Objektorientierte Techniken und UML

Unter **Objektorientierung**, kurz OO, versteht man eine Sichtweise auf komplexe Systeme, bei der ein System durch das Zusammenspiel kooperierender Objekte beschrieben wird.

Wikipedia

Vorgehensmodelle
der Softwareentwicklung

Wasserfall**modell**

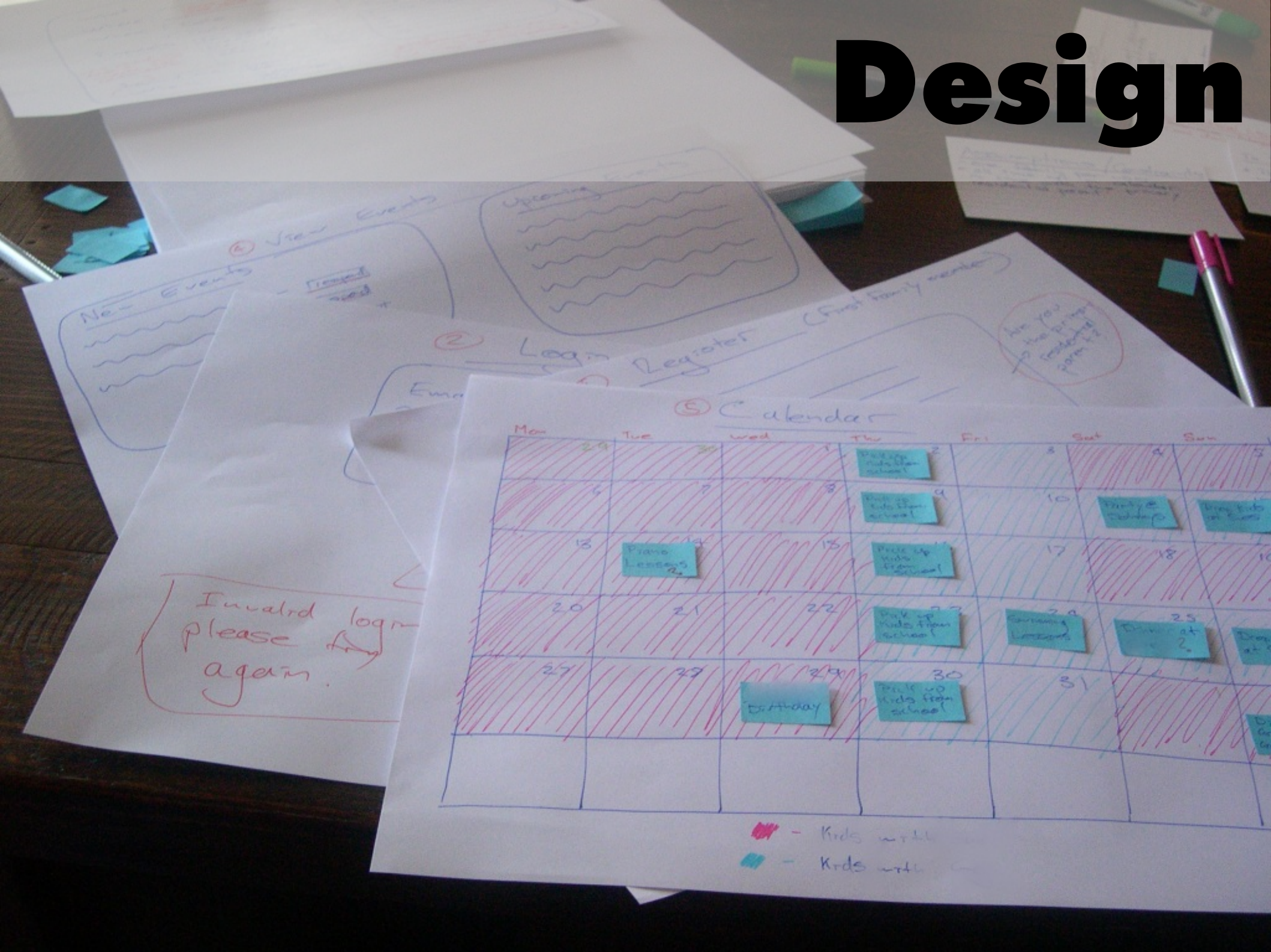
linear / nicht-iterative

Agile Softwareentwicklung iterativ und inkrementell

Phasen

Anforderungs**analyse**

Design


Upcoming
Calendar
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Piano Lessons
Birthday
Invalid login
please try
again.

Implementierung

Notationsformen

ER-Modell

UML Klassendiagramm

Elemente der Objektorientierung

Eine **Klasse** ist die Definition der **Attribute**, **Operationen** und der Semantik für eine Menge von **Objekten**. Alle Objekte einer Klasse entsprechen dieser Definition.

Bernd Oestereich

Waffeln (1/2 Rezept): (1/4 Mehr 7)

125 g	Margarine	75
125 g	Zucker	75
3	Eigelb	2
250 g	Mehl	125
1/2 Teel.	Backpulver	1/4
1/4 l	Milch angewärmt	knapp 1/8 (130 ml)
3	Eiweiß (Schnee)	2
1 Pck.	Vanillinzucker	1

Klassen

Objekte

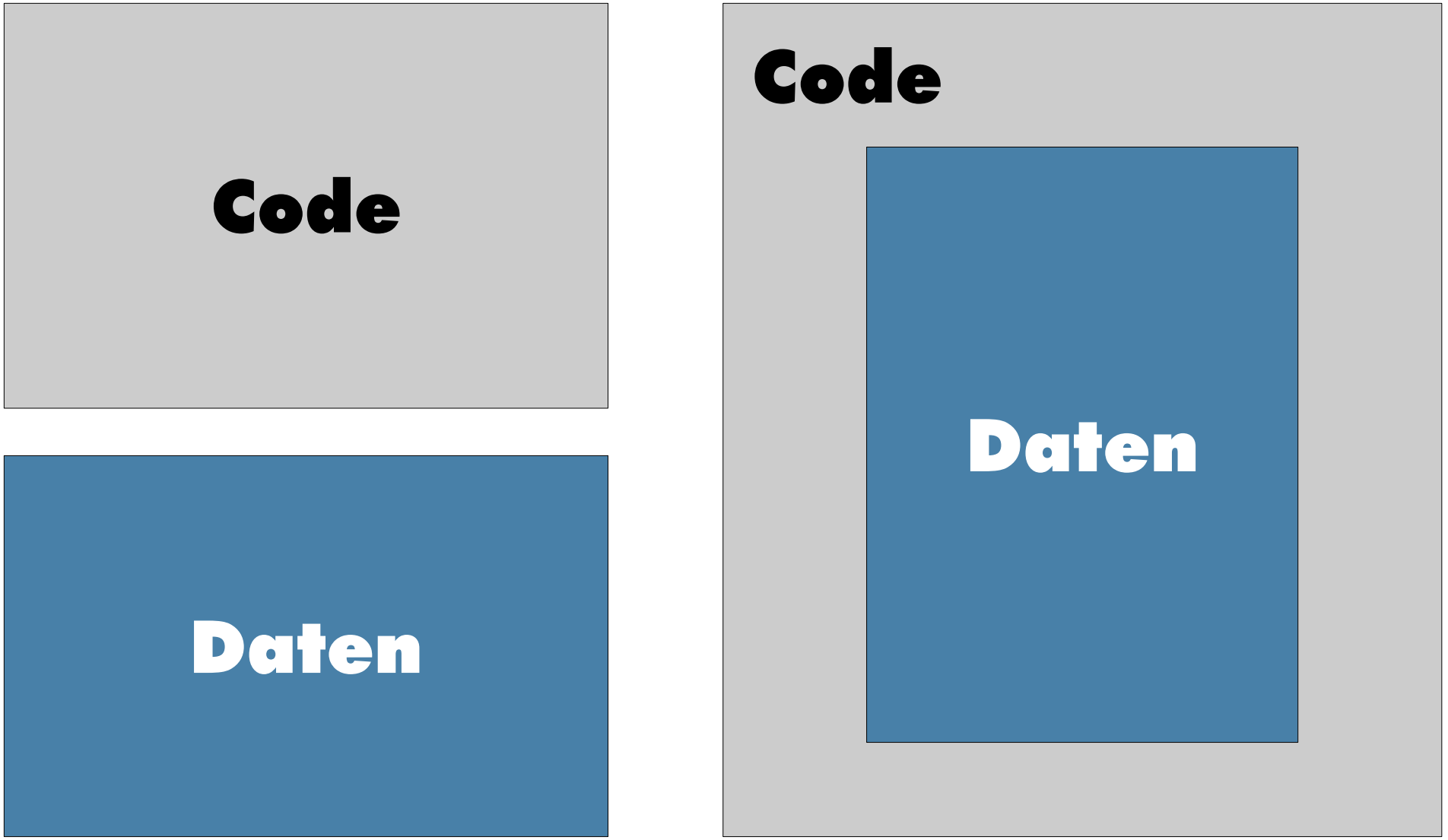

Optic 2000
160
AUDEMARS PIGUET
www.mmc-paris.fr
160
AUDEMARS PIGUET
1600 ZM 51

Attribute

Methoden

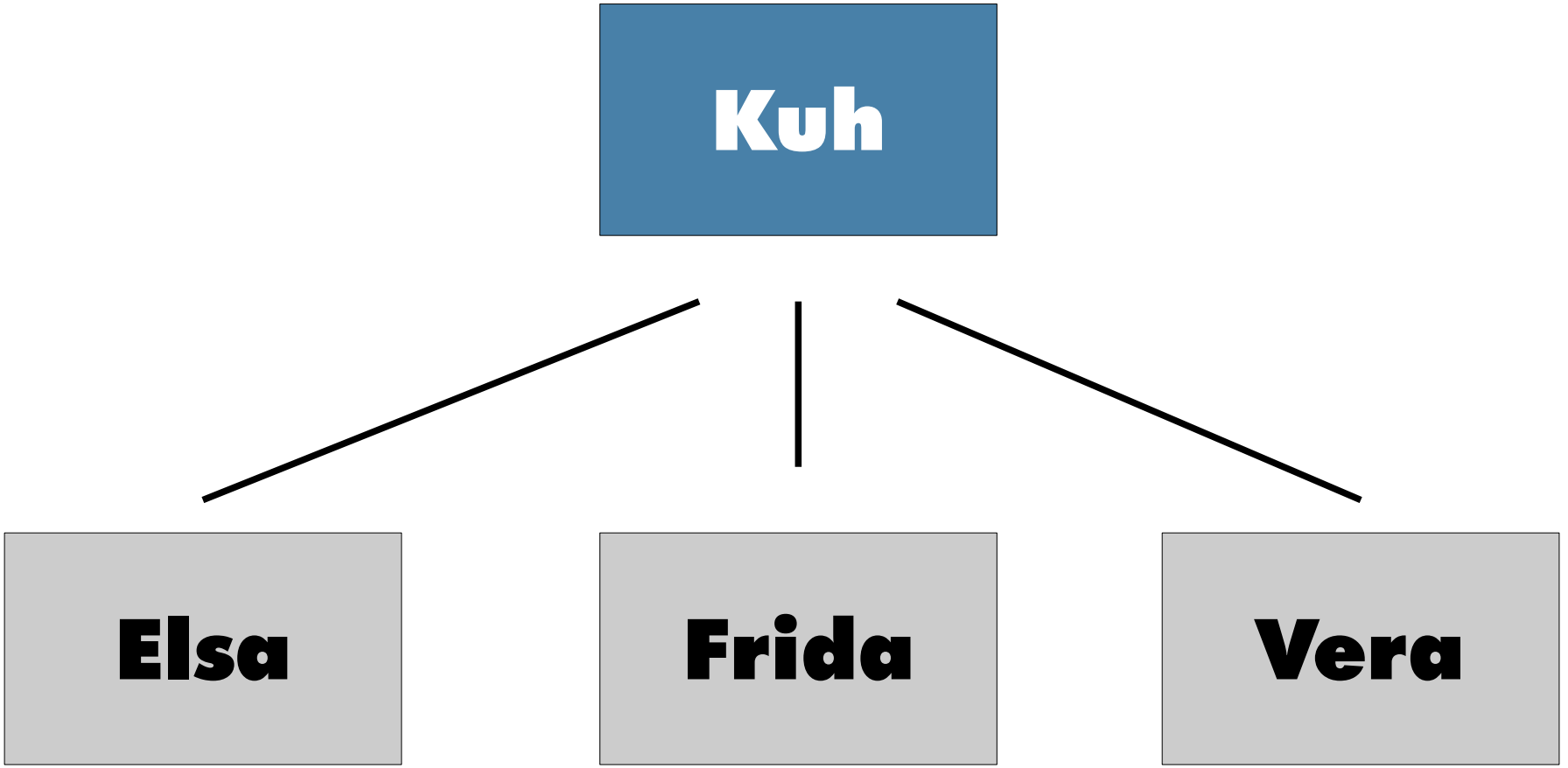
Konstruktoren

Konzepte der Objektorientierung



Kapselung

Abstraktion

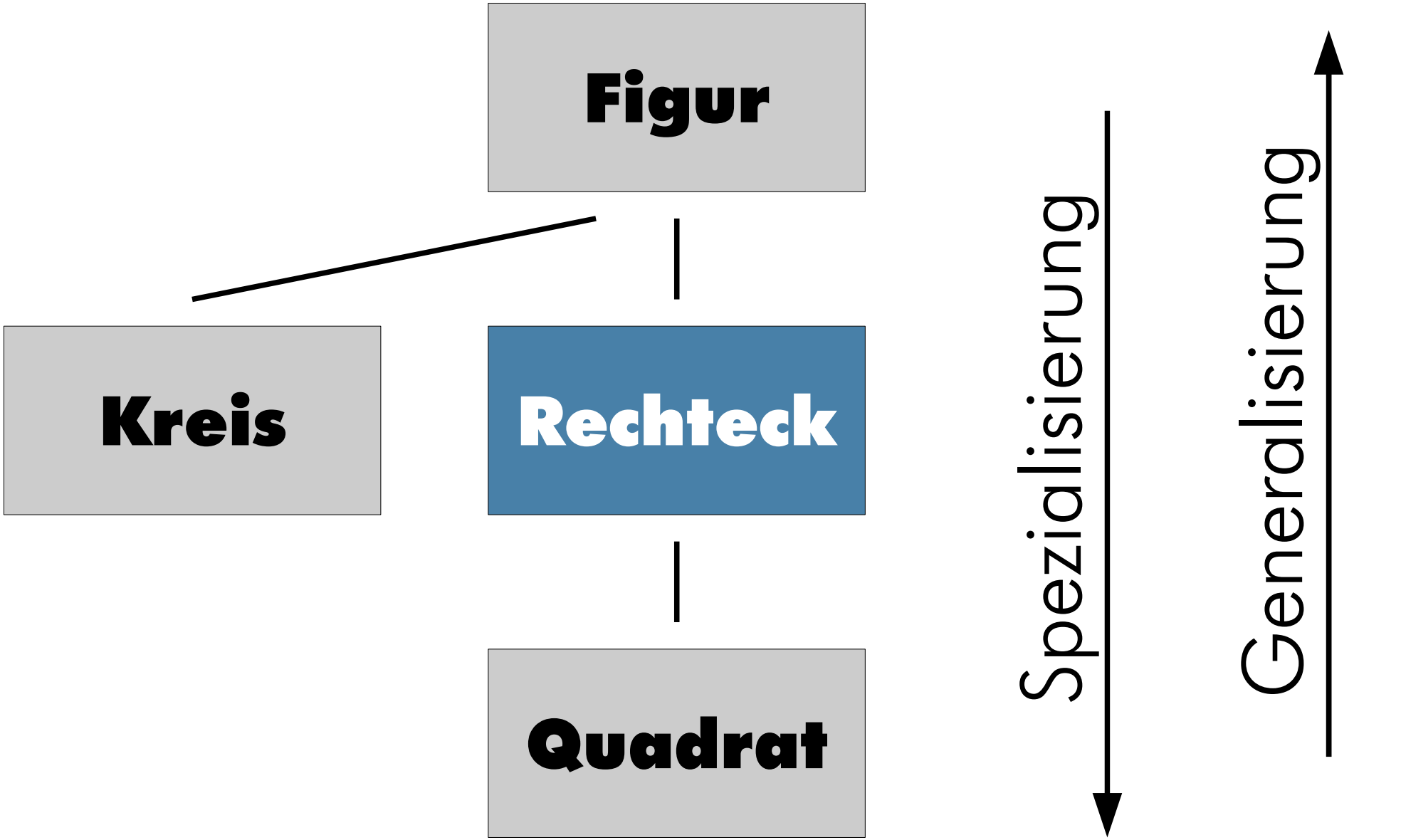


Assoziation

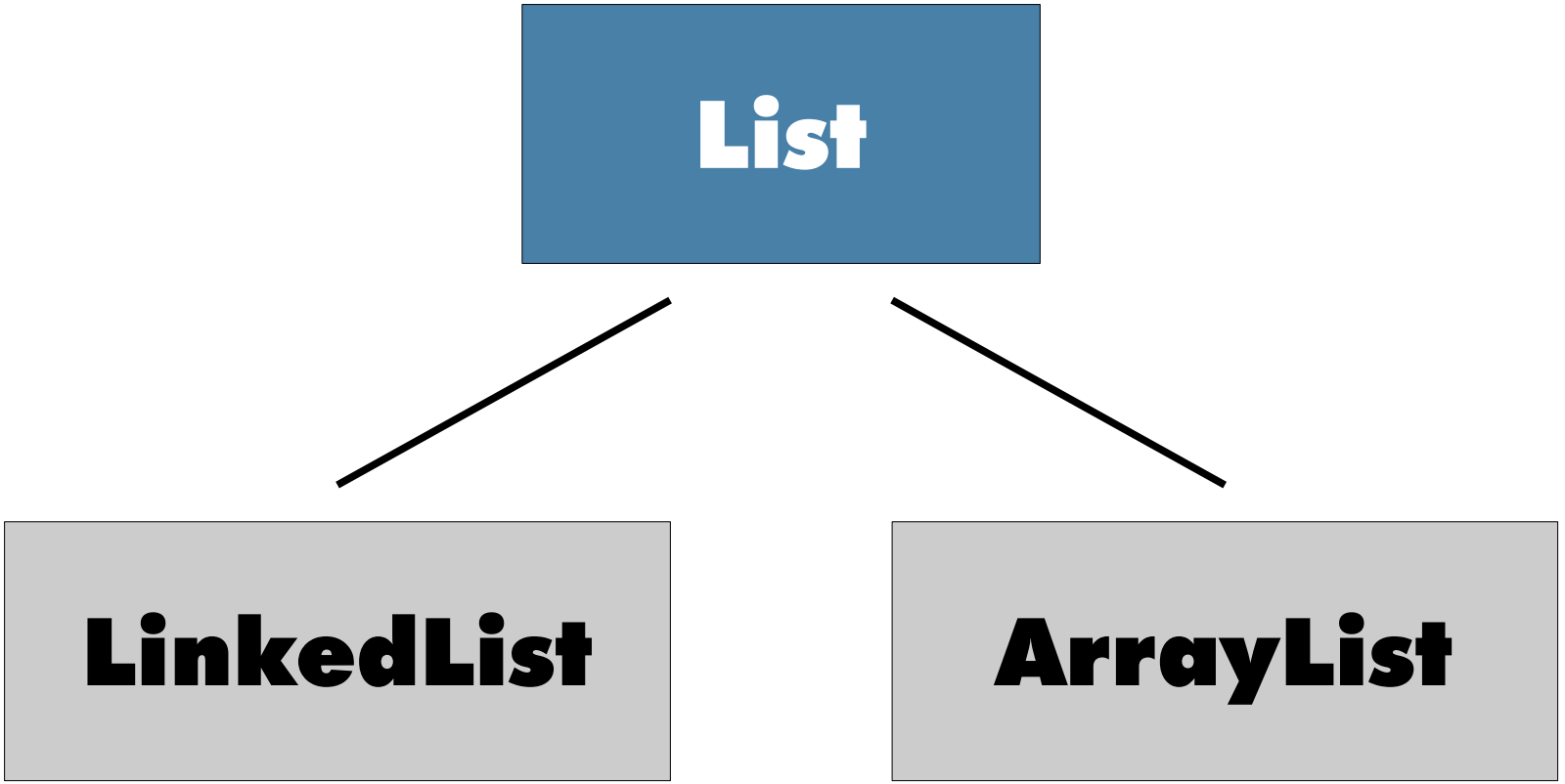
Aggregation

Komposition

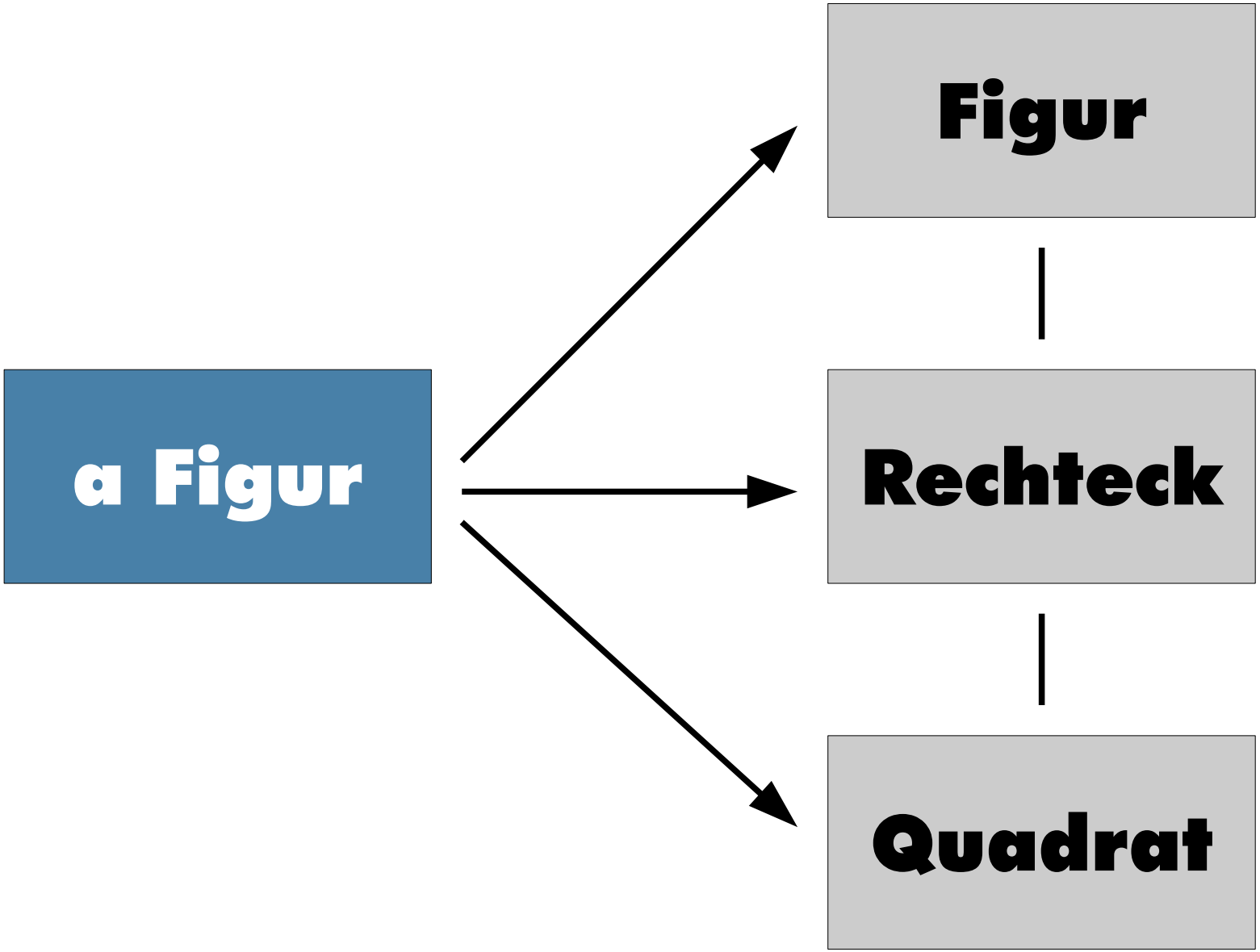
Rechnung besteht aus **Positionen**


Figur
Kreis
Rechteck
Quadrat
Spezialisierung
Generalisierung

Vererbung

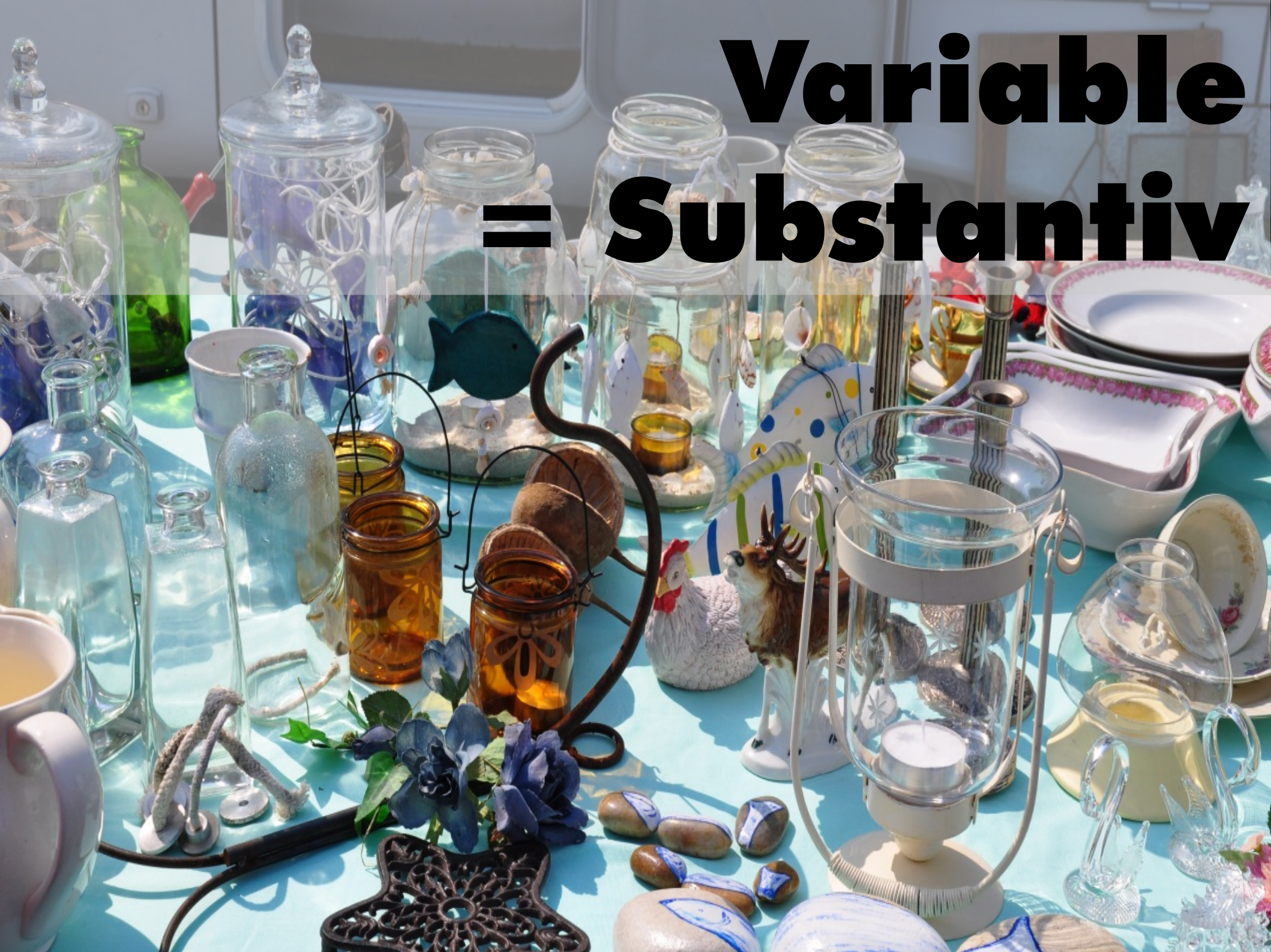


Schnittstelle



Polymorphie

Benennung von Bezeichnern


Variable
= Substantiv


Methode
= Verb

Objektorientierte **Analyse**

Verstehen der **fachlichen Domäne**

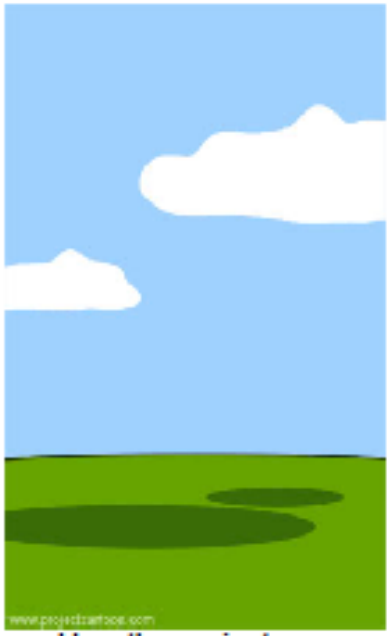
How the customer explained it

How the project leader understood it

How the analyst designed it

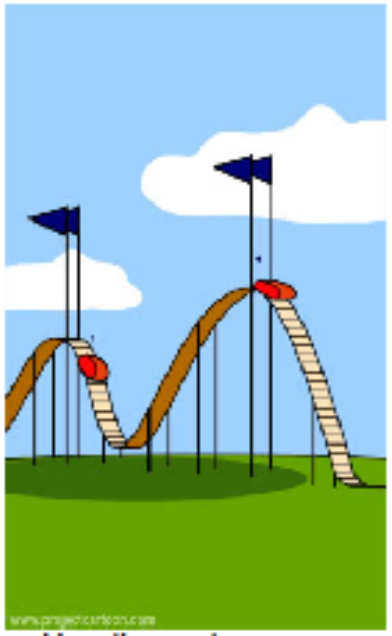
How the programmer wrote it

How the business consultant described it

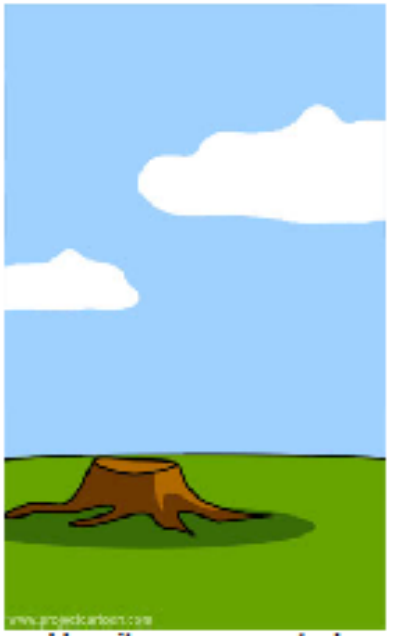


How the project was documented

What operations installed



How the customer was billed



How it was supported

What the customer really needed

Ein **Anwendungsfall** beschreibt eine Menge von **Aktivitäten** eines Systems aus der Sicht seiner **Akteure**, die für die Akteure zu einem wahrnehmbaren **Ergebnis** führen.

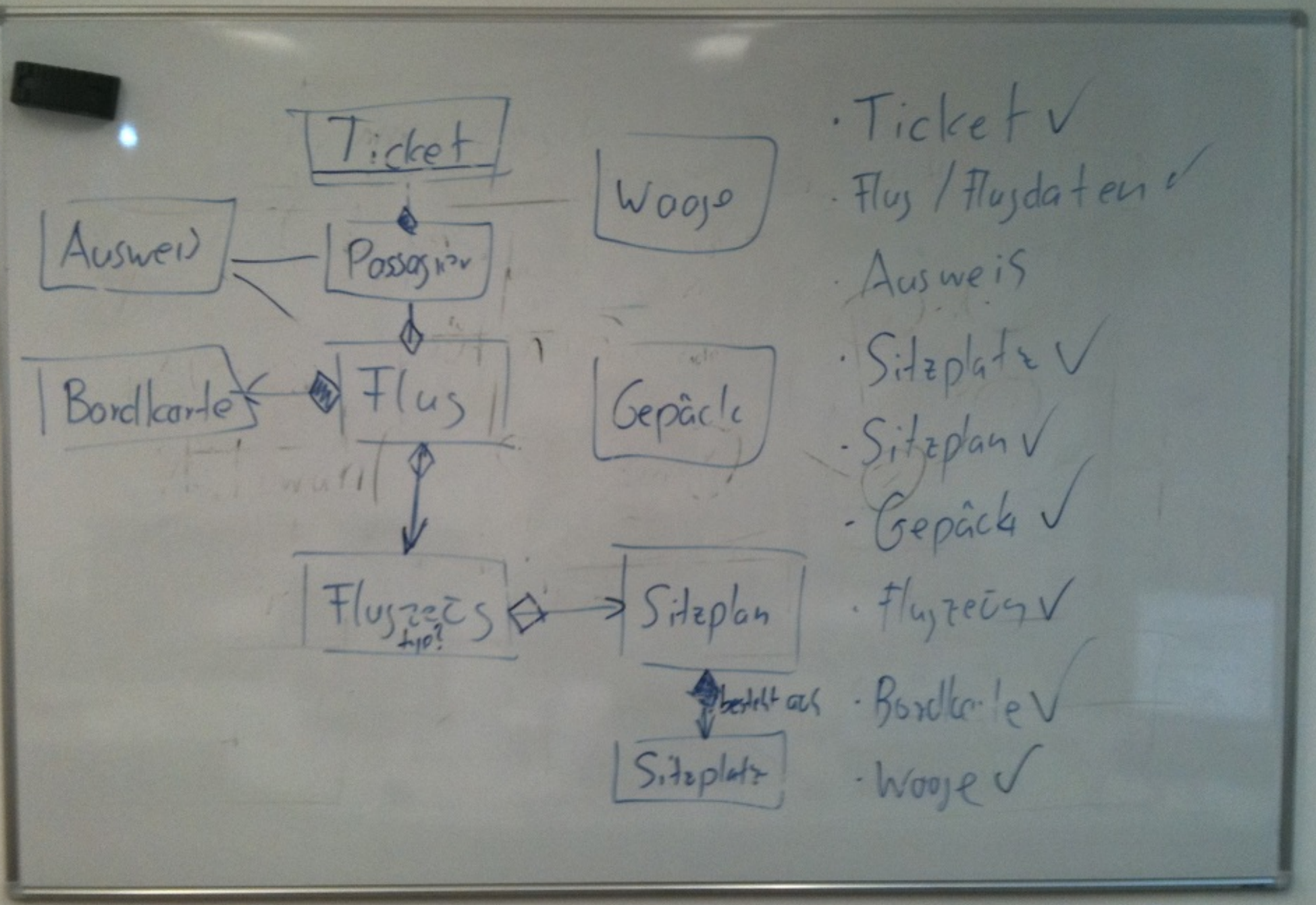
Bernd Oesterreich

Use Cases

Bei der **statischen Analyse** werden die **Geschäftsklassen** und deren **Beziehungen** untereinander identifiziert. Weiterhin findet eine Spezifikation der **Attribute und Operationen** statt.

Sebastian Hempel

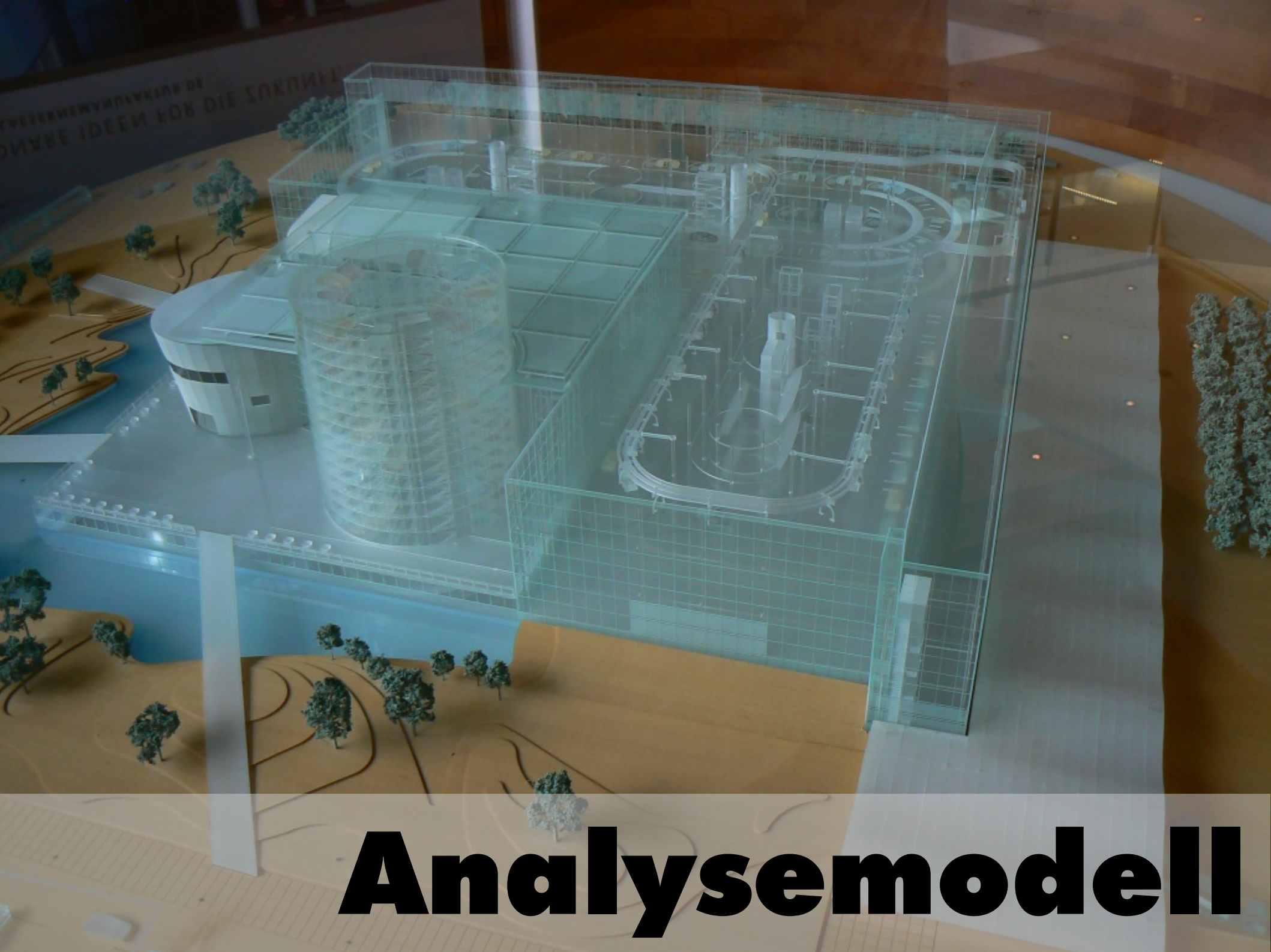
Statische Analyse


Ticket
Ausweis
Passagier
Wooje
Bordkarte
Flug
Gepäck
Flugzeugtyp
Sitzplan
bestellt aus
Sitzplatz
· Ticket ✓
· Flug / Flugdaten ✓
· Ausweis
· Sitzplatz ✓
· Sitzplan ✓
· Gepäck ✓
· Flugzeug ✓
· Bordkarte ✓
· Wooje ✓

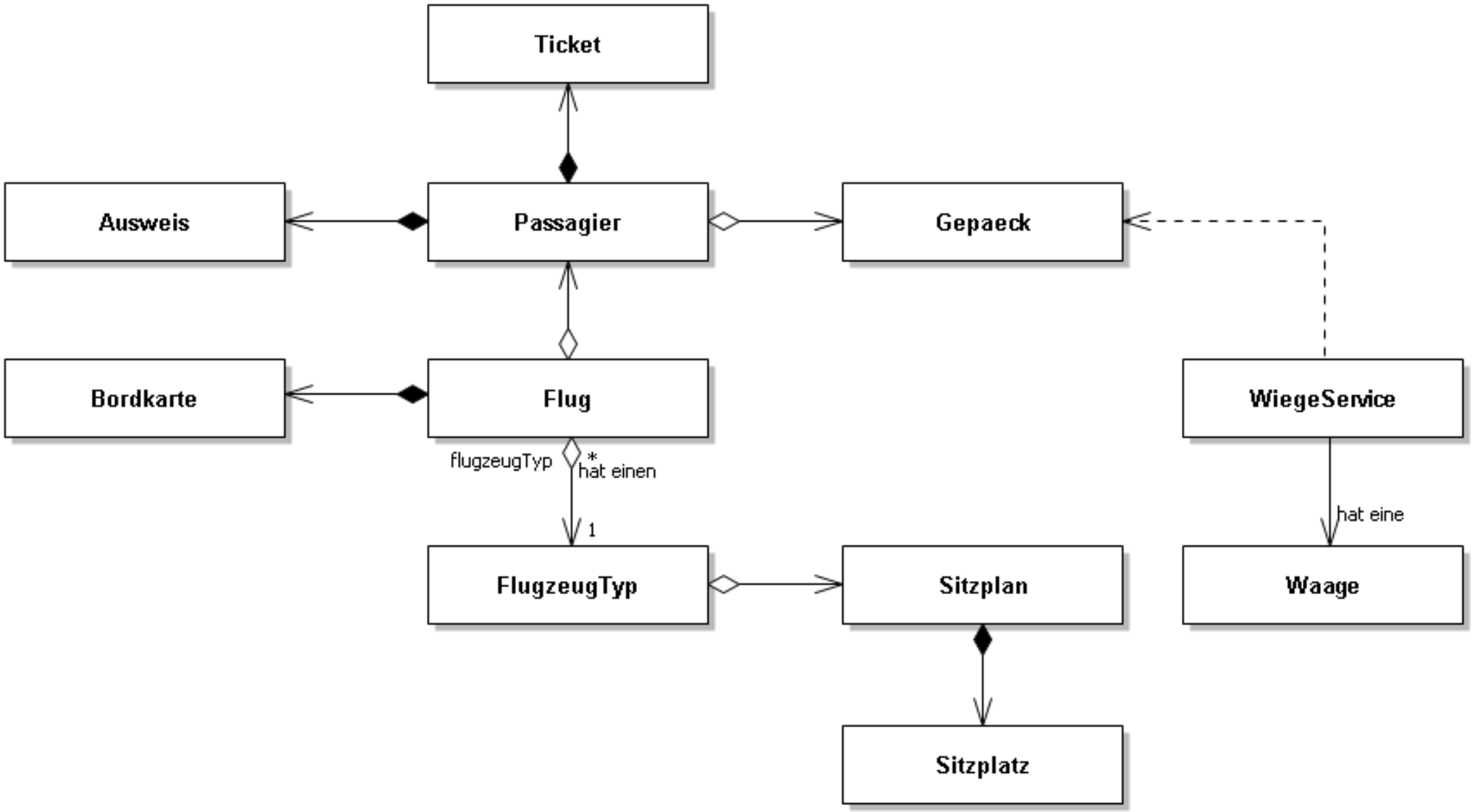
Bei der **dynamischen Analyse** werden die **Interaktionen** der erkannten **Geschäftsklassen** untereinander identifiziert und die dabei verwendeten **Operationen** beschrieben.

Sebastian Hempel

Dynamische Analyse



Analysemodell



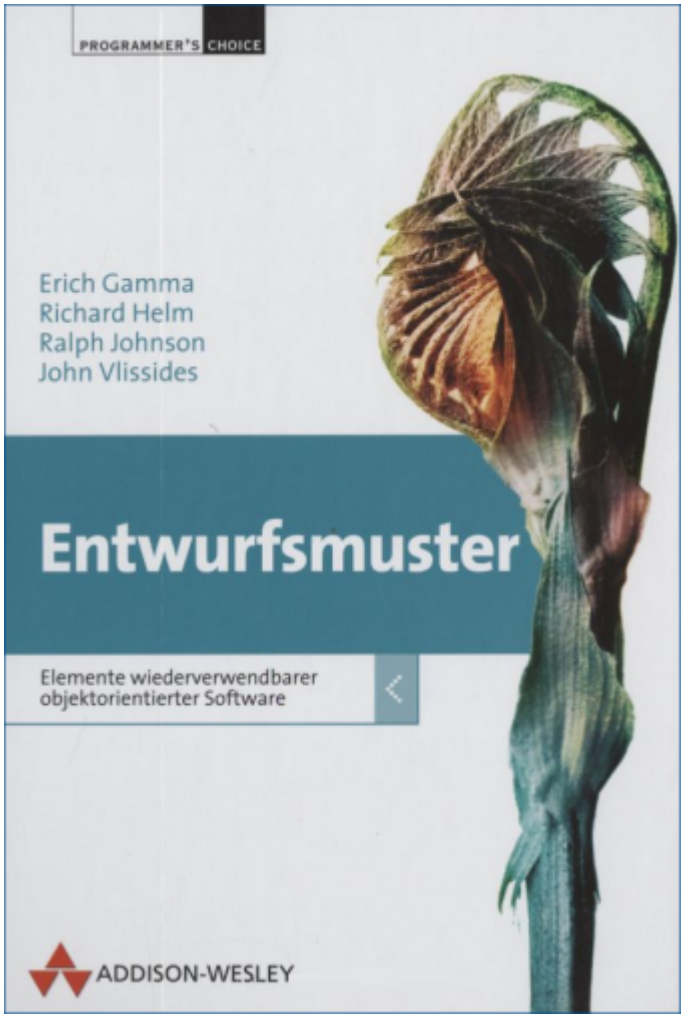
Modell für **Check-In**

Objektorientiertes **Design**

Konkretisierung des Analysemodells mit der Erweiterung um **nicht-fachliche Klassen**. Integration der Klassen in eine **Laufzeitumgebung** / Framework.

Sebastian Hempel

Komponenten

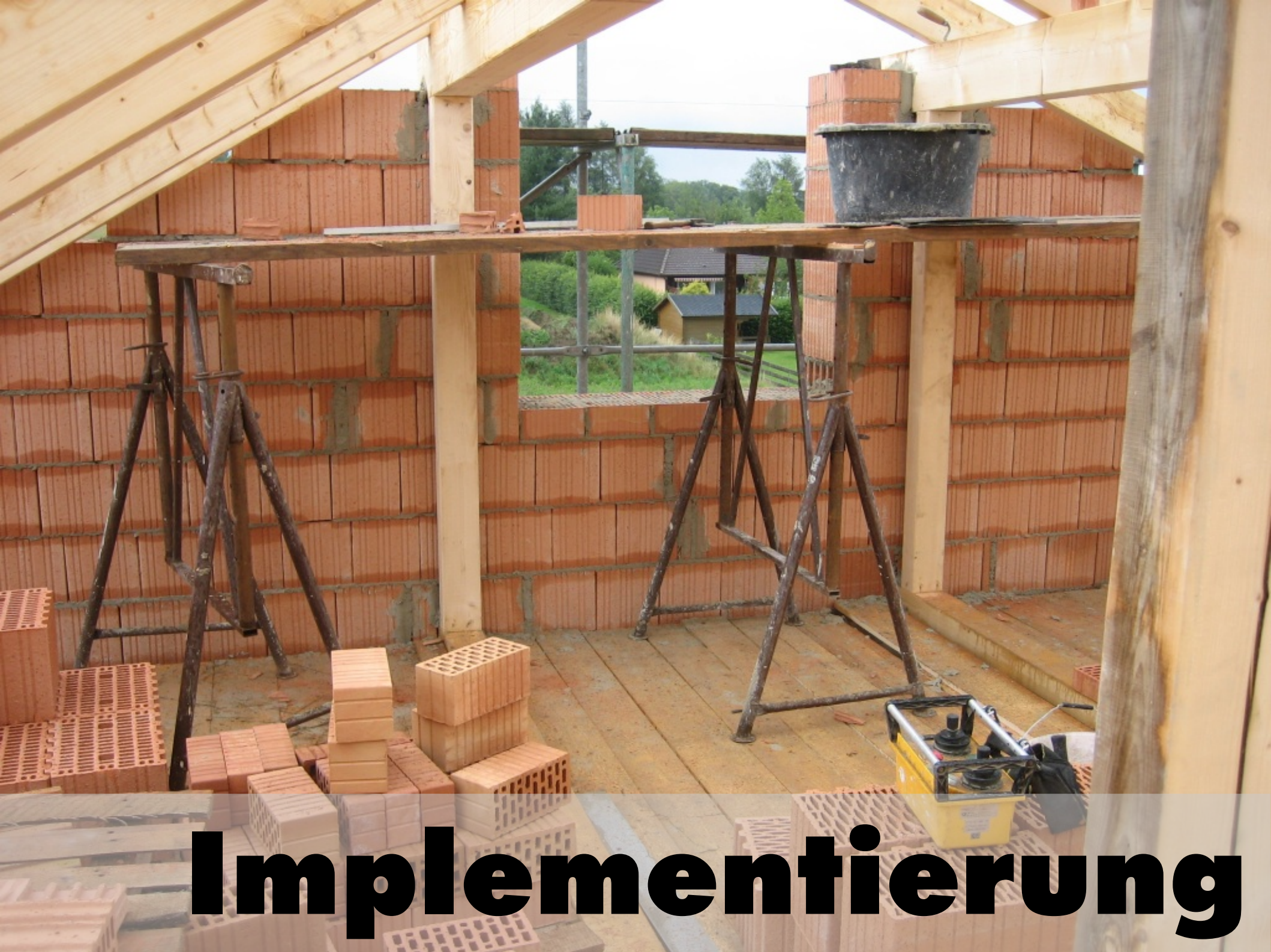

PROGRAMMER'S CHOICE
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Entwurfsmuster
Elemente wiederverwendbarer objektorientierter Software
ADDISON-WESLEY


Ein Buch zum Mitmachen und Verstehen
Entwurfsmuster
von Kopf bis Fuß
Vermeiden Sie Bindungsprobleme
Erfahren Sie, warum sich Ihre Freunde bezüglich des Factory-Musters vor multilich ihren
Laden Sie sich die entscheidenden Muster direkt ins Hirn
Entdecken Sie die Geheimnisse des Pattern-Meisters
Finden Sie heraus, wie Sternbuck-Kaffee mit dem Decorator-Muster seinen Börsenwert verdoppeln konnte
Lesen Sie, wie sich Jörgs Liebesleben verbessert hat, seitdem er das Vererben ein schränkt
O'REILLY

Klassebibliotheken

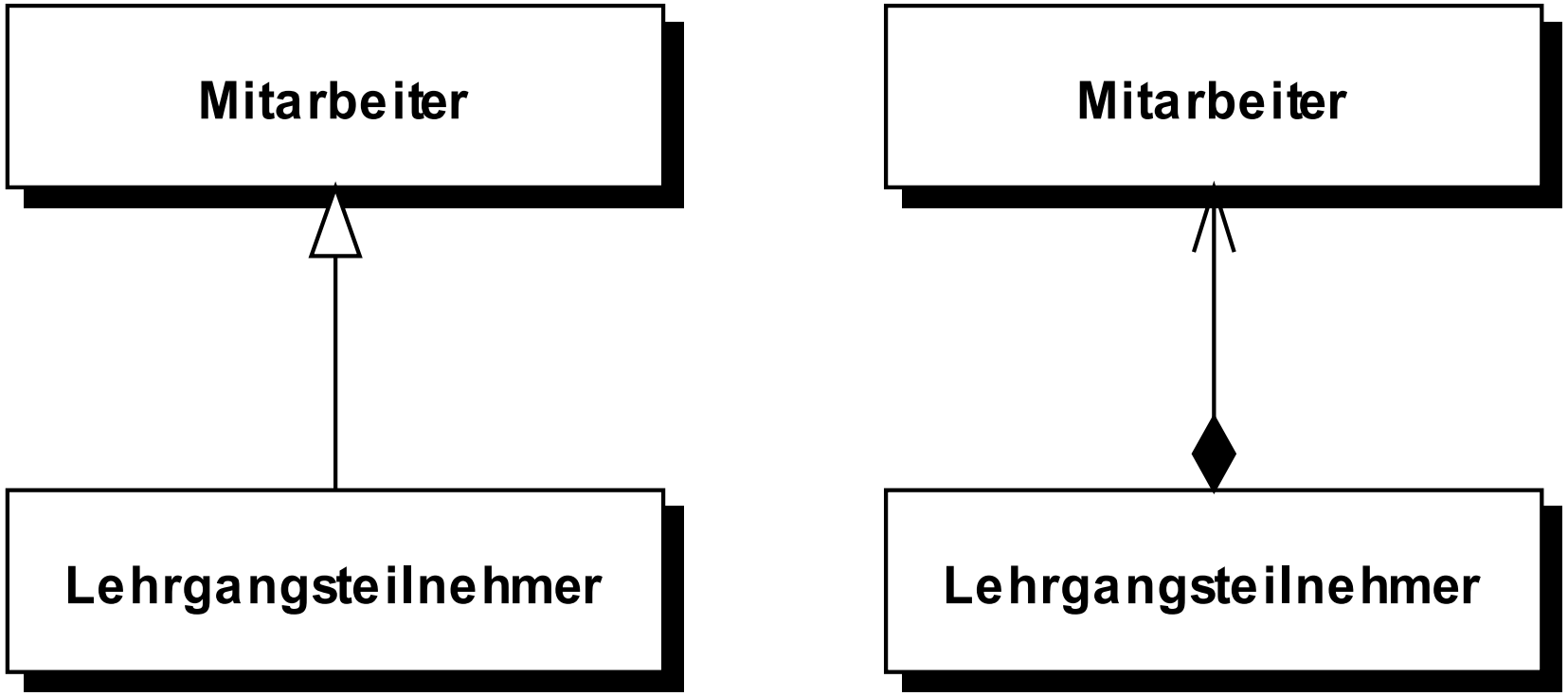
Frameworks

Objektorientierte **Programmierung**

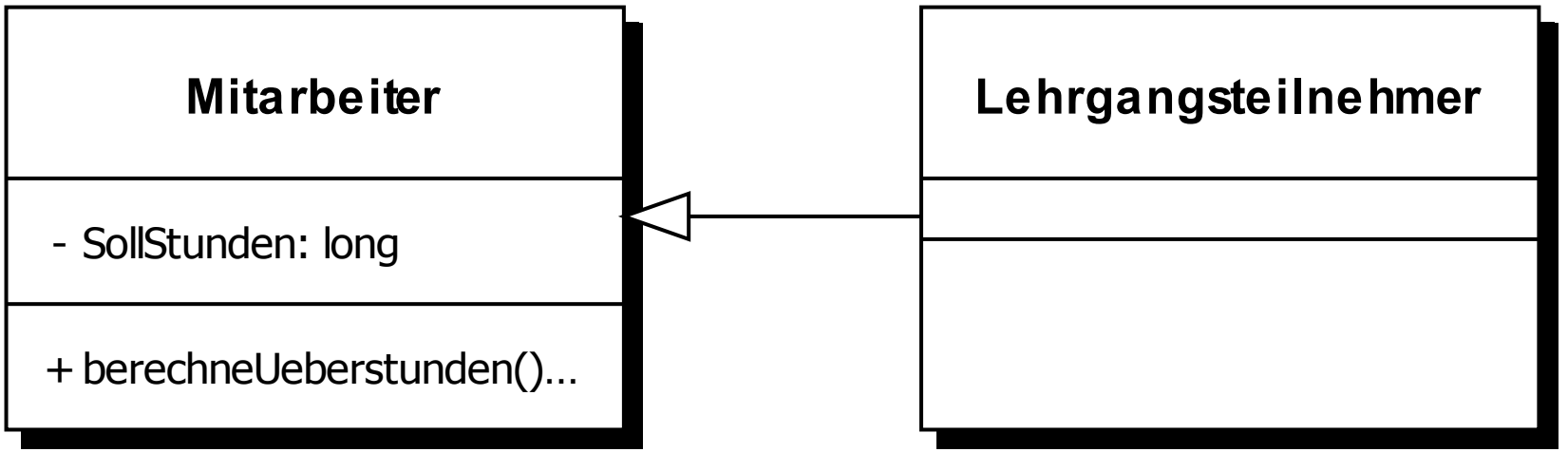


Implementierung

Favour **Composition** over **Inheritance**

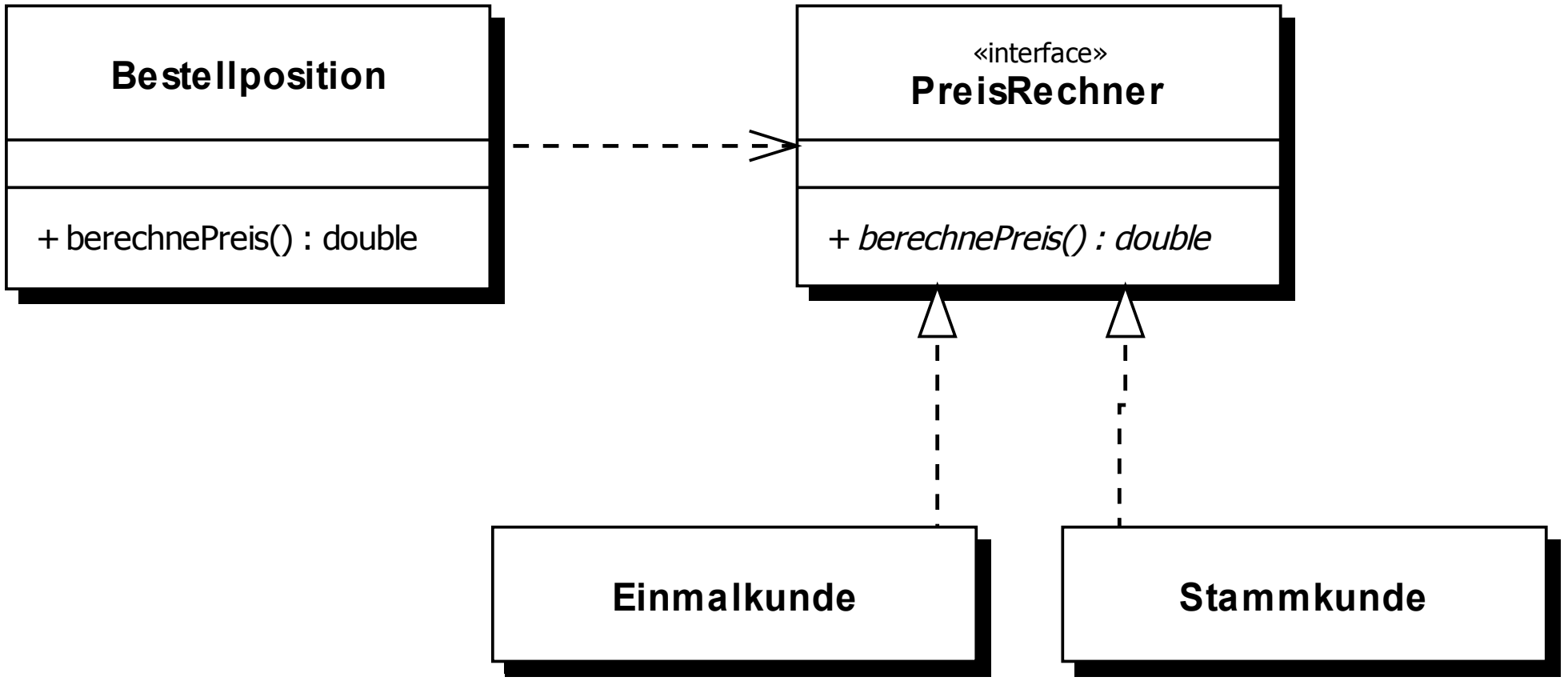


Das **Liskov Substitution Principle** besagt, dass **Subtypen** sich wie ihre **Basistypen** verhalten müssen.


Mitarbeiter
- SollStunden: long
+ berechneUeberstunden()...
Lehrgangsteilnehmer

Das **Open Closed Principle** (OCP) besagt, dass eine Klasse offen für **Erweiterungen** sein muss, jedoch geschlossen gegenüber **Modifikationen**.

```
public double Preis() {
    const decimal StammkundenRabatt = 0.95m;
    switch(kundenart) {
        case Kundenart.Einmalkunde:
            return menge * einzelpreis;
        case Kundenart.Stammkunde:
     return menge * einzelpreis * StammkundenRabatt;
        default:
            throw new ArgumentOutOfRangeException();
    }
}
```


Bestellposition
+ berechnePreis() : double
«interface»
PreisRechner
+ berechnePreis() : double
Einmalkunde
Stammkunde

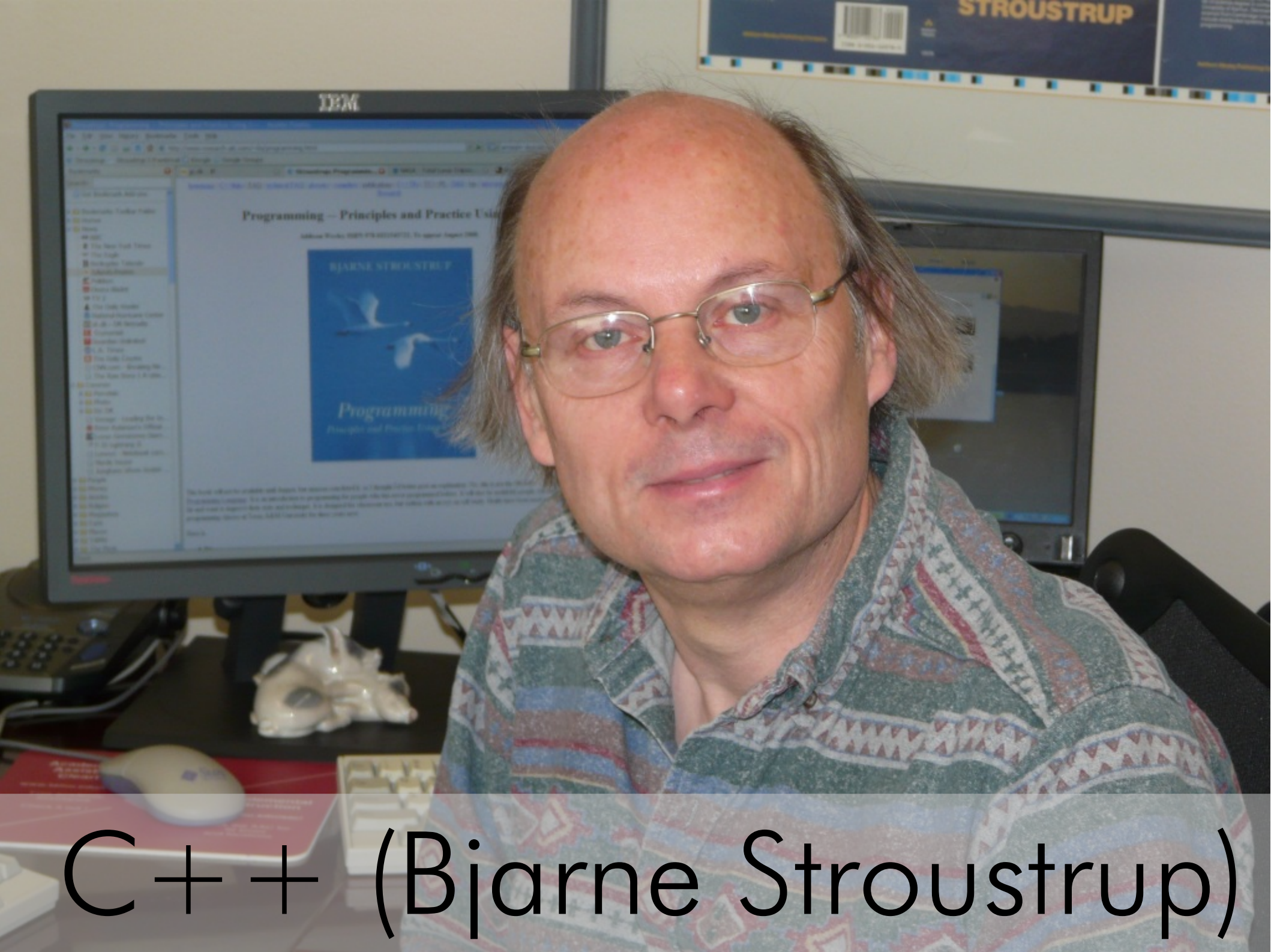
Interface Segregation bedeutet Schnittstellen klein zu halten. Teilen Sie nicht zusammengehörende **Methoden auf mehrere Schnittstellen** auf.

Nach dem **Law of Demeter** soll eine Methode nur folgende andere Methoden verwenden:

- Methoden der eigenen Klasse
- Methoden der Parameter
- Methoden assoziierter Klassen
- Methoden selbst erzeugter Objekte

Bei der **Dependency Inversion** wird darauf geachtet, dass Klassen **Abhängigkeit zu Schnittstellen** und nicht zu anderen (Low-Level) Klassen haben. Dies verbessert die Testbarkeit.

Objektorientierte Programmmier-**Sprachen**



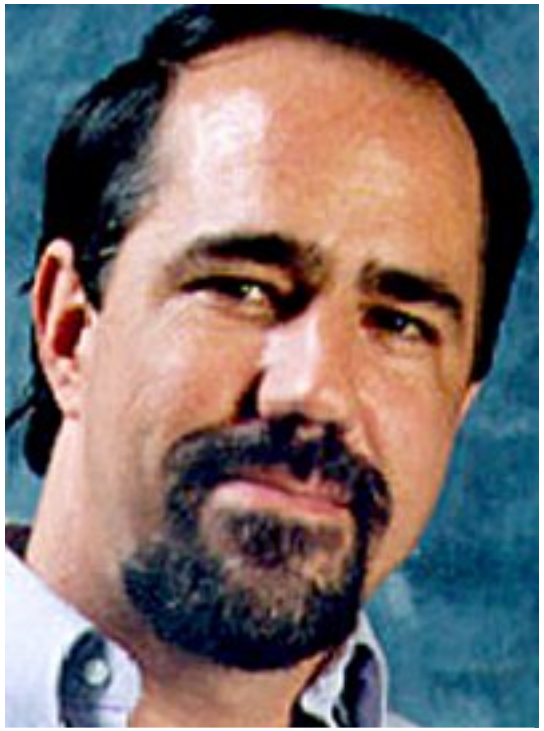
C++ (Bjarne Stroustrup)

Java (James Gosling)


C# (Anders Hejlsberg)

UML (unified modelling language)

The **Three Amigos**



Grady Booch

Ivar Jacobson

James Rumbaugh

Anwendungsfalldiagram

Klassendiagram

Verhaltendiagramme

Implementierungsdiagramme

Mit dem **Anwendungsfall**-diagramm wird beschrieben was das System leisten soll.

Ein
Klassendiagramm
besteht aus Klassen,
Attributen, Methode
und Interfaces.

Ein
Klassendiagramm
beschreibt die
Beziehungen von
Klassen.

Das
Aktivitätsdiagramm
beschreibt einen Ablauf
in einem System.

Das **Kollaborationsdiagramm** beschreibt Interaktionen zwischen Objekten.

Mit dem
Sequenzdiagramm
wird der Austausch von Nachrichten von Objekten im zeitlichen Zusammenhang beschrieben.

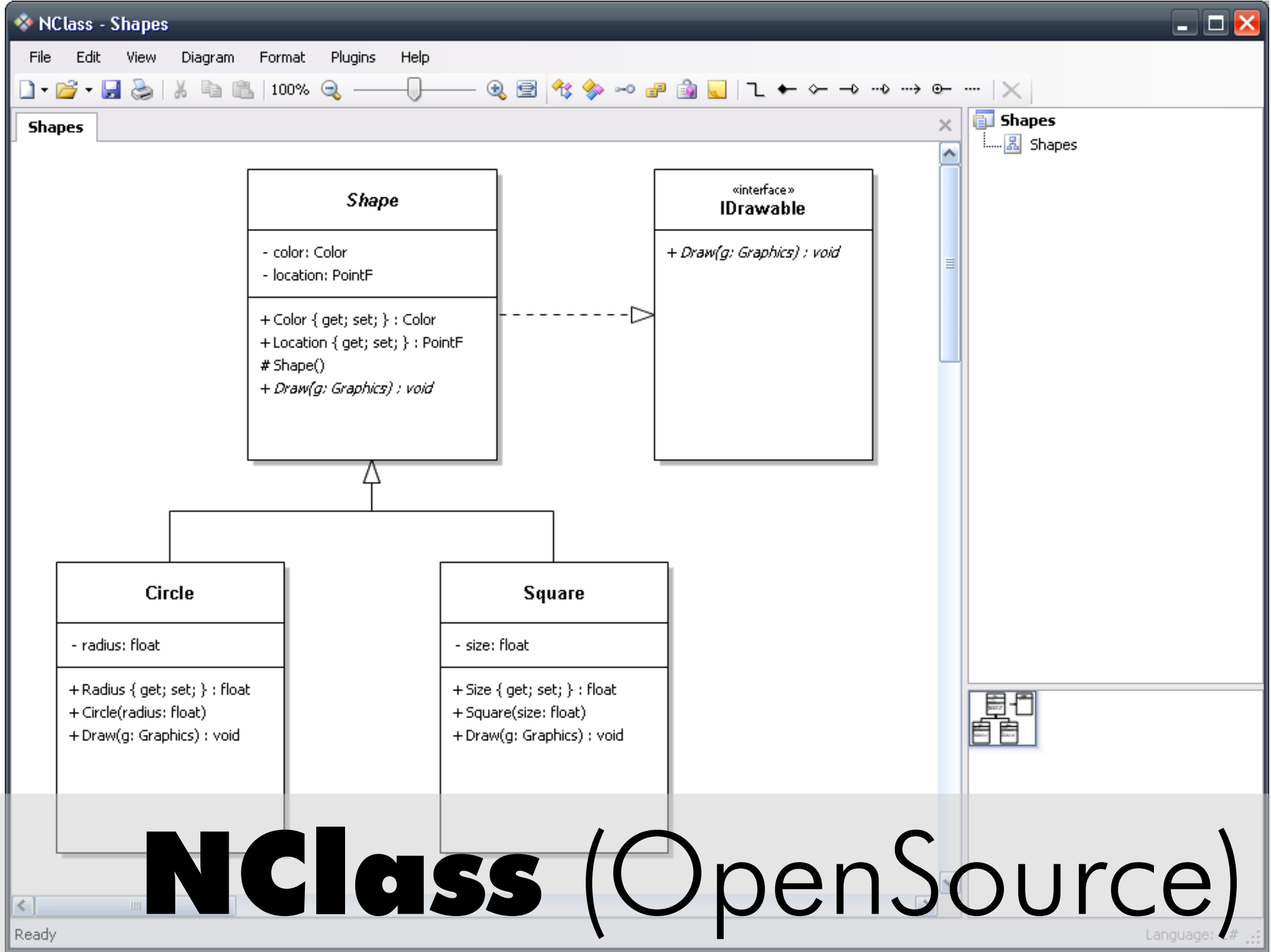
CASE

Computer

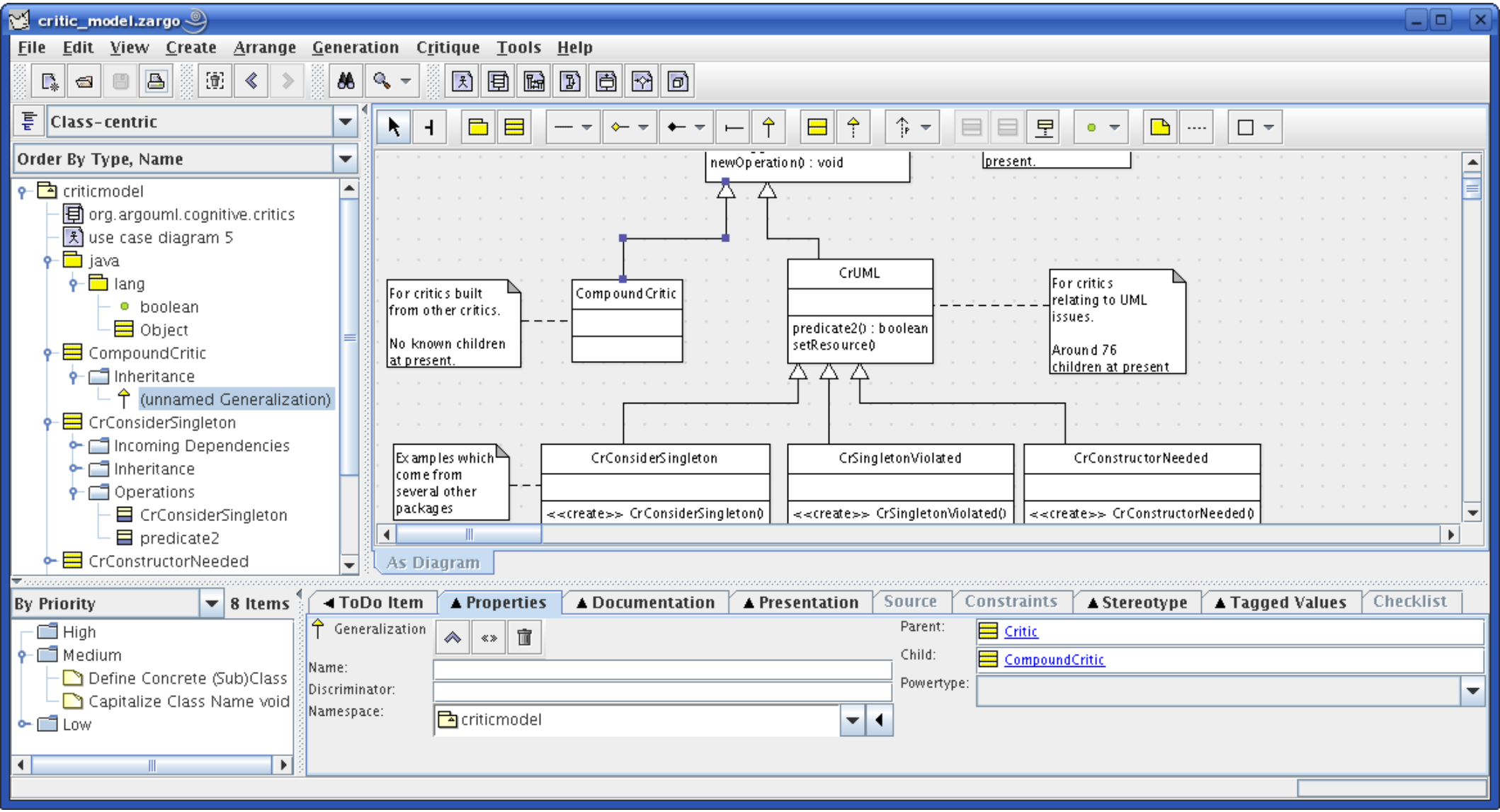
Aided

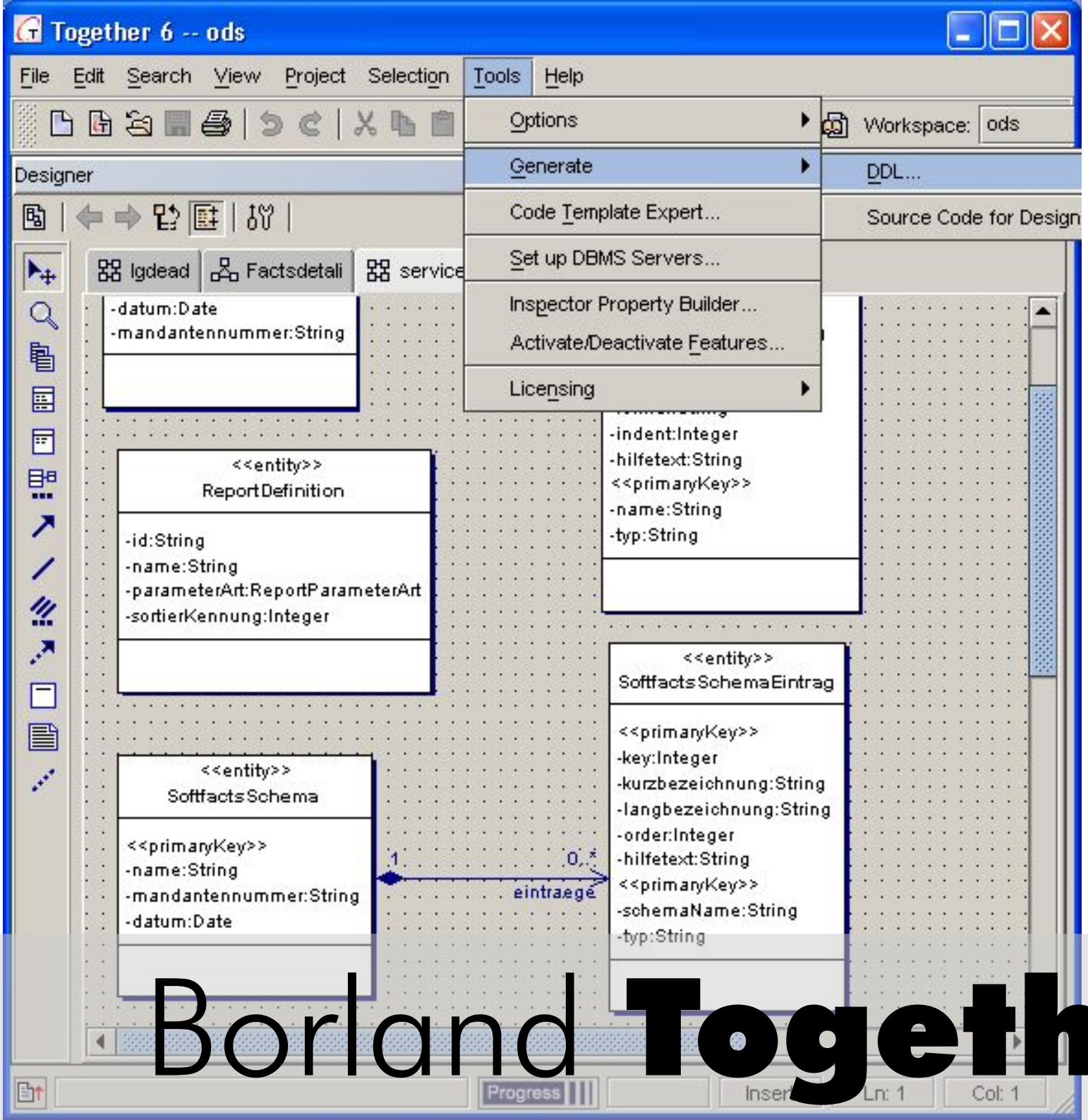
Software

Engineering

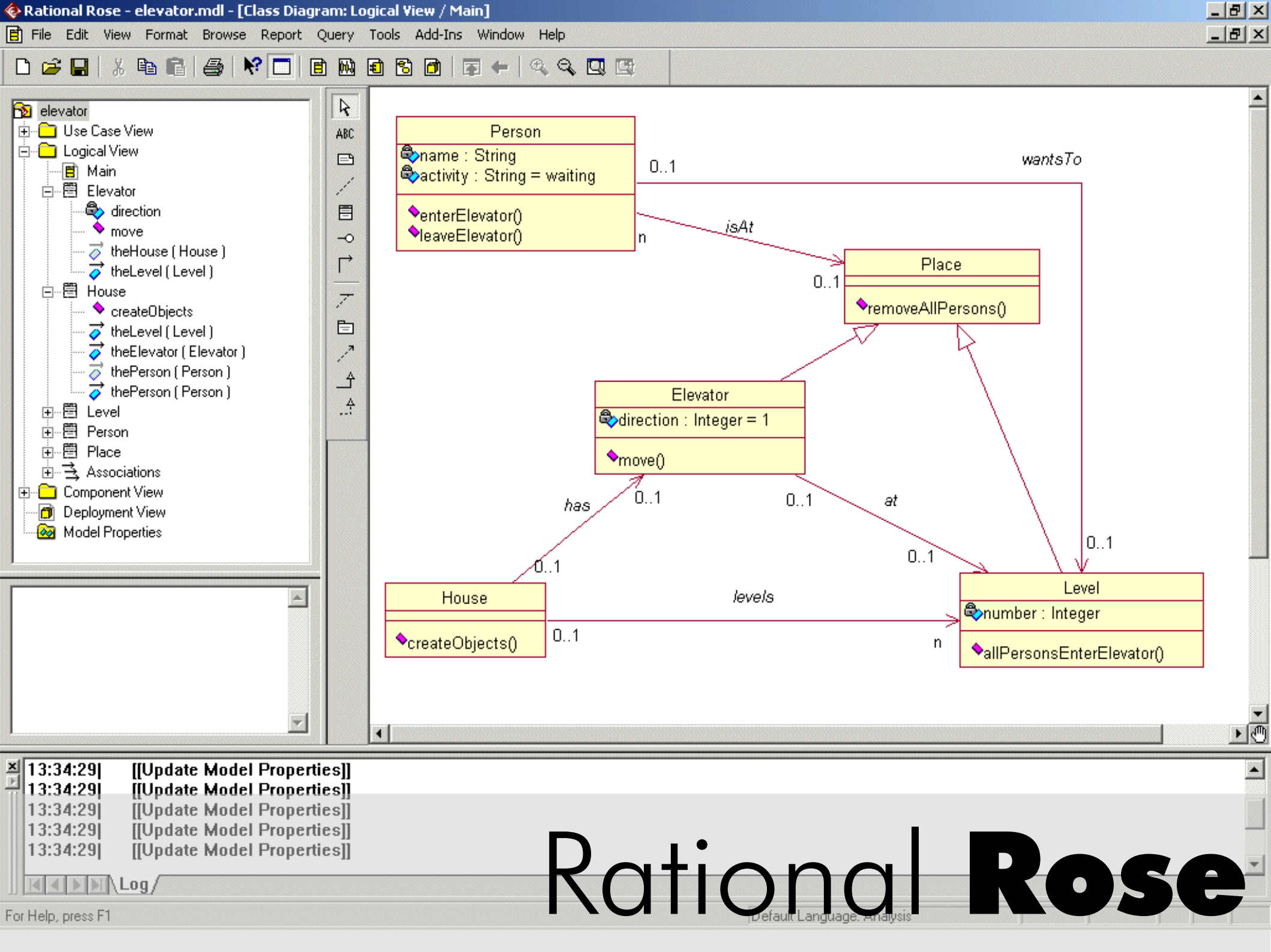

NClass - Shapes
File
Edit
View
Diagram
Format
Plugins
Help
100%
Shapes
Shape
- color: Color
- location: PointF
+ Color { get; set; } : Color
+ Location { get; set; } : PointF
# Shape()
+ Draw(g: Graphics) : void
«interface»
IDrawable
+ Draw(g: Graphics) : void
Circle
- radius: float
+ Radius { get; set; } : float
+ Circle(radius: float)
+ Draw(g: Graphics) : void
Square
- size: float
+ Size { get; set; } : float
+ Square(size: float)
+ Draw(g: Graphics) : void
Shapes
Shapes
Ready
NClass (OpenSource)

ArgoUML (OS)


critic_model.zargo
File Edit View Create Arrange Generation Critique Tools Help
Class-centric
Order By Type, Name
criticmodel
org.argouml.cognitive.critics
use case diagram 5
java
lang
boolean
Object
CompoundCritic
Inheritance
(unnamed Generalization)
CrConsiderSingleton
Incoming Dependencies
Inheritance
Operations
CrConsiderSingleton
predicate2
CrConstructorNeeded
newOperation() : void
present.
For critics built from other critics.
No known children at present.
CompoundCritic
CrUML
predicate2() : boolean
setResource()
For critics relating to UML issues.
Around 76 children at present
Examples which come from several other packages
CrConsiderSingleton
<<create>> CrConsiderSingleton()
CrSingletonViolated
<<create>> CrSingletonViolated()
CrConstructorNeeded
<<create>> CrConstructorNeeded()
As Diagram
By Priority
8 Items
High
Medium
Define Concrete (Sub)Class
Capitalize Class Name void
Low
ToDo Item
Properties
Documentation
Presentation
Source
Constraints
Stereotype
Tagged Values
Checklist
Generalization
Name:
Discriminator:
Namespace:
criticmodel
Parent:
Critic
Child:
CompoundCritic
Powertype:



Borland **Together**


Rational Rose - elevator.mdl - [Class Diagram: Logical View / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
elevator
Use Case View
Logical View
Main
Elevator
direction
move
theHouse (House)
theLevel (Level)
House
createObjects
theLevel (Level)
theElevator (Elevator)
thePerson (Person)
thePerson (Person)
Level
Person
Place
Associations
Component View
Deployment View
Model Properties
Person
name : String
activity : String = waiting
enterElevator()
leaveElevator()
0..1
wantsTo
isAt
n
Place
removeAllPersons()
0..1
Elevator
direction : Integer = 1
move()
has
0..1
0..1
at
0..1
0..1
0..1
House
createObjects()
levels
0..1
n
Level
number : Integer
allPersonsEnterElevator()
13:34:29| [[Update Model Properties]]
13:34:29| [[Update Model Properties]]
13:34:29| [[Update Model Properties]]
13:34:29| [[Update Model Properties]]
13:34:29| [[Update Model Properties]]
Log
For Help, press F1
Default Language: Analysis
Rational Rose

Bildnachweis

1. Modellbahn - Elbbrücken / Pieter & Renée Lanser / Flickr
4. Krka Wasserfälle / Karl-Hermann Loges / Flickr
5. The Zombie Attack project plan / Jez Nicholson / Flickr
7. Old couple analyzing Railway timetable / Pedro Ribeiro Simões / Flickr
8. Sketching a System / Ben and Kaz Askins / Flickr
9. Pat on his Mac / Kevin Galens / Flickr
11. Minmax_Notation_Mannschaft_Spieler / PhilippWeissenbacher / Wikipedia
15. Waffle Recipe / Brian Kelley / Flickr
16. Light Waffeln / sushiina / Flickr
17. ALFA ROMEO Giulia 1964 / Will Will / Flickr
18. Alfa Romeo Giulia Sprint GT RUD66 / Flickr
19. Ich liebe Waffeln! / Tine Steiss Flickr
30. Flohmarkt / cbronziski / Flickr
30. The Point Is im Studio der SAE Leipzig / Andre Lademann / Flickr
33. J&W Autos - Mechanic At Work 2 / Emyr Jones / Flickr
36. Self check-in at BOS / Karl Baron / Flickr
38. Everyone getting in on macroinvertebrate identification / External Affairs / Flickr
41. Marius Zierold / Marius Zierold / Flickr
42. Dresden bei VW / Dierk Schäfer / Flickr
46. highend-96 / noeffred / Flickr
48. Dombücherei Linz, 3 / Monika Bargmann / Flickr
49. Framework / jphilipg / Flickr
64. Anders Hejlsberg / D. Begley / Flickr

Sebastian Hempel

Staatlich geprüfter
Informatiker

Selbständiger Software-
Entwickler und Dozent
in den Bereichen Java
und OpenSource

Clean Code Developer

shempel@it-hempel
http://www.it-hempel.de/

cc BY NC SA