Modellierung und Metamodellierung

Seminar Model-Driven Software Development

WS09/10, 13.11.2009, S 312

Heiko Kern

Betriebliche Informationssysteme

{kern}@informatik.uni-leipzig.de

Gliederung

- Model-Driven Software Development
- Modellierung
- Metamodellierung

Model-Driven Software Development

Herausforderungen in der Software-Entwicklung

- **Beherrschung zunehmender Komplexität**
  - Technische Komplexität, Funktionale Komplexität, Entwicklungskomplexität
  - Technologiekomplexität verdeckt Fachlogik
- **Erhöhung der Software-Qualität**
  - Funktionalität, Benutzbarkeit, Zuverlässigkeit, Änderbarkeit
- **Senken von Kosten**
  - Entwicklungszeit
  - Kosten im gesamten Lebenszyklus
- **...**

Was ist MDSD?

- **Ähnliche Begriffe**
  - Model-Driven Engineering (MDE)
  - Model-Driven Development (MDD)
  - Model-Driven Software Engineering (MDSE)
  - Model Driven Architecture (MDA)
  - Model-Based Software Development

- **… consequent use of formal models as input/output of computer-based tools implementing precise operations**

- **MDE is a discipline in software engineering that relies on models as first class entities and that aims to develop, maintain and evolve software by performing Model transformations.**

- **Perhaps the closest related area to generative software development is model-driven development (MDD), which aims at capturing every important aspect of a software system through appropriate models. A model is an abstract representation of a system and the portion of the world that interacts with it. Models allow answering questions about the software system and its world portion that are of interest to the stakeholders.**

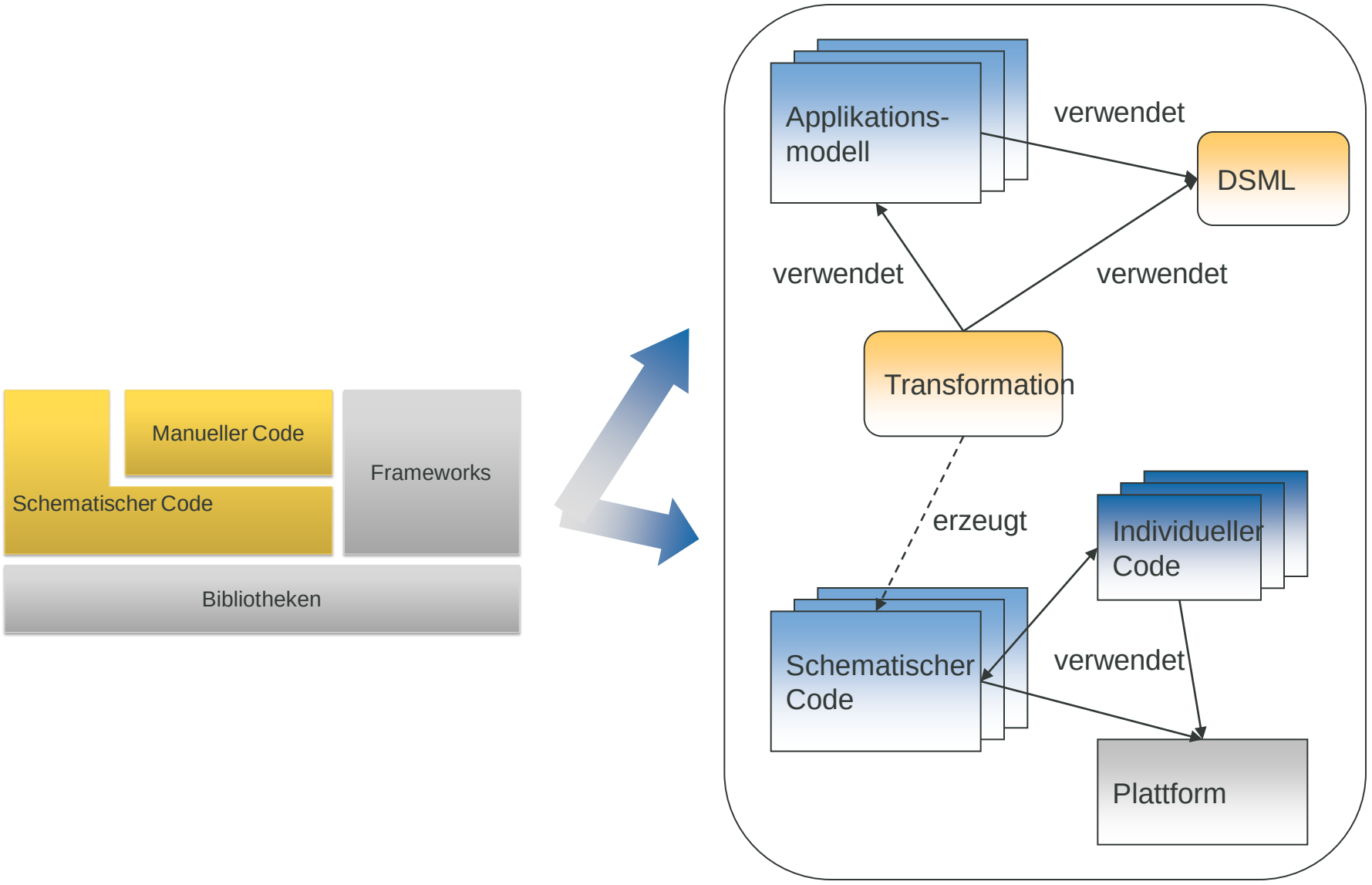
Was ist MDSD?

- **Das MSDS-Paradigma rückt im Wesentlichen zwei Kernideen in den Vordergrund**

- **(1) Modelle sind zentrales Artefakt im SW-Prozess**
  - Beschreiben die zu erstellende Software
  - Modelle sollen vorwiegend eine fachliche Semantik besitzen
  - Abstraktion gegenüber dem ausführbaren Code

- **(2) Code-Generierung und Modelltransformationen**
  - Aus Softwaremodellen wird durch Transformation ablauffähiger Code erzeugt
  - Abbildung von abstrakten Modellen auf ausführbaren Code

Was ist MDSD am Beispiel?



Ziele von MDSD

- **MDSD bietet Lösungen für die Herausforderungen in der Software-Entwicklung (siehe Folie 3)**
- **Handhabbarkeit von Komplexität**
  - Abstraktion und Strukturierung bzw. Modularisierung durch Modelle
- **Steigerung der Entwicklungsgeschwindigkeit**
  - Automatisierung im Entwicklungsprozess durch Code-Generierung und Modelltransformationen
- **Steigerung der Softwarequalität**
  - Verwendung von formal-definierten Sprachen und automatisierten Transformationen
  - Wiederverwendung durch Software-Produktionsstraßen
- **Möglichkeit zur Etablierung einer Software-Produktionsstraße (Architekturen, Modellierungssprachen, Transformationen)**
- **Verbesserung der Wartbarkeit**
  - Pflege querschnittlich verteilter Implementierungsaspekte an zentraler Stelle in Transformationen

Modellierung

Modell: Definition (1)

- **Allgemein im Duden**
  - Muster, Vorbild
  - Entwurf od. Nachbildung in kleinerem Maßstab (z.B. eines Bauwerkes)
  - vereinfachte Darstellung der Funktion eines Gegenstandes od. des Ablaufs eines Sachverhalts, die eine Untersuchung erleichtert od. erst möglich macht

- **A model is an abstraction of a physical system, with a certain purpose.**
- **A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system**
- **A model is a set of statements about some system under study (SUS).**

Modell: Definition (2)

- **In Stachowiaks „Allgemeinen Modelltheorie“ werden drei Merkmale hervorgehoben, die den Begriff in seiner allgemeinen Bedeutung näher definieren:**
  - Abbildungsmerkmal
  - Verkürzungsmerkmal
  - Pragmatisches Merkmal

Modell: Merkmale im Detail

- **Abbildungsmerkmal**
  - Modelle sind Abbilder oder Vorbilder eines vorhanden oder zu schaffenden Gebildes (Originals)
  - Zu jedem Modell gehört eine Abbildung, welche die Individuen und Attribute des Originals auf diejenigen des Modells abbildet
  - Das Modell kann selbst wieder ein Original sein
  - Es kann verschiedene Modelle des selben Originals geben

- **Verkürzungsmerkmal**
  - Modelle erfassen meistens nicht alle Individuen und Attribute des Originals
  - Ein Modell abstrahiert vom Original
  - Es wird nur das modelliert, was den Modellschaffenden relevant/wichtig/nützlich/notwendig erscheint

- **Pragmatisches Merkmal**
  - Jedes Modell ist für geschaffen für
    - **einen spezifischen Zeitraum**
    - **einem Verwendungszweck**
    - **Bestimmte erkennende und/oder handelnde, modellbenutzende Subjekte**

Modell: Beispiel

repräsentiert

repräsentiert

repräsentiert

System

repräsentiert

Modell

Wozu Modelle?

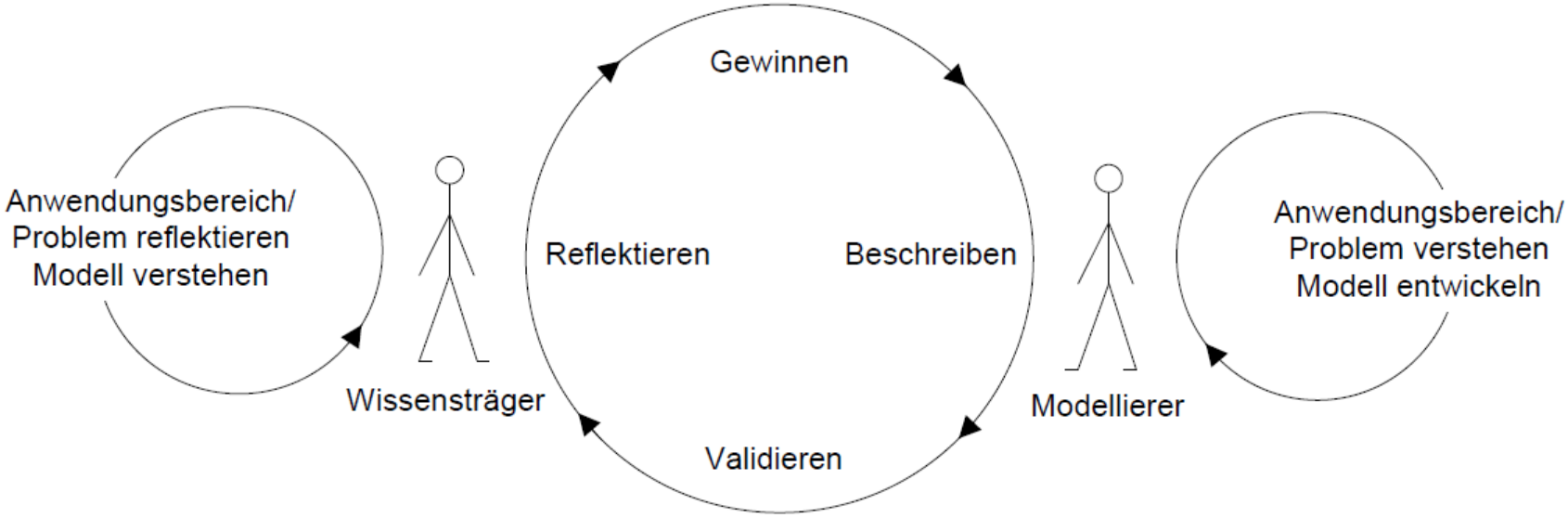
- **Verstehen eines Gebildes**
- **Kommunizieren über ein Gebilde**
- **Gedankliches Hilfsmittel zum Gestalten, Bewerten oder Kritisieren eines geplanten Gebildes oder von Varianten davon**
- **Spezifikation von Anforderungen an ein geplantes Gebilde**
- **Durchführung von Experimenten, die am Original nicht durchgeführt werden können, sollen oder dürfen**
- **Aufstellen / Prüfen von Hypothesen über beobachtete oder postulierte Phänomene**

- **wenn das modellierte Original ...**
  - nicht beobachtbar ist
  - zu groß oder zu klein ist
  - zu komplex ist
  - nicht zur Verfügung steht
  - noch nicht existiert

- **wenn die Arbeit am Original ...**
  - zu gefährlich,
  - zu teuer,
  - verboten,
  - nicht möglich ist.

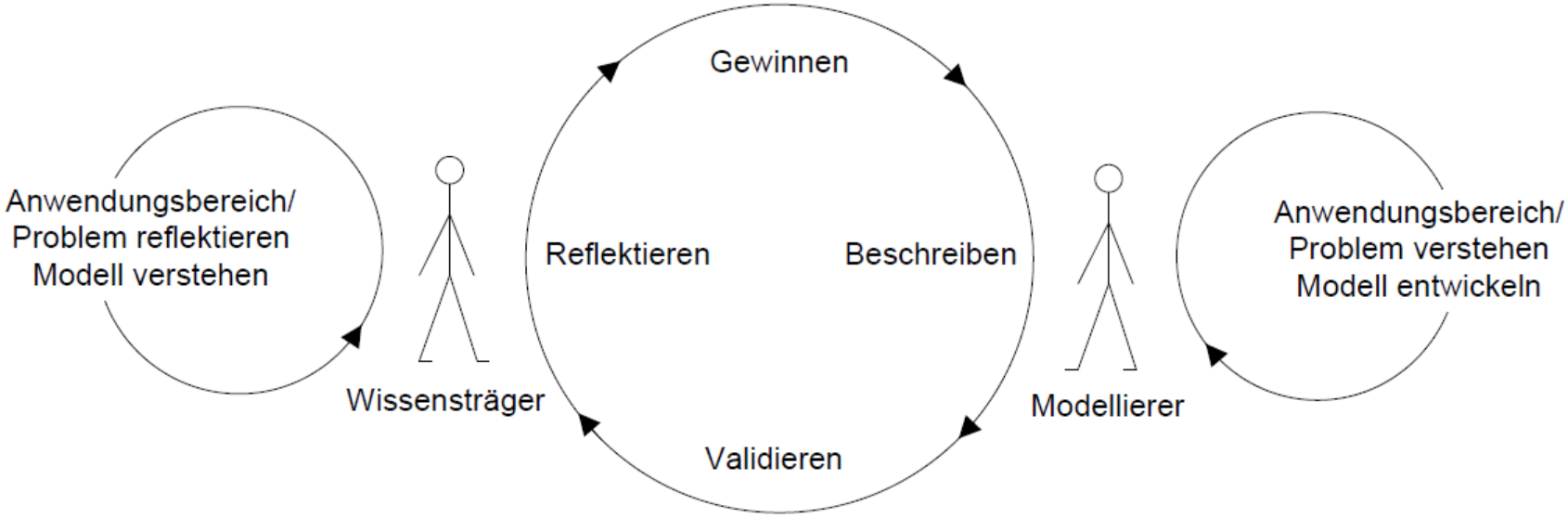
Prinzip der Modellbildung (1)

- **Prozess der Modellerstellung (meist iterativ)**
- **Zwei Rollen:**
  - Wissensträger: Rolle, welche das Wissen über den zu modellierenden Gegenstand bzw. Gegenstandsbereich hat
  - Modellierer: Rolle, welche ein Modell erstellt



Prinzip der Modellbildung (2)

- **Reflektieren**
  - Überlegen und verstehen, was modelliert werden soll (Pragmatik des Modells, abzubildende/wegzulassende Merkmale, Umfang)
- **Gewinnen**
  - Informationen über das Original und die Intentionen der Wissensträger gewinnen (Diskusionen, lesen, fragen, rückfragen, suchen, analysieren, ...).
- **Beschreiben**
  - Gewonnene Informationen verstehen, ordnen, strukturieren, bewerten, ... und mit geeigneten Mitteln beschreiben
- **Validieren**
  - Modelle (Zwischenergebnisse oder fertiges Modell) durch Wissensträger überprüfen lassen: Ist es das, was sie wollen und brauchen?



Abstraktion als Modellbildungskonzept (1)

- **Abstraktion**
  - Zentrales Mittel für das Erstellen und Verstehen von Modellen
  - Gedankliches Verfahren, das von den als unwesentlich erachteten Merkmalen absieht, um das Augenmerk auf die als wesentlich beurteilten Merkmale zu lenken.
  - Das Kriterium des Wesentlichen ist nach pragmatischen Gesichtspunkten festgelegt und variiert mit dem jeweiligen Erkenntnisinteresse.
  - Auswahl von möglichen Abstraktionskonzepten
    - **Klassifizierung**
    - **Generalisierung**
    - **Komposition**

Abstraktion als Modellbildungskonzept (2)

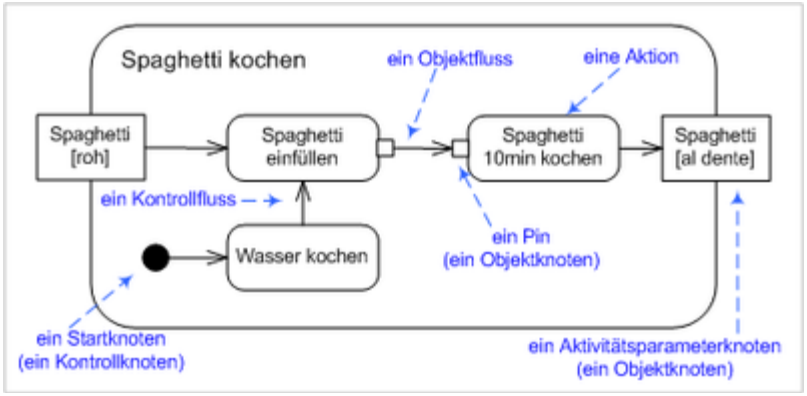
- **Klassifizierung**
  - Klassifizierung ist die systematische Einteilung oder Einordnung von ähnlichen Begriffen, Gegenständen, Erscheinungen u.a. in Klassen (Gruppen).
  - Extensional: Eine Klasse ist eine Zusammenfassung mehrerer Elemente zu einem Ganzen (der Klasse dieser Elemente).
  - Intensional: Ein Klasse beschreibt den Aufbau von zulässigen Elementen dieser Klasse.

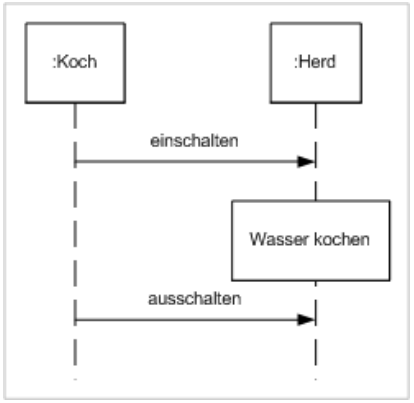
- **Generalisierung**
  - Ist in der Logik und Wissenschaftstheorie ein Verfahren, aus einer Allaussage durch Wahl eines generellen Subjektbegriffs eine neue Allaussage zu gewinnen.
  - „alle Menschen sind eigensinnig“ ist eine Generalisierung von „alle Kinder sind eigensinnig“.

- **Komposition**
  - Zusammenfassung einer Menge von Elementen (Einzelkomponenten) mit teilweise gemeinsamen Merkmalen zu einem neuen übergeordneten Element (einem Ganzen) mit neuen Merkmalen
  - Die übergeordneten Komponenten (das Ganze) ist ein in sich möglichst geschlossenes Teilmodell
  - Zusammenfassung über mehrere Stufen (Kompositionshierarchie oder Teil-Ganzes-Hierarchie)

Modellarten (1)

- **Unterschiedliche Sichten/Aspekte auf ein und das selbe System**
  - Prozesssicht, Funktionssicht, Datensicht, …
  - UML-Modellfamilien: Aktivitätsdiagramm, Sequenzdiagramm, Klassendiagramm

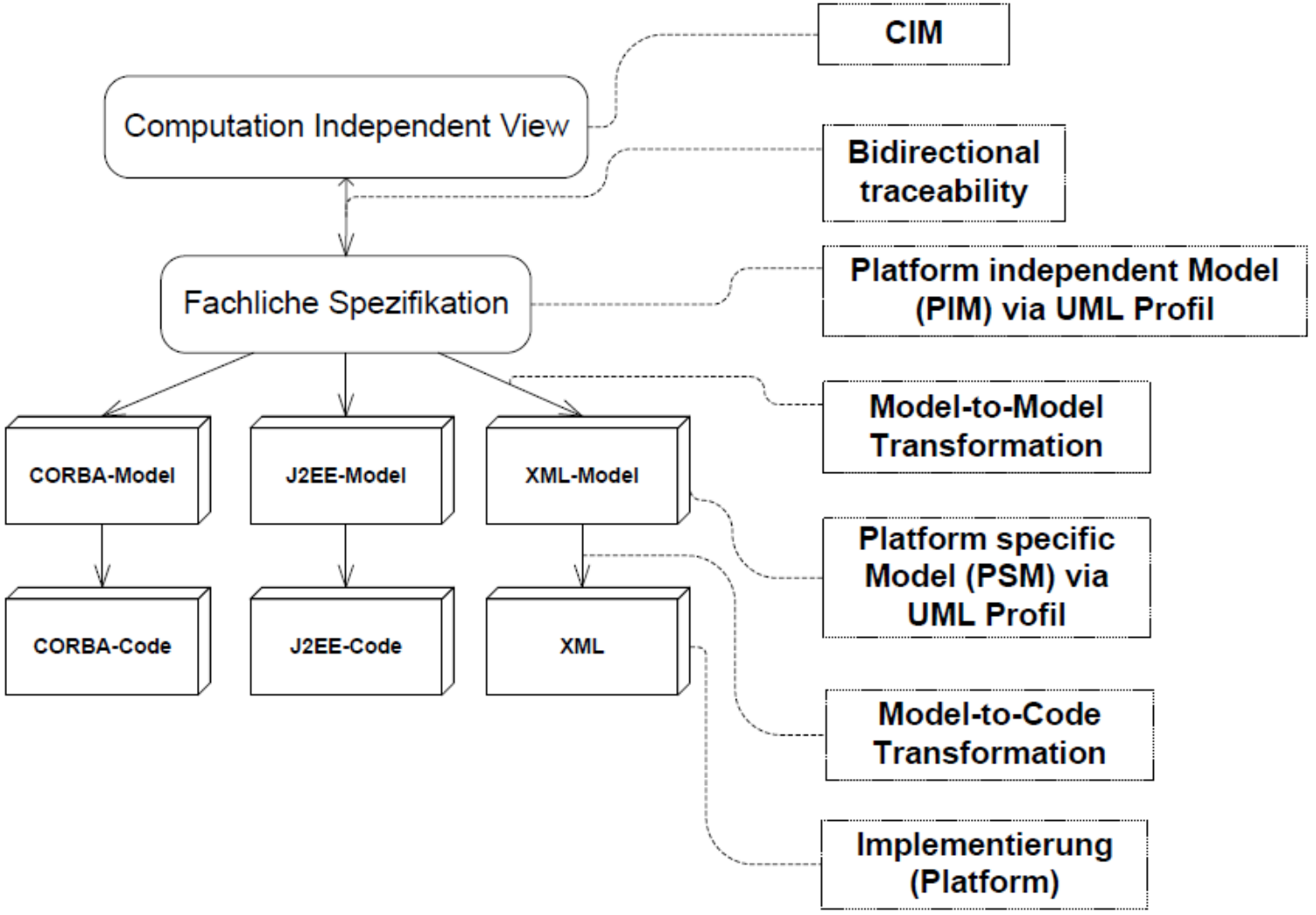




  - Architektur integrierter Informationssysteme (ARIS): Ereignisgesteuerte Prozessketten (EPK), Funktionszuordnungsdiagramm (FZD), Organigramm

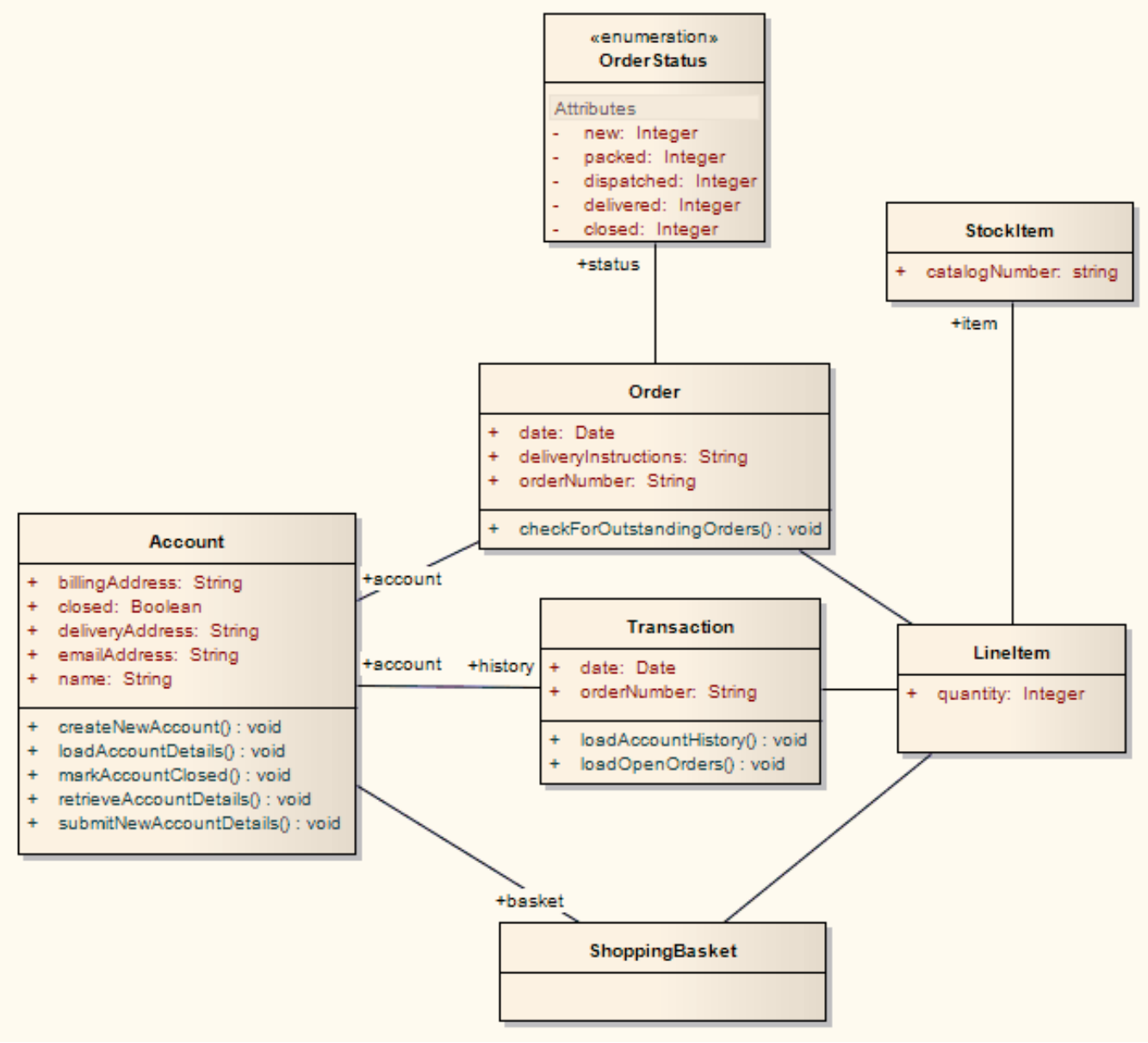
Modellarten (2)

- **Plattformabhängige Modelle (PIM) vs. plattformunabhängige Modelle (PSM)**



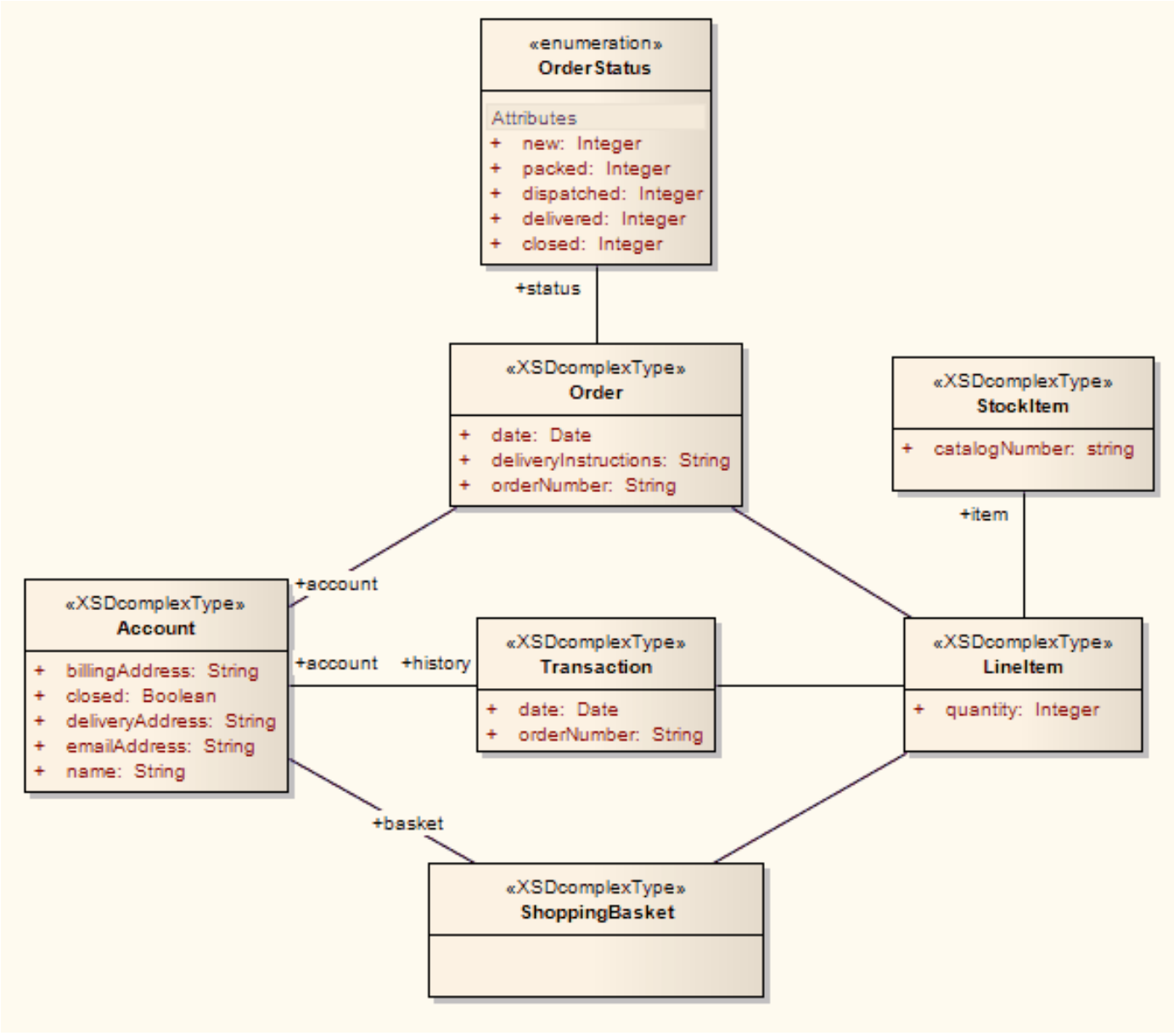
Modellarten (3)

- **PIM**



Modellarten (4)

- **PSM**



Metamodellierung

Modellierungssprache

- **Metamodellierung**
  - Die Erstellung einer Modellierungssprache im Kontext von MDSD wird als Metamodellierung bezeichnet.
  - Die „Modellierung“ einer Modellierungssprache wird zum Gegenstandsbereich.

- **Bestandteile**
  - Syntax (konkrete Syntax C und abstrakte Syntax A),
  - Semantik S
  - syntaktischen Abbildung MS : A -> C
  - semantischen Abbildung MC : A -> S

- **Semantik**
  - Die Semantik legt die Bedeutung von Symbolen bzw. der Kombination von Symbolen fest
  - Diese Bedeutung wird in einem begrenzten Bereich, der semantischen Domäne, definiert

- **Syntax: siehe nächste Folie**

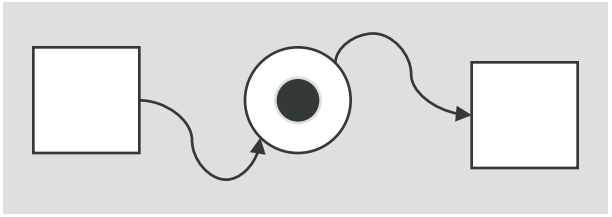
Konkrete Syntax (Textuell vs. Graphisch)

- **Konkrete Syntax**
  - Definiert die Symbole (Zeichen), welche in einer Modellierungssprache verwendet werden
  - Eine andere Bezeichnung ist Notation

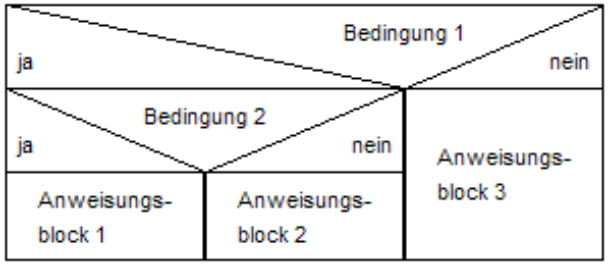
- **Textuelle Sprache**
  - Zeichen, die nach bestimmtem Mustern zu linearen Zeichenketten verknüpft werden

```
for (i=0; i<10; i++) {
    doSomething();
}
```

- **Graphische Sprache**
  - Linien, Pfeilen, geschlossenen Kurven (Kreis, Rechteck)
  - Bilden in der Regel einen Graph



- **Graphische Sprache mit Text-Elementen**

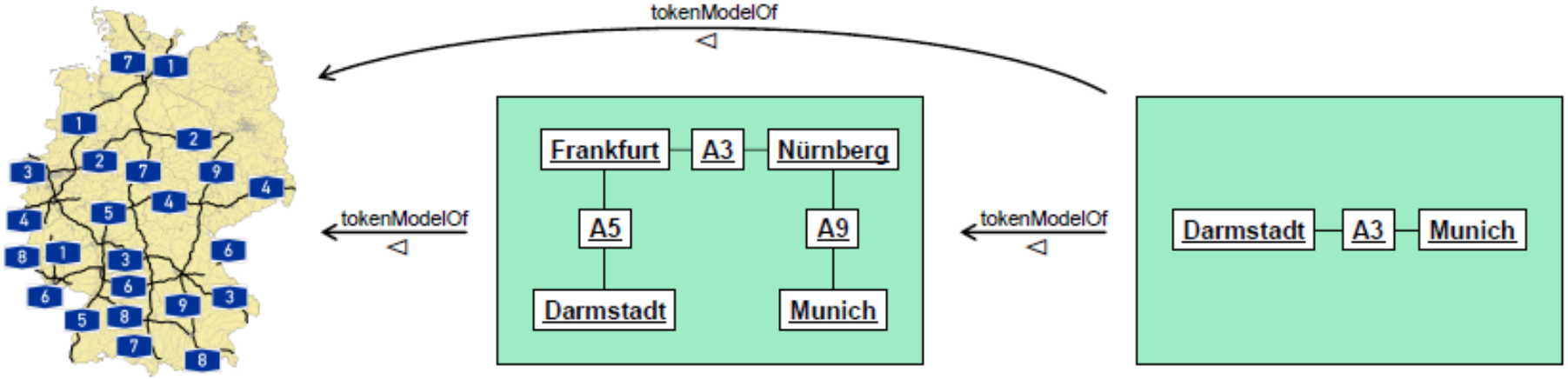


Abstrakte Syntax

- **Legt die zugrundeliegende Struktur der Notationselemente durch Konzepte, Beziehungen zwischen Konzepten und Integrationsbedingungen fest**
- **Abstrahiert von den konkreten Visualisierungen und erfasst nur die Modellierungskonzepte einer Sprache**
- **Definition der abstrakten Syntax**
  - Graphische Sprachen in der Regeln durch Metamodelle oder Graphgrammatiken
  - Textuelle Sprachen in der Regel durch Grammatiken

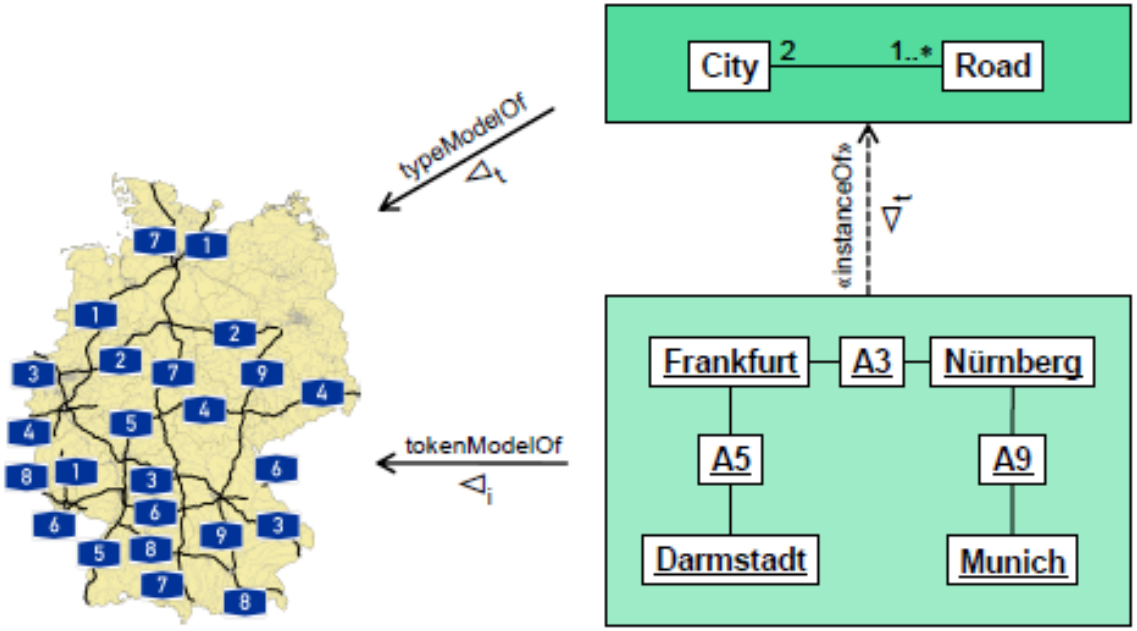
Instanz- und Typmodell (1)

- **Instanz- und Typmodell ist relativ zum Original zu betrachten**
- **Instanzmodell**
  - Auch als Token-Modell bezeichnet
  - Instanzmodelle repräsentieren Individuen sowie deren individuellen Merkmal in verkürzter Form
  - Beispiel: UML-Objektdiagramm
  - Neben Reduktion findet keine weitere Abstraktion statt
  - „Ist-Instanzmodell-von“-Relation ist transitiv



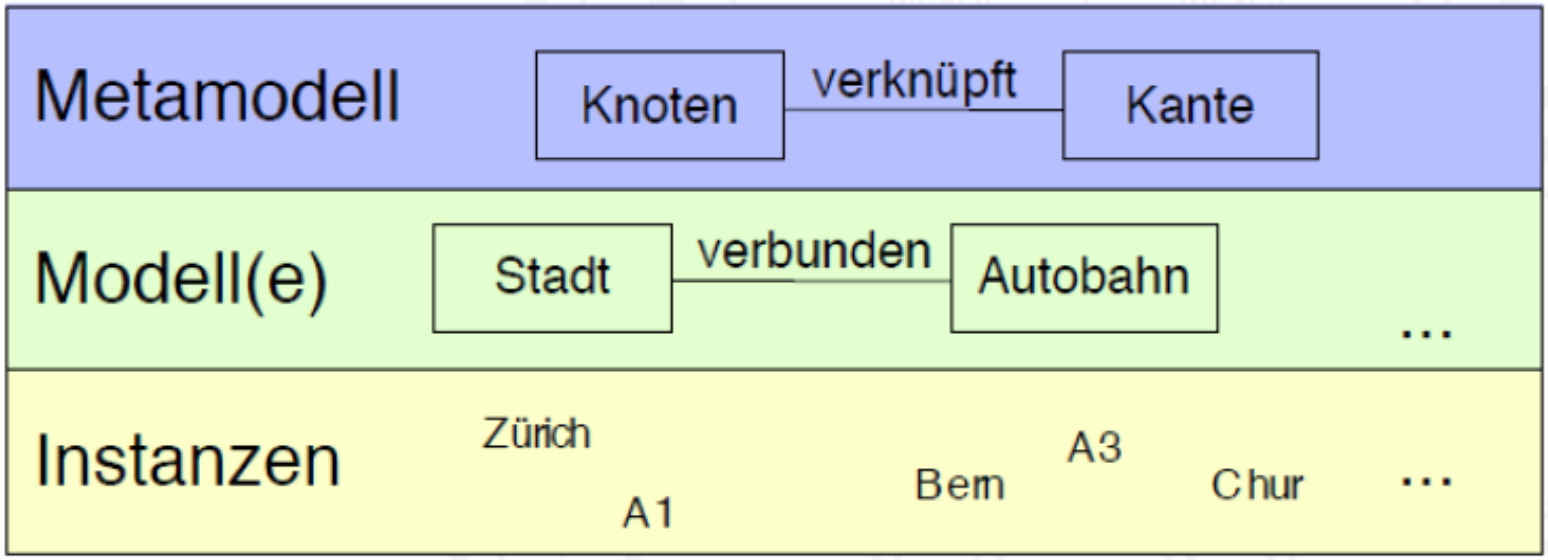
Instanz- und Typmodell (2)

- **Typmodell**
  - Typmodelle erfassen die universellen Merkmale der Individuen des Originals durch Klassifizierung
  - Mengen von Individuen werden auf Konzepte abgebildet
  - Beispiel: UML-Klassendiagramm
  - Typmodelle können auch Generalisierungen enthalten
  - „Ist-Typmodell-von“-Relation ist nicht transitiv



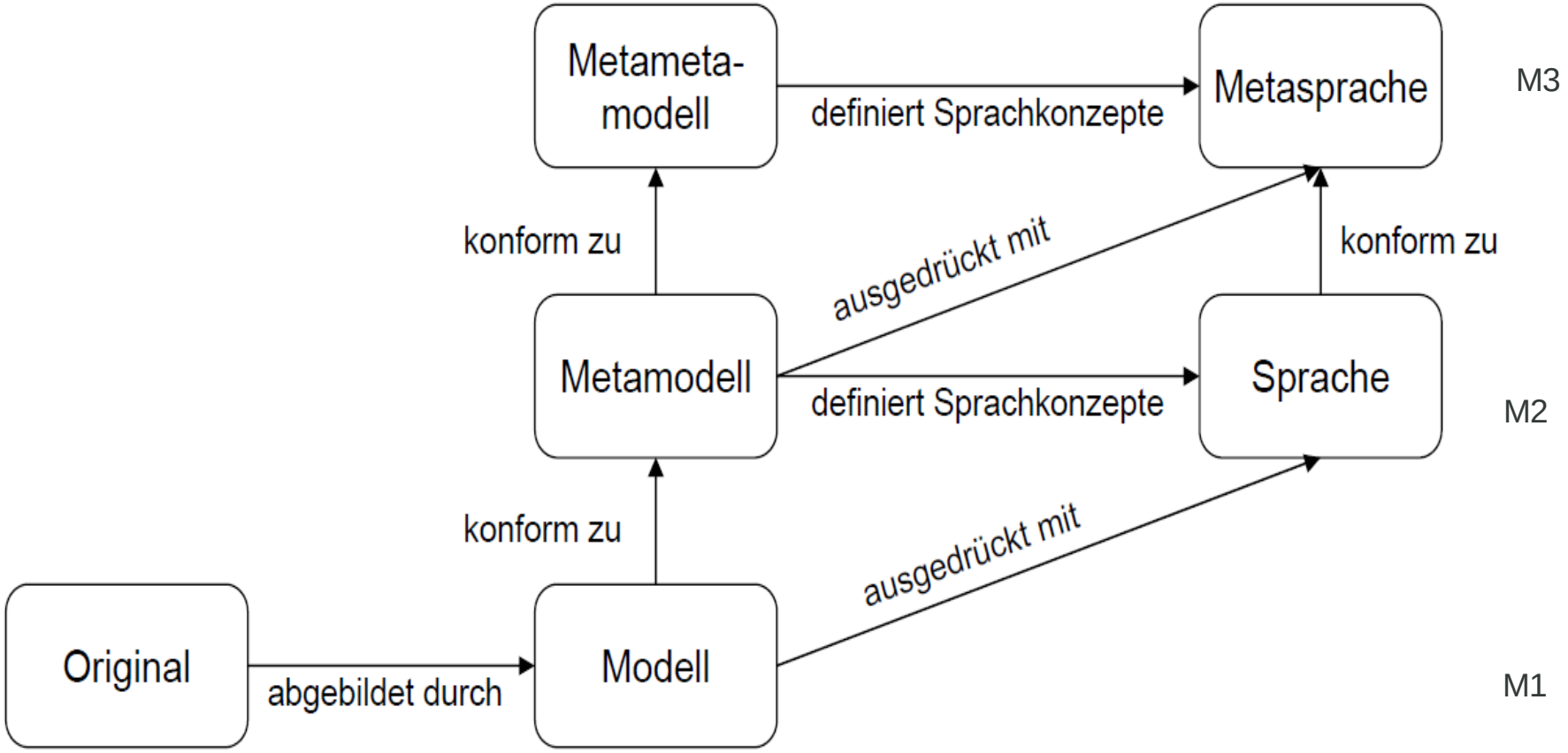
Metamodell

- **Ein durch mehrstufige Klassifizierung (nicht transitive Relation) entstandenes Modell heißt Metamodell**
- **Beschreibt die Struktur von Modellen**



- **Beispiel**
  - „Zurüch“ ist eine Instanz der Klasse „Stadt“
  - Die Klasse „Stadt“ ist eine Instanz der Klasse „Knoten“
  - „Zurüch“ ist keine Instanz der Klasse „Knoten“

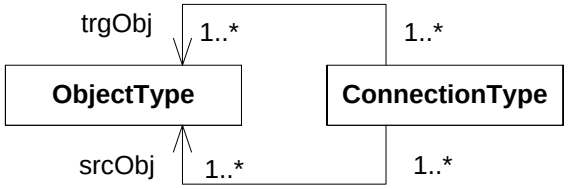
Modellhierarchie



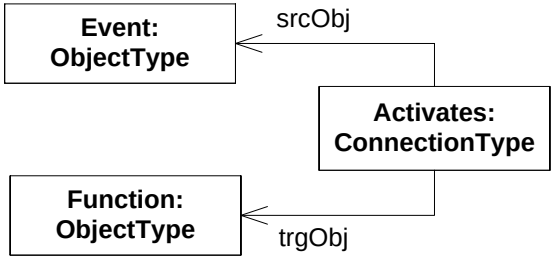
Modellhierarchie: Beispiele

ARIS

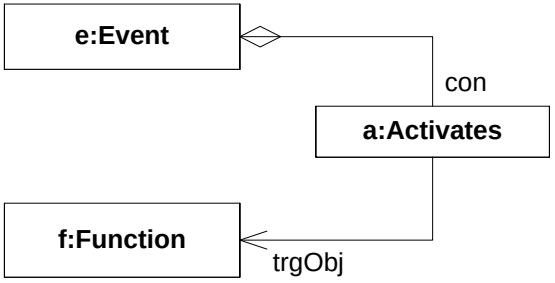
M3



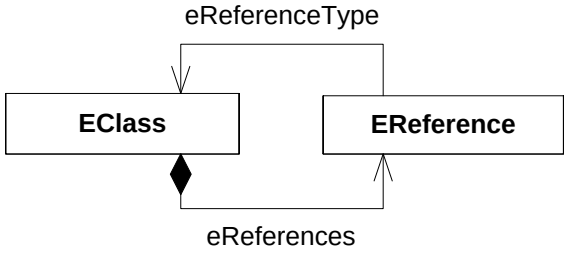
M2

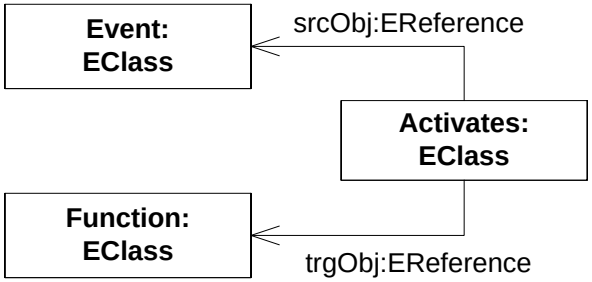


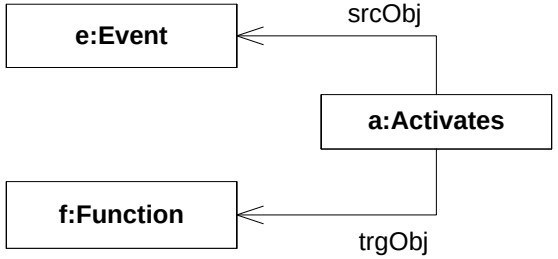
M1



Eclipse EMF

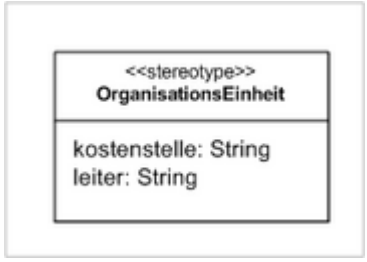




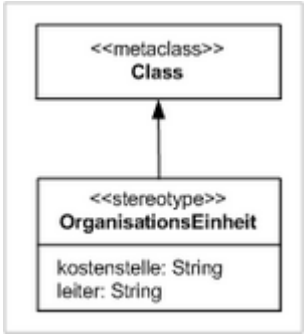


Metamodellierungsansätze

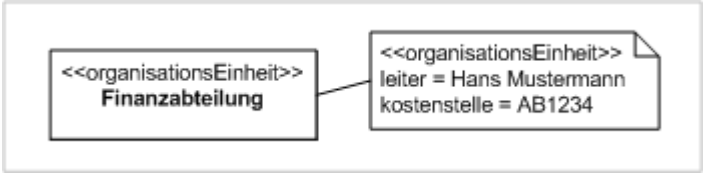
- **Leichtgewichtete Ansatz**
  - Beim leichtgewichtigen Ansatz wird ein Metamodell mit den Modellierungskonzepten des Metamodells erstellt bzw. erweitert.
  - Beispiel: UML-Stereotypen



Deklaration eines Stereotyps



Erweiterung von Metaklassen



Verwendung des Stereotyps

- **Schwergewichtete Ansatz:**
  - Beim schwergewichtigen Ansatz wird ein Metamodell mit den Modellierungskonzepten des Metametamodells erstellt.
  - Beispiel: MetaEdit+, MOF, Ecore

Metamodellierungskonzepte (1)

- **Objekttyp**
  - Klasse von eigenständigen Modellelementen
  - Andere Bezeichnung: Class, Meta-Class, Entity, Object

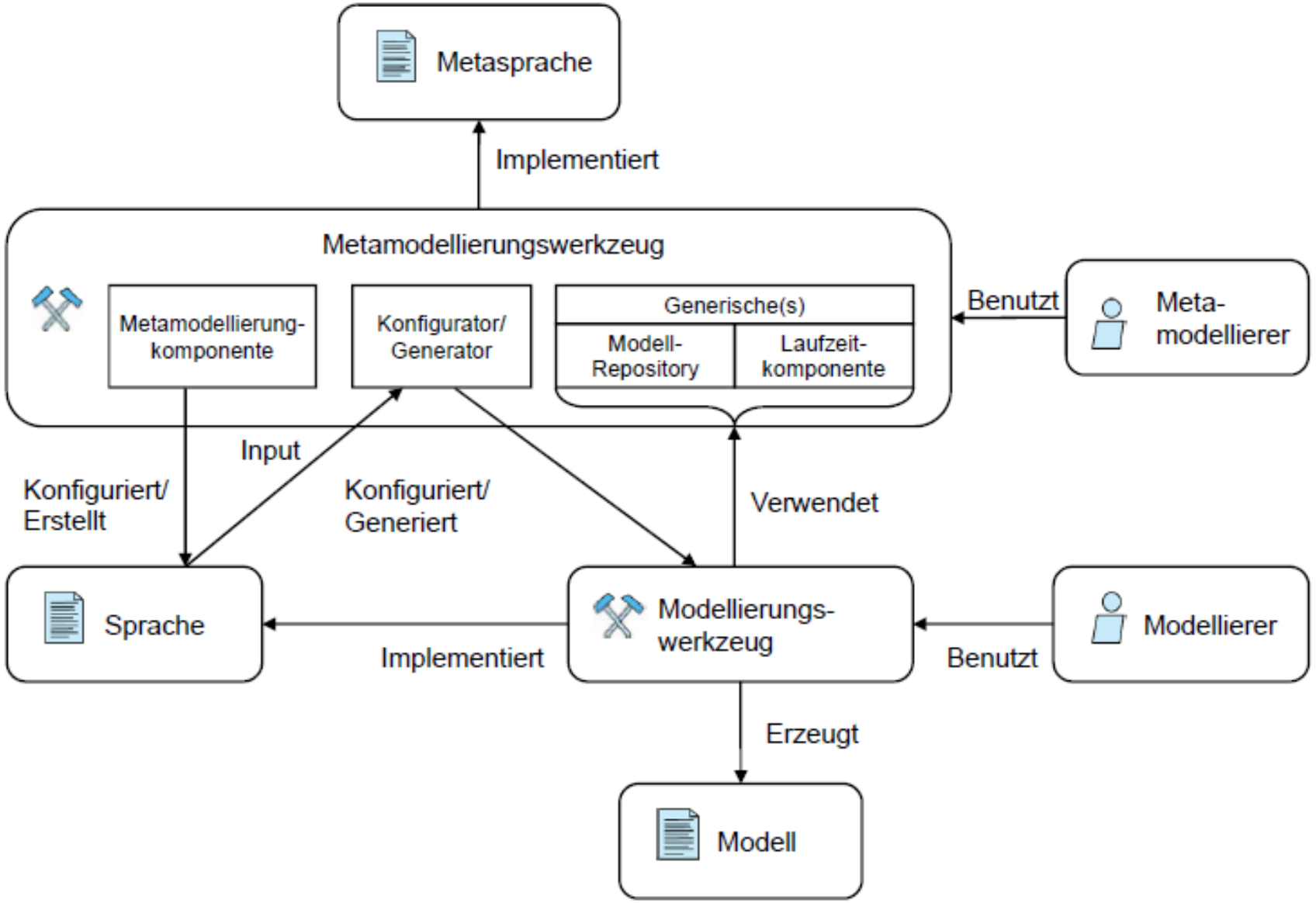
- **Relationstyp**
  - Beziehung zwischen Objekttypen
  - Andere Bezeichnungen: Referenz, Assoziation, Verbindung
  - Unidirektional, bidirektional, attributiert, (un)abhängig vom Objekttyp, Kardinalität

- **Attribut**
  - Eigenschaften von Metamodellelementen (Objekttyp, Relationstyp)
  - Besitzen meisten einen Datentyp, der den Wertebereich definiert

Metamodellierungskonzepte (2)

- **Modelltyp**
  - Umfasst verschiedene Metamodellelementen
  - Andere Bezeichnungen: Graphtyp, Domain-Modell, Metamodell
- **Vererbung**
  - Vererbungsbeziehung zwischen Metamodellelementen
- **Partitionierung**
  - Gruppierung bzw. Strukturierung von Metamodell-Elementen
  - Andere Bezeichnungen: Package, Namespace
- **Hinterlegung**
  - Beziehung zwischen Metamodellelementen (Objekttyp, Relationstyp) und einem Modelltyp
  - Andere Bezeichnung: Dekomposition, Hinterlegung

Metamodellierungswerkzeug



Auswahl einiger Werkzeuge

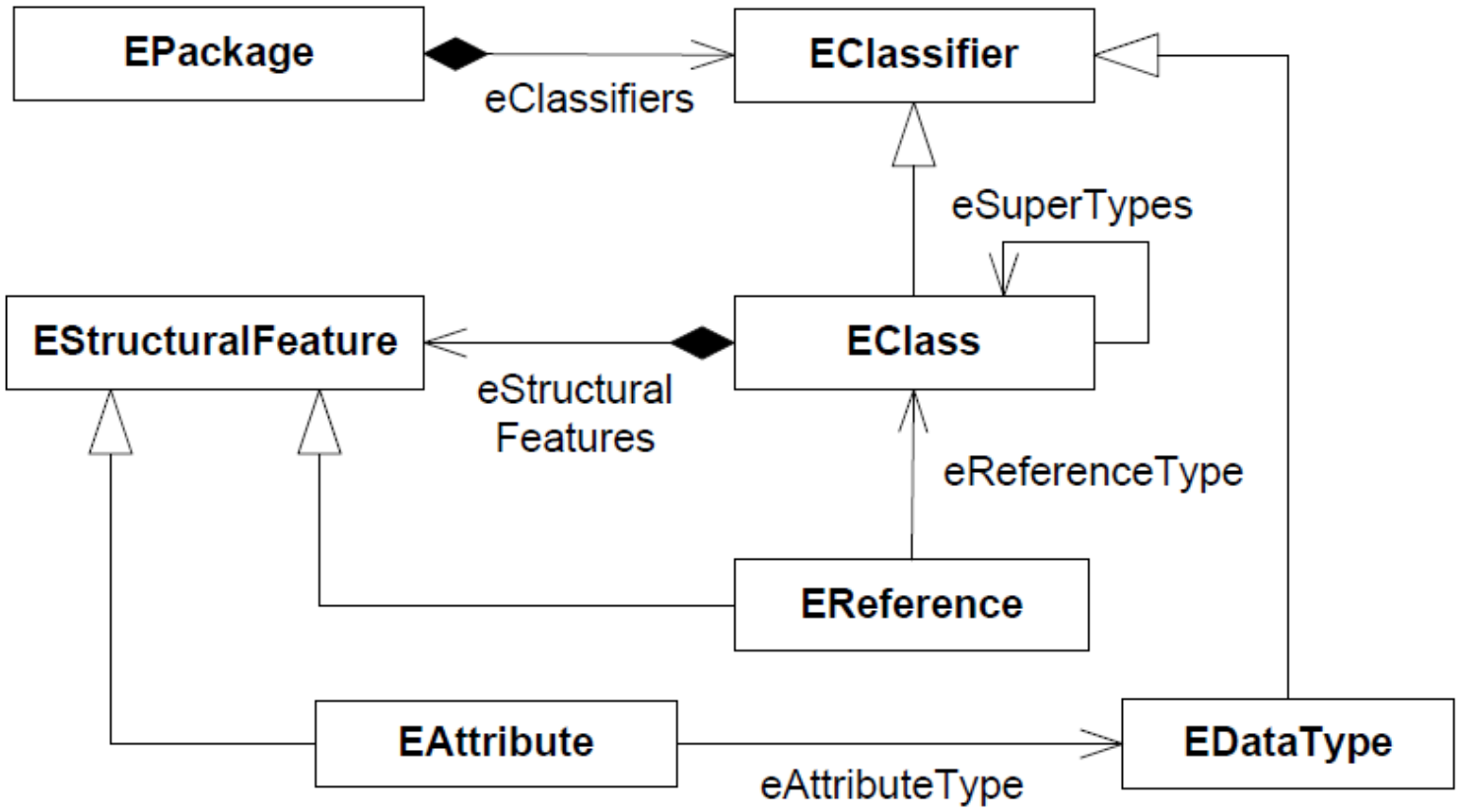
- Eclipse Modeling Project (http://www.eclipse.org/modeling/)
- MetaEdit+ von MetaCase (http://www.metacase.com/)
- Microsoft DSL Tools von Microsoft (http://www.domainspecificdevelopment.com/)
- MetaGME von University of Vanderbilt (http://w3.isis.vanderbilt.edu/projects/gme/index.html)
- Microsoft Visio

Eclipse Modeling Project

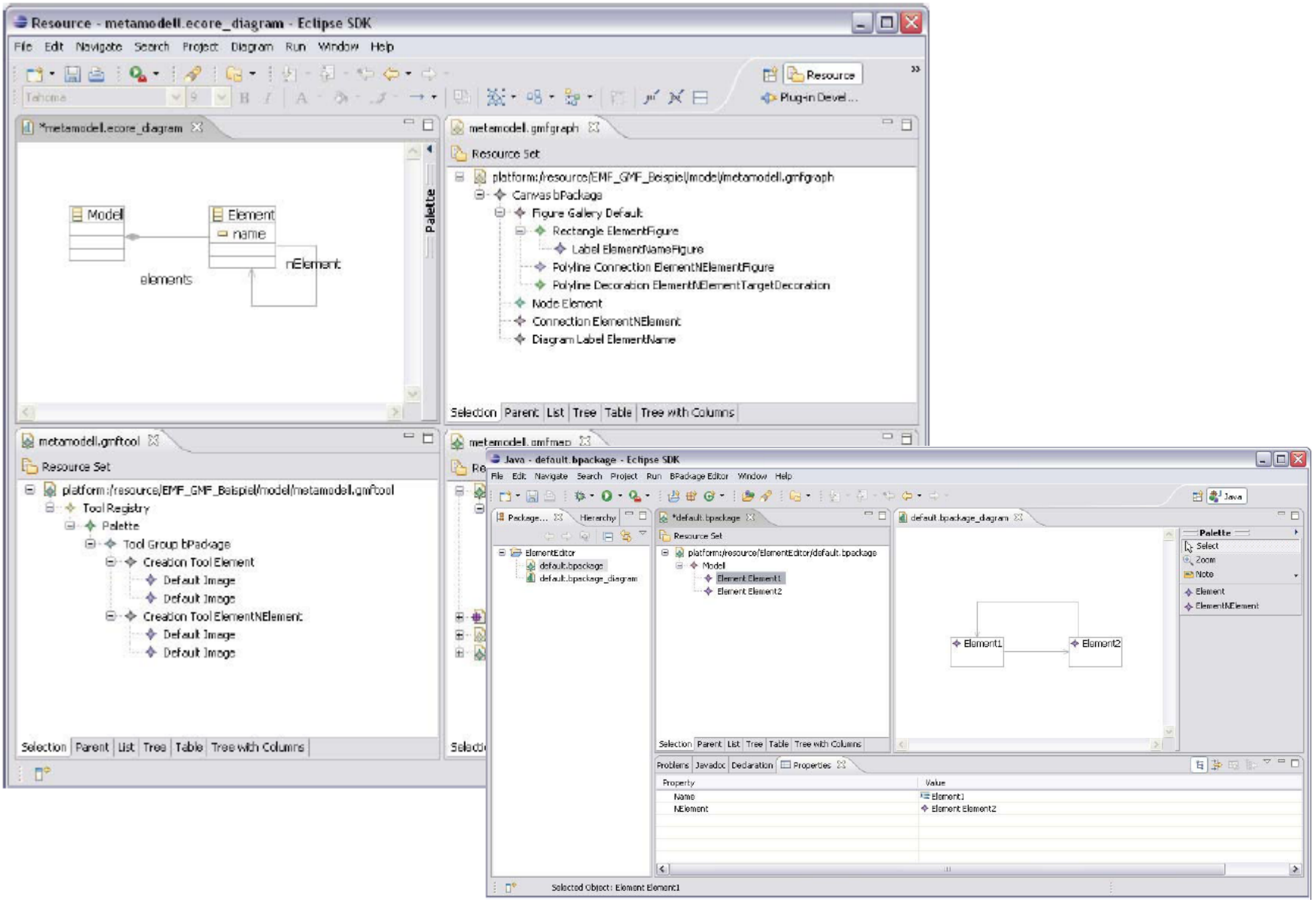
- **Zentraler Bestandteil ist das Eclipse Modeling Framework**
  - Bietet Unterstützung für die Entwicklung von (Eclipse-) Anwendungen auf der Grundlage von EMF-Modellen
  - EMF-(Meta)Modelle beschreiben die Struktur der zu entwickelten Anwendung unabhängig von der konkreten Implementierung
  - Aus EMF-Modellen kann entsprechender (Java-)Programmcode generiert werden
  - Anbindung an XML, UML, Java

- **Weitere Werkzeuge von EMP**
  - EMF Compare, Model Query, Model Transaction, Model Validation, Teneo, Connected Data Objects (CDO), **Eclipse GMF**, xText, UML2 Tools, Atlas Transformation Language, Java Emitter Templates, Xpand, ATLAS Model Weaver, Eclipse Epsilon

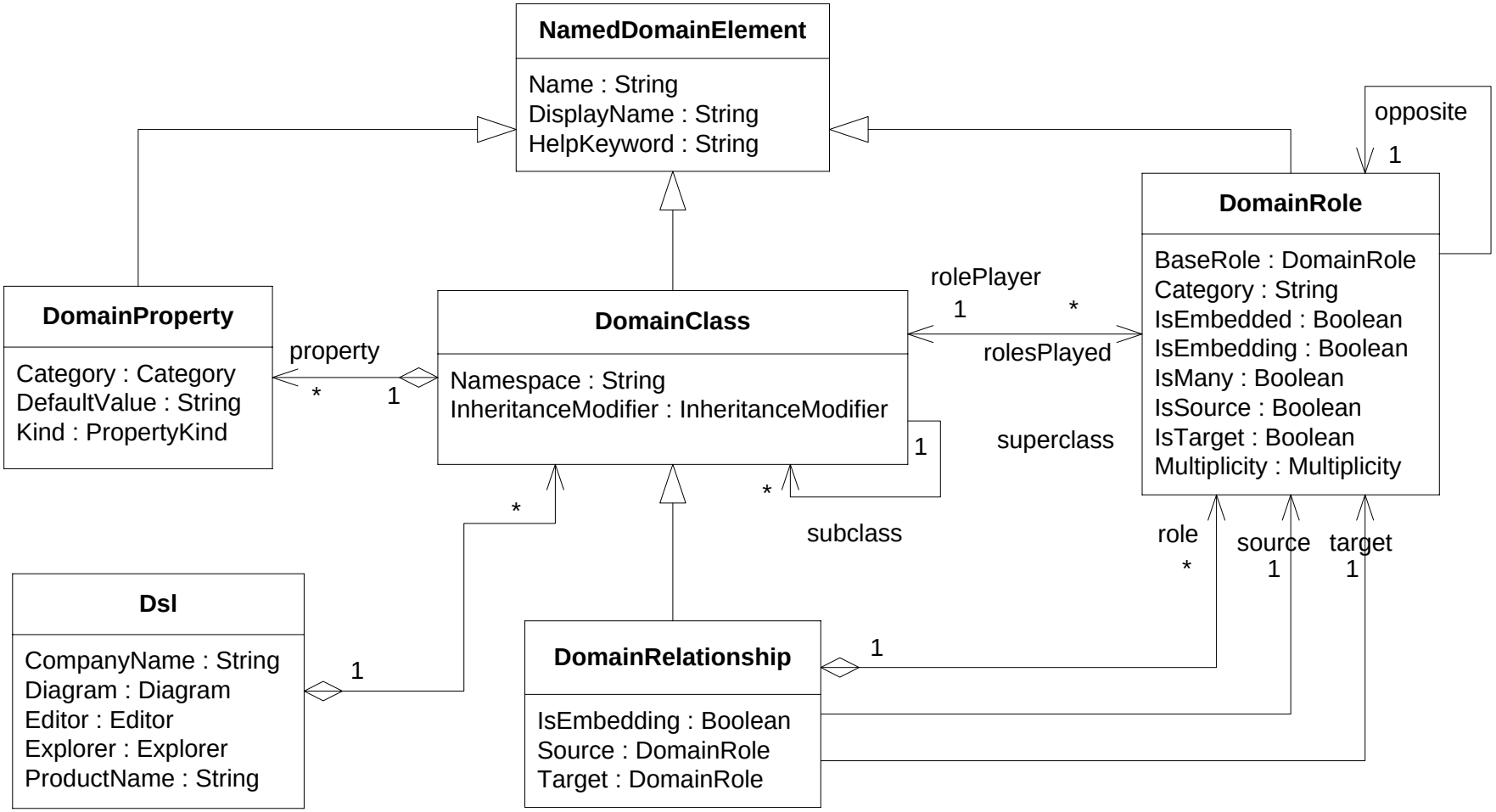
EMF: eCore


EPackage
eClassifiers
EClassifier
eSuperTypes
EStructuralFeature
eStructural
Features
EClass
eReferenceType
EReference
EAttribute
eAttributeType
EDataType

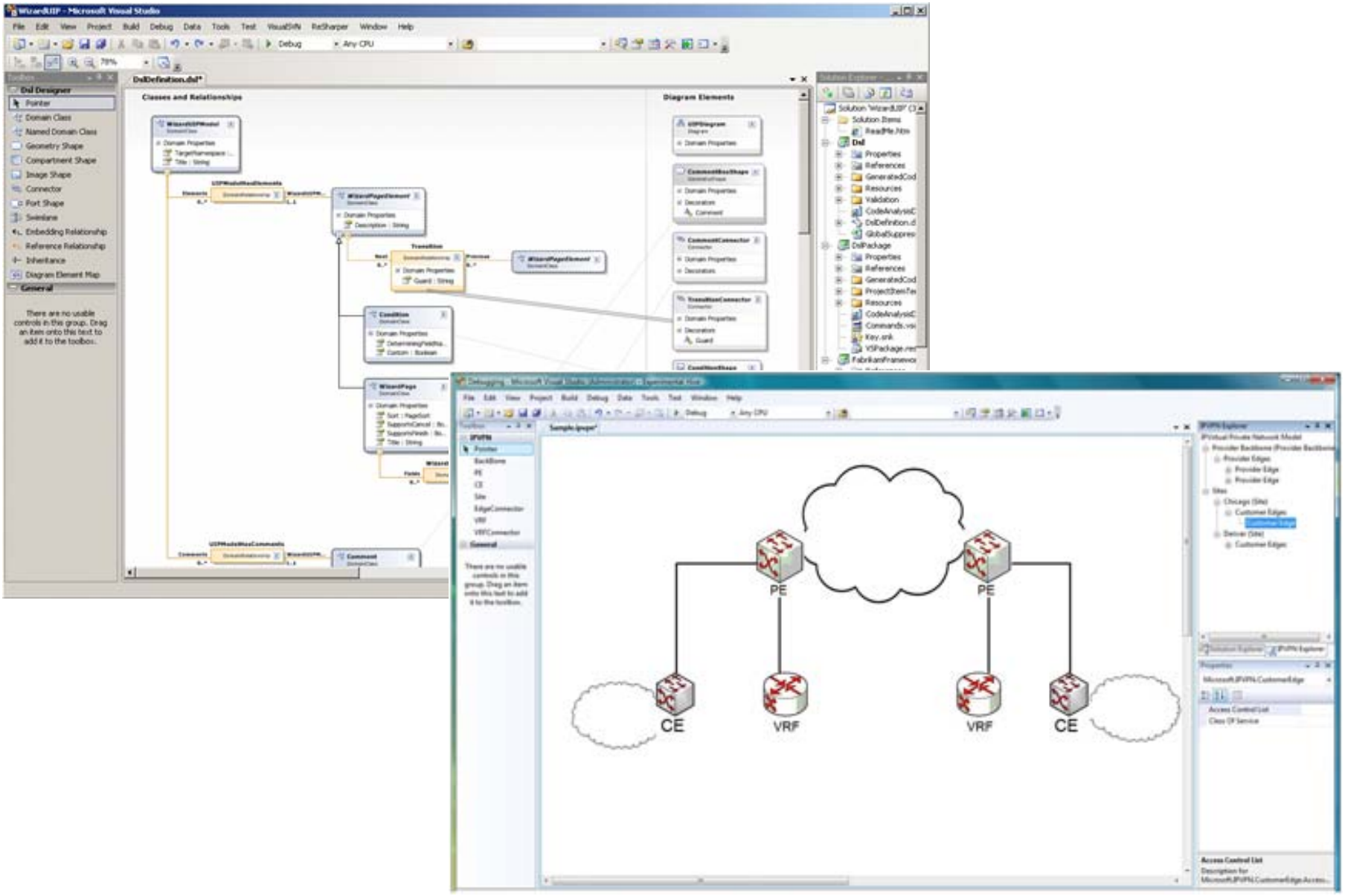
Eclipse GMF


Resource - metamodell.ecore_diagram - Eclipse SDK
File Edit Navigate Search Project Diagram Run Window Help
Resource
Plug-in Devel...
*metamodell.ecore_diagram
Model
Element
name
nElement
elements
Palette
metamodell.gmfgraph
Resource Set
platform:/resource/EMF_GMF_Beispiel/model/metamodell.gmfgraph
Canvas bPackage
Figure Gallery Default
Rectangle ElementFigure
Label ElementNameFigure
Polyline Connection ElementNElementFigure
Polyline Decoration ElementNElementTargetDecoration
Node Element
Connection ElementNElement
Diagram Label ElementName
Selection Parent List Tree Table Tree with Columns
metamodell.gmftool
Resource Set
platform:/resource/EMF_GMF_Beispiel/model/metamodell.gmftool
Tool Registry
Palette
Tool Group bPackage
Creation Tool Element
Default Image
Default Image
Creation Tool ElementNElement
Default Image
Default Image
Selection Parent List Tree Table Tree with Columns
metamodell.gmfmap
Java - default.bpackage - Eclipse SDK
File Edit Navigate Search Project Run BPackage Editor Window Help
Java
Package... Hierarchy
ElementEditor
default.bpackage
default.bpackage_diagram
*default.bpackage
Resource Set
platform:/resource/ElementEditor/default.bpackage
Model
Element Element1
Element Element2
Selection Parent List Tree Table Tree with Columns
default.bpackage_diagram
Palette
Select
Zoom
Note
Element
ElementNElement
Element1
Element2
Problems Javadoc Declaration Properties
Property
Value
Name
Element1
NElement
Element Element2
Selected Object: Element Element1

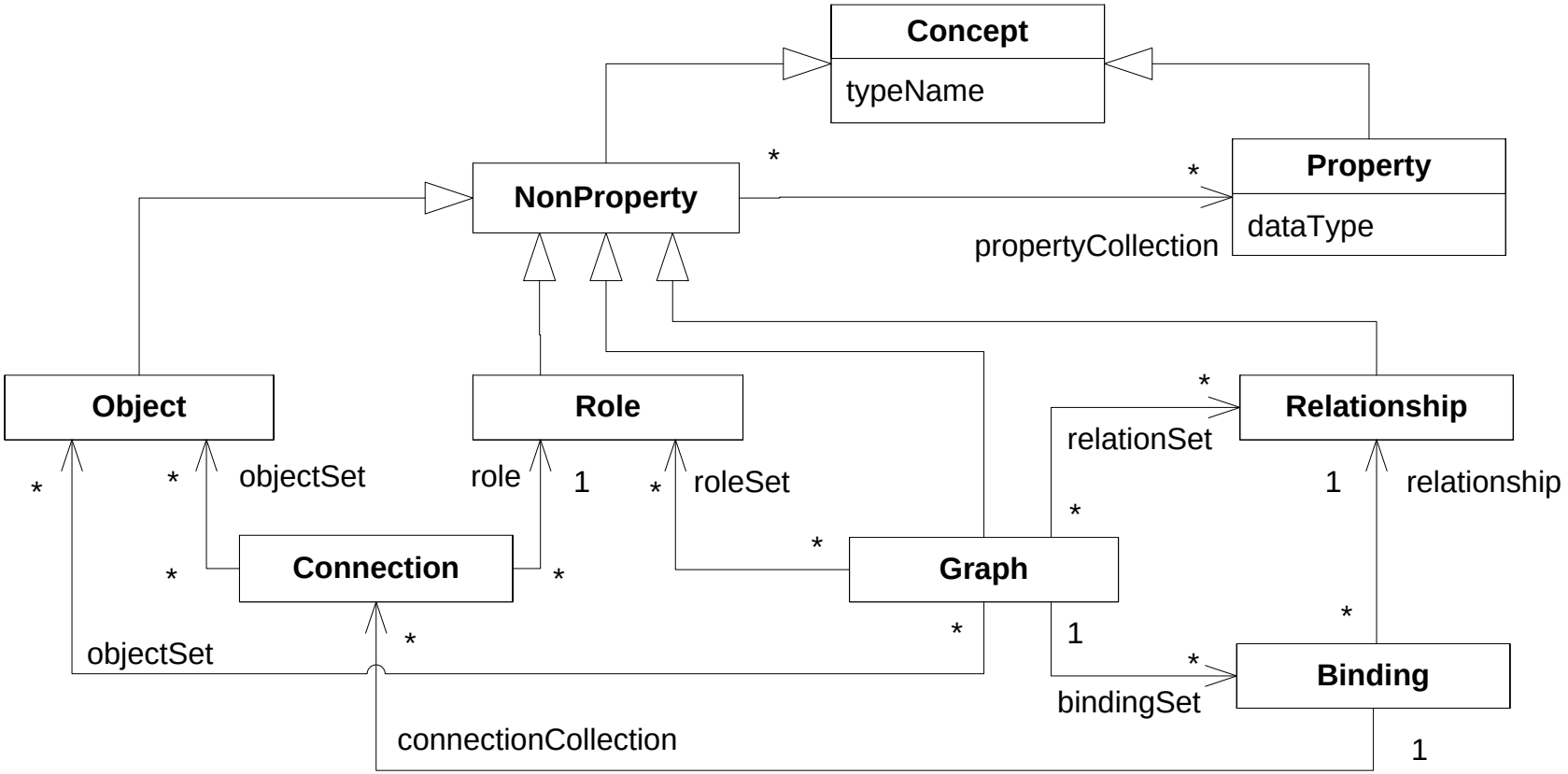
Microsoft DSL Tools: Metametamodell



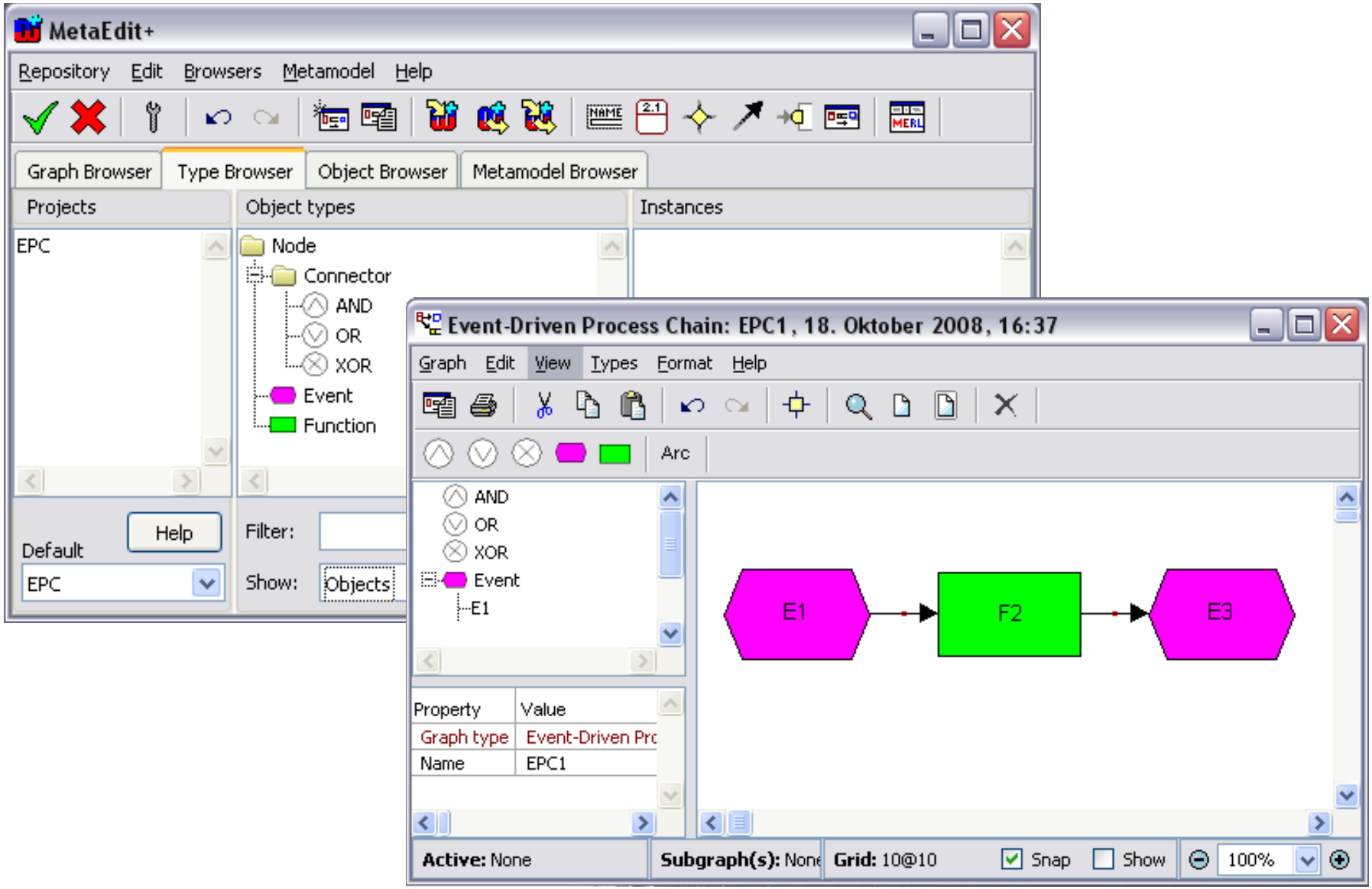
Microsoft DSL Tools



- **Graph Object Port Property Role Relationship**



MetaEdit+


MetaEdit+
Repository Edit Browsers Metamodel Help
Graph Browser
Type Browser
Object Browser
Metamodel Browser
Projects
Object types
Instances
EPC
Node
Connector
AND
OR
XOR
Event
Function
Default
Help
EPC
Filter:
Show:
Objects
Event-Driven Process Chain: EPC1, 18. Oktober 2008, 16:37
Graph Edit View Types Format Help
Arc
AND
OR
XOR
Event
E1
Property
Value
Graph type
Event-Driven Pro
Name
EPC1
E1
F2
E3
Active: None
Subgraph(s): None
Grid: 10@10
Snap
Show
100%

Microsoft Visio