Software Engineering

Vorgehensmodelle

Prof. Dr. Bodo Kraft

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Agenda der gesamten Vorlesung
Vorgehensmodelle

1. Einstieg ins Software-Engineering
2. Vorgehensmodelle
3. Anforderungsanalyse
4. Entwurf
5. Implementierung
6. Qualitätssicherung
7. Umfeld der Software-Entwicklung

Agenda für heute / diese Woche
Vorgehensmodelle

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Präsenzvorlesung

- Phasen der Software-Entwicklung
- Wasserfallmodell
- Neuere Ansätze zur Strukturierung
- Konkrete Vorgehensmodelle in der Praxis

Selbststudium

- Agile Vorgehensmodelle (Video)

Sie wissen, wie man einen Softwareentwicklungsprozess strukturieren kann.

Sie kennen die Vor- und Nachteile unterschiedlicher Vorgehensmodelle und Sie können für Ihr Projekt entscheiden, welches Vorgehensmodell am Besten passt.

Welchen Nutzen bringen Vorgehensmodelle?
Phasen der Software-Entwicklung

Einsicht

Man sollte die Gesamt-Vorgehensweise nicht in jedem Projekt neu erfinden
→ sondern sich auf **vorhandene Erfahrungen** abstützen

Prinzipien:

Planung und Koordination

→ Es senkt das Risiko, wenn alle Beteiligten im Voraus erkennen können, was wann getan werden muss

Iteration

→ Es senkt das Risiko, wenn das Projekt in kurzen Abständen einsetzbare Versionen der Software hervorbringt

Kontinuierliche Verbesserung

→ Versuche, den Prozess so zu gestalten, dass die unvermeidlich auftretenden Fehler gut ausgeglichen werden können

Welchen Nutzen bringen Vorgehensmodelle?
Phasen der Software-Entwicklung

→ **Softwareentwicklungsprozess wird transparent**
- planbar
- nachvollziehbar
- kontrollierbar
- lehrbar

→ **Softwareprodukt wird besser**
- höhere Qualität
- effizientere Produktion
- bessere Wartbarkeit

- und somit
  - schnellere Fehlerbehebung
  - erhöhte Änderungsfreundlichkeit

Erste Idee eines Vorgehensmodells

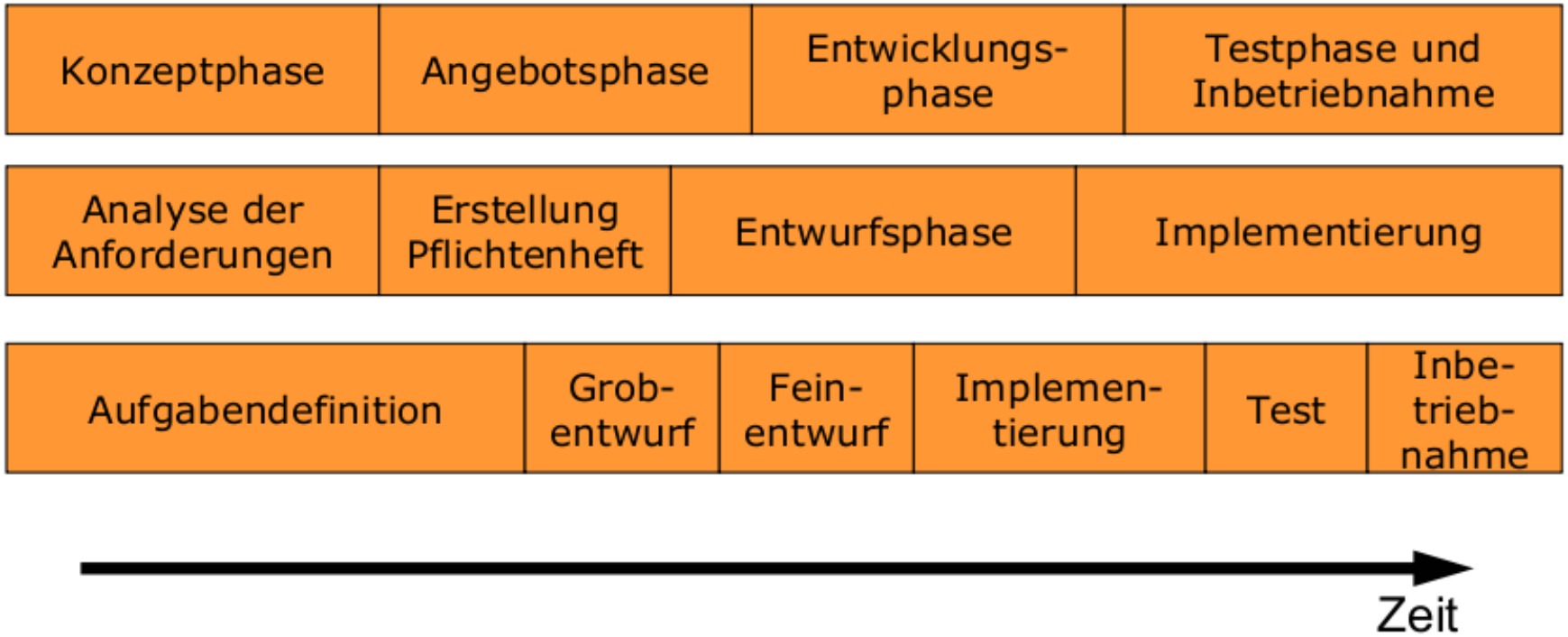
Definition von Phasen, d.h. zeitlich begrenzten Aktivitäten mit einer speziellen Aufgabe, die von Mitarbeitern mit geeigneten Rollen bearbeitet werden, um basierend auf vorgegebenen Artefakten neue, definierte Artefakte zu produzieren.

Zu einem Zeitpunkt wird nur genau eine Phase durchlaufen

→ **Phasenmodelle**

Phasen und Bezeichnungen nicht festgelegt
Phasen der Software-Entwicklung

Modelle sind oft firmenspezifisch entwickelt



Phasenmodelle entsprechen i.d.R. nicht der Realität

Klassisches sequentielles Vorgehensmodell
Wasserfallmodell

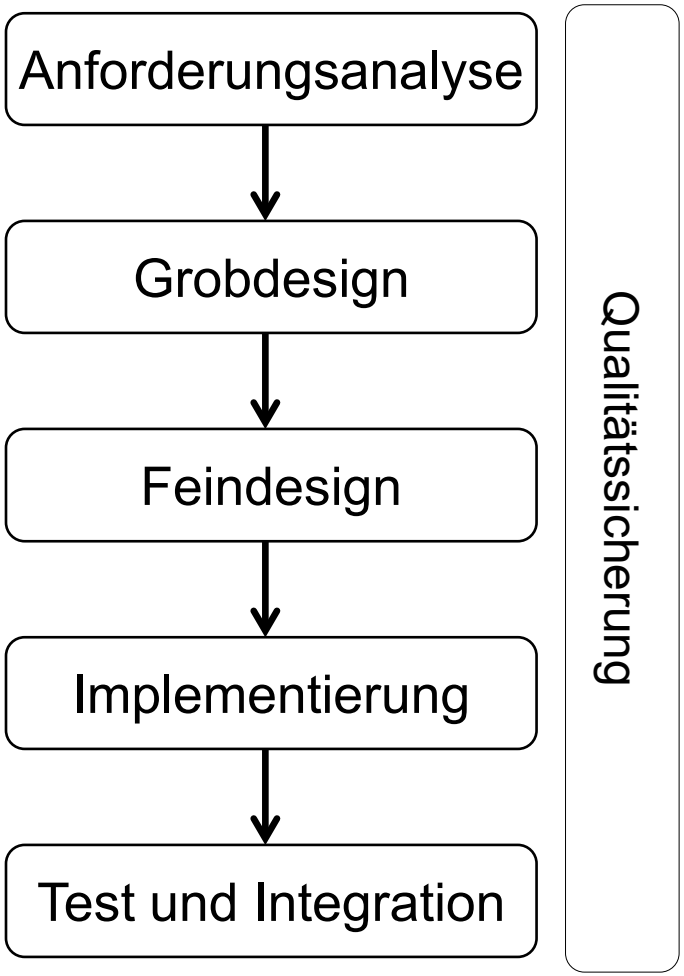
Erhebung und Festlegung des WAS mit Rahmenbedingungen

Klärung der Funktionalität und der Systemarchitektur durch erste Modelle

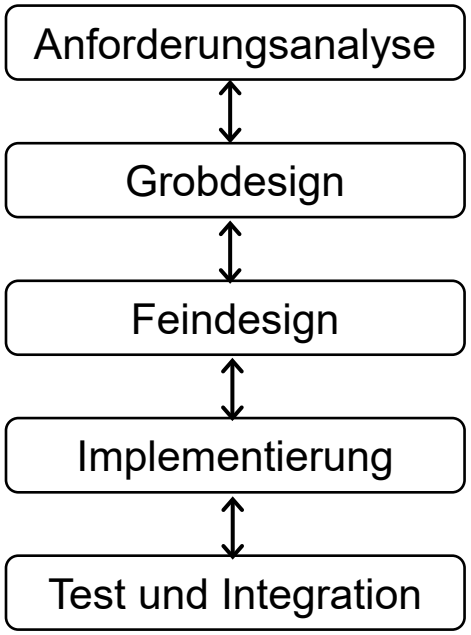
Detaillierte Ausarbeitung der Komponenten, der Schnittstellen, Datenstrukturen, des WIE

Ausprogrammierung der Programmiervorgaben in der Zielsprache

Zusammenbau der Komponenten, Nachweis, dass Anforderungen erfüllt werden, Auslieferung



Bewertung des Wasserfallmodells
Wasserfallmodell



Vorteile

- Plan auch für Nichtexperten verständlich
- einfache Planung

Nachteile

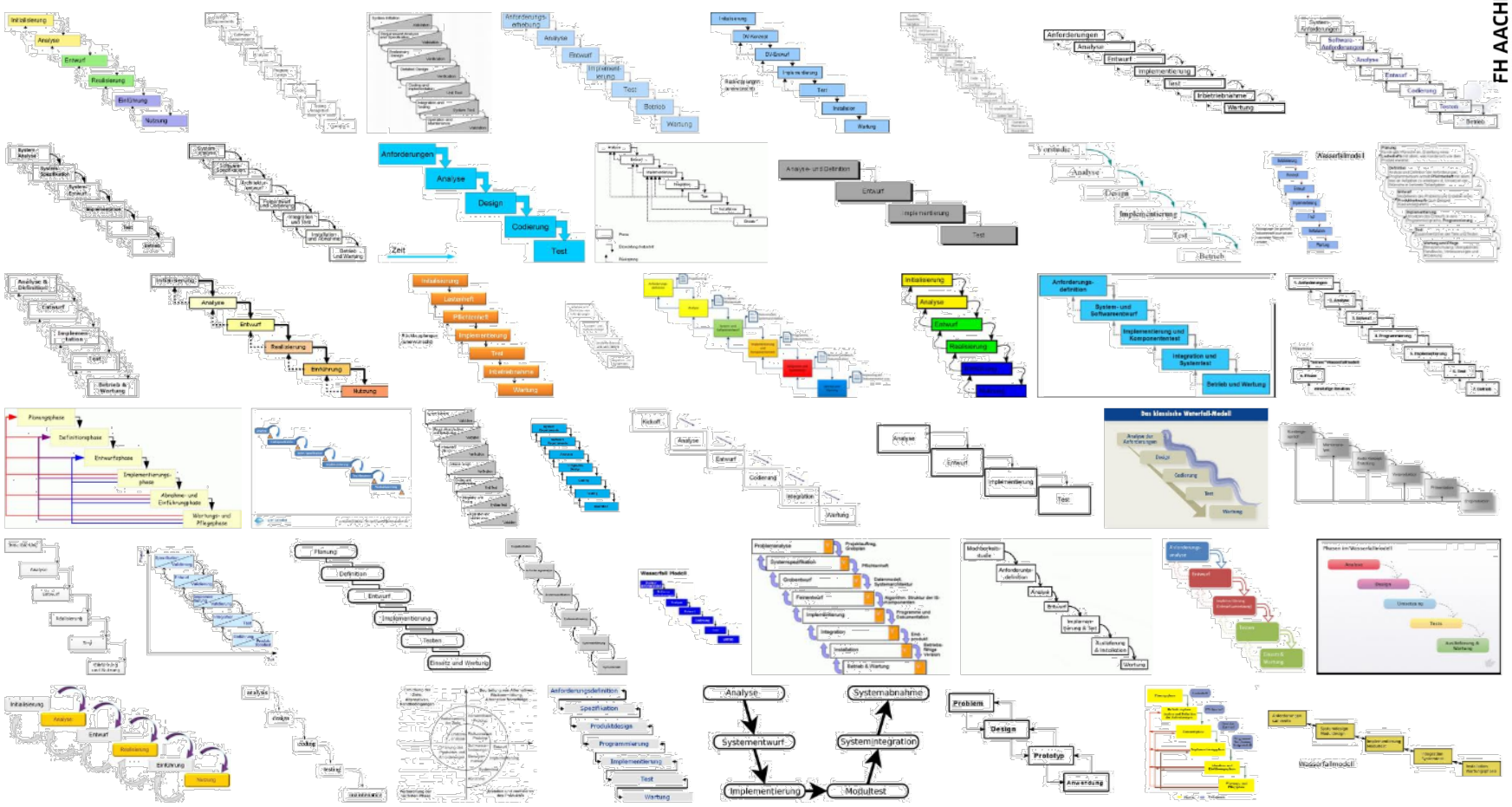
- Anforderungen müssen 100% abgeschlossen sein
- Auftraggeber ist nur in der ersten Phase eingebunden

- Entwicklungsrisiken werden spät erkannt
- Lauffähige Version des Systems erst am Ende

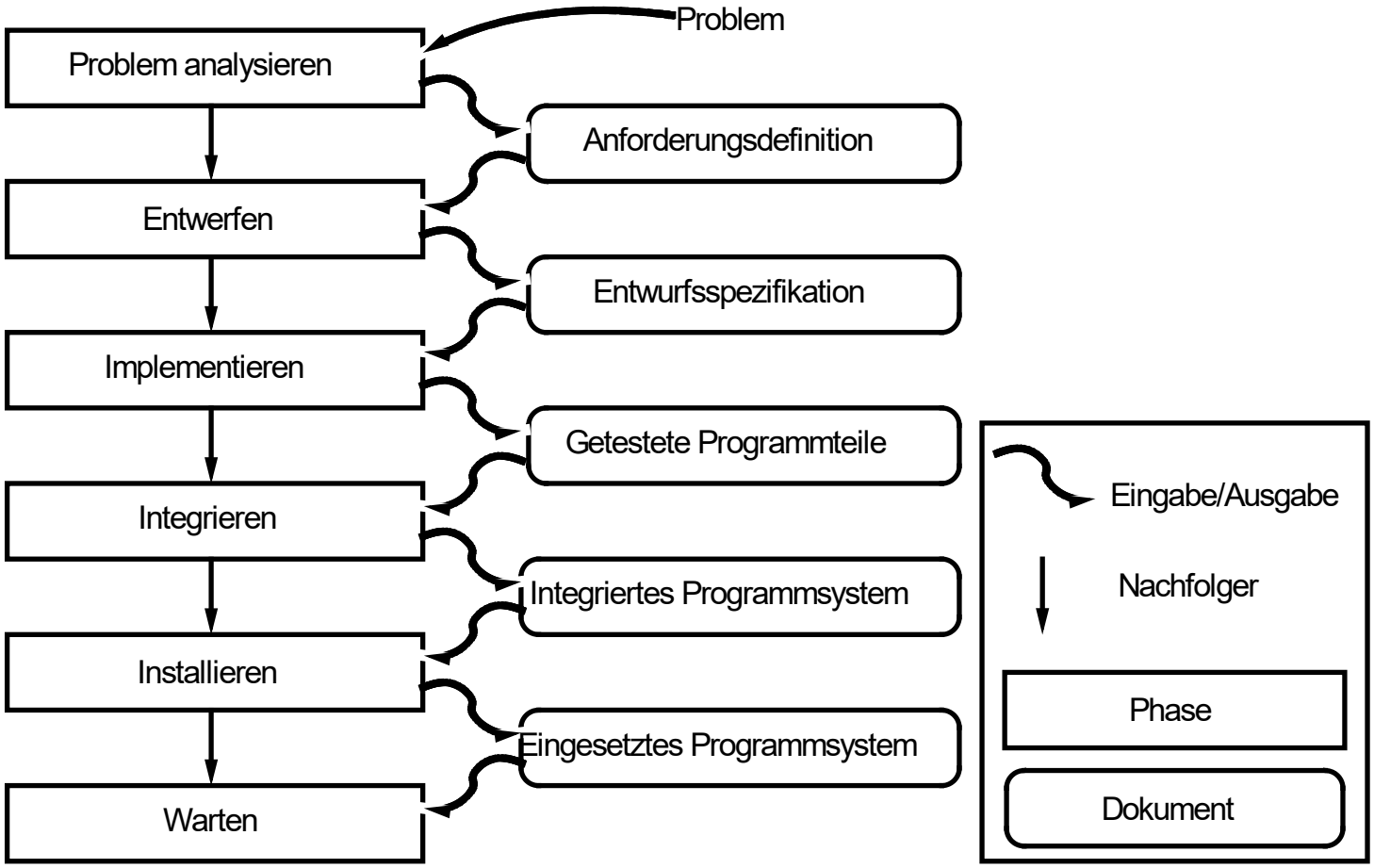
- Zeitverzug im Projekt geht oft zu Lasten späterer Phasen (z.B. beim Testen)

- Testen nur am Ende vorgesehen

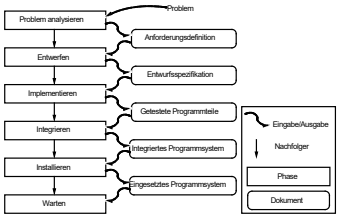
Trotz diverser Nachteile enorme Verbreitung
Wasserfallmodell



Sicht auf Aktivitäten und Artefakte
Wasserfallmodell im Detail



Phase: Problem analysieren
Wasserfallmodell im Detail



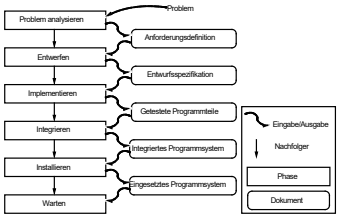
- **Kernprobleme bestimmen**
  - Welche Funktionen soll das System anbieten?
  - Wie soll sich die Bedieneroberfläche verhalten?
  - Wie effizient, sicher, … muss das System sein?
- **Relevantes Umfeld bestimmen**
  - Art, Anzahl der Benutzer
  - vorgesehene Hardware, vorhandene Software
- **Durchführbarkeit abschätzen**
  - technische, personelle Kapazität
  - Kosten (und Nutzen)

Requirements Analysis

Anforderungsdefinition (Requirements Specification, Pflichtenheft)

- Zweck des Systems, gewünschte Funktionen
- korrekte / falsche Eingaben, entsprechende Reaktionen
- Gestaltung der Bedieneroberfläche
- Effizienz-Ziele
- Anforderungen an Dokumentation
- zu verwendende Standards, Software, Hardware
- Zeitplan, Aufwands- und Risikoschätzung (grob)

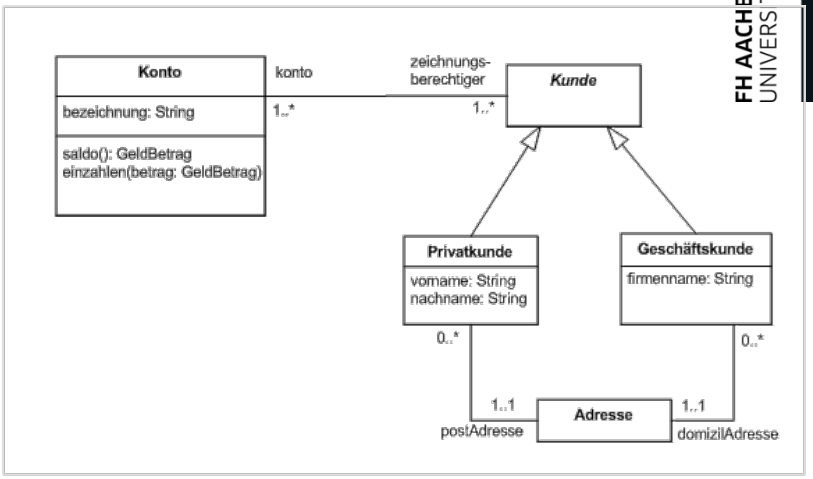
Phase: Entwerfen
Wasserfallmodell im Detail



Entwerfen (Architekturmodellierung, Design):

System grob strukturieren

- In Komponenten zerlegen
- Zusammenspiel definieren
- Entwurf vs. Anforderungen prüfen



Entwurfsspezifikation (Architektur, Design Specification):

Beschreibung der Komponenten

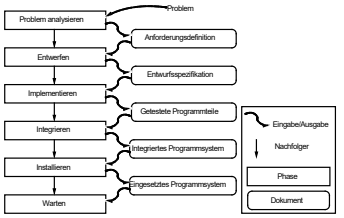
- Zweck und Rollen einer Komponente
- Von einer Komponente angebotene Dienste

Beziehungen zwischen Komponenten

- welche Komponente darf welche anderen benutzen

Begründung von Entwurfsentscheidungen

Phase: Implementieren
Wasserfallmodell im Detail



Implementieren (Coding)

Komponenten implementieren
- Datenstrukturen wählen
- Algorithmen wählen
- In Programmiersprache formulieren

Komponenten dokumentieren
- wie erledigt die Komponente ihre Aufgabe?
- Implementierungsalternativen begründen

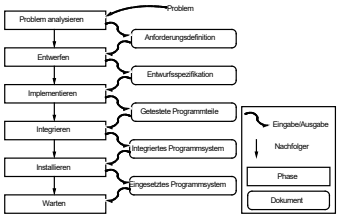
Komponenten prüfen (vs. Entwurf)
- Testumgebung einrichten, Testdaten erfassen
- Testläufe durchführen
- Verifizieren

Source-Code

```
//Datenabstraktionsmodul für Konto

public class Konto {
   // Bezeichnung der Kontoart
   // einfache Objektvariable
   private String bezeichnung;

public String getBezeichnung() {
        return bezeichnung;
   }

   ...
}
```

Getestete Programmteile (Code)
- ausprogrammierte Komponenten
- Dokumentation (meistens enthalten: Kommentare)

Phase: Integration
Wasserfallmodell im Detail



Integrieren

Komponenten zusammenfügen

- Komponenten zu Paketen
- Pakete zu Gesamtsystem

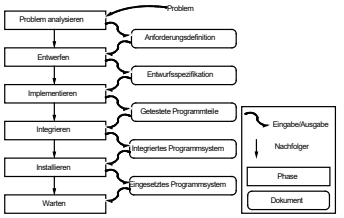
Zusammenspiel prüfen (vs. Entwurf, Anforderungen)

- Funktionalität prüfen
- Effizienz, Sicherheit prüfen

Integriertes Programmsystem

- lauffähiges Gesamtsystem
- i. a. auch Benutzerhandbuch und weitere Dokumente
- Auslieferung als Quelltext oder Objektcode

Phase: Installation / Auslieferung
Wasserfallmodell im Detail



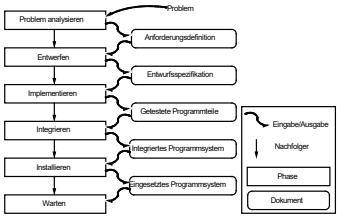
Installieren

- System beim Auftraggeber installieren
- Daten migrieren
- Anpassen
- Altsystem deinstallieren
- Abnahme durch Auftraggeber
- System übergeben
- Benutzer schulen

Eingesetztes Programmsystem

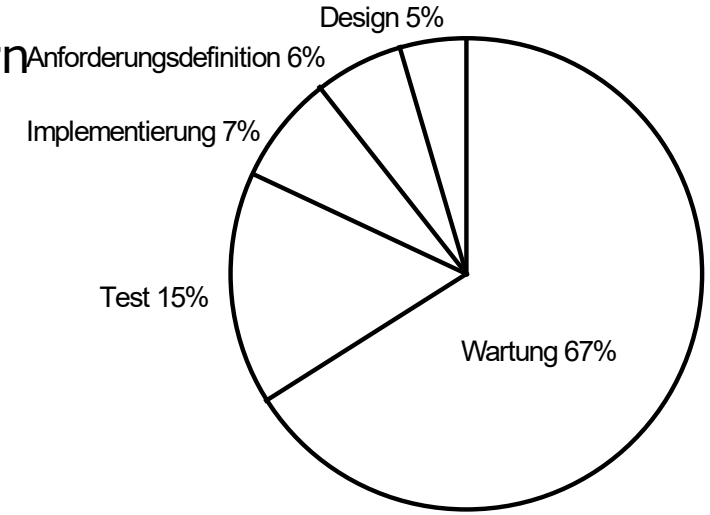
- im Betrieb befindliches Gesamtsystem
- Benutzerhandbuch und weitere Dokumente

Phase: Installation / Auslieferung
Wasserfallmodell im Detail



Warten (Maintaining)

- Fehler beheben
  - algorithmische Fehler
  - Fehler bezüglich Anforderungen
- Modifizieren
  - Auf andere Hardware portieren
  - Funktionalität erweitern / verbessern



Agenda für heute
Vorgehensmodelle

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Phasen der Software-Entwicklung

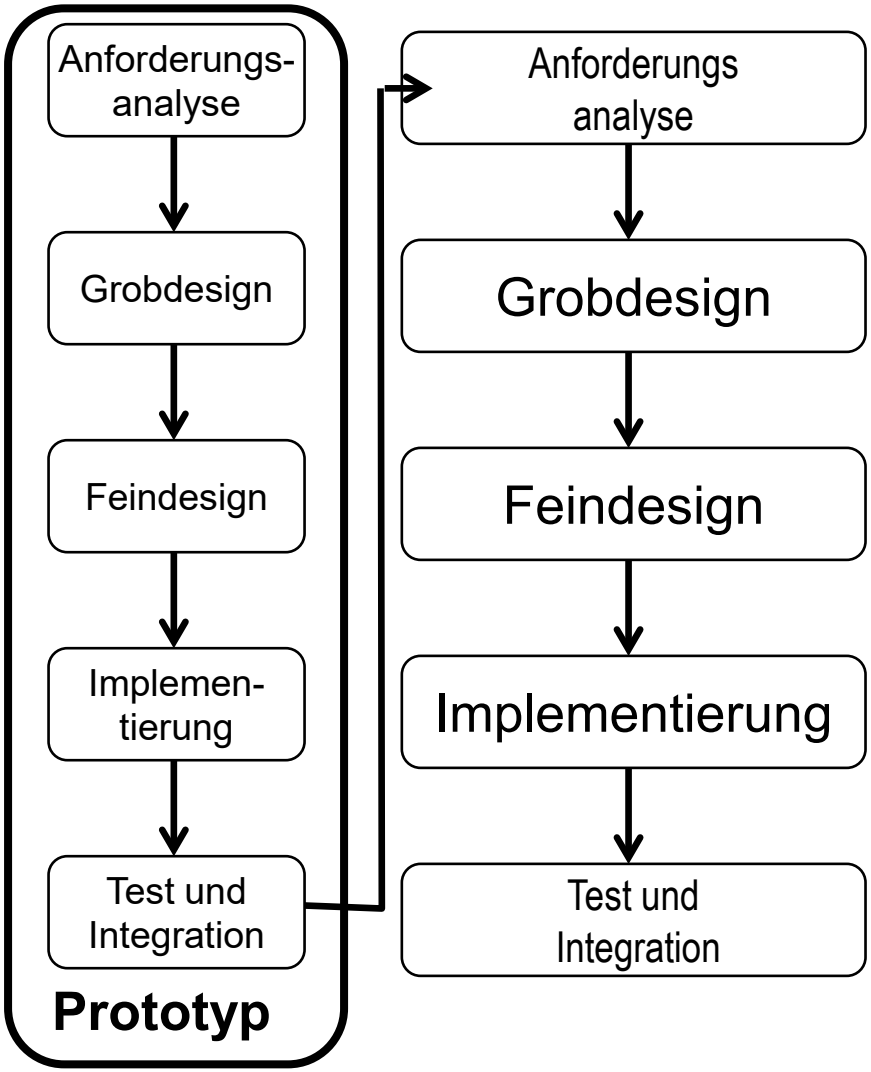
Wasserfallmodell

Neuere Ansätze zur Strukturierung

Konkrete Vorgehensmodelle in der Praxis

Agile Vorgehensmodelle

Prototypische Entwicklung
Neuere Ansätze zur Strukturierung

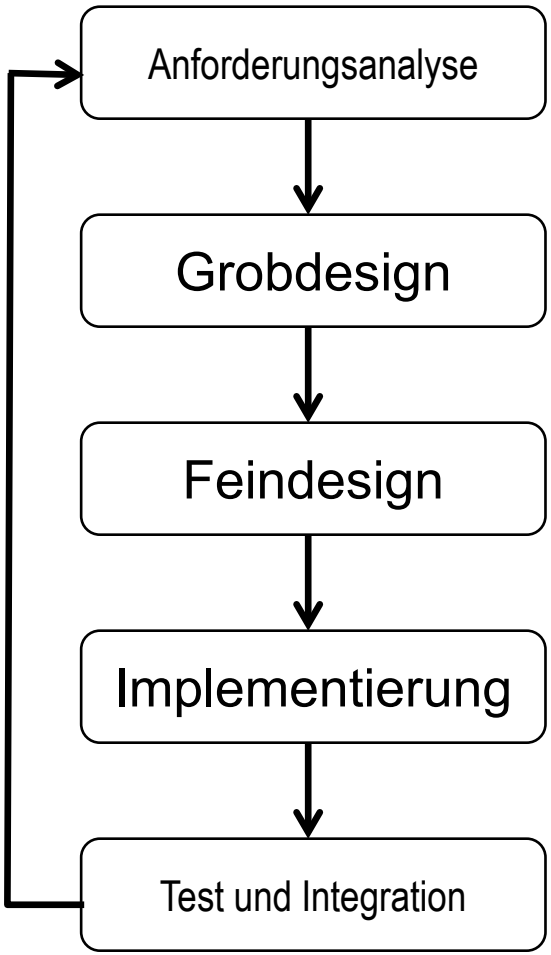


Merkmale
- potenzielle Probleme frühzeitig identifiziert
- Lösungsmöglichkeiten im Prototypen gefunden, daraus Vorgaben abgeleitet

Vorteile
- frühzeitige Risikominimierung
- schnell erstes Projektergebnis

Nachteile
- Anforderungen müssen 100%-tig sein
- Prototyp (illegal) in die Entwicklung übernommen
- Kunde erwartet schnell Endergebnis

Iterative Entwicklung
Neuere Ansätze zur Strukturierung



Merkmale

- Erweiterung der Prototypidee
  → SW wird in Iterationen entwickelt
- In jeder Iteration wird System weiter verfeinert
- In ersten Iterationen Schwerpunkt auf Analyse und Machbarkeit; später auf Realisierung

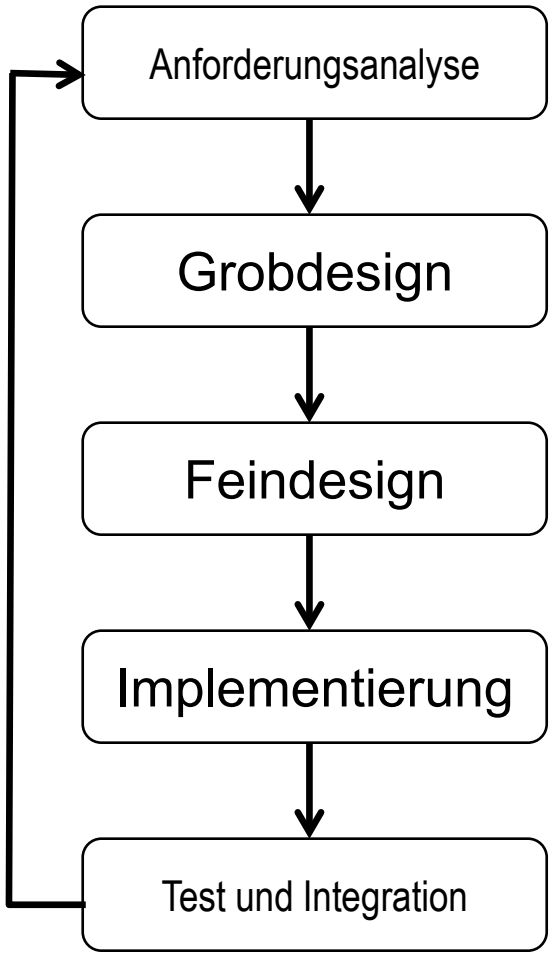
große Vorteile

- dynamische Reaktion auf Risiken
- Teilergebnisse mit Kunden diskutierbar

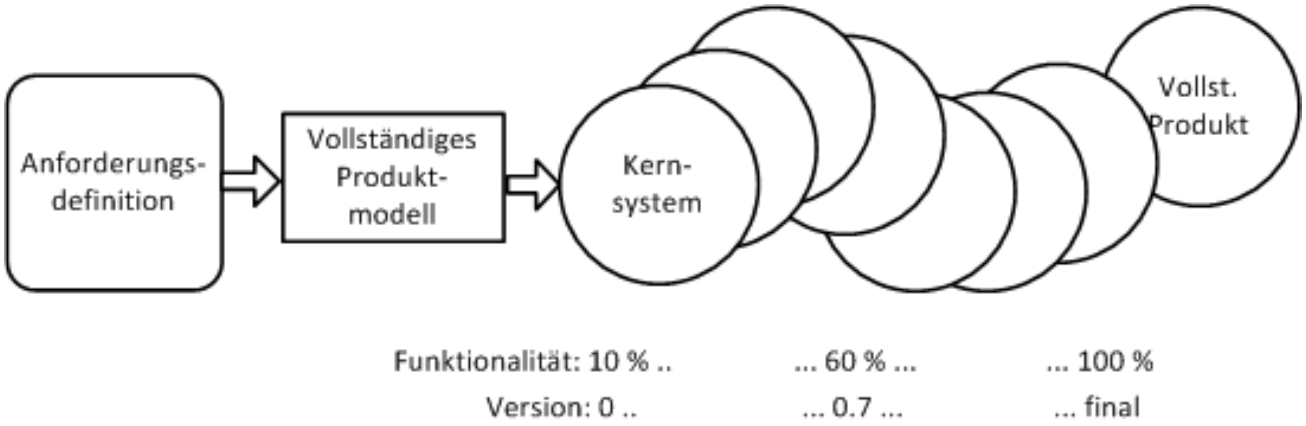
Nachteile im Detail

- schwierige Projektplanung
- schwierige Vertragssituation
- Kunde erwartet zu schnell Endergebnis
- Kunde sieht Anforderungen als beliebig änderbar

Iterative Entwicklung
Neuere Ansätze zur Strukturierung



Inkrementelle Entwicklung
Neuere Ansätze zur Strukturierung

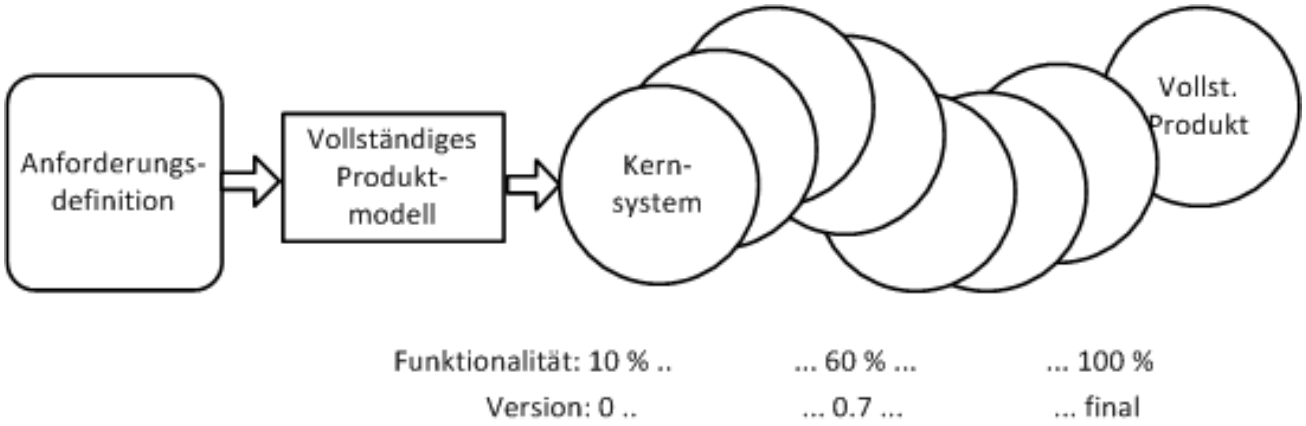


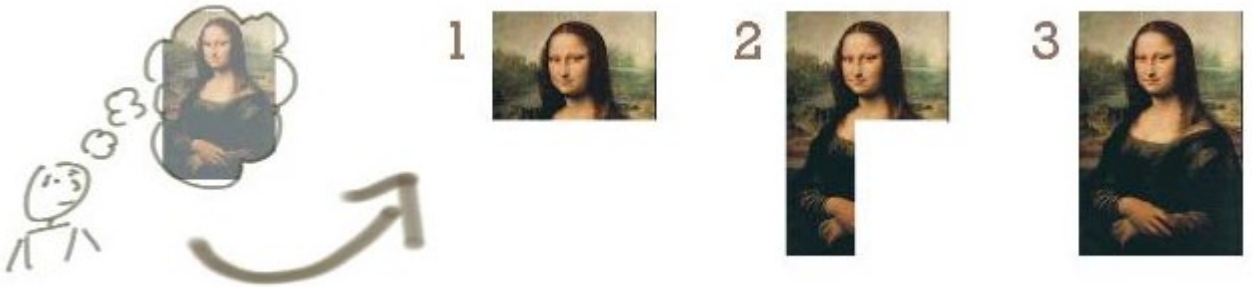
Anforderungen vollständig erfassen und modellieren

Auf Kernanforderungen des Auftraggebers konzentrieren

- Zunächst nur Kernsystem entwickeln (Null-Version)
- Kernsystem ausliefern
- Anwender sammeln Erfahrungen und äußern Wünsche

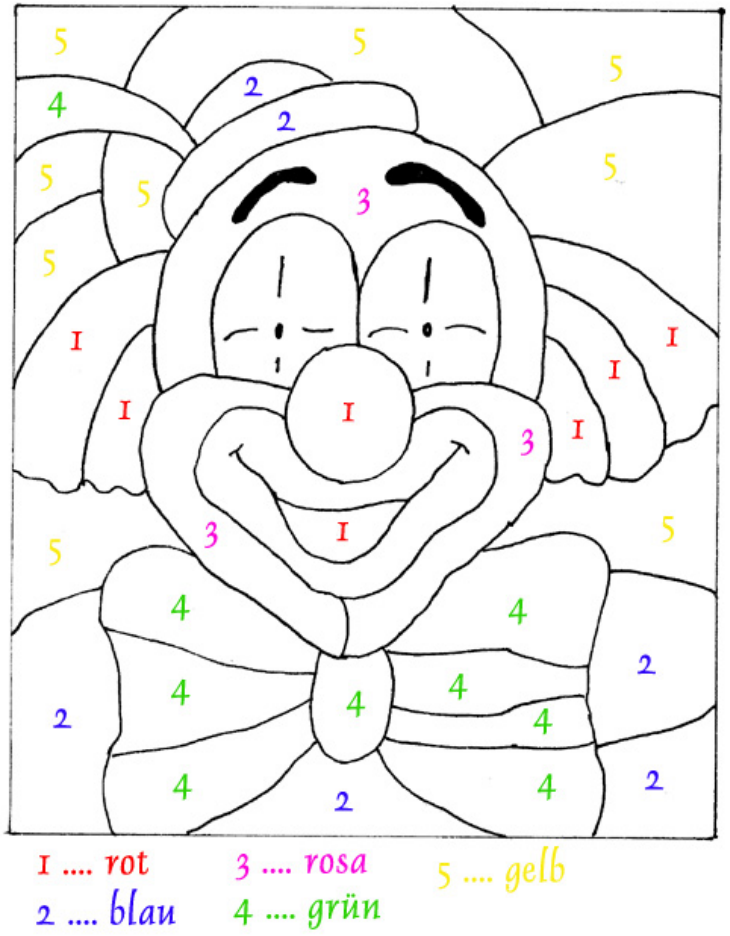
Inkrementelle Entwicklung
Neuere Ansätze zur Strukturierung





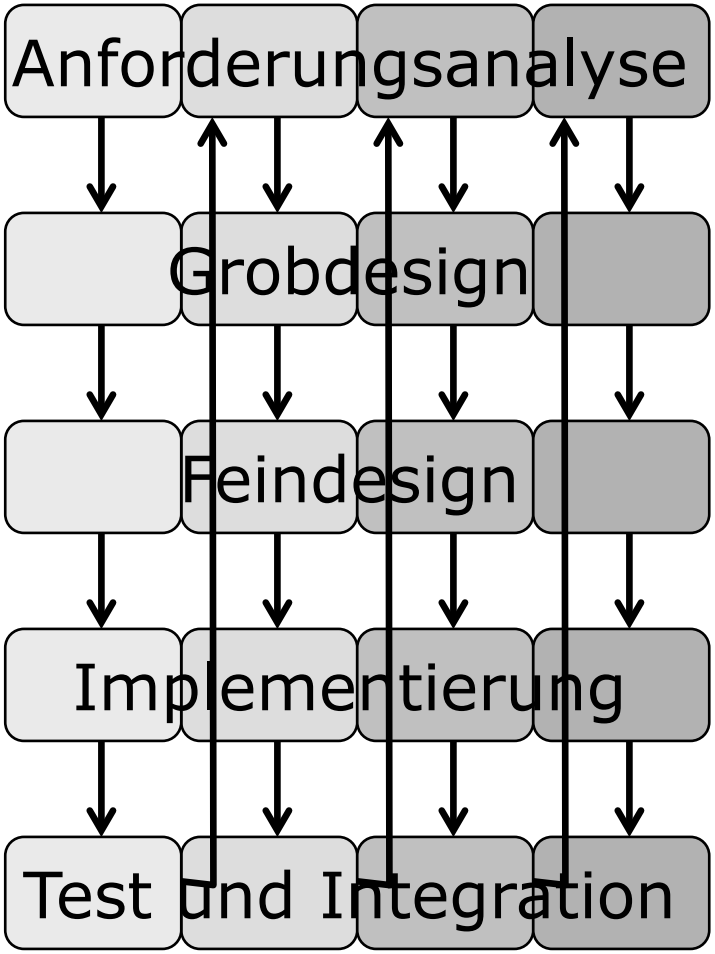
Webshop: Welche Inkremente entwickeln Sie zuerst?

Iterativ oder Inkrementell?



Iterativ Inkrementelle Entwicklung (State of the Art)

Neuere Ansätze zur Strukturierung



Bsp.: vier Inkremente

Merkmal

- Projekt in kleine Teilschritte zerlegt
- pro Schritt neue Funktionalität (Inkrement) + Überarbeitung existierender Ergebnisse (Iteration)
- n+1-ter Schritt kann Probleme des n-ten Schritts lösen

Vorteile

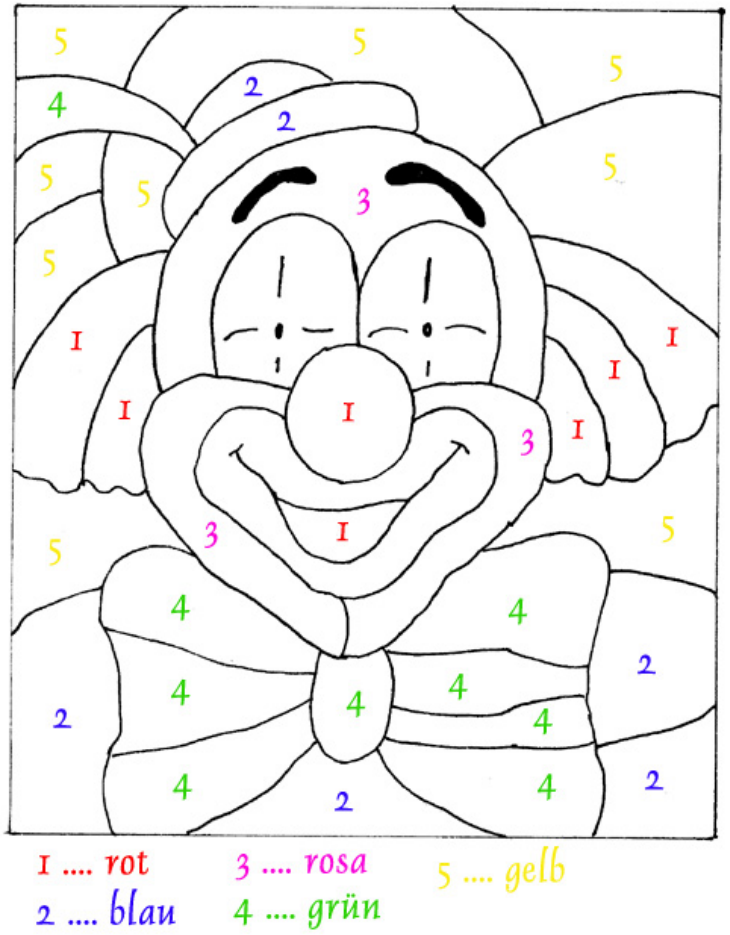
- siehe „iterativ"
- flexible Reaktion auf neue funktionale Anforderungen

Nachteile

- siehe „iterativ" (etwas verstärkt)

Anwendungsbeispiel

Iterativ oder Inkrementell oder inkrementell iterativ?



Agenda für heute
Vorgehensmodelle

Phasen der Software-Entwicklung

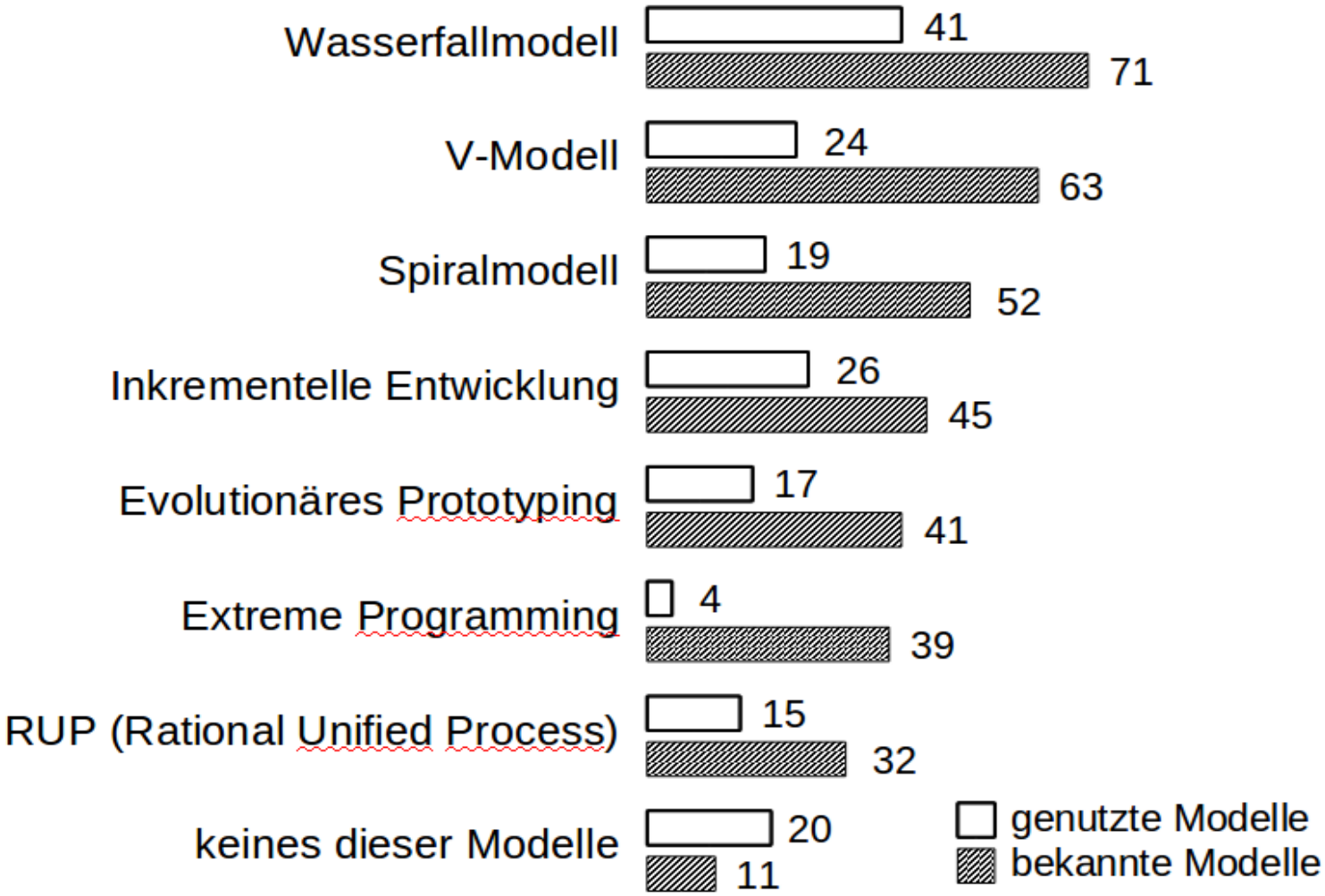
Wasserfallmodell

Neuere Ansätze zur Strukturierung

Konkrete Vorgehensmodelle in der Praxis

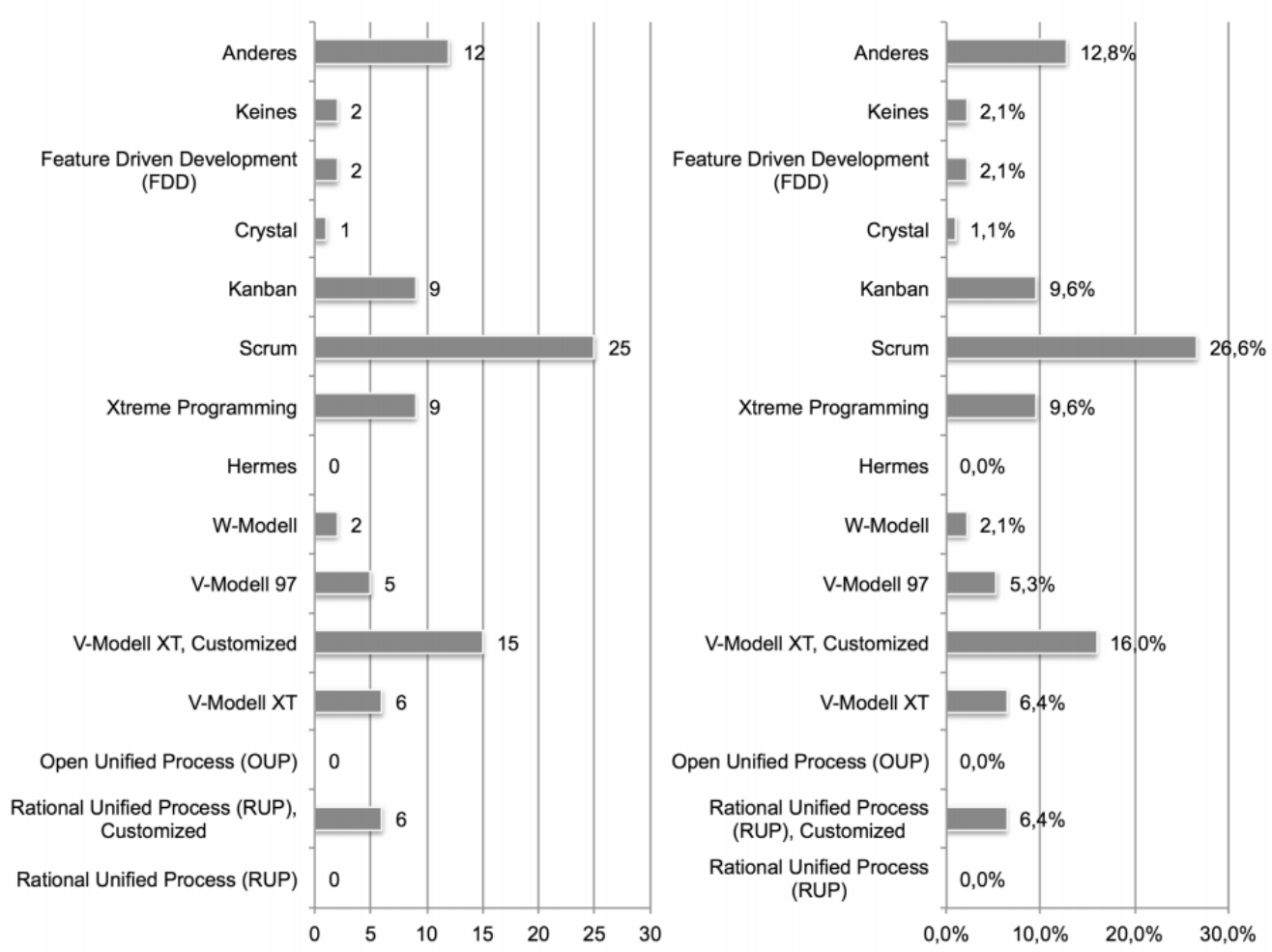
Agile Vorgehensmodelle

Bekanntheit von Vorgehensmodellen
Konkrete Vorgehensmodelle in der Praxis



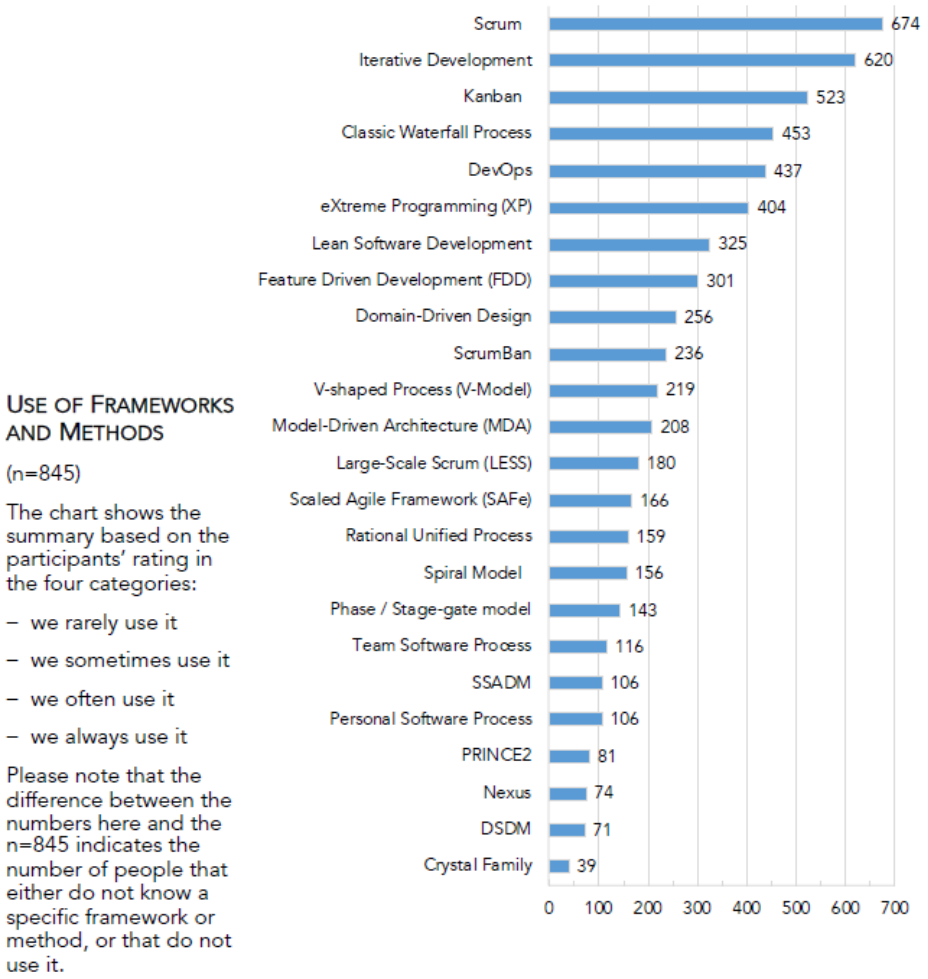
Quelle: Softwareentwicklung läuft nicht auf Zuruf, Computer Zeitung Nr. 46/05

Bekanntheit von Vorgehensmodellen
Konkrete Vorgehensmodelle in der Praxis



Quelle: Kategorisierung etablierter Vorgehensmodelle und ihre Verbreitung in der deutschen Software-Industrie, 2007
Martin Fritzsche, Patrick Keil, TUM

Bekanntheit von Vorgehensmodellen
Konkrete Vorgehensmodelle in der Praxis



Quelle: HELENA Stage 2 Results: Marco Kuhrmann und andere, 2018

Grundidee alle Vorgehensmodelle sehr ähnlich
Konkrete Vorgehensmodelle in der Praxis

Alle Vorgehensmodelle, wie

V-Modell XT des Bundes

(Rational) Unified Process

OEP (Object Engineering Process)

enthalten

- **Aktivitäten** (was soll gemacht werden),
- **Rollen** (wer ist wie an Aktivität beteiligt) und
- **Produkte** (was wird benötigt; bzw. ist Ergebnis)

Vorgehensmodelle sind als Rahmenwerke zu verstehen

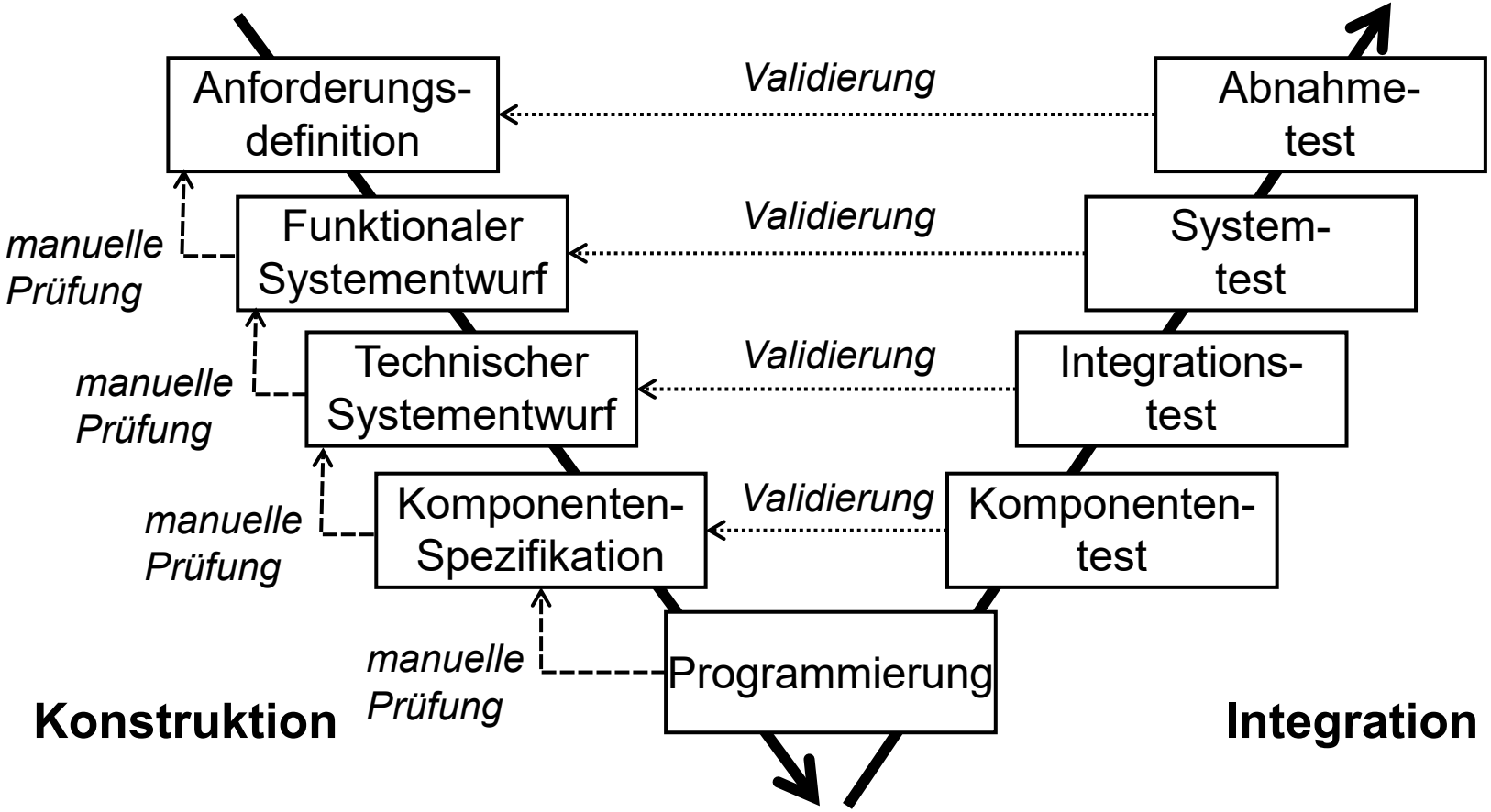
→ Zu Projektbeginn „tailoring" durchführen

→ Nutzen verstehen und pragmatisch anwenden

→ Prozess kontinuierlich verbessern

Phasen des V-Modells

Konkrete Vorgehensmodelle in der Praxis



Anmerkung: wird iterativ / inkrementell zum W-Modell

V-Modell des Bundes
Konkrete Vorgehensmodelle in der Praxis

Regelung der Softwarebearbeitung

- Standard im Bereich der Bundeswehr, des Bundes und der Länder
- einheitliche und (vertraglich) verbindliche Vorgabe von
  - Aktivitäten und
  - Produkten (Ergebnissen)

Historie

V-Modell 92 (Wasserfall im Mittelpunkt),

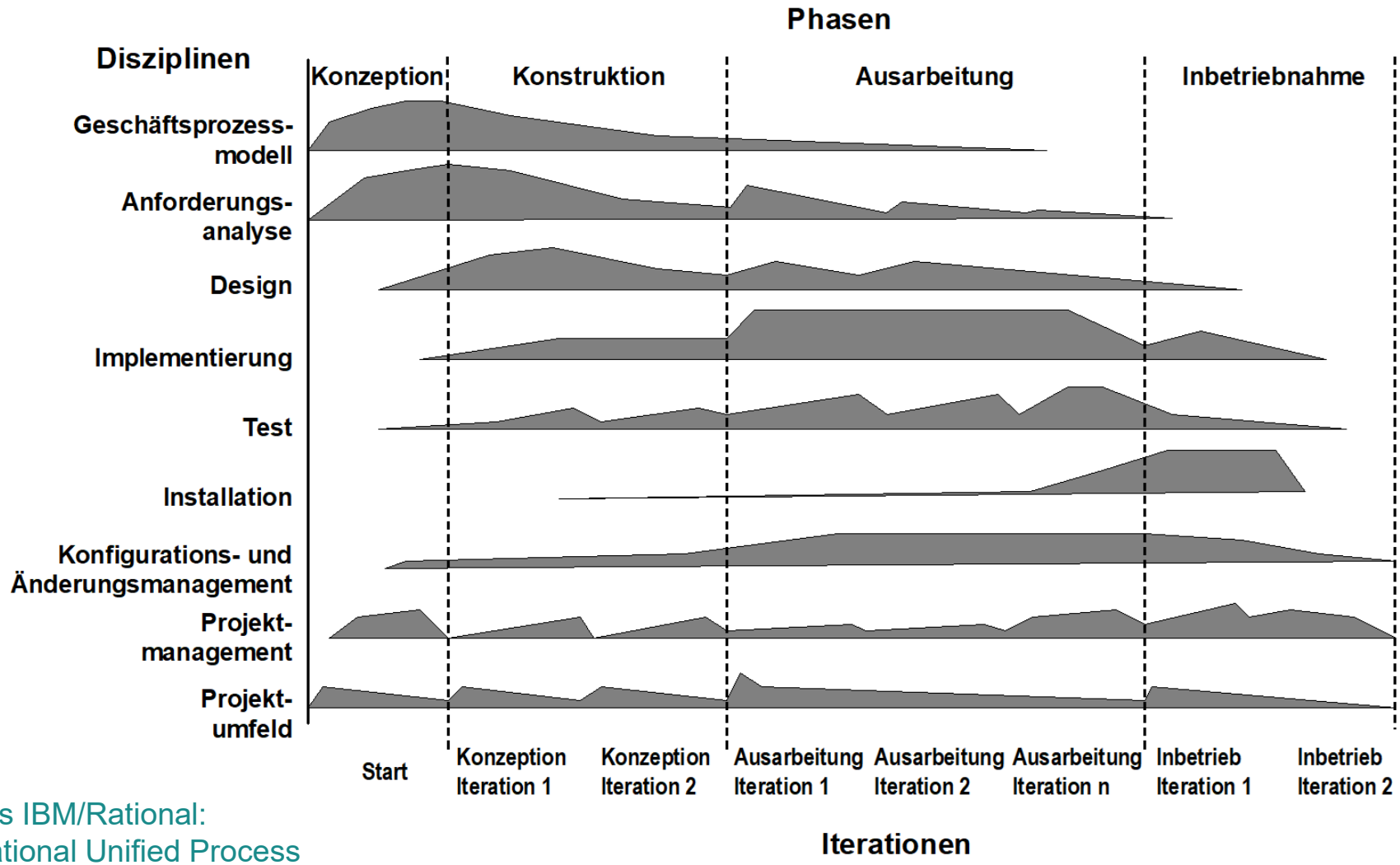
Überarbeitung **V-Modell 97** (Anpassung an inkrementelle Ideen (W-Modell); Forderung nach zu früher Festlegung von Anforderungen)

aktuell: **V-Modell XT** (eXtreme Tailoring)

Fokus auf Verhältnis von Auftragnehmer und Auftraggeber (starker akademischer Einfluss bei Entwicklung)

http://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/2.3/Dokumentation/V-Modell-XT-HTML/4ffe144f97a502a.html

Rational Unified Process (RUP)
Konkrete Vorgehensmodelle in der Praxis



aus IBM/Rational:
Rational Unified Process

Phasen des RUP
Konkrete Vorgehensmodelle in der Praxis

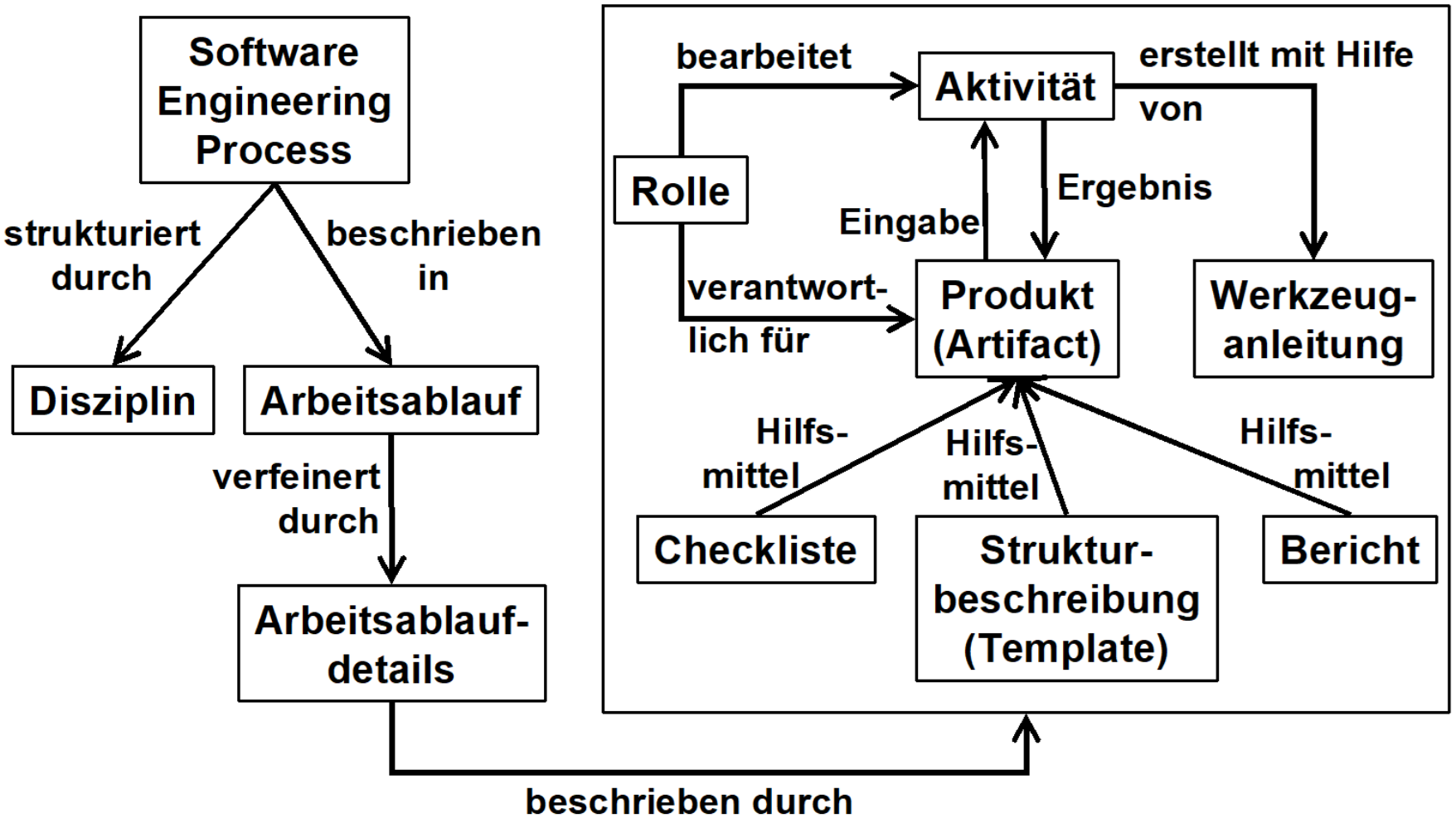
inception (Konzeption): Ermittlung zentraler Anforderungen, Projektumfang definieren, erste Entwurfs- und Implementierungsansätze, Identifikation der Projektrisiken und Aufwände

elaboration (Ausarbeitung): stabile, möglichst vollständige Anforderungen, Entwurfsspezifikation, detaillierter Projektplan mit aktivem Risikomanagement

construction (Konstruktion): Implementierung, Integration, auslieferbare Version

transition (Inbetriebnahme): Beta-Test, Endabnahme, Inbetriebnahme, Endlieferung

Struktur des RUP
Konkrete Vorgehensmodelle in der Praxis



http://www.utm.mx/~caff/doc/OpenUPWeb/index.htm

Kritik an Vorgehensmodellen
Konkrete Vorgehensmodelle in der Praxis

- Es müssen viele Dokumente erzeugt und gepflegt werden
- Eigene Wissenschaft, Modelle wie V-Modelle und RUP zu verstehen und zurecht zu schneidern
- Prozessbeschreibungen hemmen Kreativität
- Anpassung an neue Randbedingungen, z. B. neue Technologien (Web-Services) in Prozessen und benutzten Werkzeugen ist extrem aufwändig

Alternativer Ansatz

„Menschen machen Projekte erfolgreich, traue den Menschen"

→ agile Prozesse

(vorherige Name: leichtgewichtige Prozesse)