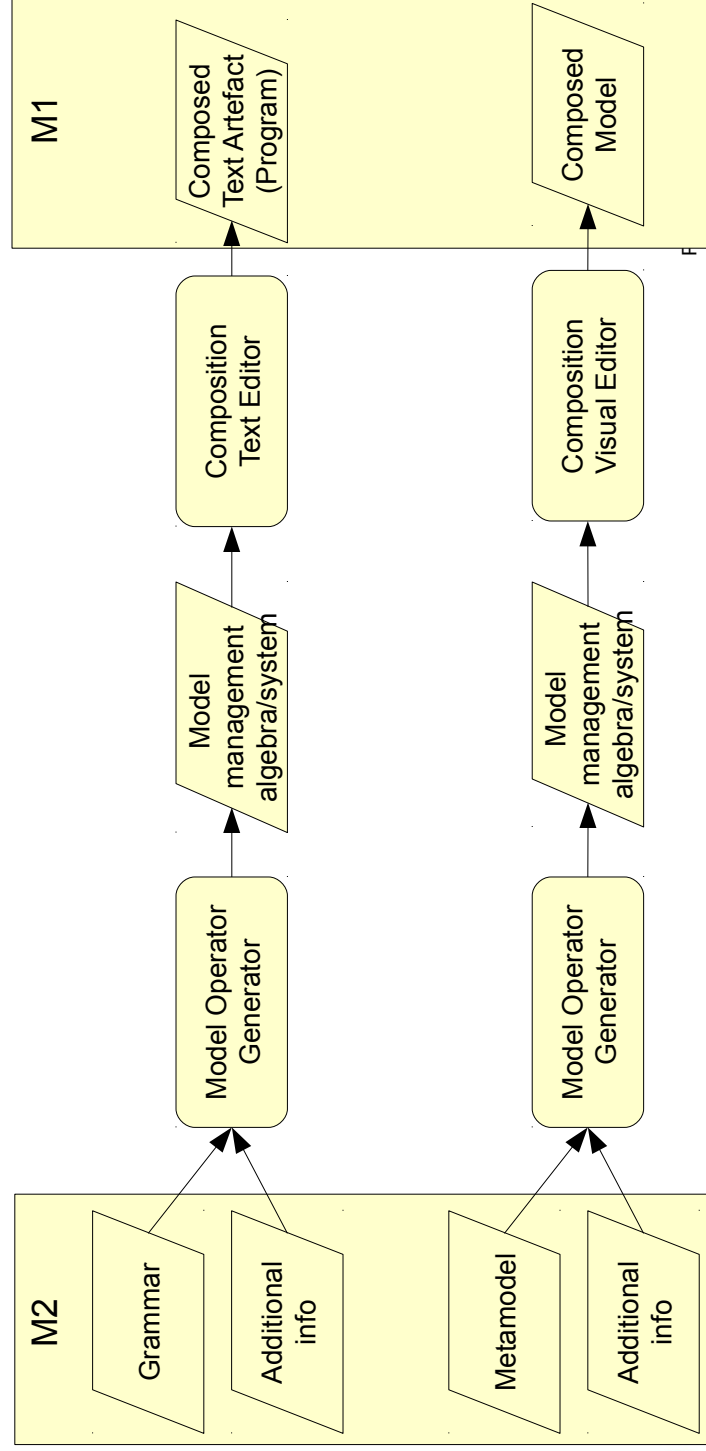
Werkzeuge nur dann kombinierbar, wenn sie im gleichen Technikraum leben

Technikräume über der Metahierarchie

	Gramm arware (Strings)	Text	Text- Tabell e	Table ware	Treeware (Bäume)	Graphw are/Mo delware	Ontology- ware
M3	Strings	Text	Text- Tabell e	Relation ale Algebra	XML	MOF	OWL-Ware
	EBNF	EBNF		CWM (common warehouse model)	XSD	MOF	MetaEdit +
							GOPPR
M2	Grammati k einer Sprache	Grammat ik mit Zeilenre nnern	csv- header	Relation ales Schema	XML Schema- beschreib ung, z.B. xhtml	UML- CD, -SC, OCL	HTML XML MOF UML DSL
							UML, many others
							ERD
M1	String, Program m	Text in Zeilen	csv Datei	Relation en	XML- Dokumente	Klassen, Program me	Klassen, Program me
							CDIF- Sprache
							Eclipse
M0	Objekte				dynamische Semantik im Browser	Objektne tze	A-Box (RDF- Graphen)
							RDFS OWL

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



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

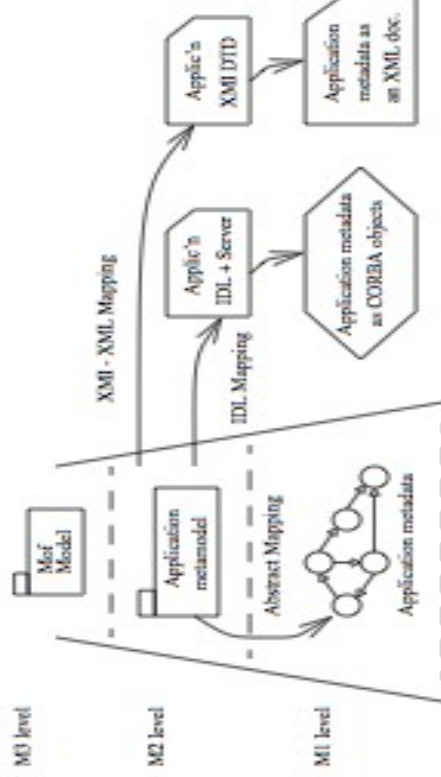


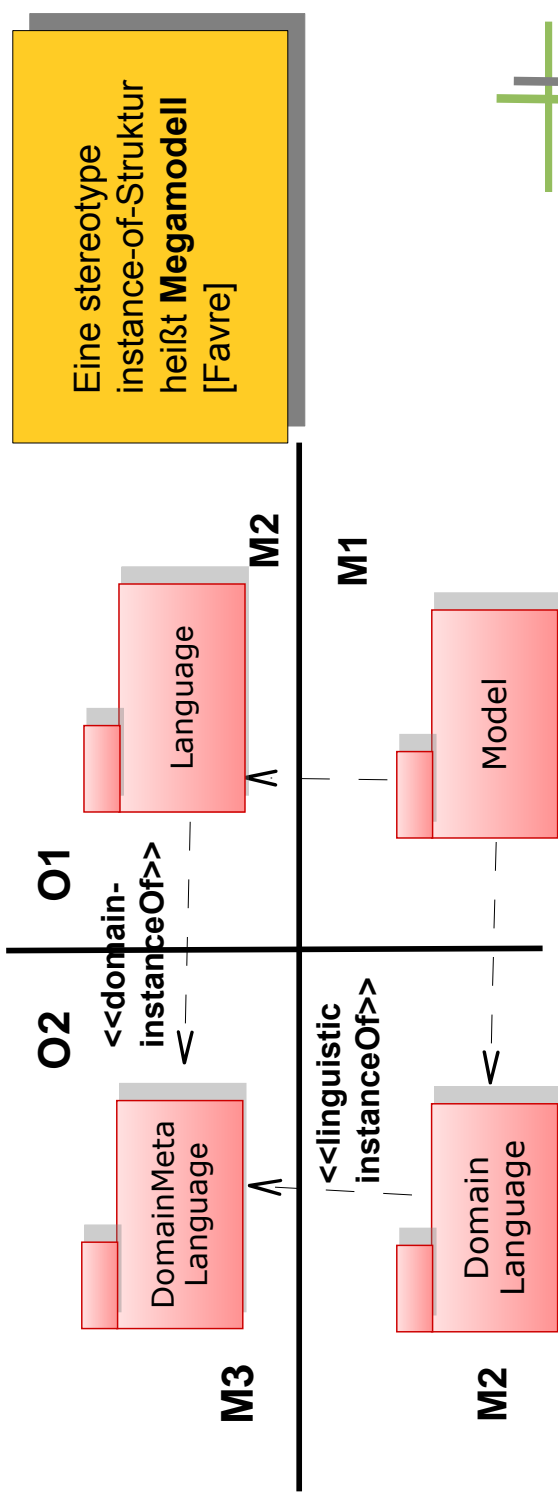
Figure 2-6 The function of MOF Technology Mappings

11.6 Megamodelle



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]



The End