



Teil II

Objektorientierte Programmierung (OOP) mit Objektnetzen

20. Software-Entwicklung im V-Modell

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
Technische Universität Dresden
Version 22-0.1, 30.04.22

- 21) Verfeinern von Assoziationen mit dem Java-2 Collection Framework
- 22) Einführung in Entwurfsmuster
- 23) Teams und Kanäle
- 24) Entwurfsmuster für Produktlinien
- 25) Graphen in Java

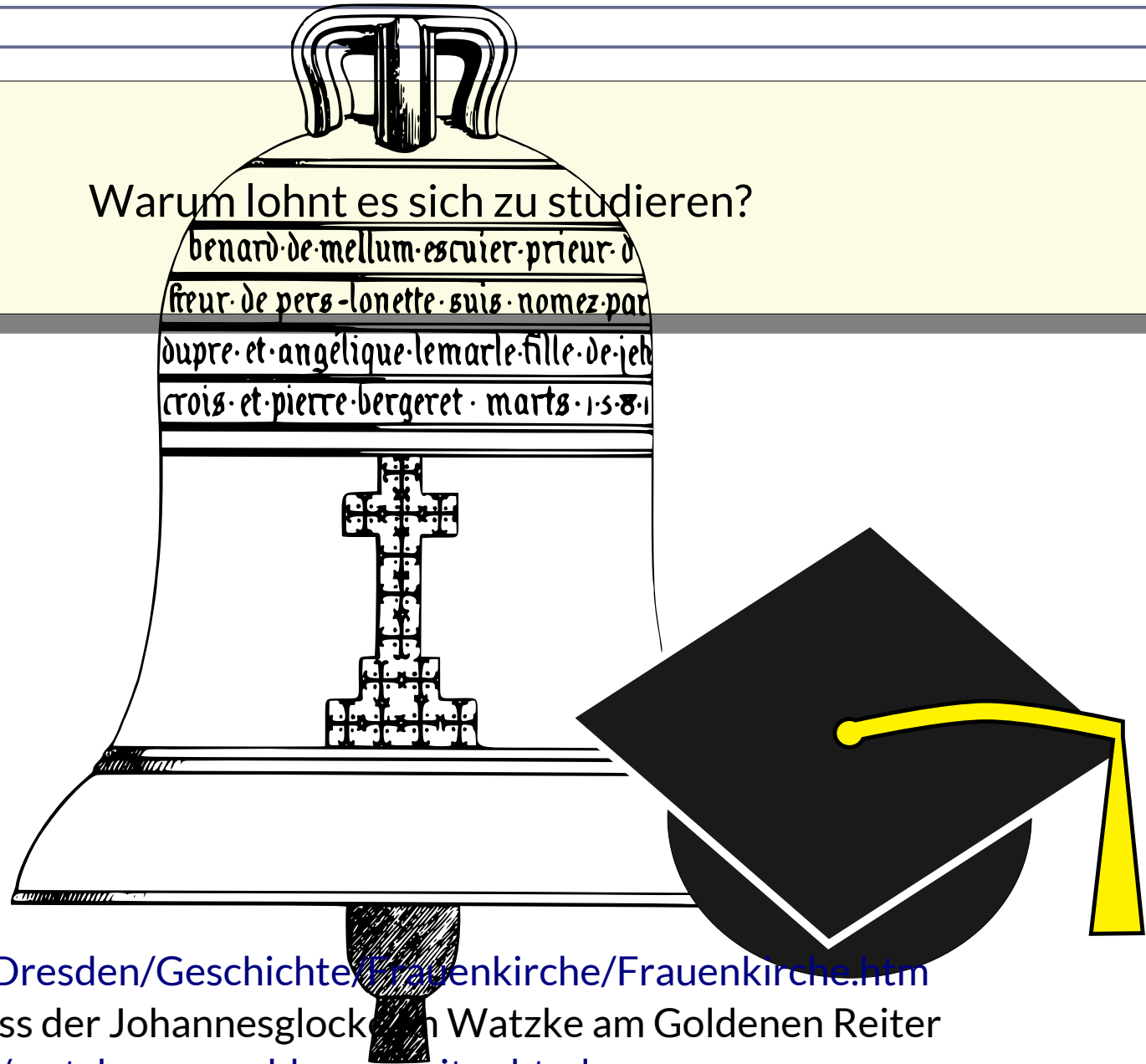
Wichtige Literatur

- ▶ Java language spec
 - https://de.wikipedia.org/wiki/Java_Language_Specification
 - <https://docs.oracle.com/javase/specs/jls/se8/html/index.html>
- ▶ Diffs in Version 12
 - <https://cr.openjdk.java.net/~iris/se/12/latestSpec/java-se-12-annex-3.html>
- ▶ JDK Tutorial für J2SE oder J2EE, <https://docs.oracle.com/javase/tutorial/>

Das Ziel des Studiums: Schöne und große Dinge erschaffen können

Warum lohnt es sich zu studieren?

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



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

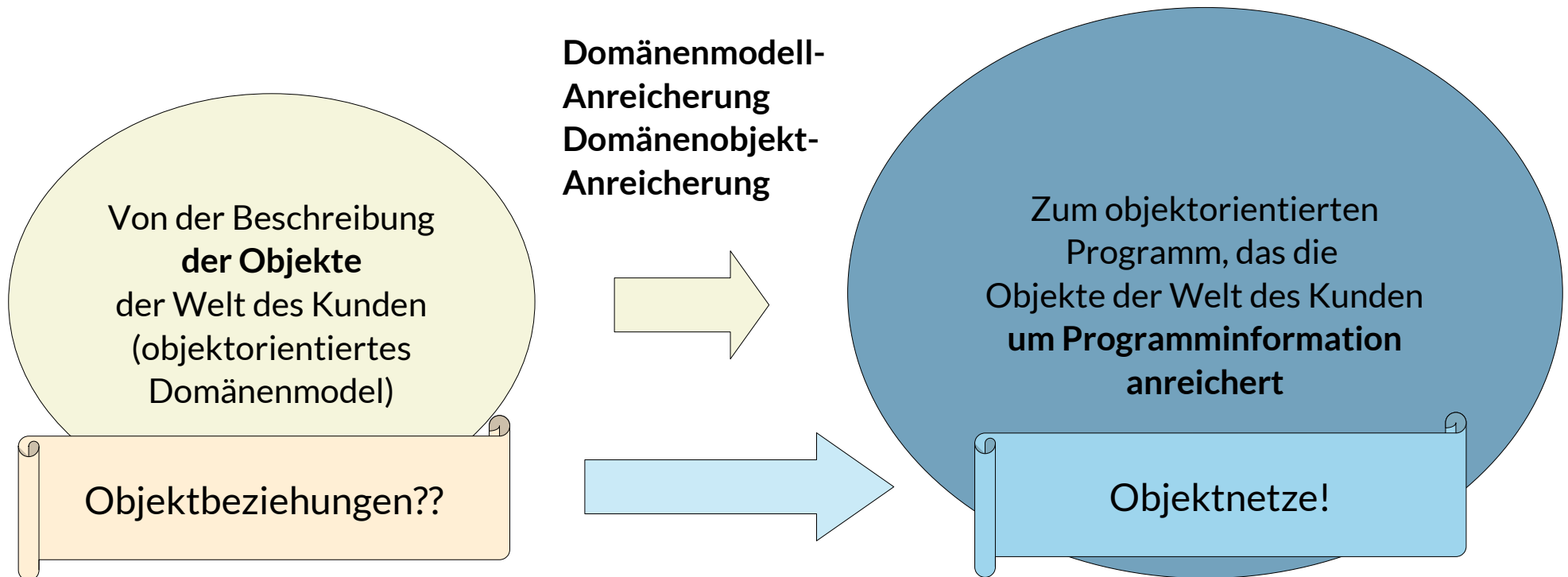
Fehltönender Probeguss der Johannesglocke in Watzke am Goldenen Reiter

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

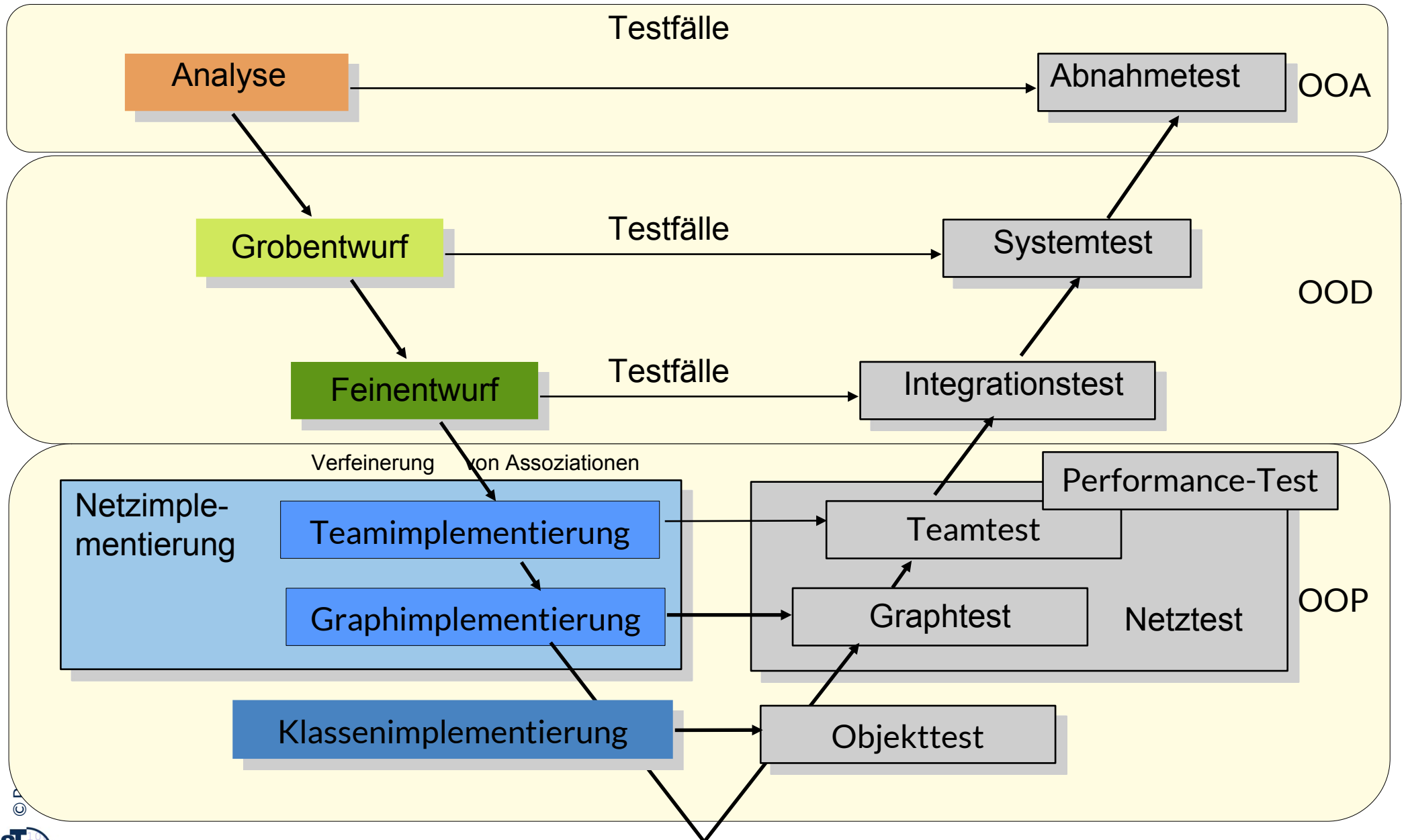
Ziel von Teil II der Vorlesung

- ▶ Wie kann man **Objektnetze** (“Assoziationen”, Graphen, Dags, Bäumen, Listen) abstrakt und ausdrucksstark beschreiben?
- ▶ Wie kann man deren **Test** vereinfachen? (Das Programmieren von Objektnetzen ist sehr fehleranfällig)
- ▶ Wie kann man den Aufbau von Objektnetzen durch **Verfeinerung von Assoziationen** konkret auf den Rechner zuschneiden?
 - Graphen, Iteratormethoden, Iteratoren, und Streams
 - Große Objekte (Bobs) mit internen Netzen
 - Endo- und Exoassoziationen
 - Wie man Graphen erweitert

OUR GOALS

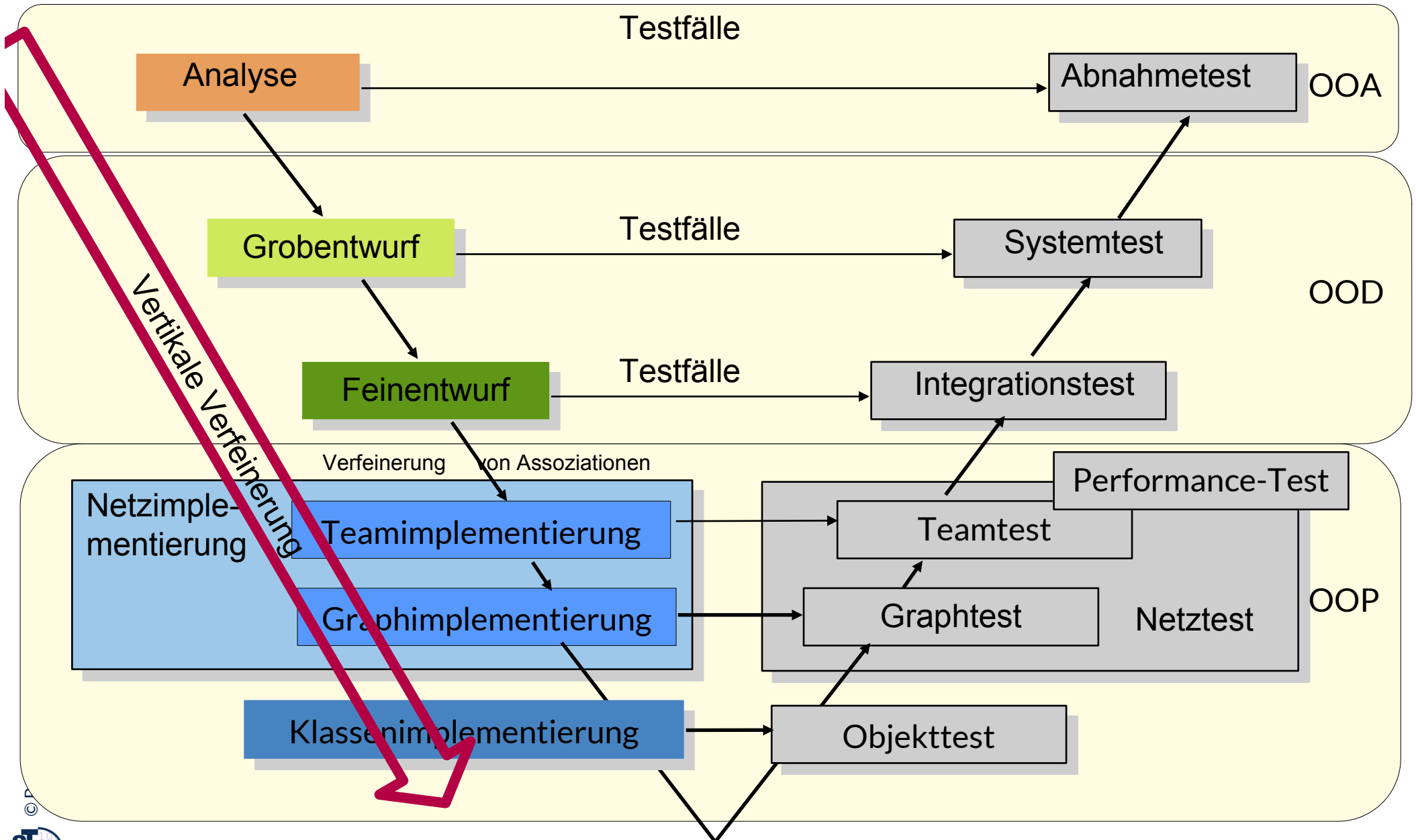
Q4: Softwareentwicklung im V-Modell

[Boehm 1979]



Q4: Softwareentwicklung im V-Modell

[Boehm 1979]



Modellierung im V-Modell

- ▶ Im V-Modell werden mehrere Sprachen gleichzeitig benutzt (hier aUML, dUML, jUML)
- ▶ Die Modelle werden durch verschiedene Verfeinerungsoperationen aus dem initialen Anforderungsmodell entwickelt

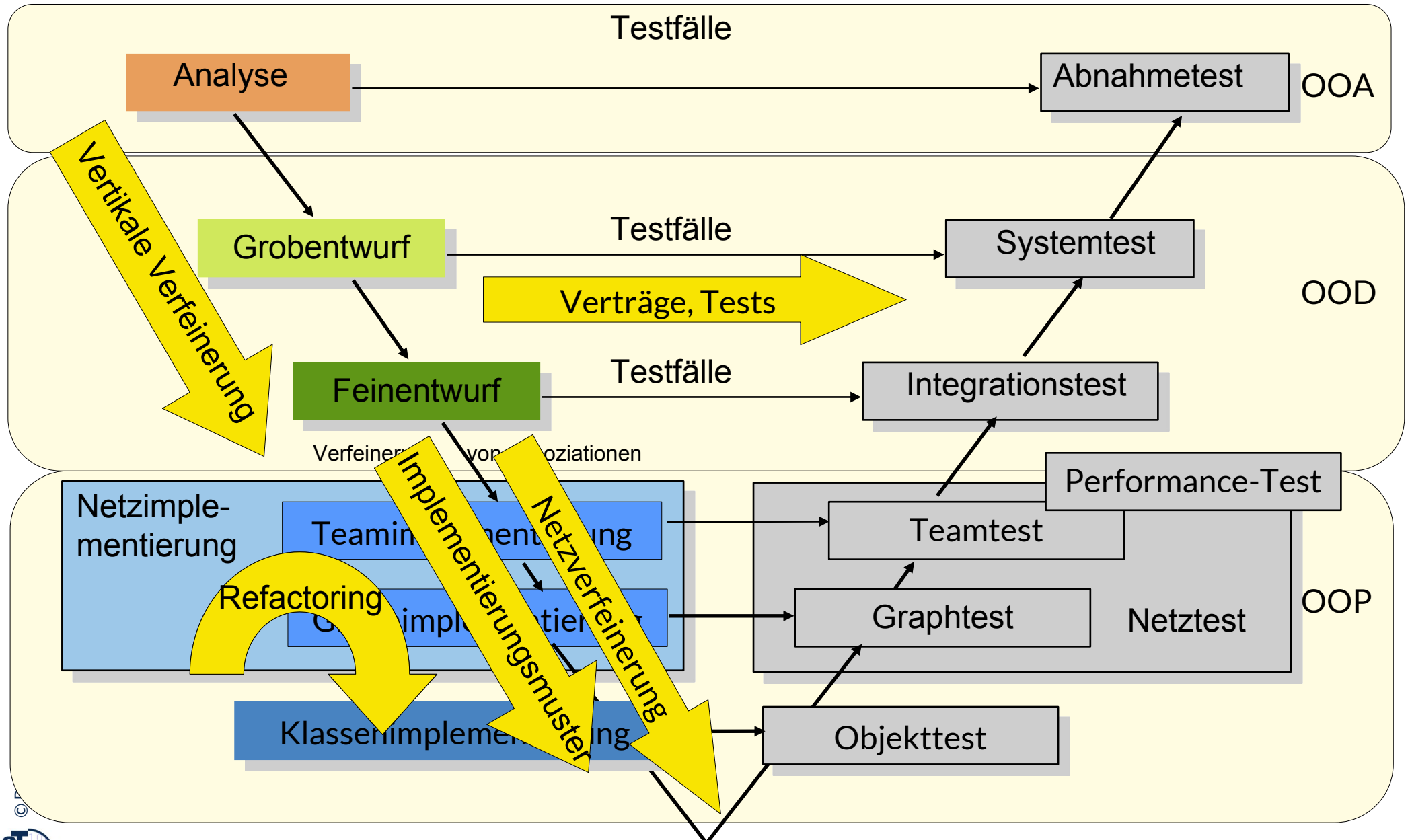
- ▶ Ein **Sprachkonstrukt (Sprachelement)** bezeichnet ein Konstrukt bzw. Konzept einer Sprache.
- ▶ Ein **Programm-/Modellelement** bezeichnet ein Element eines Programms bzw. eines Anforderungs- oder Entwurfsmodells bezogen auf ein Sprachkonstrukt.
- ▶ Ein **Fragment (Snippet)** eines Programms oder Modells ist ein partieller Satz der Sprache, d.h. ein Netz aus Programm- oder Modellelementen.
- ▶ Ein **generisches Fragment (generisches Snippet, Fragmentformular)** eines Programms oder Modells ist ein partieller Satz der Sprache mit Platzhaltern ("Lücken"), der mit Füllseln gefüllt werden muss.
- ▶ 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, Verfeinerung)** ist das Anfügen von Details. Verfeinerung kann *horizontal* oder *vertikal* erfolgen.

Software-Detaillierung (Verfeinerung) im V-Modell

- ▶ **Horizontale Verfeinerungsoperationen** ersetzen Fragmente und Fragmentgruppen **auf gleicher Sprachebene**:
 - **Detaillierung (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) vereinfachen Fragmente und **wechseln dabei die Sprache**, z.B. von UML nach Java:
 - **Abflachen** von Fragmenten (Flachklopfen, Realisierung, lowering): Realisierung ersetzt ausdrucksstarke Konstrukte durch weniger ausdrucksstarke, implementierungsnähere
 - Einsatz von Implementierungsmustern

Q4: Softwareentwicklung im V-Modell

[Boehm 1979]

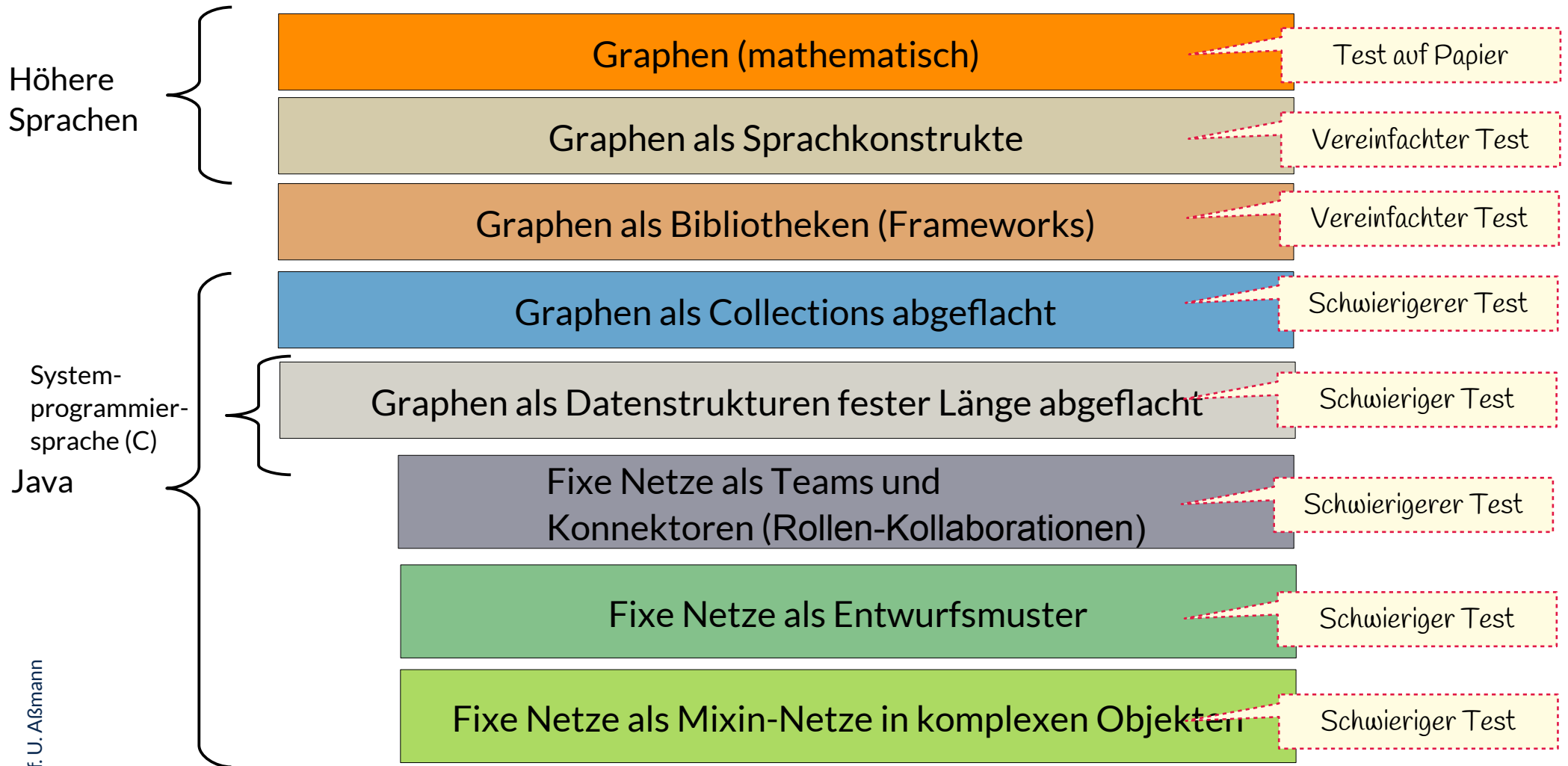


Verfeinerung: Schritte von UML zur Implementierung

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 flexiblen und fixen Objektnetzen als Datenstrukturen (Netzverfeinerung)



- ▶ Wieso kann man den Klang einer Glocke nicht testen?
- ▶ Wieso braucht man viel Erfahrungen, um die Parameter einer Glocke korrekt zu schätzen?
- ▶ Warum ist das Programmieren von Objektnetzen so schwierig?
- ▶ Welche Möglichkeiten gibt es fürs Testen, wenn man Objektnetz-Bibliotheken verwendet?
- ▶ Warum ist Performance-Test von Netz- und Objekttest verschieden?
- ▶ Warum ist Testautomatisierung für die Programmierung von Objektnetzen so wichtig?

A blue rectangular banner with the word "SUMMARY" in large, white, 3D-style capital letters. The letters have a slight shadow and are set against a background of horizontal blue lines.

SUMMARY