

Teil II

Objektorientierte Programmierung (OOP)

20. Objektnetze

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
Technische Universität Dresden
Version 17-0.2, 28.04.12

- ▶ 21) Verfeinern von Assoziationen mit dem Java-2 Collection Framework
- ▶ 22) Netze mit Datenfluss: Iteratoren, Kanäle
- ▶ 23) Graphen in Java
- ▶ 25) Entwurfsmuster



Obligatorische Literatur

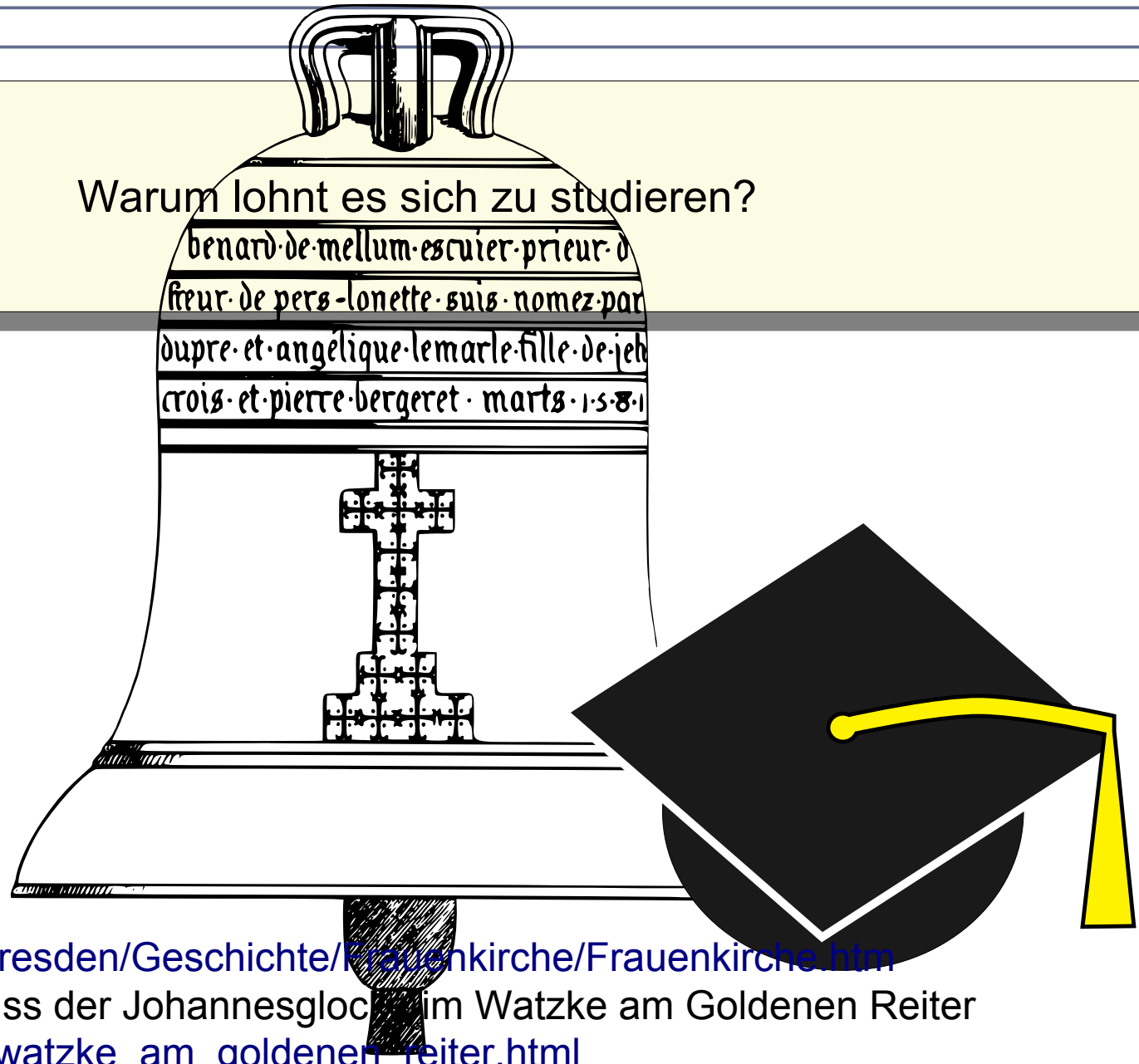
2 Softwaretechnologie (ST)

- ▶ JDK Tutorial für J2SE oder J2EE, www.java.sun.com

Das Ziel des Studiums: Als Meister handeln können

Warum lohnt es sich zu studieren?

benard·de·mellum·escuier·prieur·d
feur·de·pers·lonette·suis·nomez·par
dupre·et·angélique·lemarle·fille·de·jeh
crois·et·pierre·bergeret·marts·1·5·8·1



<http://www.kprdd.de/Dresden/Geschichte/Frauenkirche/Frauenkirche.htm>

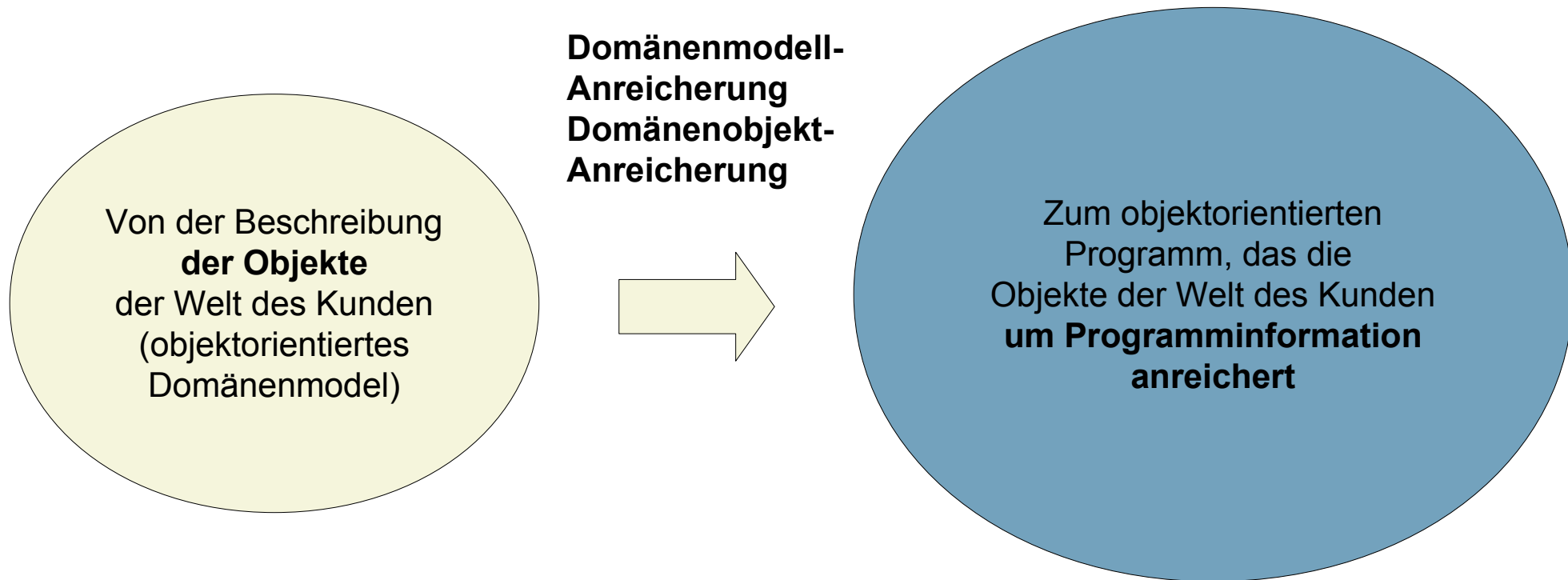
Fehltönender Probeguss der Johannesglocke im Watzke am Goldenen Reiter

http://www.watzke.de/watzke_am_goldenen_reiter.html

Geschichte: <http://www.md-pro.de/depot/mitteilungen/mit0401.pdf>

Die zentralen Fragen des objektorientierten Ansatzes

Wie kommen wir vom Problem des Kunden zum Programm (oder Produkt)?



Anreicherung/Verfettung: Anreicherung durch technische Programminformation
„object fattening“: Anreicherung von Objekten des Domänenmodells

Ziel von Teil II der Vorlesung

- ▶ Aufbau von Objektnetzen (Graphen, Dags, Bäumen, Listen) durch Datenverfeinerung von Assoziationen
 - Graphen, Iteratormethoden, Iteratoren, und Streams
 - Große Objekte (Bobs) mit internen Netzen
 - Endo- und Exoassoziationen
 - Wie man Graphen erweitert

Q4: Softwareentwicklung im V-Modell

[Boehm 1979]

Testfälle

Analyse

Abnahmetest

OOA

Grobentwurf

Testfälle

Systemtest

OOD

Feinentwurf

Testfälle

Integrationstest

Verfeinerung von Assoziationen

Netzimplementierung

Teamimplementierung

Teamtest

Graphimplementierung

Graphtest

Netztest

OOP

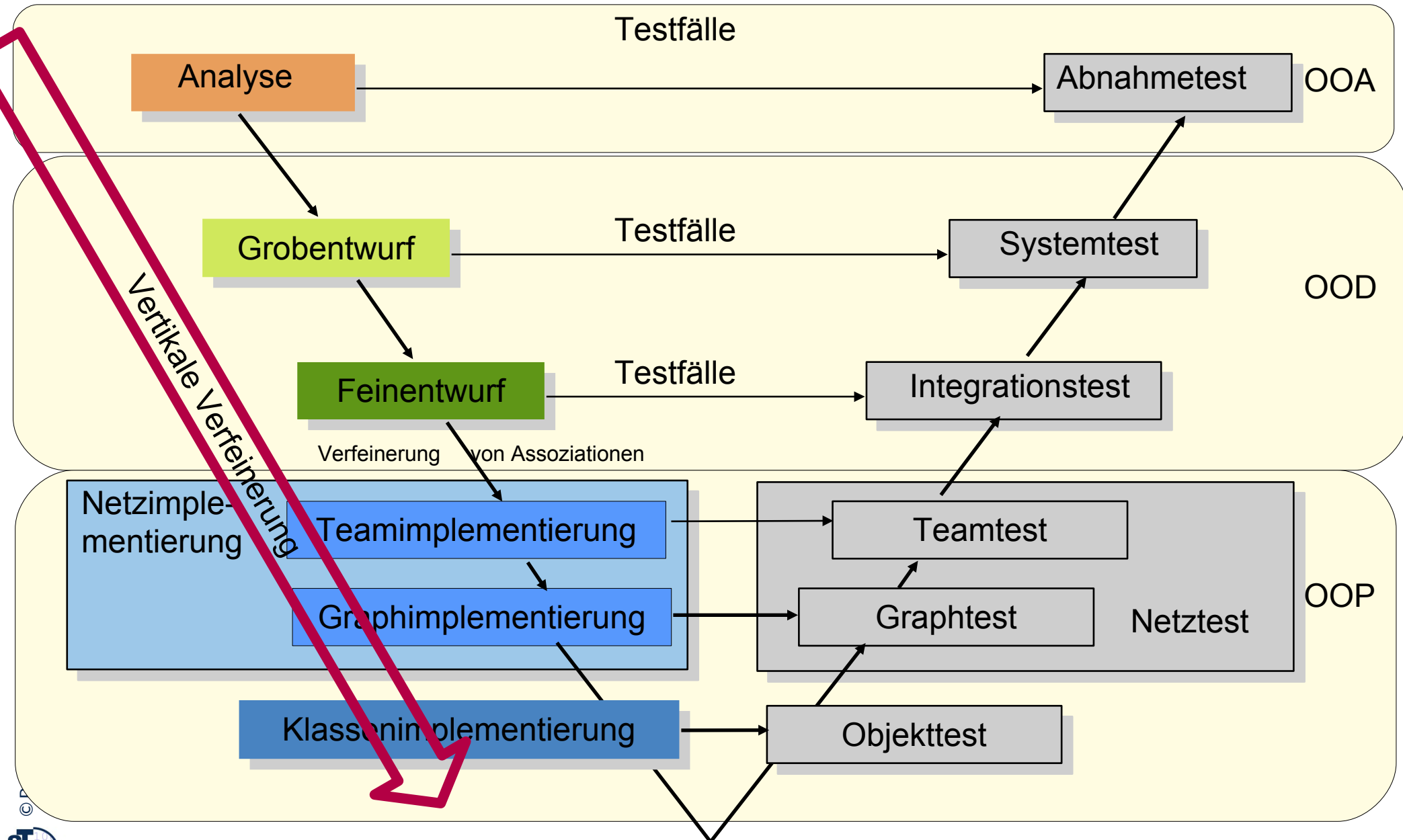
Klassenimplementierung

Objekttest



Q4: Softwareentwicklung im V-Modell

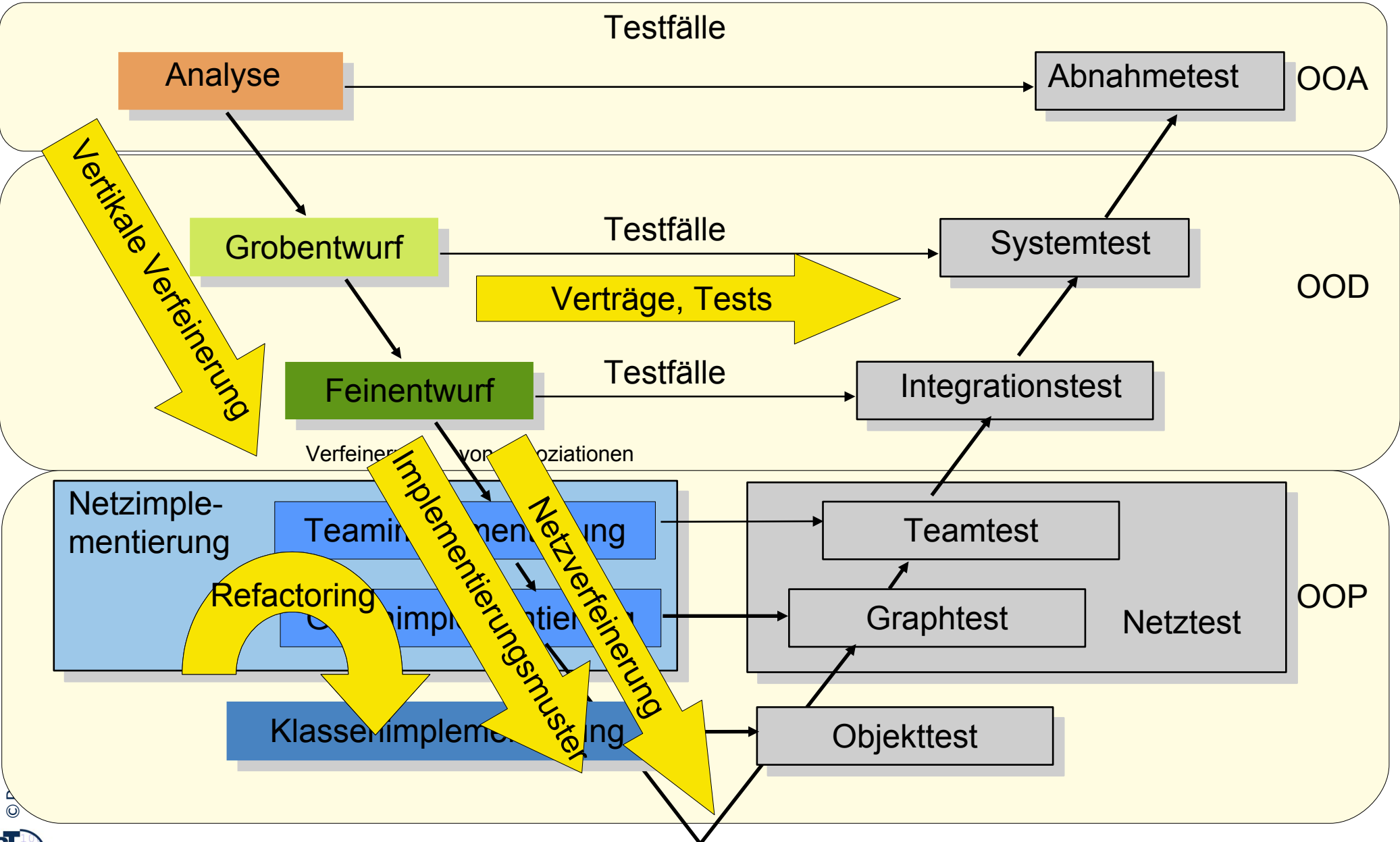
[Boehm 1979]



- ▶ **Horizontale Verfeinerungsoperationen** ersetzen Fragmente auf gleicher Sprachebene:
 - **Detailierung (Anreicherung):** Ergänzung von Einzelheiten
 - **Vervollständigung (Elaboration)** von Fragmenten zu Sätzen der Modellierungssprache
 - **Erhöhung Zuverlässigkeit:** Ergänzung von qualitätssteigernden Fragmenten (Typisierung, Verträge, Tests)
 - Einführung des **Architektur-Aspektes** des Systems
 - **Strukturierung** und **Restrukturierung**
 - **Refaktorisierung (Refactoring)** ist semantische Restrukturierung
- ▶ **Vertikale Verfeinerungsoperationen** (von abstrakter Ebene zu konkreter Ebene):
 - **Abflachen von Fragmenten** (Flachklopfen, Realisierung, lowering):
 - **Realisierung** ersetzt ausdrucksstarke Konstrukte durch weniger ausdrucksstarke, implementierungsnähere

Q4: Softwareentwicklung im V-Modell

[Boehm 1979]



Verfeinerung: Schritte von UML zur Implementierung

Ein **Implementierungsmuster** (*workaround, Idiom*) beschreibt die vertikale Verfeinerung eines Sprachkonstruktes einer Modellierungs- oder Spezifikationsprache durch ein Fragment einer Implementierungssprache

- ▶ Verfeinerung von Sprachkonstrukten (Realisierung, Abflachen, lowering)
 - Netzentwurf
 - Implementierung von Methoden (von Statecharts und Aktivitätsdiagrammen)
 - Datenverfeinerung
 - Kontrollverfeinerung
 - Syntaktische Verfeinerung
 - Semantische Verfeinerung

Repräsentation von Objektnetzen als Datenstrukturen (Netzverfeinerung)

- ▶ Auf Ebene der Anforderungen werden Objektnetze als math. Graphen dargestellt
- ▶ Im Grobentwurf und Feinentwurf werden sie repräsentiert (verfeinert zu)
 - durch Graphen als **Sprachkonstrukte** (für Sprachen, die das eingebaut haben)
 - Rapid Application Development (RAD)
 - durch **Graphen** aus Java-Graph-Bibliotheken
 - durch **Collections**, nach dem Abflachen/Flachklopfen von bidirektionalen Assoziationen in gerichteten Links
 - durch **Datenstrukturen** fester Länge (Arrays, Matrizen) (speicher-bewusstes Programmieren)

Graphen (mathematisch)

Graphen als Sprachkonstrukte

Graphen als Bibliotheken

Graphen als Collections abgeflacht

Graphen als Datenstrukturen fester Länge abgeflacht

Netzverfeinerung von fixen Netzen als Konnektoren (Kollaborationen)

- ▶ Fixe Netze mit statisch festem n, m (z.B. 1:1-Assoziationen) können durch **Konnektoren** (Kollaborationen, Teams) verfeinert werden
 - Assoziationen tragen Rollentypen als Assoziationsenden
- ▶ Einfache Objektnetze werden durch **Entwurfsmuster** beschrieben
 - Listen: Decorator, Chain
 - Bäume: Composite
 - Dags, Graphen: Modifiziertes Composite
 - Konnektoren: Observer, Mediator, Visitor
- ▶ Große Objekte (Bobs) haben oft interne Subobjektnetze
 - Subobjektnetze beruhen auf Endo-Assoziationen

Graphen als Konnektoren (Rollen-Kollaborationen)

Graphen als Entwurfsmuster

Graphen als Subobjektnetze in Bobs

Höhere
Sprachen

Graphen als Sprachkonstrukte

Graphen als Konnektoren (Rollen-Kollaborationen)

Graphen als Entwurfsmuster

Java

Netze mit Datenfluss

Graphen als Bibliotheken (Java)

Graphen als Collections abgeflacht (Java)

Graphen als Endoassoziationen:
Netze von Unterobjekten in komplexen Objekten (Java)

System-
programmier-
sprache (C)

Graphen als Datenstrukturen fester Länge abgeflacht

Anhang

- ▶ Ein **Sprachkonstrukt (Sprachelement)** bezeichnet ein Konstrukt bzw. Konzept einer Sprache.
- ▶ Ein **Programm-/Modellelement** bezeichnet ein Element eines Programms/Modells.
- ▶ Ein **Fragment (Snippet)** eines Programms oder Modells ist ein partieller Satz der Sprache.
- ▶ Ein **generisches Fragment (generisches Snippet, Fragmentformular)** eines Programms oder Modells ist ein partieller Satz der Sprache mit Platzhaltern ("Lücken").
- ▶ Eine **Fragmentgruppe** ist eine Menge von (ggf.generischen) Fragmenten eines Programms oder Modells.
- ▶ Eine **Fragmentkomponente** ist eine Fragmentgruppe zur Wiederverwendung.
- **Abstraktion** ist das Vernachlässigen von Details
- **Detaillierung (Anreicherung)** ist das Anfügen von Details