Semesterprojekt

“Entwicklung von Automatik-Funktionen in einer Fahrsimulation“

WS 2012/13

Realisierung der Automatiken: Entwurf, Implementation, Test

K. Bothe

26. 11. 2012

Aufgaben im Überblick

Automatiken für eine ausgewählte Situation (Gabelung)

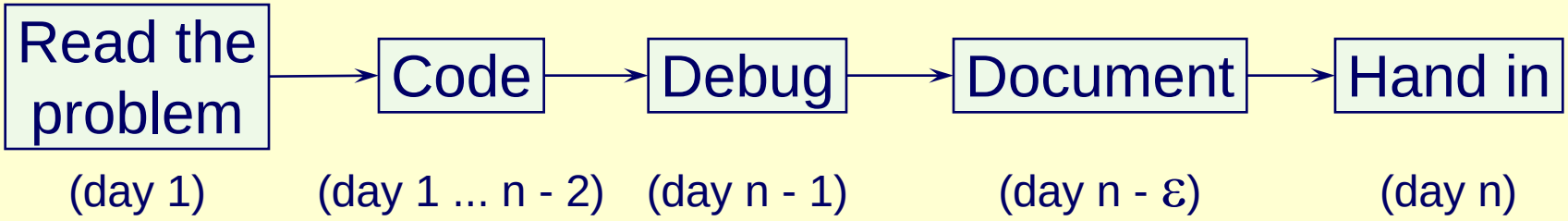
- Aufgabe 1: 22. – 29. 10. 2012
  Vertrautmachen mit Versuchsaufbau

- Aufgabe 2: 29. 10. – 18. 11. 2012
  Automatiken konzipieren → Pflichtenheft

- Aufgabe 3: 18. 11. – 16. 12. 2012
  Realisierung der Automatiken → Entwurf, Implementation, Test

- Aufgabe 4: 16. 12. – 11. 2. 2013
  Weitere Automatiken umsetzen

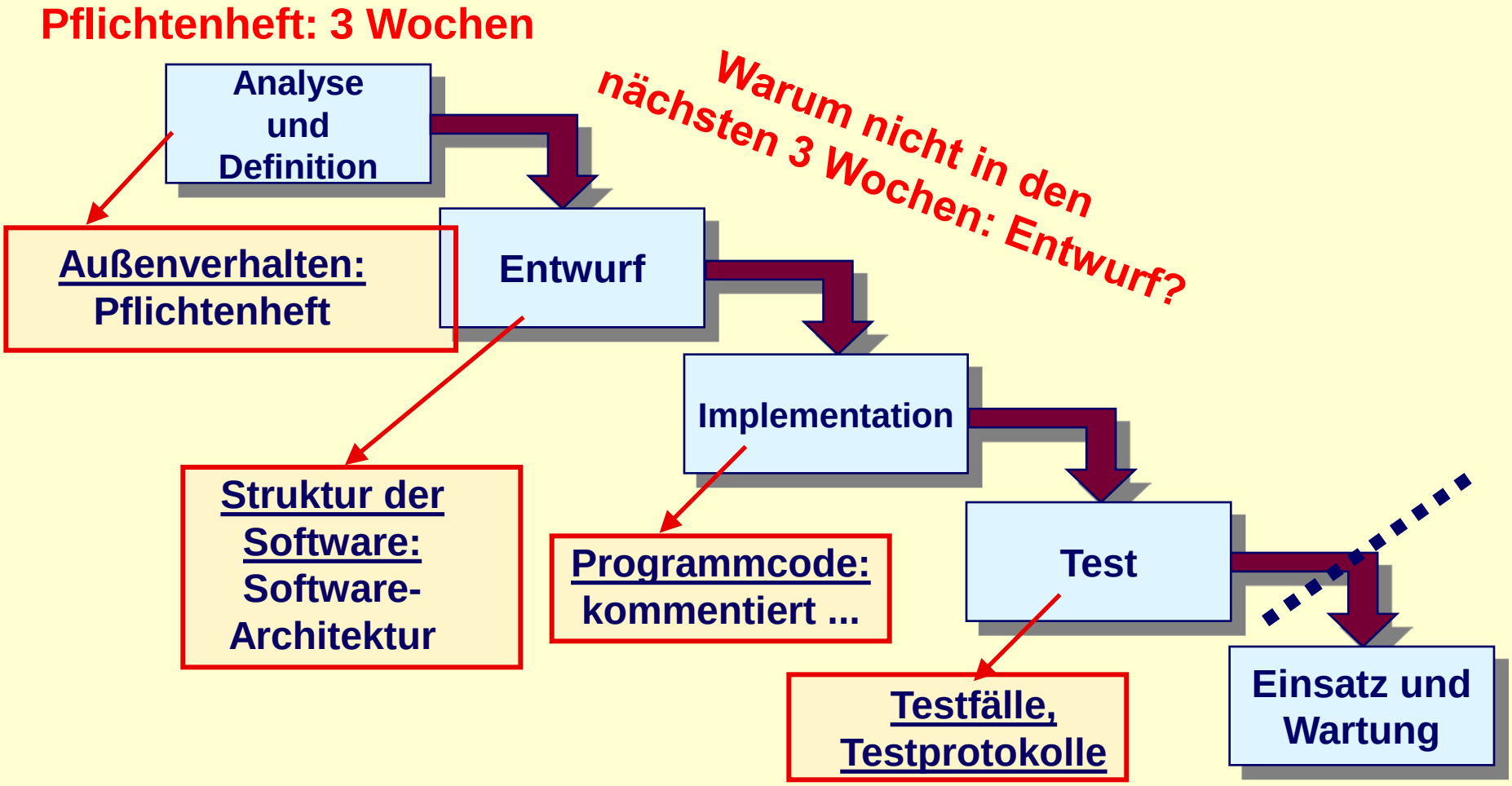
Gefahr:
Studentische Softwareentwicklung



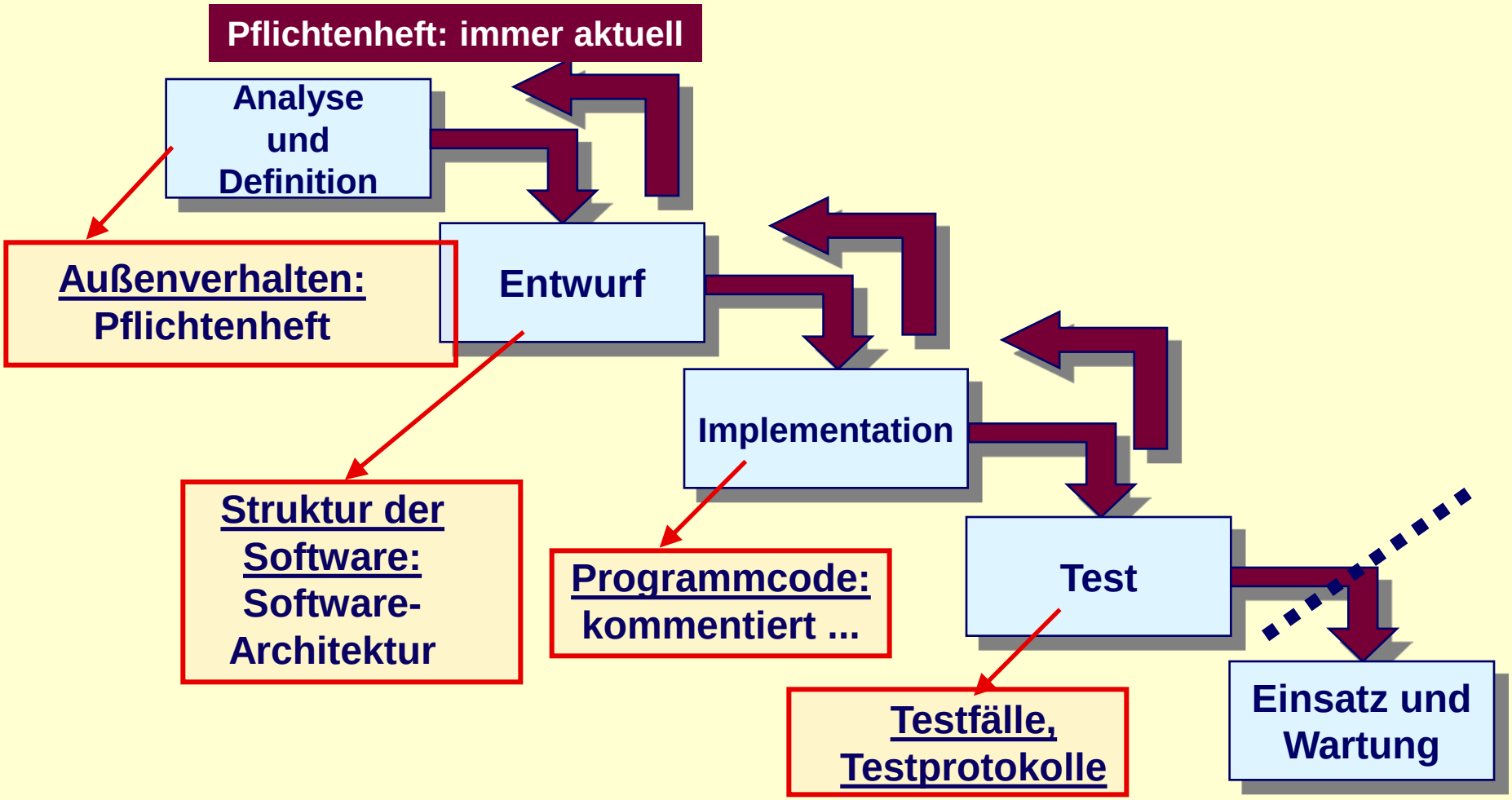
„Normal“ student view of software life cycle

Modesitt, LNCS 750, S 42

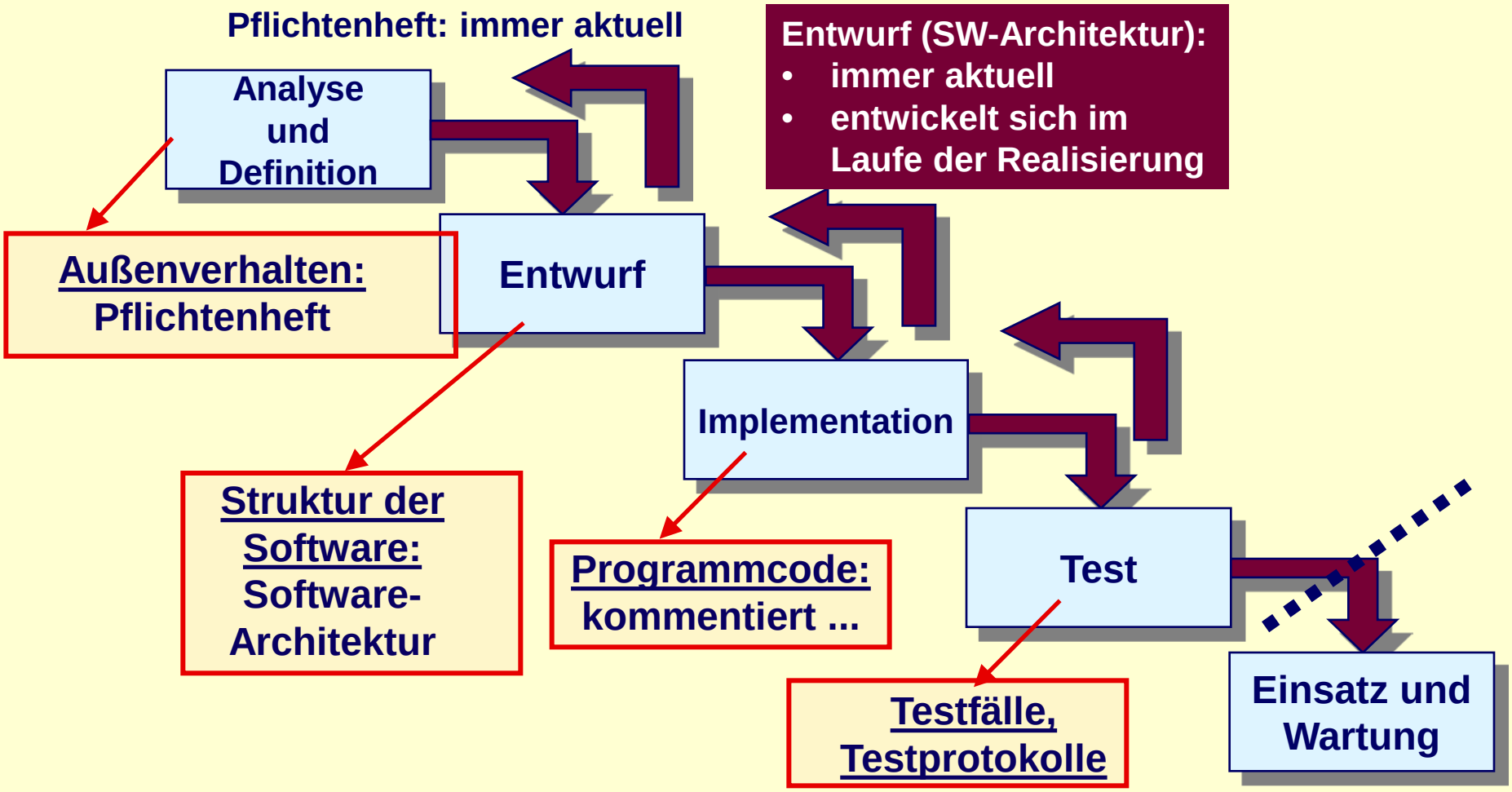
Klassisches Wasserfallmodell: entstehende Dokumente



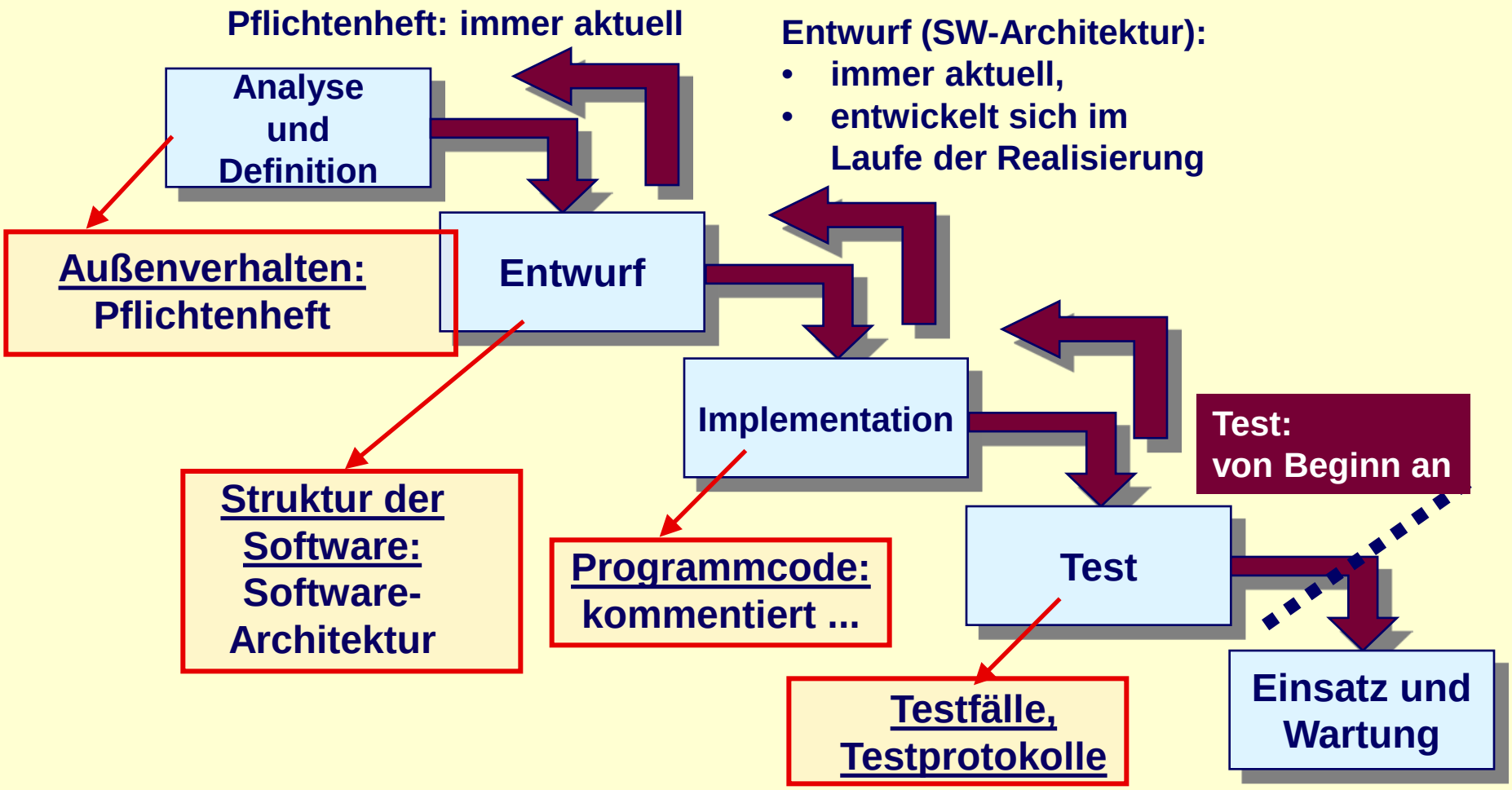
Klassisches Wasserfallmodell: entstehende Dokumente



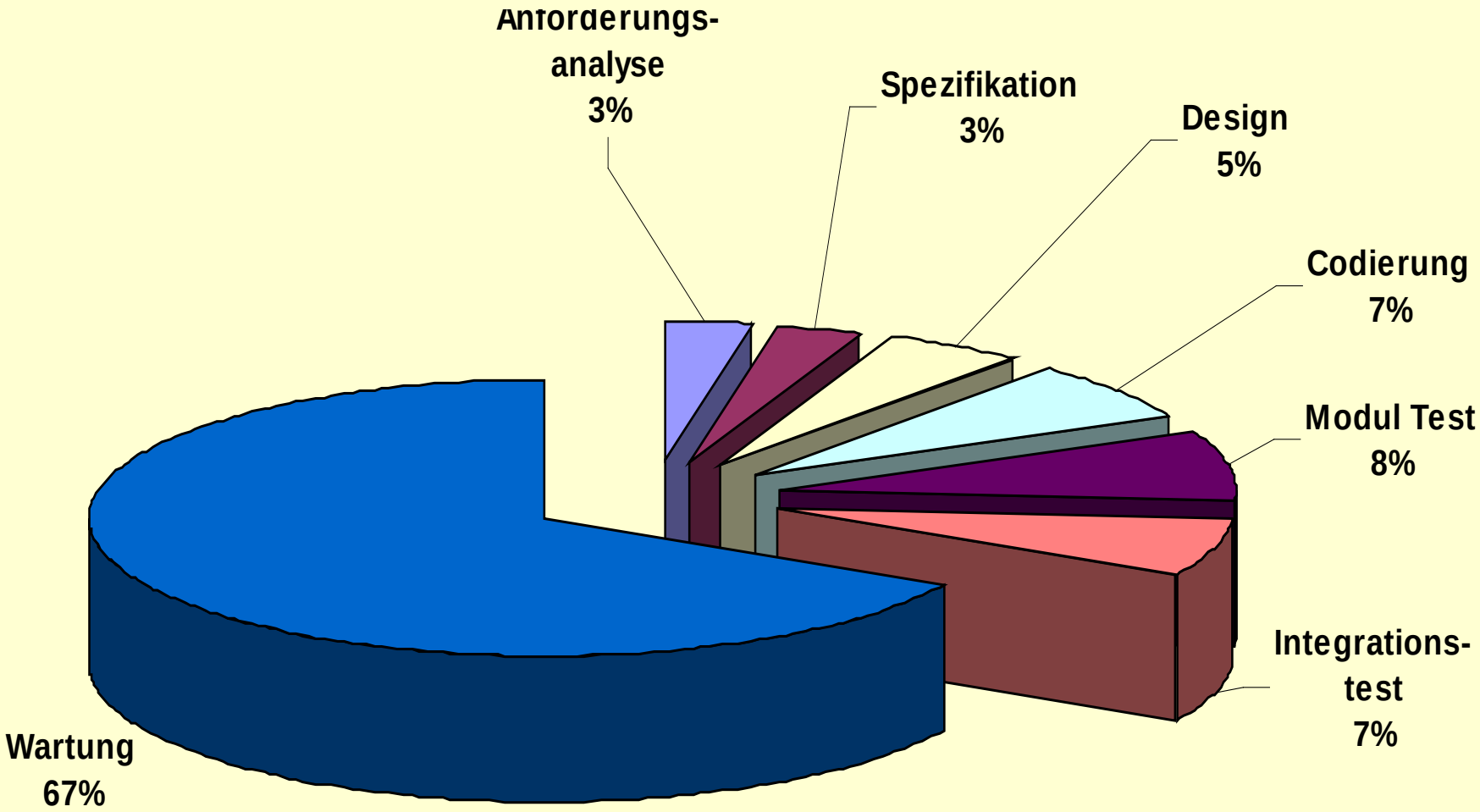
Klassisches Wasserfallmodell: entstehende Dokumente



Klassisches Wasserfallmodell: entstehende Dokumente



Kostenverteilung im Software-Life-Cycle



Quelle: R. Klösch, H.Gall: Objektorientiertes Reverse Engineering. Springer, 1995

Grundlagen zum Entwurf

vgl. K. Bothe: Vorlesung Software Engineering

Entwurf: Worauf kommt es im Projekt an?

- Ziel: Entwicklung einer Softwarearchitektur:
  Struktur der Software = Komponenten + Konnektoren

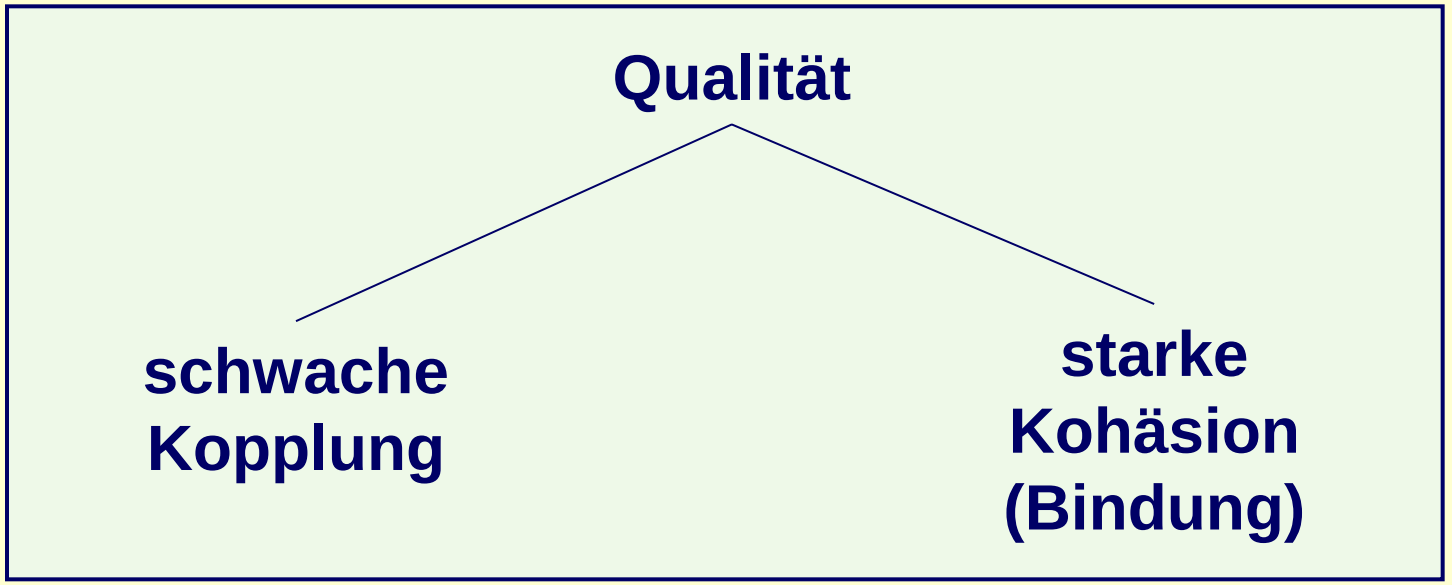
- Problembereich ATEO (Automatiken):
  neues Anwendungsgebiet

  → Architektur muss erst „erlernt“ werden
  – jedes Anwendungsgebiet hat eigene Ansätze
  → entwickelt sich schrittweise zusammen mit der Implementation
  → d.h. Qualität „reift“
  → SW-Architektur ständig verbessern
  (Refactoring: Klassen entstehen und verändern sich)

- Dokumentation: UML (Komponenten sind Klassen)

- Qualitätsmerkmale: Kohäsion, Kopplung

„Gute“ SW-Architekturen: Kernmerkmale



Möglichst wenige Beziehungen zwischen den Komponenten

Eine Komponente dient der Realisierung einer Teilaufgabe

Trennung von Zuständigkeiten

Kohäsion (Bindung)

Logische Beziehung zwischen Elementen
(Daten, Operationen) *innerhalb einer Komponente*

- *Ziele:*
  - Elemente der Komponente dienen der Lösung einer gemeinsamen Teilaufgabe (**starke Kohäsion**: Bindung der Elemente an diese Teilaufgabe)
  - Keine Vermischung von mehreren Teilaufgaben innerhalb einer Komponente
  - problembezogene Komponentenbildung

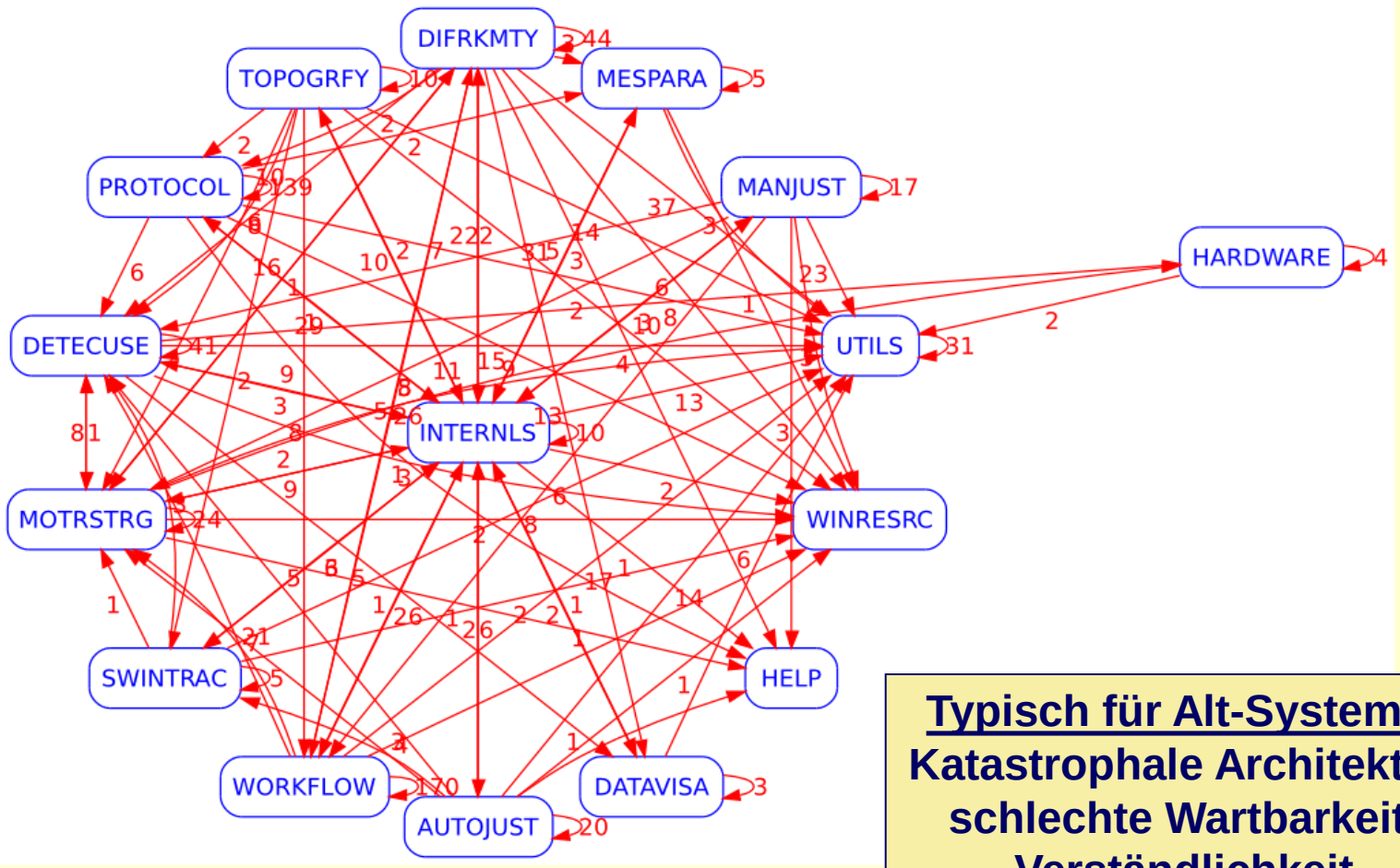
Kopplung

Schnittstellenbeziehung *zwischen Komponenten*

- *Ziele:*
  - möglichst wenige Beziehungen zwischen Komponenten (**schwache Kopplung**: geringer Informationsaustausch)
  - leichter zu verstehen und zu modifizieren (Konzentration auf eine Komponente)

Wichtig:
- Viele unabhängige Komponenten günstig
- Möglichst keine zyklischen Beziehungen.

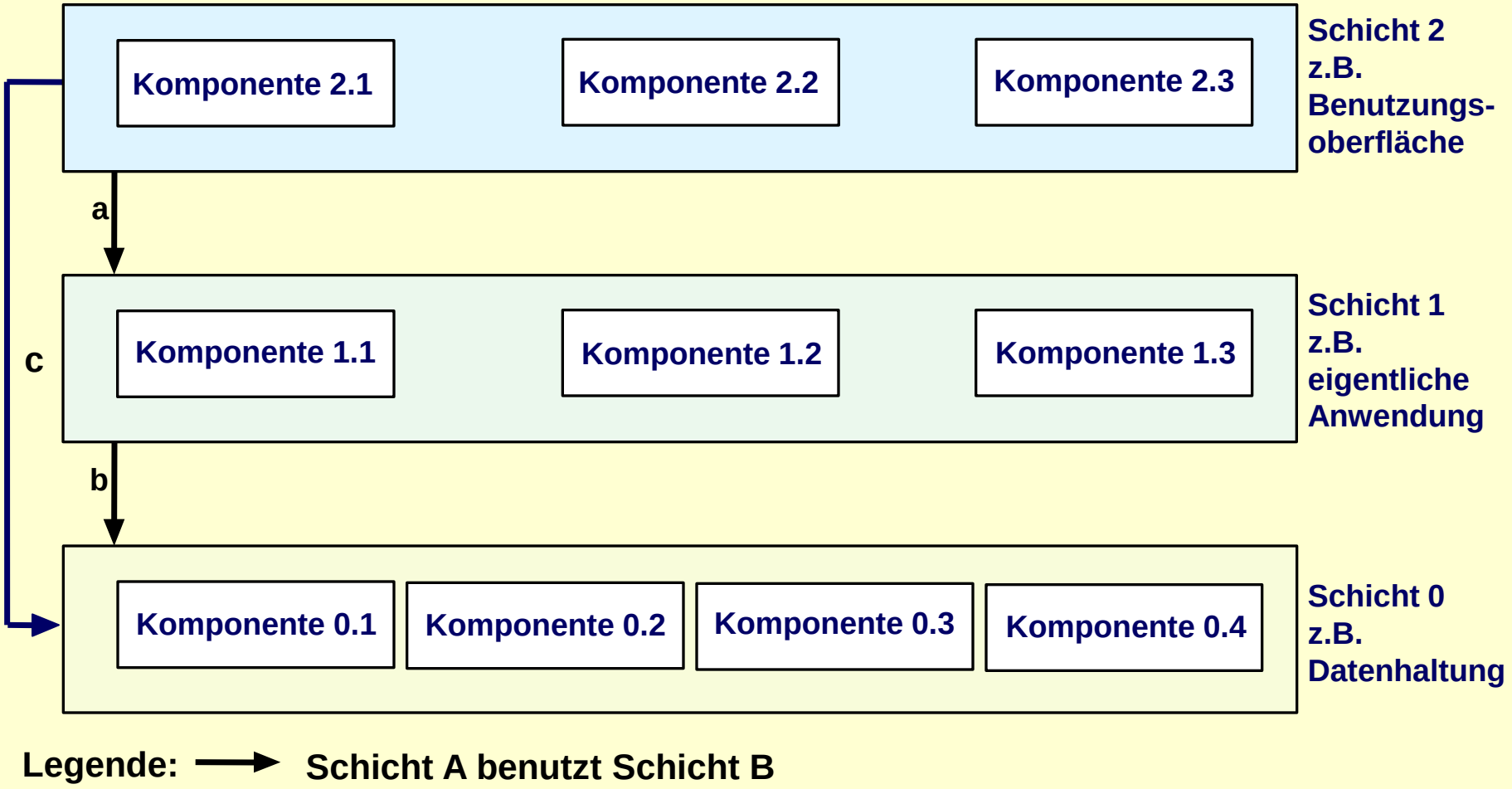
So nicht: reale Architektur von XCTL



Es müsste Schichtenarchitektur sein

<u>Typisch für Alt-Systeme</u>:
Katastrophale Architektur, schlechte Wartbarkeit, Verständlichkeit
→ zyklische Abhängigkeiten

Schichtenarchitektur: Hierarchiebildung



Grundlagen zum Test

Vgl. Vorlesung SE

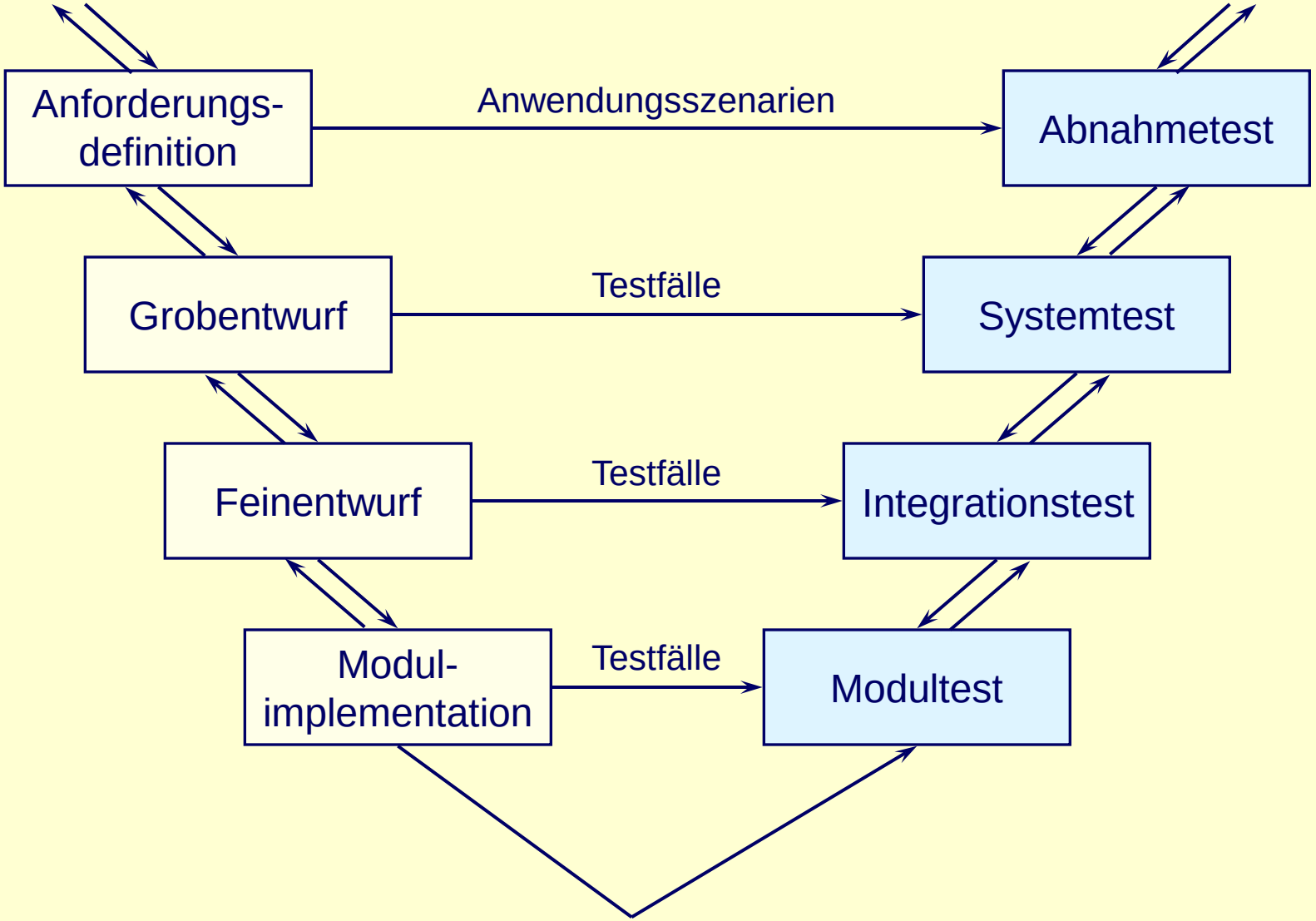
Test: Worauf kommt es im Projekt an?

- Testaktivitäten nicht am Ende, sondern „parallel“ zum gesamten Entwicklungsprozess

- Test auf unterschiedlichen Abstraktionsebenen:
  Systemtest … Modultest

  → Test der Automatik als Ganzes
  → Test von einzelnen Methoden
  (zur Realisierung der Automatik)

V-Modell: Test auf unterschiedlichen Abstraktionsebenen



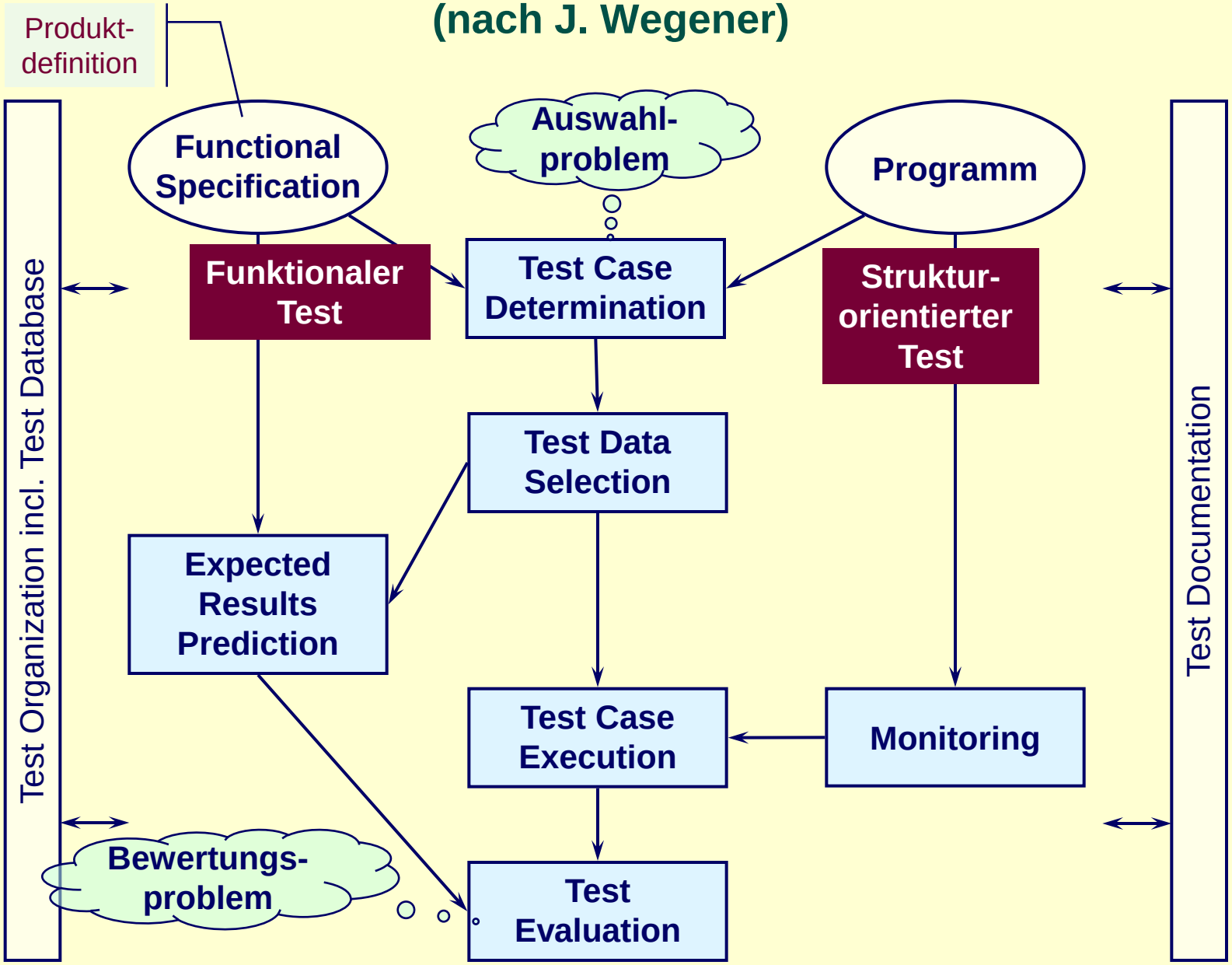
Test: Worauf kommt es im Projekt an?

- Testaktivitäten nicht am Ende, sondern „parallel“ zum gesamten Entwicklungsprozess
- Test auf unterschiedlichen Abstraktionsebenen: Systemtest … Modultest
- Grundproblem: Haben wir die richtigen Testfälle (nicht zu viele – keine vergessen)?

→ unser „Problem“ (Motivation):
Software nicht sicherheitskritisch bzw. kommerziell
(Experimentalsoftware in der Psychologie,
Automatik nicht im Flugwesen /
XCTL: kostspielige Hardware wird gesteuert /
dBase fehlerhaft ausgeliefert → Firma bankrott)

Detaillierte Testaktivitäten

(nach J. Wegener)



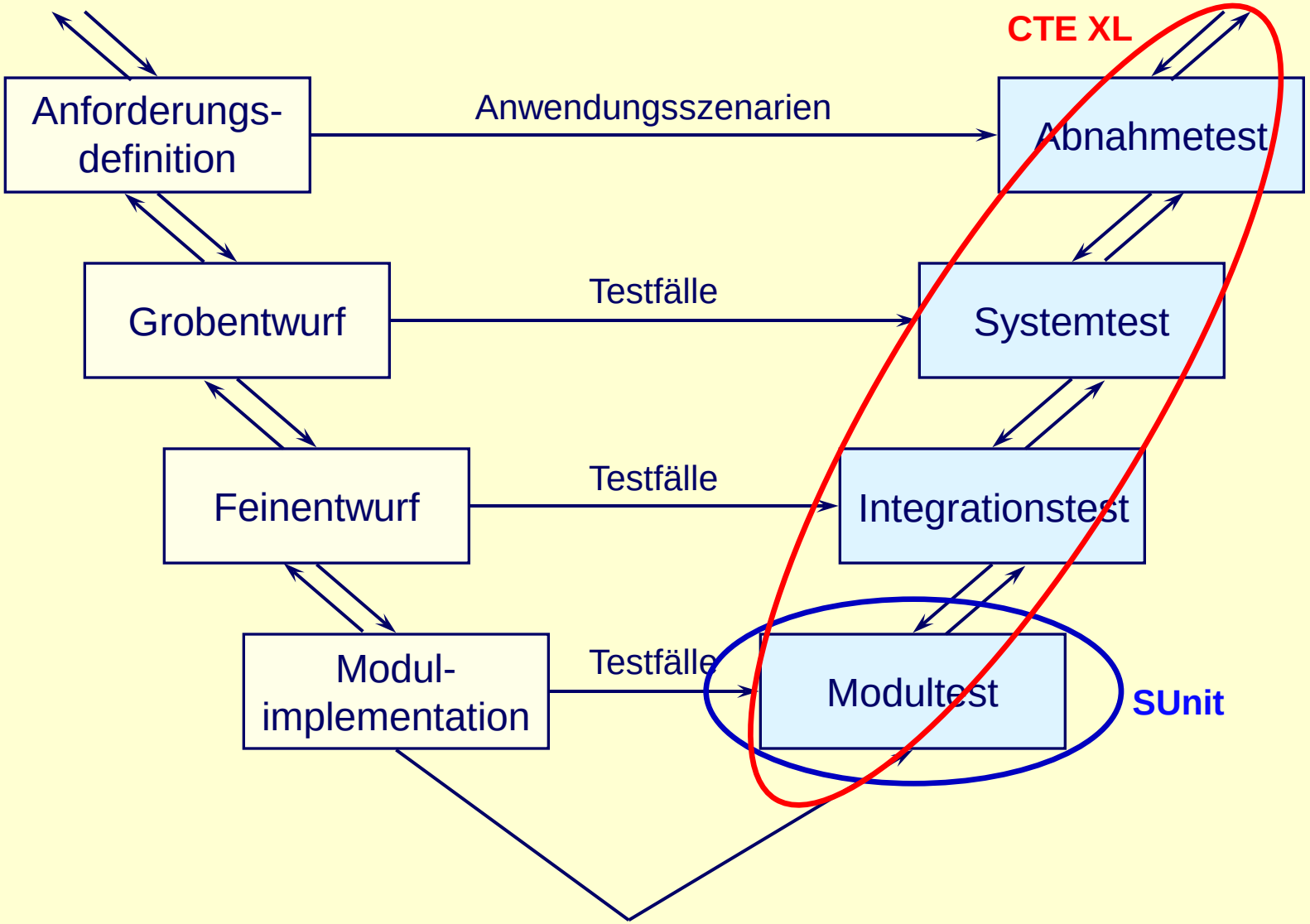
Test: Worauf kommt es im Projekt an?

- Testaktivitäten nicht am Ende, sondern „parallel“ zum gesamten Entwicklungsprozess
- Test auf unterschiedlichen Abstraktionsebenen: Systemtest … Modultest
- Grundproblem: Haben wir die richtigen Testfälle (nicht zu viele – keine vergessen)?
- Automatisierung durch Toolunterstützung: CTE XL, SUnit

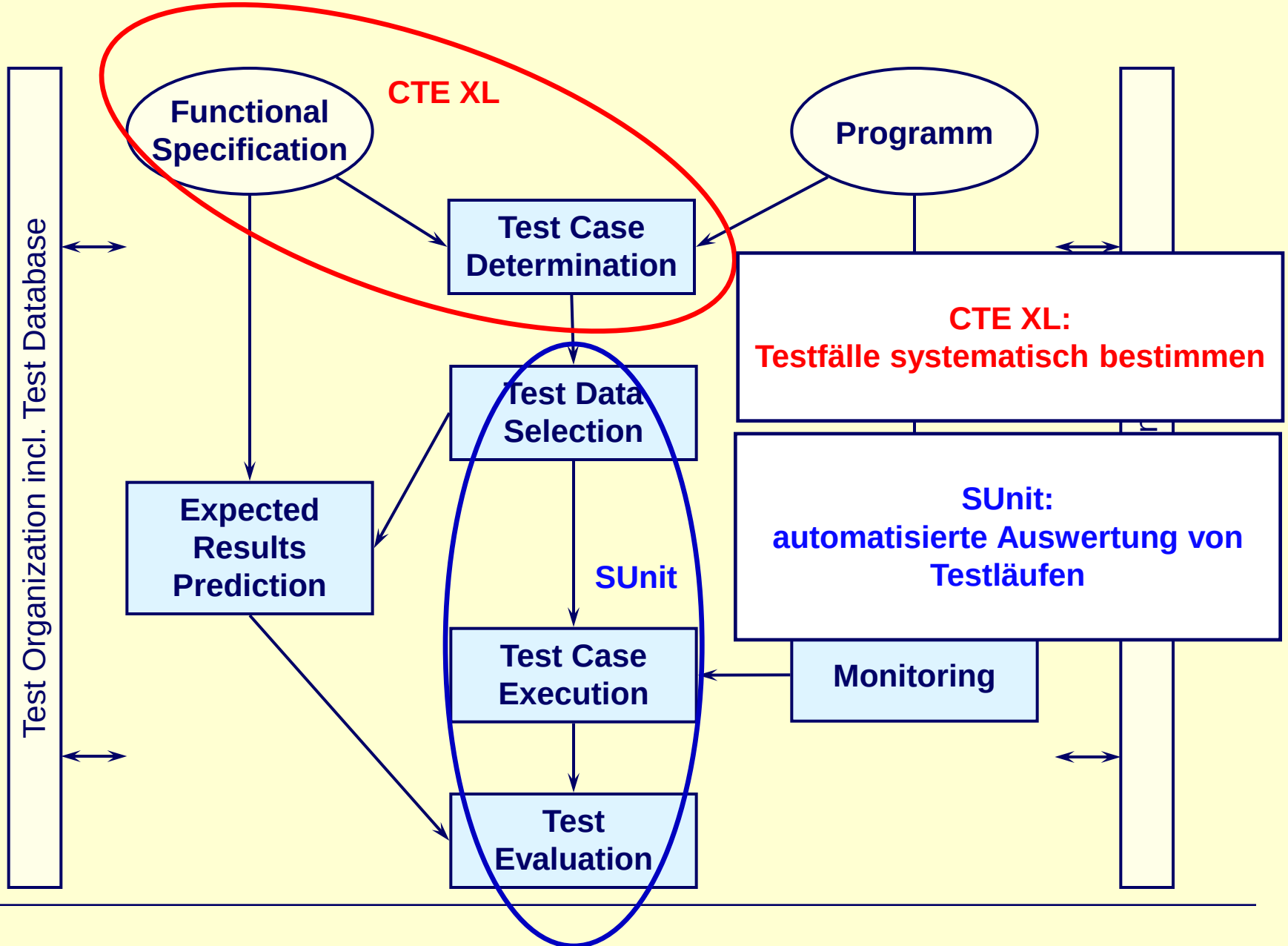
CTE XL:
Tool für den funktionalen Test

Auswahl von Testfällen aufgrund der funktionalen Beschreibung (Pflichtenheft)

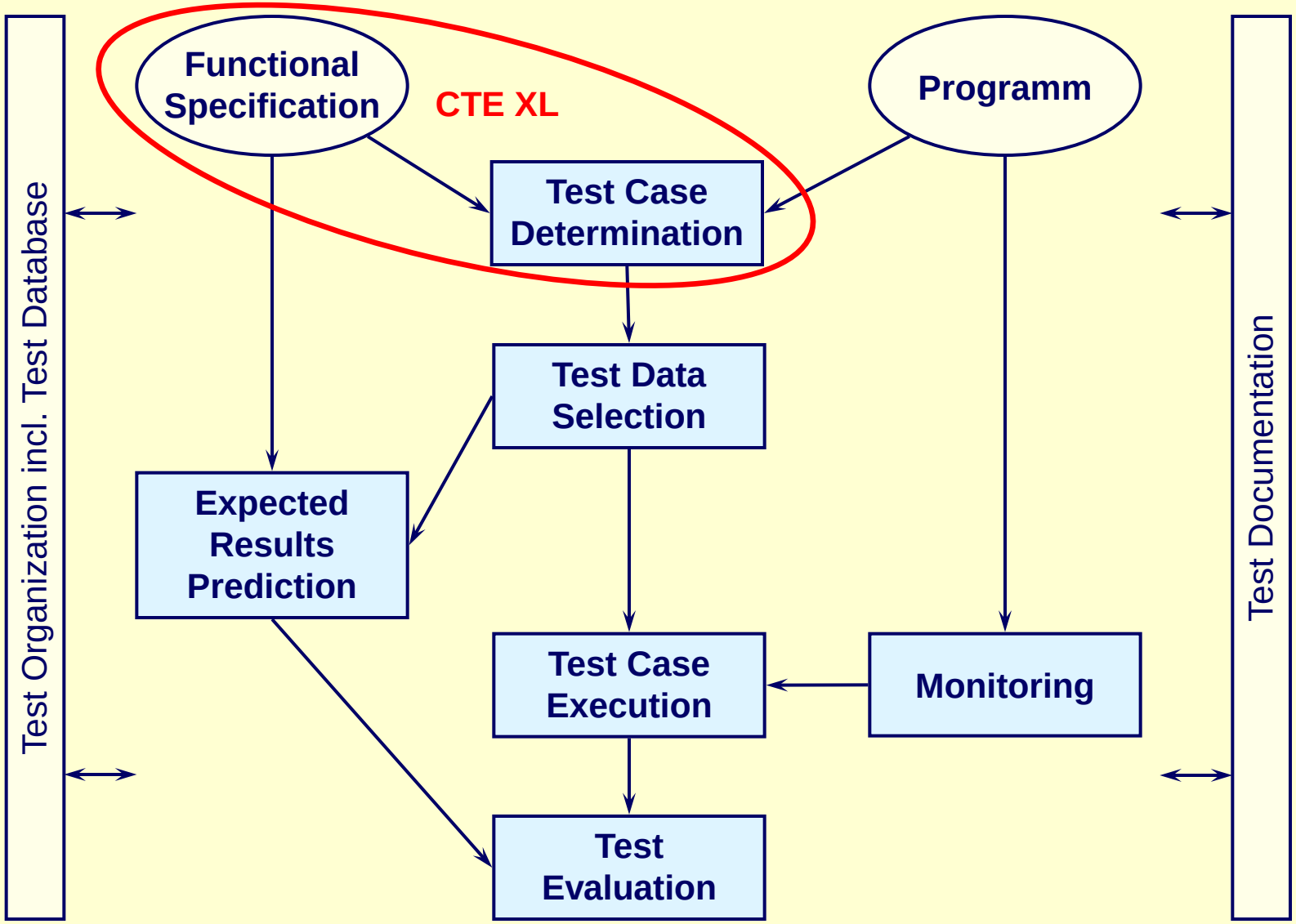
V-Modell: Einordnung von CTE XL und SUnit



CTE XL und SUnit

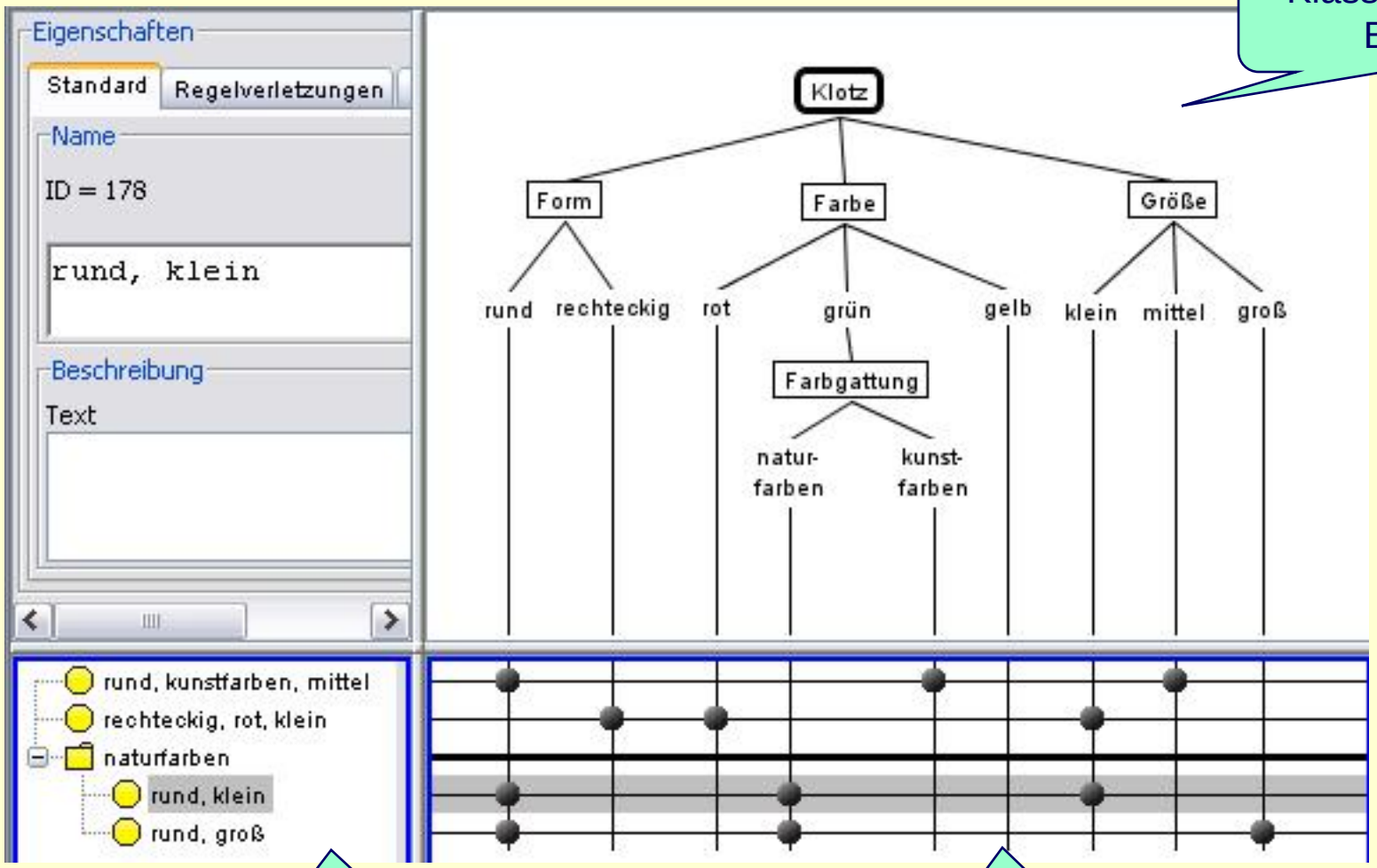


CTE XL: Testfälle systematisch auswählen durch Klassifikation des Eingabedatenraums



CTE-XL: Methode und Beispiel

- CTE-XL-Screenshot -



Klassifikations-Baum

Testfälle und Gruppierung

Auswahl derTestfälle

Beispiel

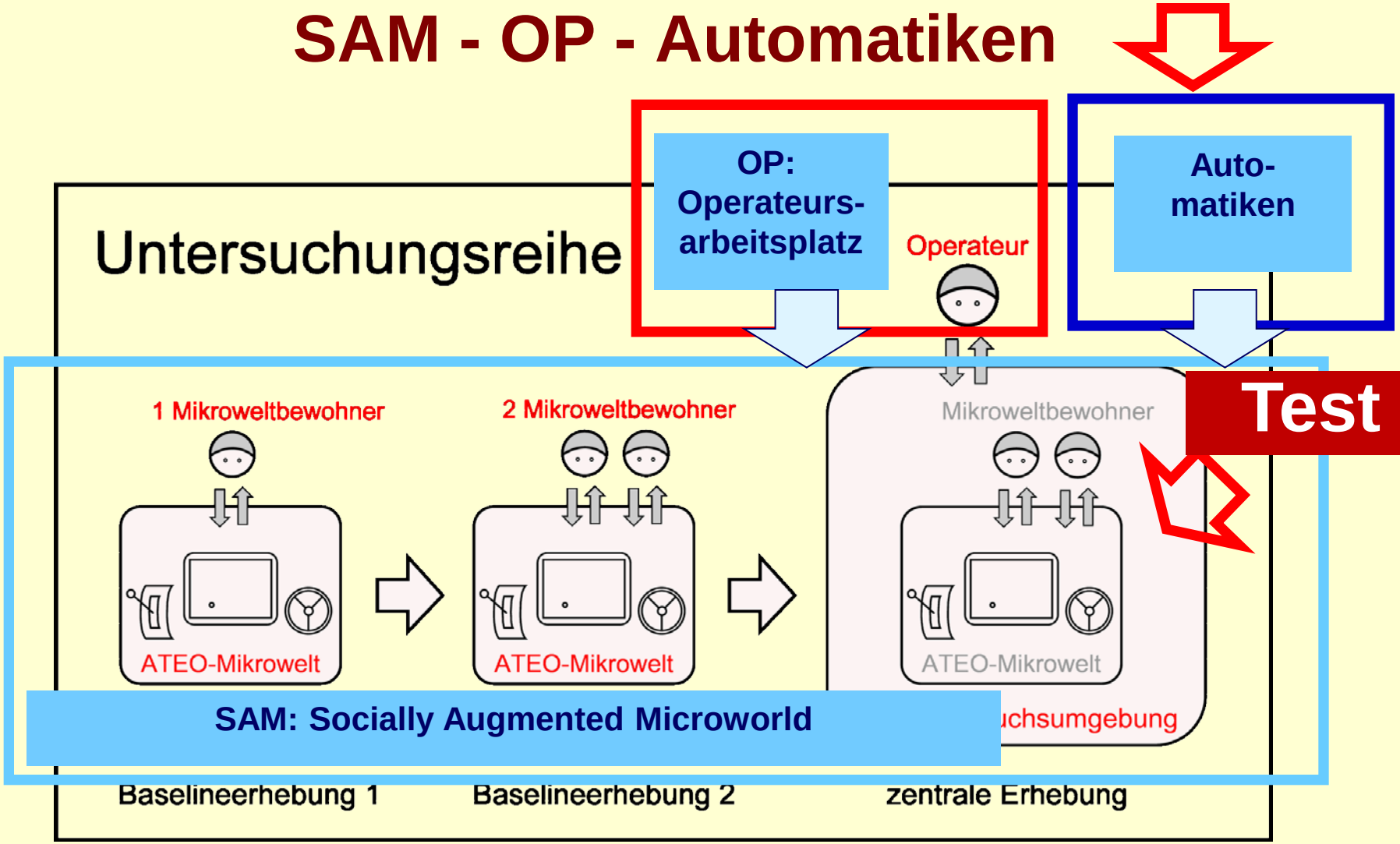
CTE XL:
Testfälle ableiten für SAM

Aufgabe: Projektseminar 2009

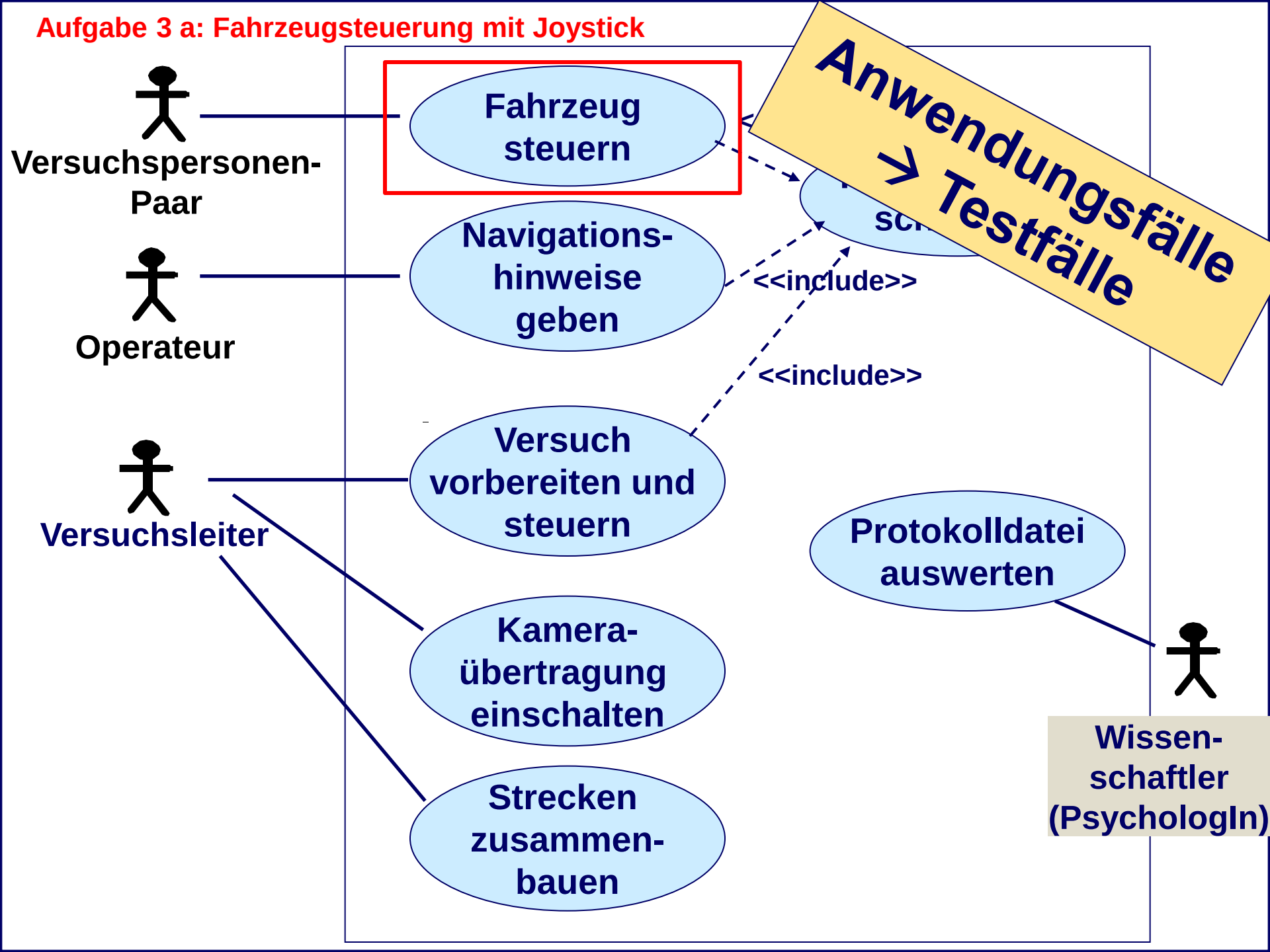
Semesterprojekt

→ Testfälle für Automatiken

Softwarekomponenten von ATEO: SAM - OP - Automatiken

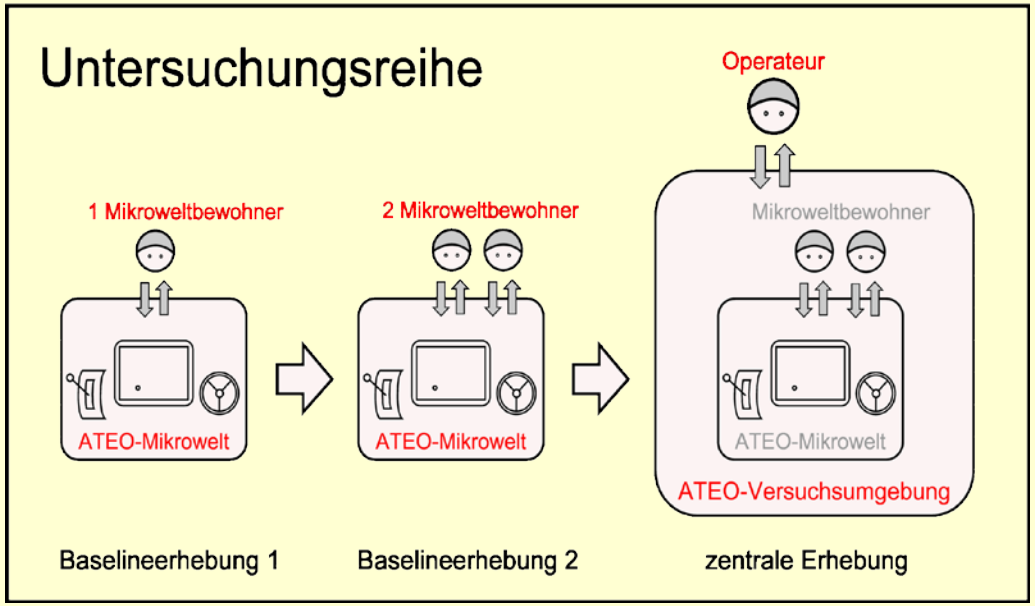


Aufgabe 3 a: Fahrzeugsteuerung mit Joystick



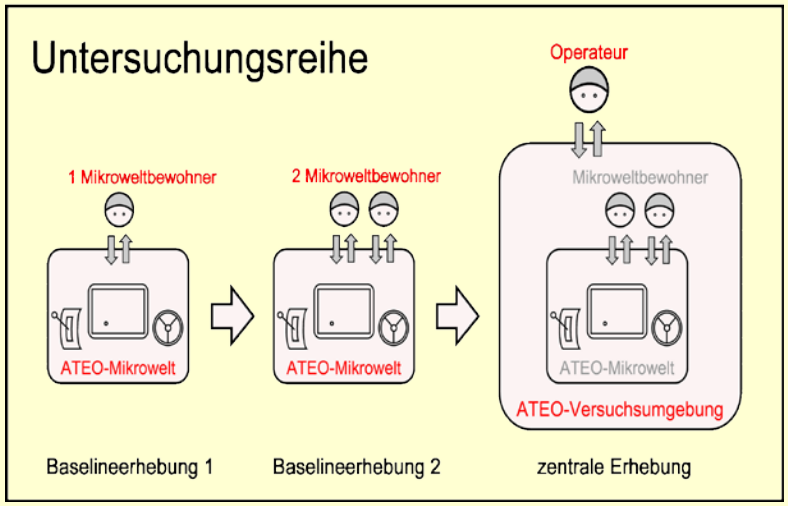
Fahrobjektsteuerung mit Joystick

- Was sind Eingabedaten?
- Aspekte und Klassen?
- Was sind die Effekte (Ausgabedaten)?



Fahrobjektsteuerung mit Joystick

- Was sind Eingabedaten?
- Aspekte und Klassen?
- Was sind die Effekte (Ausgabedaten)?



Eingabedaten:

- 1. MWB: Joystick 1
- 2. MWB: Joystick 2
- Gewichtung
- Streckenposition

Wirkung, Ausgabedaten:

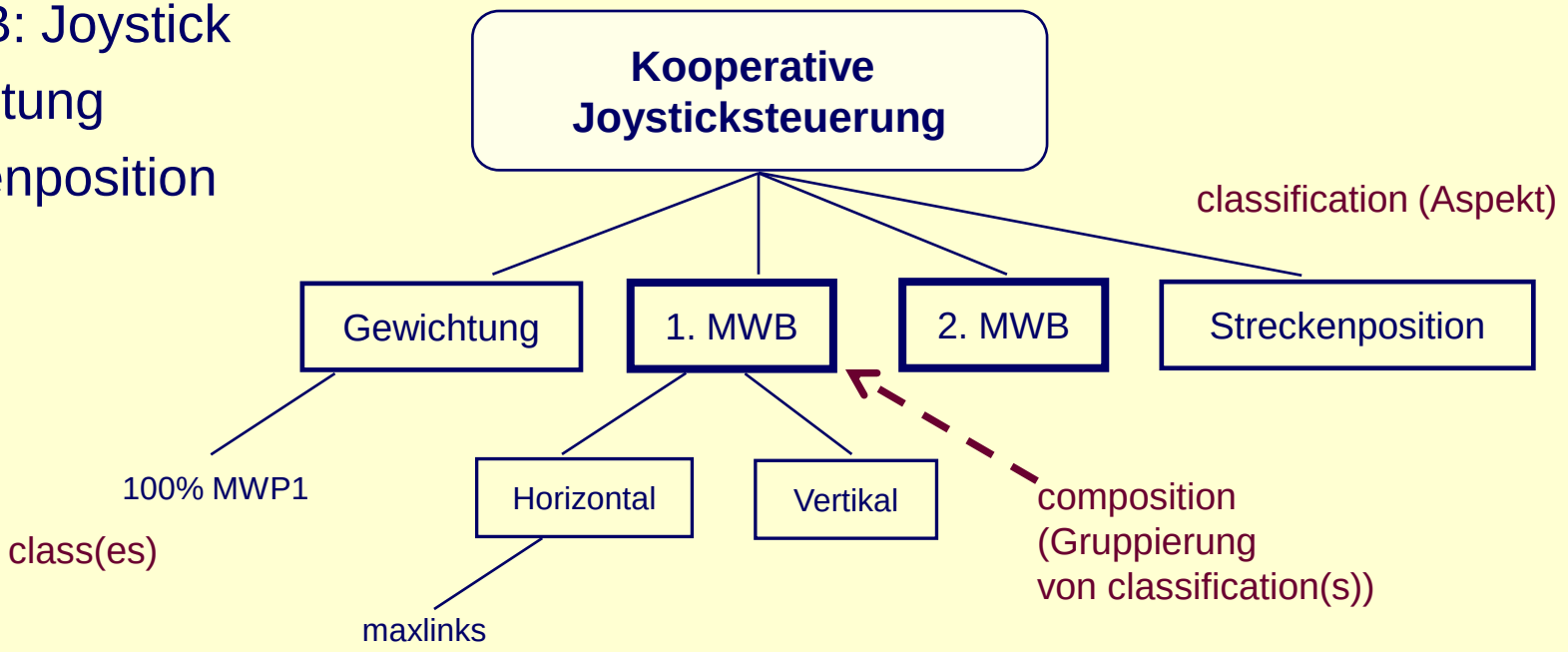
- Position des Fahrzeugs (Richtung)
- Position der Strecke (Geschwindigkeit)
- Geschwindigkeitsanzeige
- Protokolldatei

Fahrobjektsteuerung mit Joystick: Aspekte und Klassen?

Eingabedaten:

- 1. MWB: Joystick
- 2. MWB: Joystick
- Gewichtung
- Streckenposition

Ansatz für Klassifikationsbaum:



SUnit:
Tool für den Modultest

→ **Michael Hildebrandt**