M e t a-Modelle

Vortrag im Arbeitskreis Objektorientierung
9. 12. 2002

Übersicht

Vorstellung Person/ Firma

Teil 1: Meta-Modelle in Reinkultur
Begriffsbestimmung/ Beispiele/ Einsatzgebiete

Teil 2: ein bisschen Theorie und viel Praxis
die 4-Ebenen-Architektur
Meta-Bestandteile in Anwendungs-Software:
Vorteile/ Nachteile/ Konsequenzen

Infos zur Person und Firma

Beate Ritterbach

Methodenberaterung und Software-Entwicklung im Umfeld Objekt-Orientierung/ Java

seit 1990 freie Beratung/ Entwicklung

seit 2001: Logimod GmbH

Kooperation mit weiteren Firmen im OO-Beratungs-Umfeld

Interessenschwerpunkte

Techniken zur Entwicklung von **verständlichem** Code

Agile Prozesse, insbes.Refactoring

Modellierung

Programmiersprachen

zugrundeliegende Paradigmen und Konzepte

Qualitätskriterien und Messbarkeit

Aktuelle „Arbeitsgebiete“

Werte und Objekte

Qualitätskriterien für Sprachen

Meta-Modelle

Sprachkonzepte, die Code-Verständlichkeit „erzwingen“

Zielsetzung

Erkennen/ Sensibilisieren für Themen wie

- Was sind Meta-Modelle?
- Wofür sind sie einsetzbar?
- Wann bewegt man sich auf der Meta-Ebene?
- Welche Vorteile/ Welche Gefahren birgt es?

Teil 1: Meta-Modelle in Reinkultur

Gliederung

- Begriffsbestimmung
- Beispiele für Meta-Modelle „in Reinkultur“
- Nutzen und Einsatzgebiete für Meta-Modelle

Begriffsbestimmung: Meta-...

Meta (griech) = über, nach, neben, zwischen, ...

Meta-Ebenen sind auch in anderen Disziplinen gebräuchlich , z. B.

Ökonomie: Meta-Geschäft = Geschäft über die Aufteilung des Gewinns aus zukünftigen Geschäften

Kommunikationstheorie (Schulz v. Thun): Meta-Kommunikation = Kommunikation über Art und Inhalt des Gesagten

Philosophie : Meta-Kritik = Kritik der Kritik

Begriffsbestimmung: Modell

1. Eine inhaltliche Beschreibung von Datenstrukturen, Prozessen, Regeln, etc. für ein konkretes Anwendungsgebiet
z. B. das Klassenmodell von Projekt X,
das UDM der Firma Y, teilweise auch: Schema

2. ein Ausdrucksmittel mit formal festgelegten Symbolen und Regeln, das den Rahmen absteckt, um einen Gegenstandsbereich zu beschreiben
z. B. „das ER-Modell“
-> eine Sprache

Begriffsbestimmung: Sprache

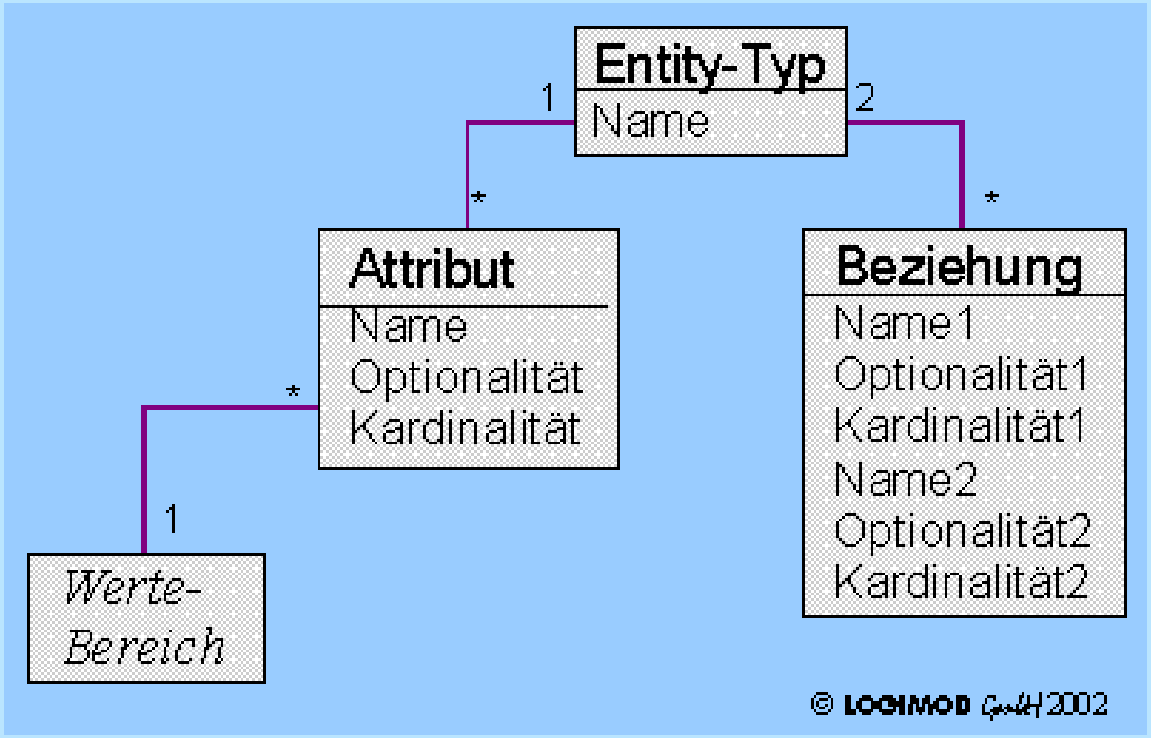
Ein Ausdrucksmittel mit

- Vokabular
- Syntax
- Semantik
- Pragmatik

Modellierungs-Sprachen und Programmier-Sprachen verfügen über diese 4 Aspekte.

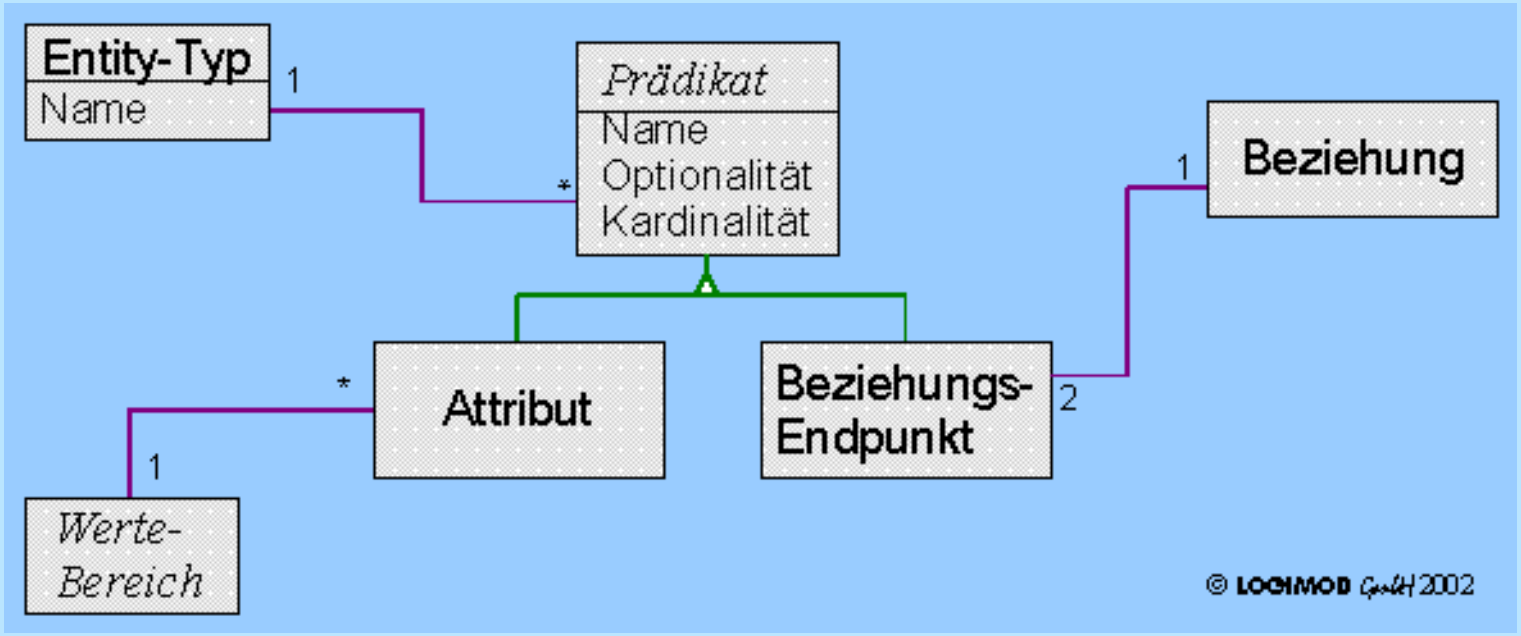
Beispiel 1:
Meta-Modell des ER-Modells

Binäres Entity-Relationship-Modell



Meta-Modell des ER-Modells

Variante 2 (desselben Modells)



Beispiel 2: Meta-Modell der UML

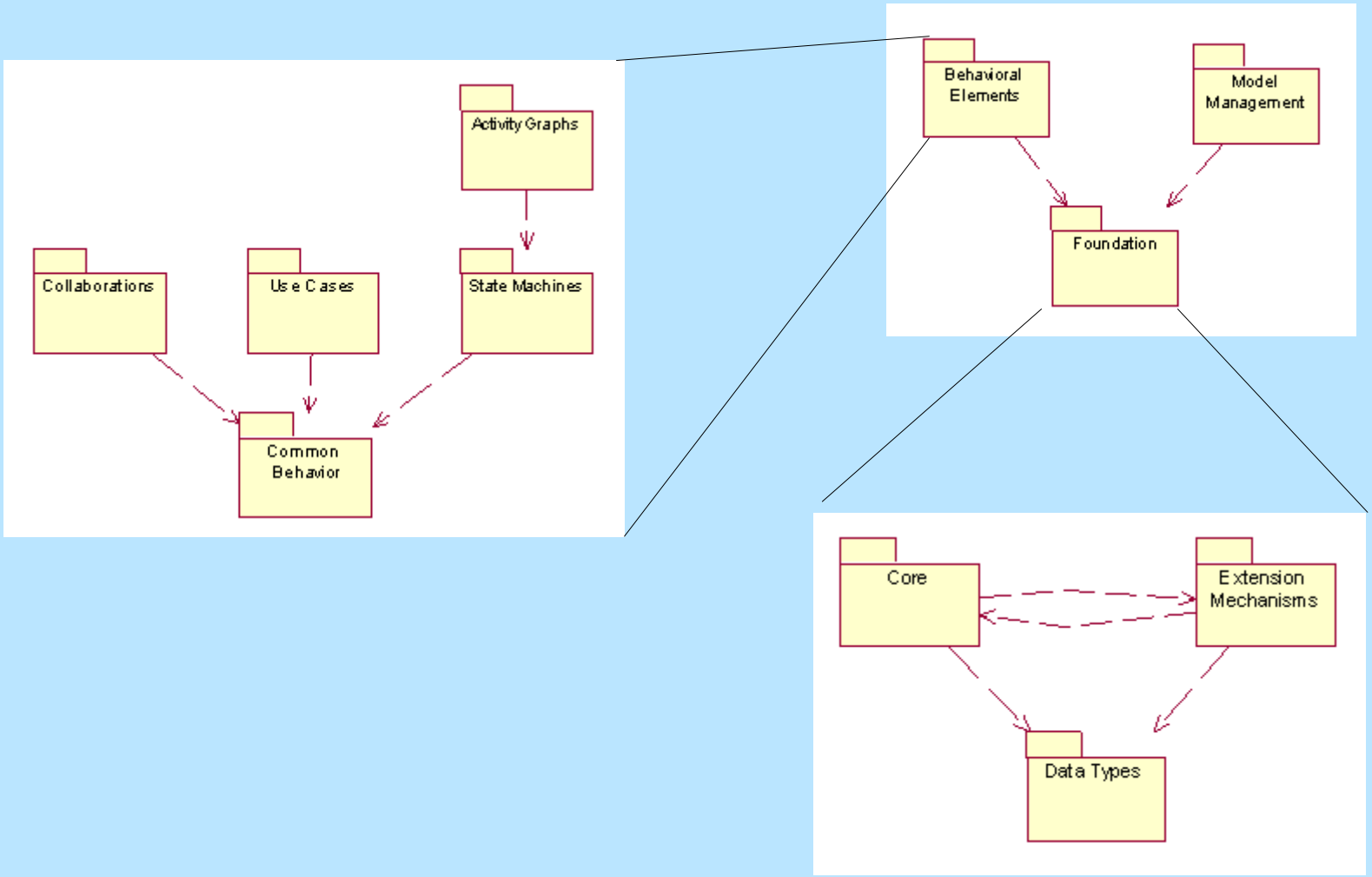
Überblick:
Das UML-Meta-Modell enthält

... ca. 130 (Meta-)Klassen und zugehörige Attribute, Assoziationen, Constraints, ...

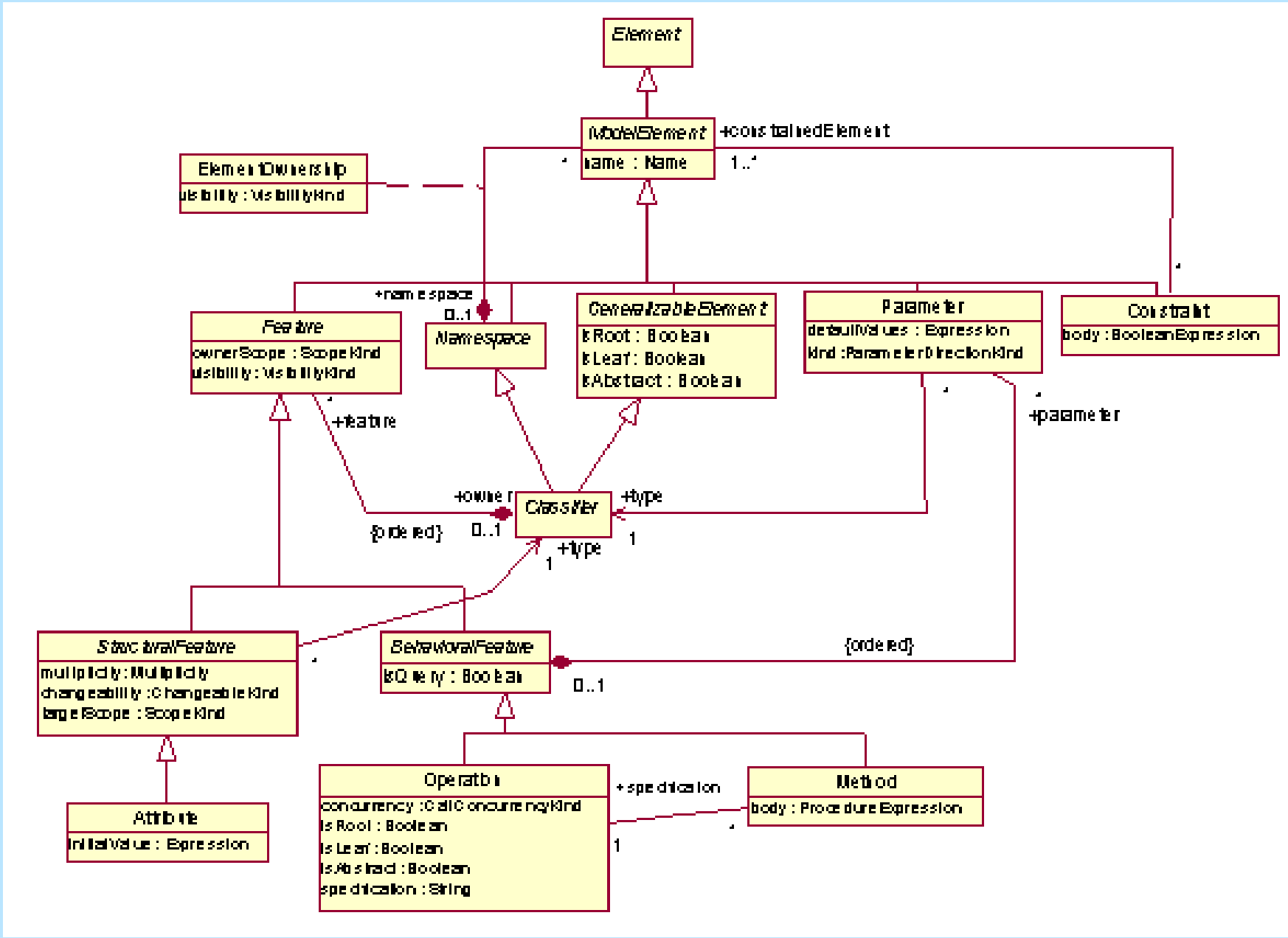
... in 11 Paketen/ Unterpaketen, teilw. nochmals aufgeteilt auf mehrere Diagramme

das beschreibende Dokument umfasst über 500 Seiten

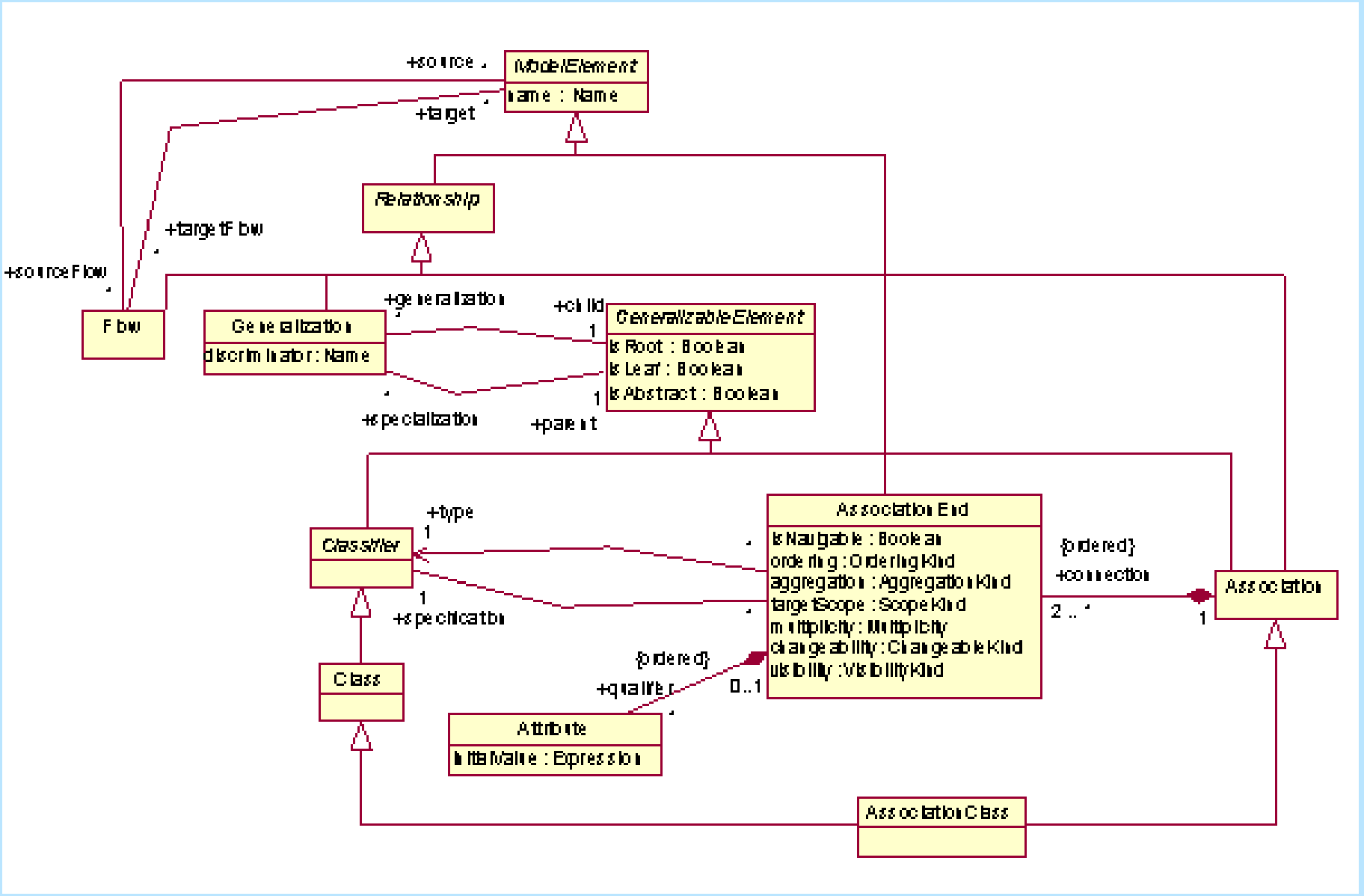
UML-Meta-Modell: Packages



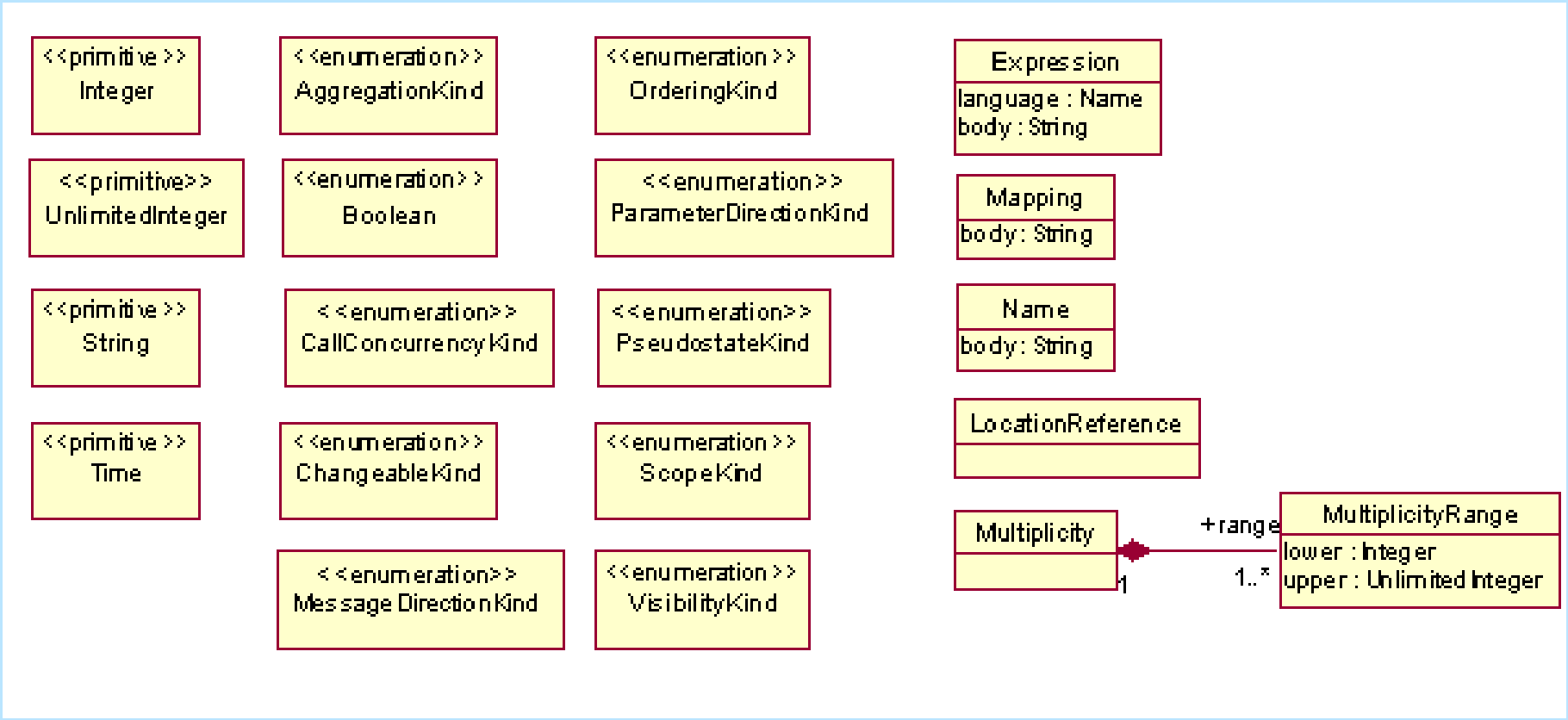
UML-MM: Core/ Backbone



UML-MM: Core/ Relationships



UML-MM: Data-Types



Meta-Modell der UML

Das Meta-Modell ist selbst in UML-Notation

Zu jeder Meta-Klasse gehört:

- eine verbale Beschreibung seiner Exemplare
- eine Anlistung der zugehörigen (Meta)Attribute sowie (Meta)Assoziationen
- zugehörige Regeln/ Einschränkungen („Constraints“)

Beispiel: Association End/ Beschreibung

An association end is an endpoint of an association, which connects the association to a classifier. Each association end is part of one association. The association-ends of each association are ordered.

In the metamodel, an AssociationEnd is part of an Association and specifies the connection of an Association to a Classifier. It has a name and defines a set of properties of the connection (e.g., which Classifier the Instances must conform to, their multiplicity, and if they may be reached from another Instance via this connection).

In the following descriptions when referring to an association end for a binary association, the source end is the other end. The target end is the one whose properties are being discussed.

...

Beispiel: Association End/ Attribute

aggregation (none/ aggregate/ composite) When placed on one end (the target end), specifies whether the class on the target end is an aggregation with respect to the class on the other end (the „source”end). Only one end can be an aggregation ...

changeability (changeable/ frozen/ addOnly) ...

ordering (unordered/ ordered) ...

isNavigable ...

multiplicity ...

name (Inherited from ModelElement) ...

targetScope (instance/ classifier) ...

visibility (public/ protected/ private/ package) Specifies the visibility of the association end from the viewpoint of the classifier on the other end.

...

Constraints

Auch genannt: „Well-Formedness Rules“

Können verbal abgefasst sein, oder formal mit der OCL (Object Constraint Language)

ein grosser Teil dessen, was die Bedeutung des Meta-Modells ausmacht, steckt in diesen Constraints

Beispiele für Constraints

AssociationEnd:
An Instance may not belong by composition to more than one composite Instance.
OCL: *self.aggregation = #composite implies self.multiplicity.max <= 1*

The Classifier of an AssociationEnd cannot be an Interface or a DataType if the association is navigable away from that end.

Class:
If a Class is concrete, all the Operations of the Class should have a realizing Method in the full descriptor.

Data Type:
A DataType can only contain Operations, which all must be queries.

Programmiersprachen

Beobachtungen:

- Erhebliche Unterschiede im Programmierstil bei Verwendung der gleichen Sprache möglich
- Trotz gleicher Sprache fällt es einem Programmierer oft schwer, den Code eines anderen zu verstehen und damit zu warten
- Die verwendete Programmiersprache beeinflusst die Denkweise des Programmierers
- Der Umstieg von einer Programmiersprache zu einer anderen ist aufwändig und wird nach Möglichkeit vermieden

Hilferuf

Hi Beate,

habe das Gefühl, ich verstehe überhaupt nichts mehr.
Also, wie nennt man den u.a. Text? Das steht in einem Texteditor, bzw. in einer Datei CSparbuchNutzen1.java.
Ist das jetzt eine KLasse? wobei ich nicht weiß, was dann die nicht public classes bedeuten??
wobei doch aber einzahle, abheben, ertrag und ertragZ methoden sind.
Und was wären Objekte?? bzw. Instanzen??
Ich mag nicht mehr, das kann doch nicht so schwierig sein, andere haben das doch auch gelernt....

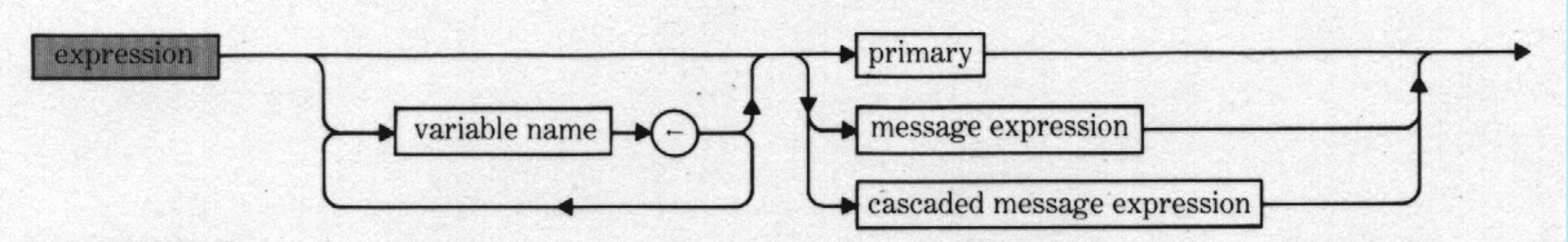
```
class CSparbuch
   {
   double kapital;
...
```

Annäherung an eine Programmiersprache

Learning by Doing, Beispiele, „Teach yourself...“
Auswendiglernen aller Keywords

Sprachspezifikation:
sehr lang, grösstenteils verbal,
Java-Language Specification: > 800 Seiten

Syntax-Diagramme



starke Betonung der Syntax, nur noch wenig gebräuchlich, Semantik tritt in den Hintergrund

Programmiersprachen

Es kommt es primär auf die zugrundeliegenden Konzepte an

Äusserlichkeiten wie Syntax oder Befehlsumfang sind zweitrangig

Meta-Modell einer PROGRAMMIER-Sprache:
-> bisher in der Literatur nicht gefunden

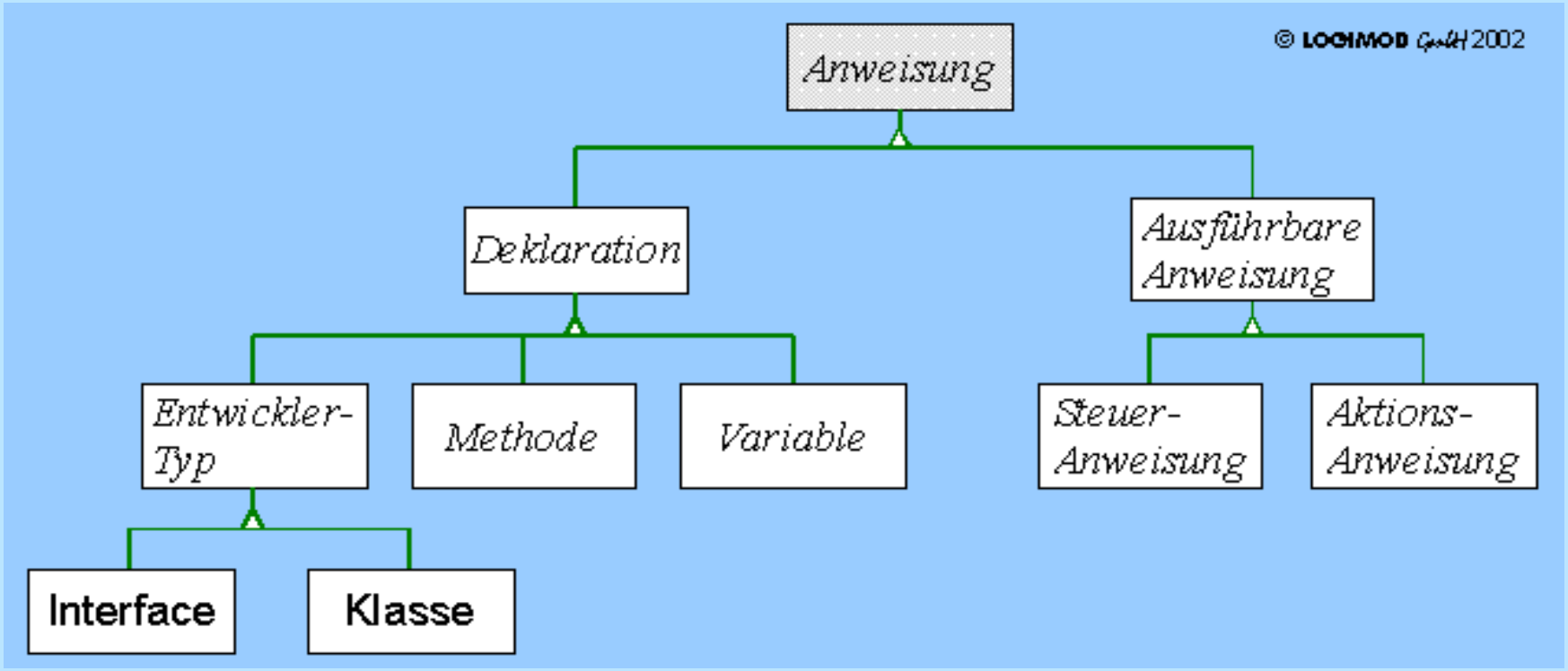
Experiment:
Meta-Modell einer PROGRAMMIER-Sprache erstellen -> Java

Beispiel 3:
Meta-Modell der Sprache Java

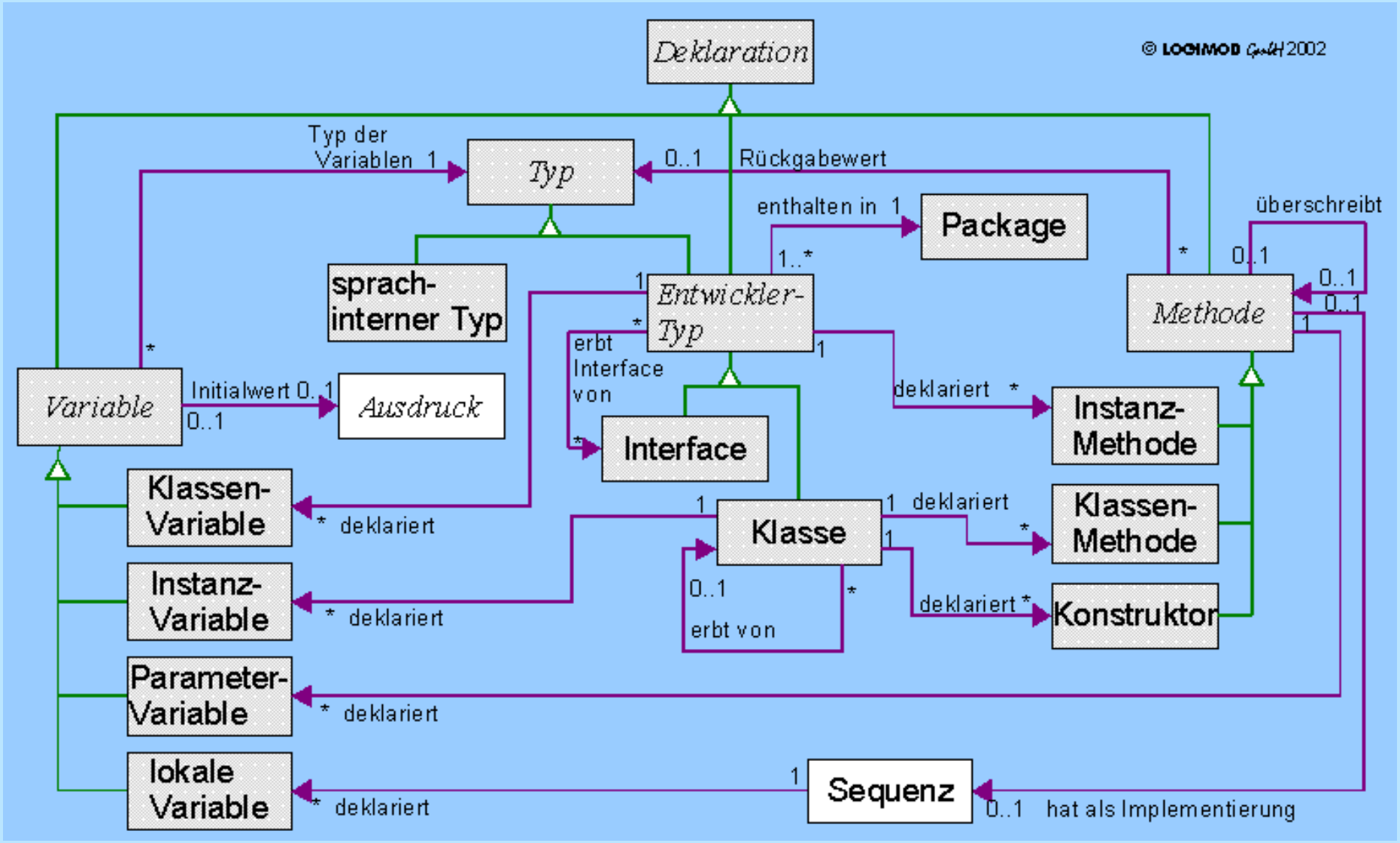
Überblick

- 40 Klassen (davon 16 abstrakt)
- mit zugehörigen Vererbungsbeziehungen, Assoziationen
- Verteilt auf 4 Diagramme
- Zugehörige verbale/ formale Beschreibung (Attribute, Constraints, ...)

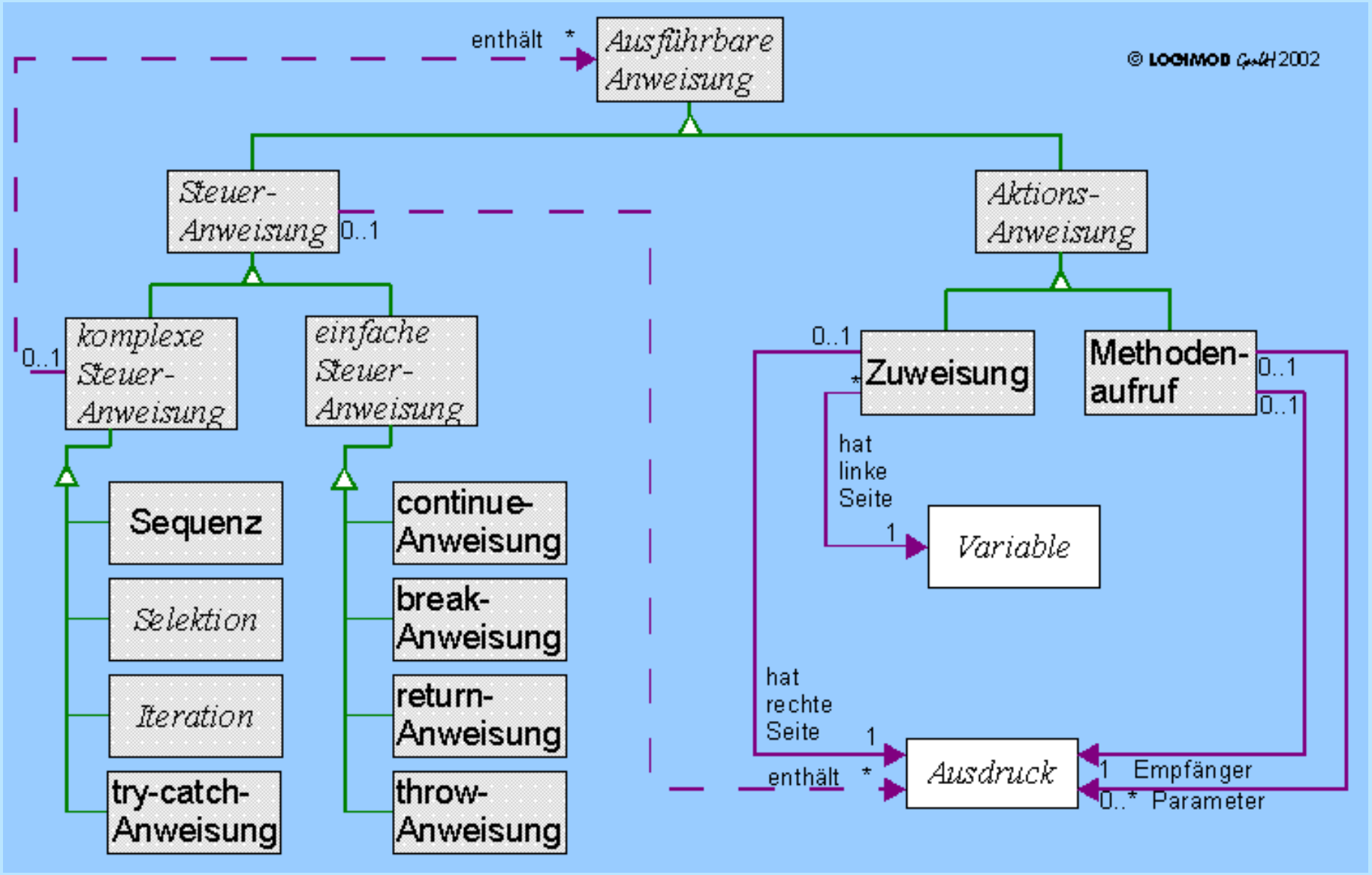
Java-MM: Übersicht



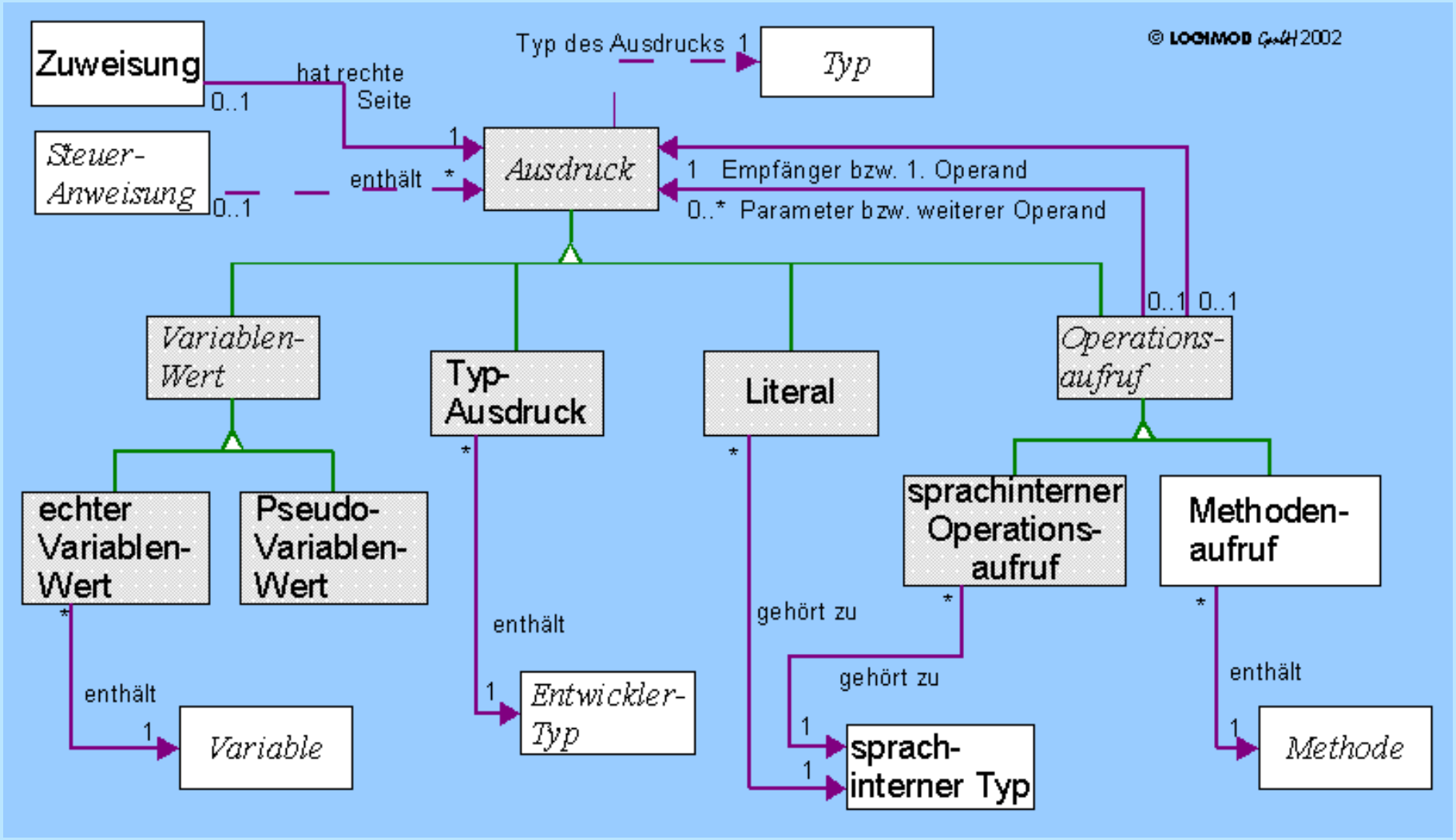
Java-MM: Deklarationen



Java-MM: Ausführbare Anweisungen



Java-MM: Ausdrücke



Java-Meta-Modell: Kernpunkte

Überblick über verfügbare Sprachelemente, z. B.

- Klassen / Interfaces/ Primitive Datentypen
- Statische Typisierung
- Exception Handling
- Einfach/ Mehrfach-Vererbung
- Keine Zusicherungen
- Keine Generischen Typen (ausser Arrays)

Java-Meta-Modell: Kernpunkte

Konzentration auf die Sprach-Konzepte

Grafische Darstellung von deren Zusammenhang

Keine Abbildung der Sprachsyntax

Kein Nachschlagewerk für Schlüsselwörter und Anweisungen der Sprache

(Meta-)Modell nie eindeutig - immer auch Sichtweise des Modellierers

Java-MM: Veranschaulichung von Sprach-Konzepten

von der Sprache mitgelieferte Typen und vom Entwickler deklarierbare Typen als Spezialfälle desselben Konzeptes

Sprachsyntax unterschiedlich für:
Methodenaufrufe - Operationen f. Primitive Typen
Literale - Objekte

Vererbungskonzept von Java: einfache Implementierungs-Vererbung, mehrfache Schnittstellenvererbung mit Interfaces

Sprachsyntax nutzt unterschiedliche Schlüsselwörter (extends/ implements) bei gleicher Vererbungsart

Java-MM: Veranschaulichung von Sprach-Konzepten

Konzeptionelle Trennung von Instanzmethoden, Klassenmethoden, Konstruktoren

Sprachsyntax: Konstruktoren gleicher Name wie Klasse, kein eigenes Schlüsselwort für Instanzmethoden, Schlüsselwort static für Klassenmethoden,

Trennung von Variable als Ziel einer Zuweisung und Variablenwert durch Verwendung von verschiedenen Meta-Klassen

Sprachsyntax: Variable als Ziel einer Zuweisung und als Ausdruck, mit dem auf den Wert der Variablen bezug genommen wird, sind syntaktisch gleich

Java-MM: Veranschaulichung von Sprach-Konzepten

Meta-Modell dokumentiert das Vorhandensein diverser prozeduraler Elemente (Steueranweisungen, Schleifen, Selektionen, ...)
der zugehörige Ausschnitt des Meta-Modells ist mit dem Meta-Modell einer prozeduralen Sprache fast identisch

nur 2 grundlegende Arten von Aktionen: Zuweisung und Aufruf einer Operation/ Methode
Sprachsyntax: Unterschiede zwischen Aufruf Methode/ spracheigene Operation, Zuweisung und Operationen haben syntaktisch gleiche Form

Java-MM - Details hinter der Grafik

Einige Details wurden in den Diagrammen weggelassen (-> (Meta-)Attribute)

Einige Details sind in den Diagrammen nicht darstellbar (-> Constraints)

Java-MM
Meta-Attribute / Beispiele

Klasse
Sichtbarkeit
isAbstract
isFinal

Variable
Sichtbarkeit
isFinal
isTransient
isArray
dimension

Methode
Sichtbarkeit
isAbstract
isFinal

Java-MM Constraints Beispiele

Vererbung darf nicht kreisförmig verlaufen.

Eine Methode muss entweder abstrakt sein oder eine Implementierung haben.

Eine Methode in einem Interface ist immer abstrakt.

Eine final Methode darf in einer Unterklasse nicht überschrieben werden.

Eine Methode darf nicht abstrakt und final sein.

Java-MM Constraints
Beispiele

Eine Methode darf in ihrer Implementierung nur auf Variablen zugreifen, die

- in der eigenen Klasse deklariert sind
- in einer Oberklasse deklariert und mindestens protected sind
- als Parameter dieser Methode übergeben werden
- auf die eigenen lokalen Variablen

Java-MM Constraints
Beispiele

Der Typ des Ausdruck rechts einer Zuweisung muss zum Typ der Variablen links der Zuweisung passen (d. h. davon erben).

In einem Methodenaufruf darf als Parameter nur ein Ausdruck verwendet werden, dessen Typ zum Typ der zugehörigen Parametervariablen passt.

Java-MM Constraints

Wie beim UML-Meta-Modell wird durch die Constraints ein grosser Teil der Bedeutung des Modells (bzw. der Sprache) festgelegt

Die Constraints bilden ein Regelwerk, wie der Programmierer die Sprachelemente nutzen/ verknüpfen darf und wie nicht

Constraints sind vom Compiler - und damit bereits zur Entwicklungszeit - überprüfbar

Zusammenfassung Meta-Modelle

Meta-Modelle beschreiben:

- Modellierungs- Sprachen
- Programmiersprachen
- Werkzeuge (soweit sie Unterstützung für Sprachen bieten)

d. h. Beschreibung des Ausdrucksmittels selbst, nicht eines konkreten Systems

Nutzen von Meta-Modellen

Visualisierung
-> erleichtert den Blick für das Wesentliche und für **Zusammenhänge**

Klare und formale Beschreibung auf semantischer Ebene, explizite Konzepte
-> geringere Gefahr von Missverständnissen/ Halbwissen
-> Teamübergreifend **gleiches** Verständnis für eine Sprache bzw. die davon einzusetzenden Elemente

Trennung der Kernkonzepte von Äusserlichkeiten
-> Entfernen des „Syntax Sugar“
-> kein „man sieht den Wald vor Bäumen nicht“

Einsatzgebiete für Meta-Modelle

Entwerfen einer Sprache

Entwickeln eines Modellierungs-Werkzeuges

Erlernen/ Unterrichten einer Sprache

Aufgaben im Bereich Architektur, Standards und Methoden:
-> Auswahl/ Erweiterungen/ Einschränkungen von bestehenden Sprachen und Werkzeugen

Beurteilung der Qualität

Ist ein Meta-Modell

- vorhanden? (bzw. leicht erstellbar?)
- einfach oder unübersichtlich?
- überschaubar oder umfangreich?
- Elegant oder voller Sonderfälle?
- einheitlich oder zusammenhanglos?

Vergleichen verschiedener Ansätze

Werden die gleichen Konzepte abgedeckt?

Passen die einzelnen Konzepte zueinander?

Passen die vorliegenden Konzepte zum Bedarf?

Enthält das Meta-Modell zu viele (und damit ggf. unerwünschte) Elemente?

Integrieren verschiedener Ansätze

Verschiedene Entwicklungsphasen,
heterogene Systemlandschaft, ...

- Passen die einzelnen Konzepte zueinander?
- Ergänzen sie sich? Wo sind die Nahtstellen?
- Überschneiden sie sich?
- Stehen sie isoliert nebeneinander?

Glaubwürdigkeit/ Konsequenz

Um ein komplexes System zu verstehen, benötigt man ein Modell davon

Da ist es konsequent, diese Forderung auf die eigenen Arbeitsmittel zu übertragen

„Nachdenken über das eigene Tun“

Was leisten Meta-Modelle NICHT?

Kein Ersatz für praktische Erfahrung im Modellieren und Programmieren

Kann komplexe Mechanismen nur begrenzt erläutern (z. B. Interfaces, ExceptionsHandling,..)

Kein Nachschlagewerk für Bedeutung und Handhabung einzelner Sprachbefehle
-> keine Syntax

Keine Richtlinie, in welchem Kontext welche Sprachelemente sinnvoll einsetzbar sind
-> keine Pragmatik

Übungsaufgabe

;-)

Erstellen Sie ein Meta-Modell Ihrer Lieblings-Sprache!