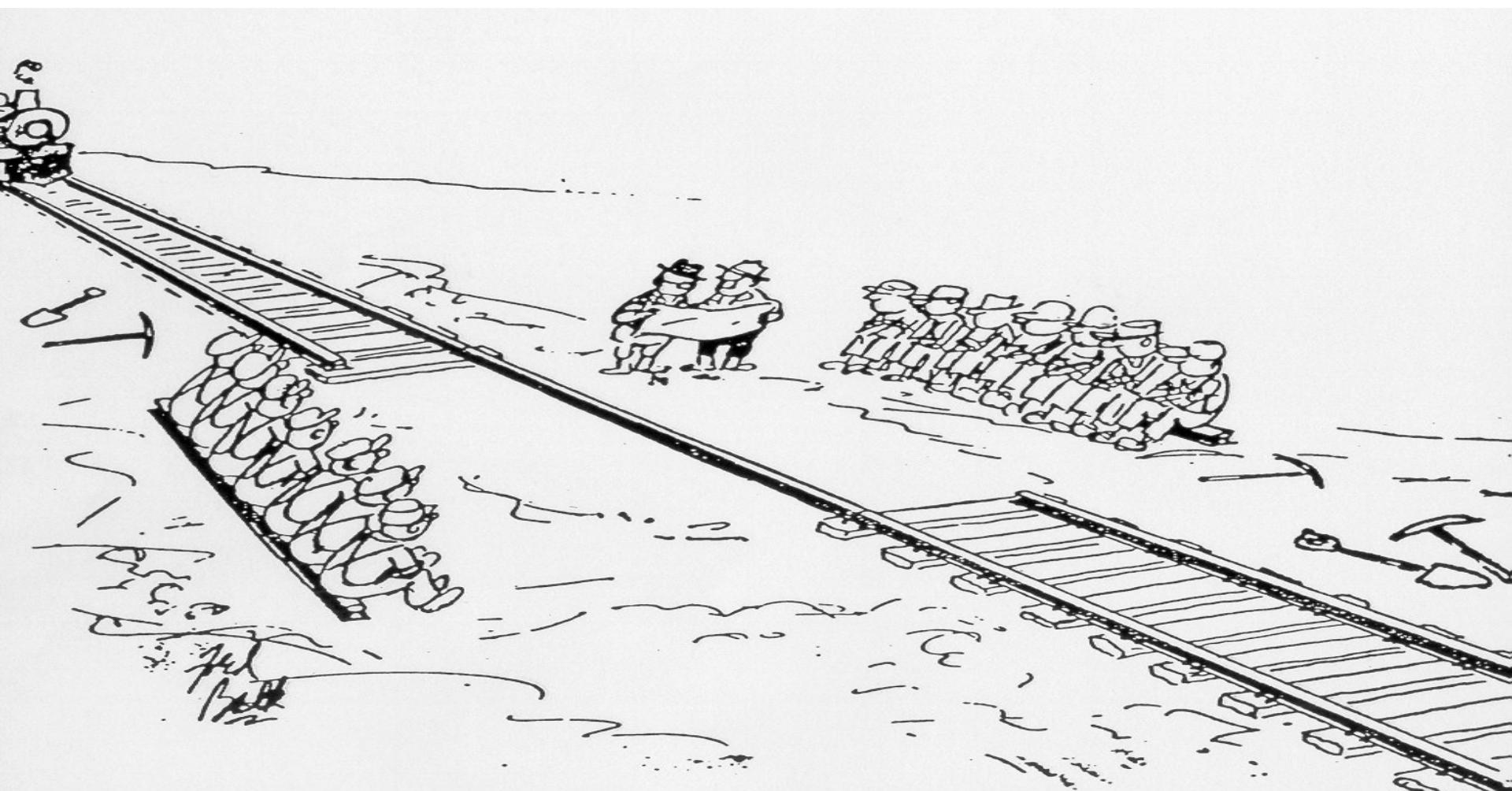


Software Engineering **Motivation**

2009/2010

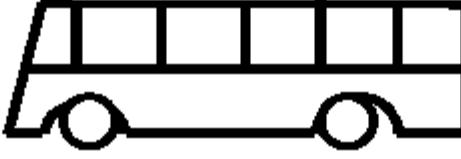
Dr. Thomas Stütz

Projektmanagement - Rückblick



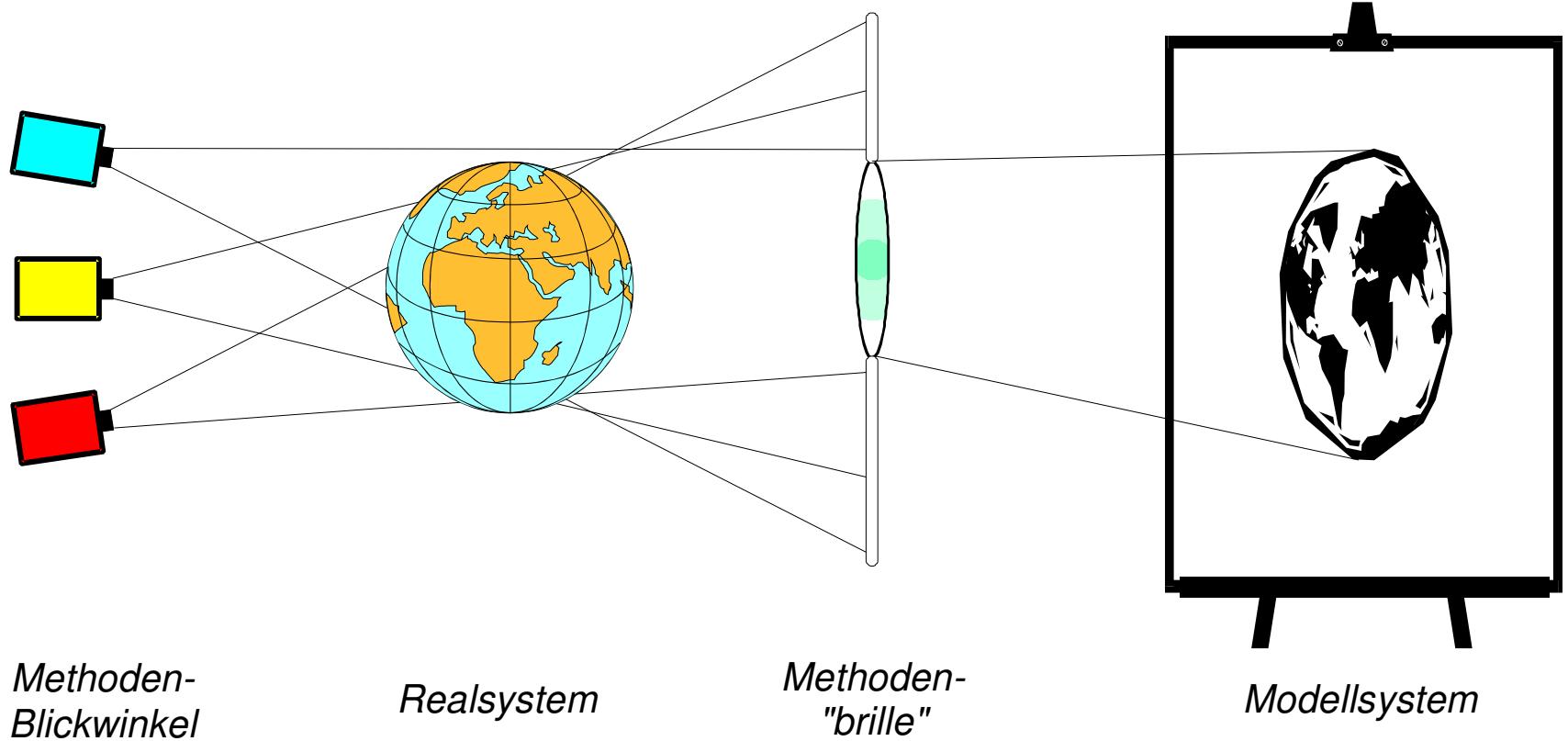
**Was ist im 3.JG schief
gelaufen?**

Anforderungsdefinition - Problematik

<p>Was der Anwender wollte</p> 	<p>Wie es der Anwender dem Programmierer sagte</p> 	<p>Wie es der Programmierer verstanden hat</p> 
<p>Was der Programmierer bauen wollte</p> 	<p>Was der Programmierer tatsächlich gebaut hat</p> 	<p>Was der Anwender tatsächlich gebraucht hätte</p> 

Anforderungsanalyse aus allen **Blickwinkeln** betrachtet?

4



Business Process Reengineering durchgeführt?

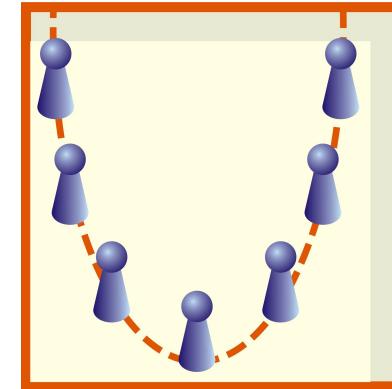
1985

Mauern zwischen
Abteilungen



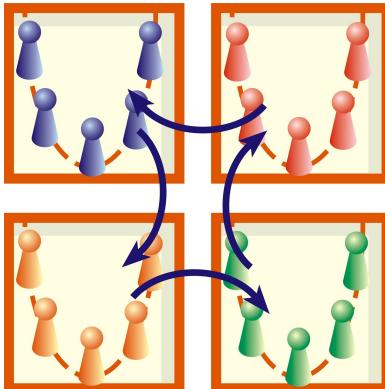
1995

Geschäftsprozesse



1999

Mauern zwischen
Unternehmen



Business Process
Reengineering

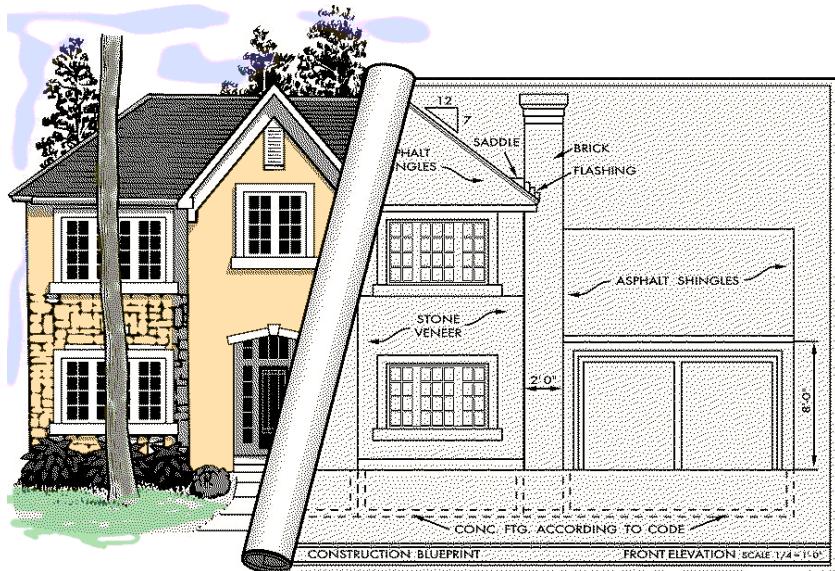
Seit 2000

E-Business

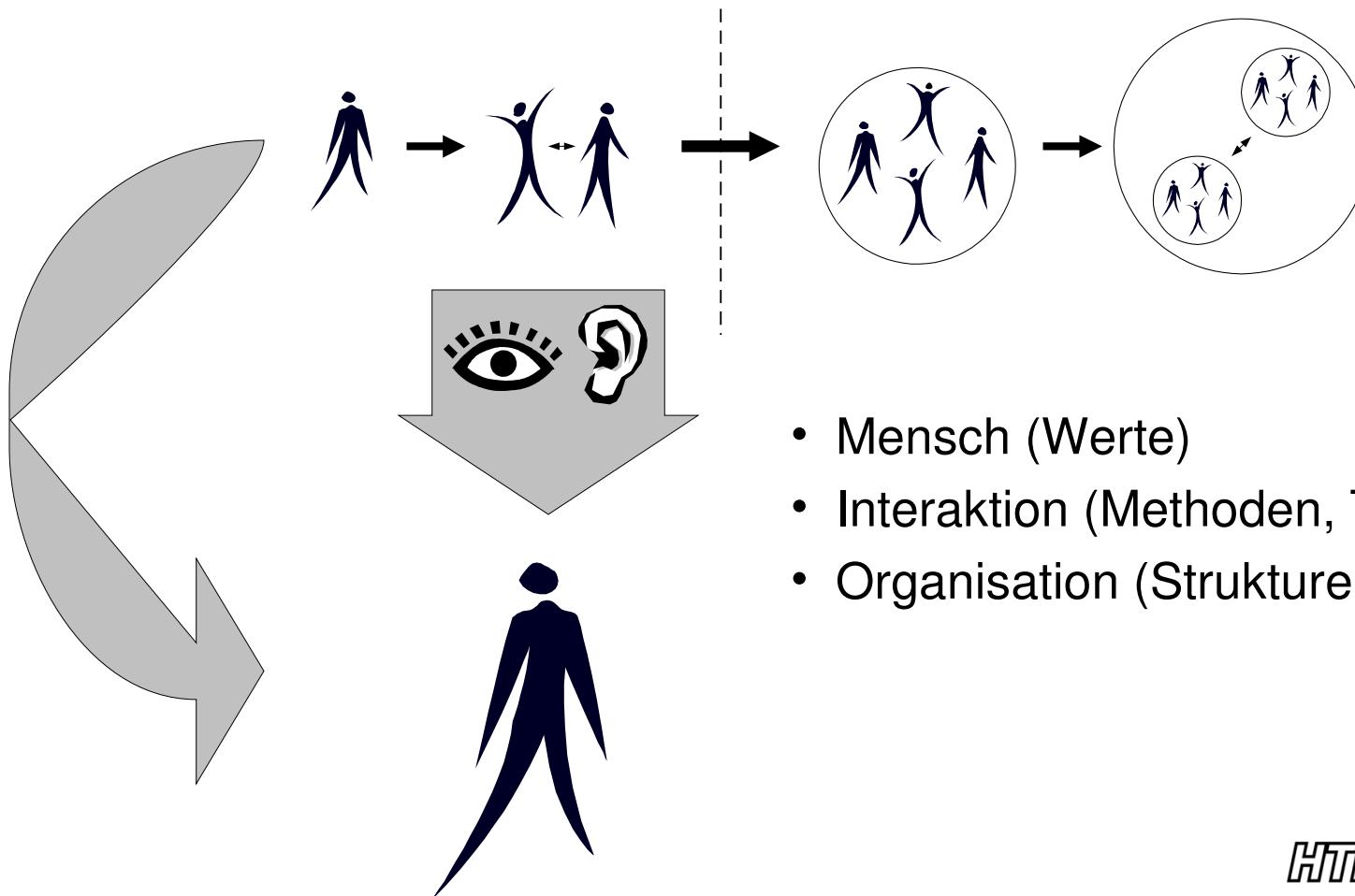
E-Business
Engineering



Planungskomponente ausreichend?



Change Management definiert und exekutiert?



- Mensch (Werte)
- Interaktion (Methoden, Tools, Medien)
- Organisation (Strukturen, Regeln)

Vorgehensmodelle (Prozessmodelle)

Wiederholung und Vertiefung

Vorgehensmodelle

- **Vorgehensmodelle** beschreiben organisatorischen Rahmen für eine Entwicklung
 - Was ist wann zu tun?
- **Vorgehensmodelle** legen fest:
 - durchzuführende **Aktivitäten**
 - **Reihenfolge** des Arbeitsablaufs (Entwicklungsstufen, Phasen)
 - Definition der **Teilprodukte / Ergebnisse** (Inhalt, Layout)
 - **Fertigstellungskriterien**
 - **Verantwortlichkeiten und Kompetenzen**,
 - **Notwendige Mitarbeiterqualifikationen**,
 - Anzuwendende **Standards, Richtlinien, Methoden** und **Werkzeuge**
- **Vorgehensmodelle** sollen zu einer **disziplinierten, sichtbaren** und **kontrollierbaren** Entwicklung führen

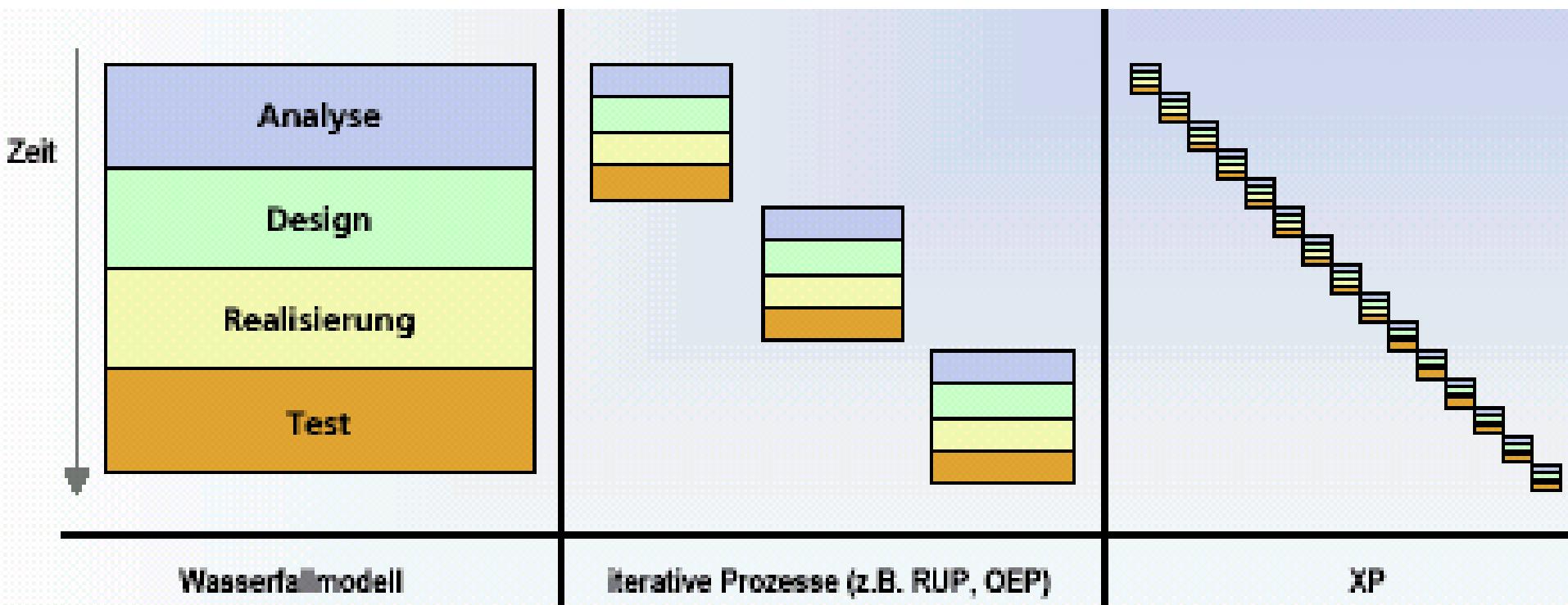
Vorgehensmodelle - Überblick

Prozeß-Modell	Primäres Ziel	Antreibendes Moment	Benutzerbeteiligung *)	Charakteristika
Wasserfall-Modell	minimaler Management-aufwand	Dokumente	gering	sequentiell, volle Breite
V-Modell	maximale Qualität (<i>safe-to-market</i>)	Dokumente	gering	sequentiell, volle Breite, Validation, Verifikation
Prototypen-Modell	Risiko-minimierung	Code	hoch	nur Teilsysteme (horizontal oder vertikal)
Evolutionäres Modell	minimale Entwicklungszeit (<i>fast-to-market</i>)	Code	mittel	sofort: nur Kernsystem
Inkrementelles Modell	minimale Entw.-zeit (<i>fast-to-market</i>), Risikominimierung	Code	mittel	volle Definition, dann zunächst nur Kernsystem
Objekt-orientiertes Modell	Zeit- und Kostenminimierung durch Wieder-verwendung	Wiederverwendbare Komponenten	?	volle Breite in Abhängigkeit von wiederverwendbaren Komponenten
Nebenläufiges Modell	minimale Entwicklungszeit (<i>fast-to-market</i>)	Zeit	hoch	volle Breite, nebenläufig
Spiralmodell	Risiko-minimierung	Risiko	mittel	Entscheidung pro Zyklus über weiteres Vorgehen

[Balz98]

*) Marketing, Vertrieb, Auftraggeber, Endbenutzer

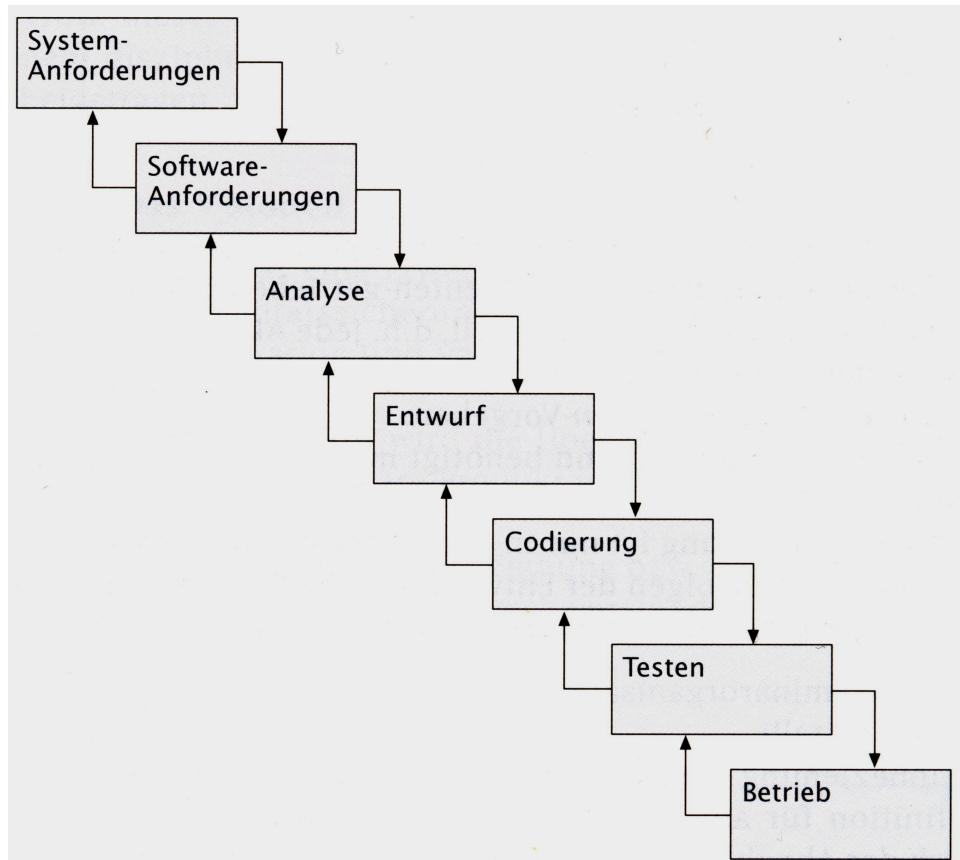
Vorgehensmodelle



- Grundlegende Prozessmodelle
- Erweiterte Prozessmodelle
- Aktuelle Prozessmodelle

Das Wasserfallmodell

- Ältester bekannter Ansatz
- Entwicklung in **sukzessiven Schritten** (Aktivitäten, Phasen)
- Teilweise ist **Rückkopplung** zwischen Phasen erlaubt
- Vielzahl von Variationen in der Anzahl und den Inhalten der Phasen
- Weit verbreiteter Ansatz, Grundlage anderer Ansätze



z.B. Royce 1970

Das Wasserfallmodell

Vorteile:

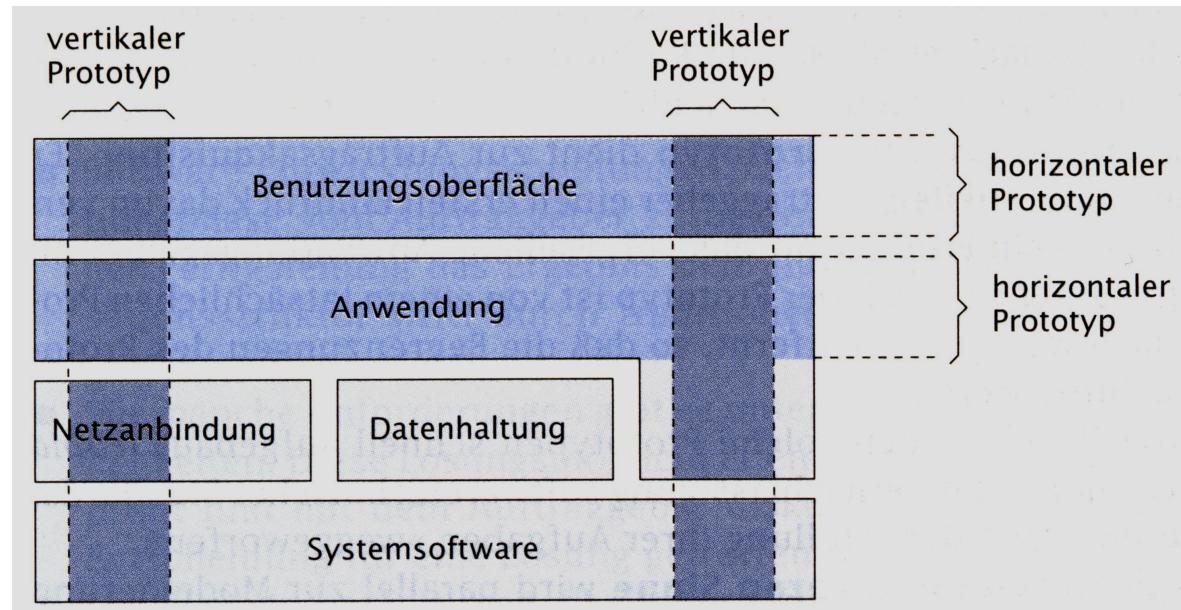
- Phasen werden in der richtigen Reihenfolge und vollständig durchlaufen
- Am Ende einer Phase steht ein Dokument
- Unterstützt ein Top-Down-Vorgehen
- Ist einfach, verständlich
- Benötigt wenig Managementaufwand

Nachteile:

- Feste, sequentielle Reihenfolge und vollständiges Durchlaufen einer Phase ist nicht immer sinnvoll
- Dokumentation ist gegebenenfalls wichtiger als das System selbst.
- Eventuell werden Risikofaktoren zu wenig berücksichtigt.

Das Prototypenmodell

- **Prototypen**
 - bei **unklaren Anforderungen** oder
 - bei **alternativen Lösungen**
- **Prototypen**
 - zur **Klärung** der Anforderungen
 - zum **Experimentieren**
 - um **praktische Erfahrungen** zu sammeln
 - als **Diskussionsbasis**



[Balz98]

- Prototypen lassen sich unterteilen in:
 - **Horizontale** Prototypen,
 - **Vertikale** Prototypen.

Das Prototypenmodell

- Prototyping unterstützt auf systematische Weise die frühzeitige Erstellung ablauffähiger Modelle (Prototypen) des zukünftigen Systems
- Arten von Prototypen
 - Demonstrationsprototyp (erster Eindruck),
 - Prototyp im engeren Sinne (erste Funktionalität),
 - Labormuster (technische Umsetzbarkeit),
 - Pilotsystem (Kern des Systems)
- Zusammenhang zwischen Prototyp und Produkt:
 - Prototyp zur Klärung von Problemen
 - Prototyp als Teil der Produktdefinition
 - Prototyp wird inkrementell weiterentwickelt

Das Prototypenmodell

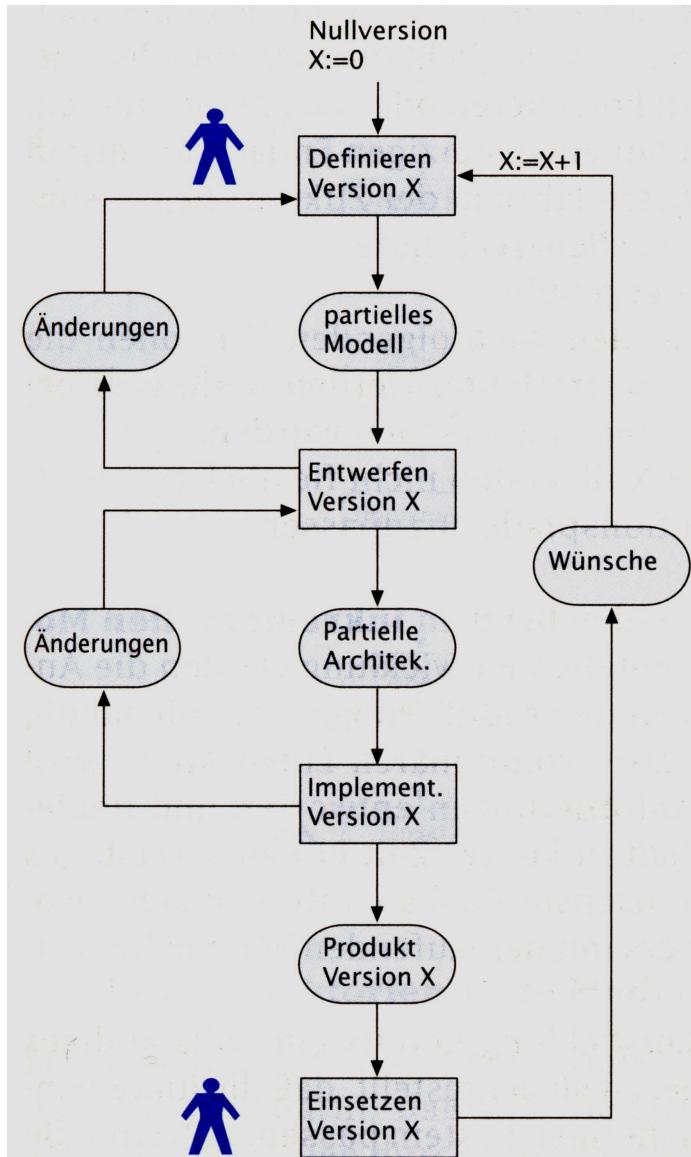
Vorteile:

- Prototypen reduzieren **Entwicklungsrisiko**
- Prototypen können in andere Prozessmodelle „**integriert**“ werden
- **Schnelle Prototypentwicklung** durch entsprechende Werkzeuge möglich
- Prototypen fördern die **Kreativität**

Nachteile:

- Höherer **Entwicklungsaufwand**, da Prototypen i.a. zusätzlich erstellt werden
- Gefahr der **Weiterverwendung** von (Wegwerf-)Prototypen
- **Grenzen** von Prototypen sind oft nicht erkennbar
- Bei Fremdentwicklung **vertraglich** schwer zu fassen

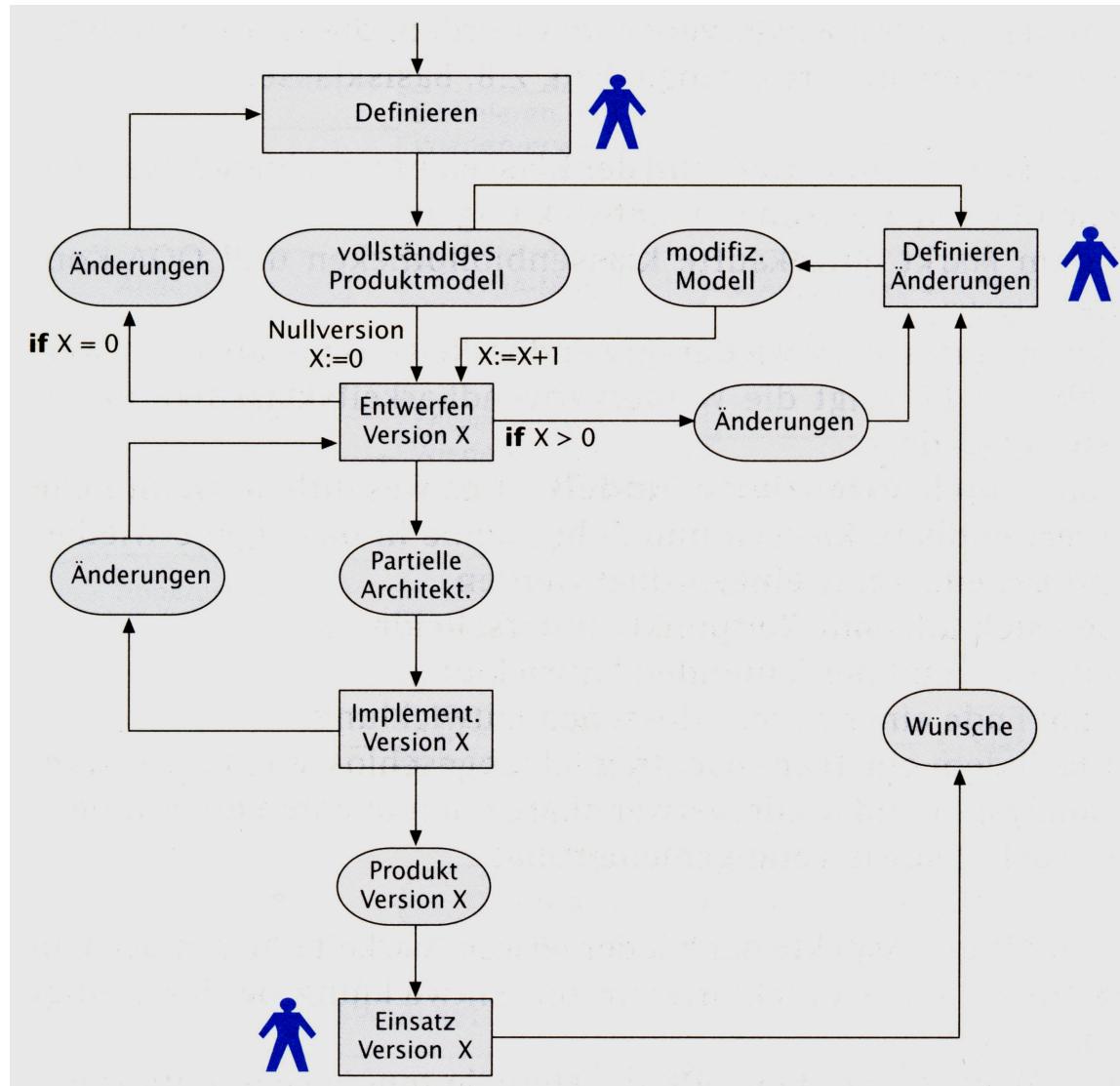
Das evolutionäre Modell



Evolutionäres Modell (Versionsentwicklung):

- Ausgangspunkt sind Kernanforderungen
- Diese führen zu einem Kernsystem (Nullversion)
- Erfahrungen mit der Version führen zu neuen, ergänzenden Anforderungen
- Entwicklung einer neuen Version
- Entwicklung ist daher Code-getrieben

Das inkrementelle Modell



[Balz98]

Variation des evolutionären Modells:

- Hier werden zunächst die **Anforderungen vollständig** erfasst
- Anschließend werden **wiederholt** Teile der Anforderungen umgesetzt

Das evolutionäre / inkrementelle Modell

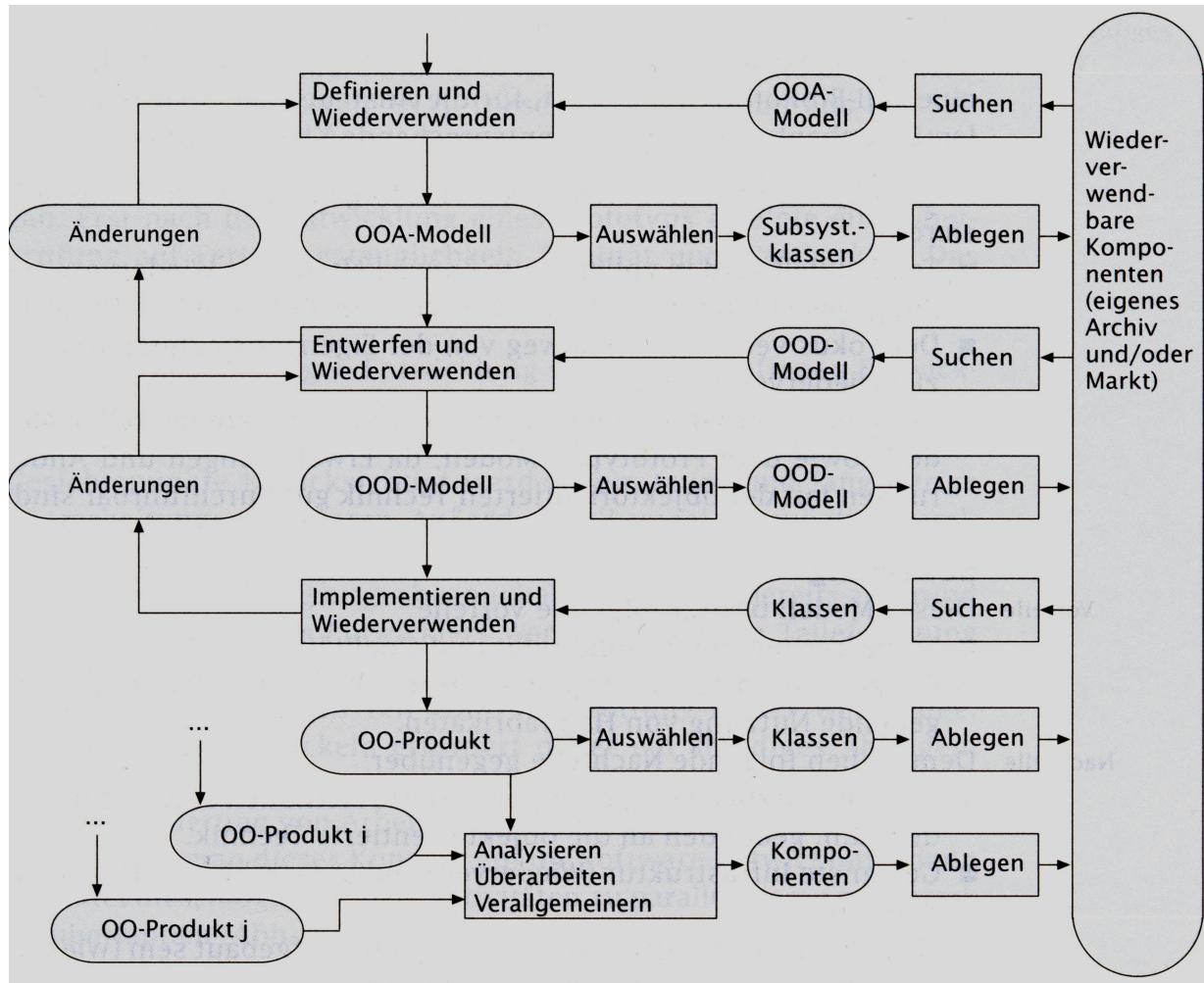
Vorteile:

- Einsatzfähige Produkte in kürzeren Zeitabständen
- Kombinierbar mit Prototypen-modell
- Auswirkungen des Produkt-einsatzes lassen sich untersuchen
- Die Richtung der Entwicklung ist leichter steuerbar
- Statt einem Ergebnis gibt es mehrere (Zwischen)-Ergebnisse

Nachteile:

- Falls Kernanforderungen übersehen werden, muß ggf. die Systemarchitektur überarbeitet werden
- Sind die Zwischenergebnisse an die Anforderungen ungeplanter Evolutionspfade leicht anpassbar?

Das objektorientierte Modell



- Charakteristisches Merkmal der Objektorientierung ist die **Wiederverwendung**
- Wiederverwendung, bezogen auf:
 - Ebenen (Modelle, Entwürfe, Klassen)
 - Gebiete (Branchen, individuell)
 - Herkunft (eigen, fremd)
- **Bottom-Up-Aspekt**

Das objektorientierte Modell

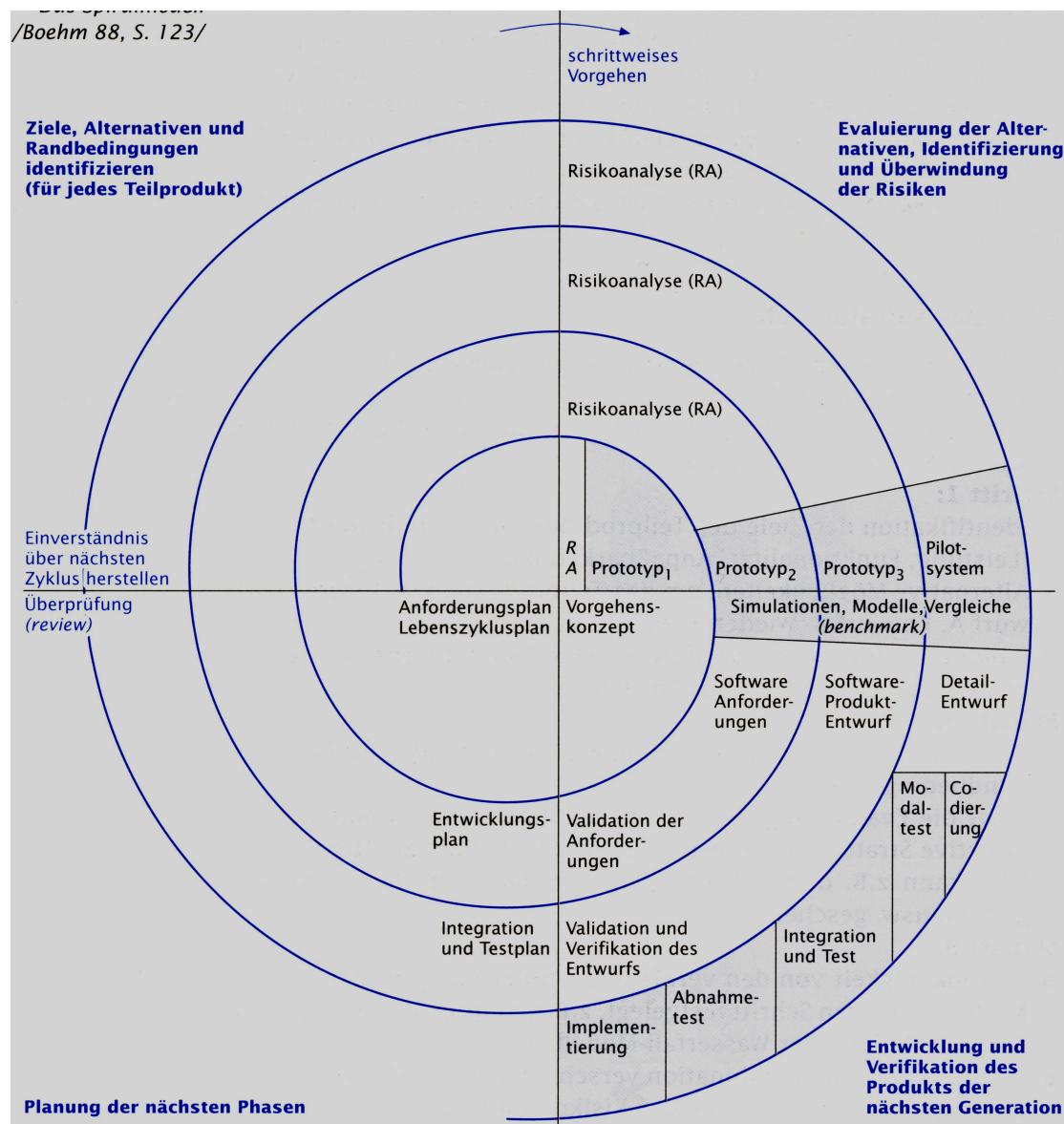
Vorteile:

- Verbesserung von Produktivität und Qualität
- Konzentration auf die eigenen Stärken
- Nutzung von Halbfabrikaten

Nachteile:

- Gebunden an objektorientierte Technik
- Geeignete Infrastruktur muß vorhanden sein
- Firmenkultur der Wiederverwendung muß aufgebaut werden
- Inkompatibilitätsprobleme müssen gelöst werden

Das Spiralmodell



- **Risikogetriebenes Prozessmodell**
- Für jedes Teilprodukt und für jede Verfeinerungsebene sind vier **zyklische Schritte** zu durchlaufen
- **Ziele** eines Zyklus ergeben sich aus den Ergebnissen des letzten Zyklus

Das Spiralmodell

Schritt 1

- Identifikation der **Ziele** des Teilprodukts (Leistung, Funktionalität, ...)
- Identifikation alternativer **Realisierungsmöglichkeiten** (Entwurf A, Entwurf B, Wiederverwendung, Kauf)
- Identifikation der **Randbedingungen**, die bei den verschiedenen Alternativen zu beachten sind (Kosten, Zeit, Schnittstellen usw.)

Schritt 2

- **Evaluierung** der Alternativen unter Berücksichtigung der Ziele und Randbedingungen
- Zeigt die Evaluierung, daß es **Risiken** gibt, dann ist eine kosteneffektive **Strategie** zu entwickeln, um die Risiken zu überwinden
- Dies kann z.B. durch **Prototypen**, Simulationen oder Benutzerbefragungen geschehen

Das Spiralmodell

Schritt 3

- In Abhängigkeit von den verbleibenden Risiken wird das **Prozessmodell** für diesen Schritt festgelegt, z.B. evolutionäres Modell, Prototypenmodell oder Wasserfallmodell
- Es kann auch eine **Kombination** verschiedener Modelle vorgenommen werden, wenn dadurch das Risiko minimiert wird

Schritt 4:

- **Planung** des nächsten Zyklus einschließlich der benötigten **Ressourcen**. Dies beinhaltet auch eine mögliche Aufteilung eines Produktes in Komponenten, die dann unabhängig weiterentwickelt werden
- **Überprüfung** (Review) der Schritte I bis 3 einschließlich der Planung für den nächsten Zyklus durch die betroffenen Personengruppen oder Organisationen
- **Einverständnis** (Commitment) über den nächsten Zyklus herstellen

Das Spiralmodell

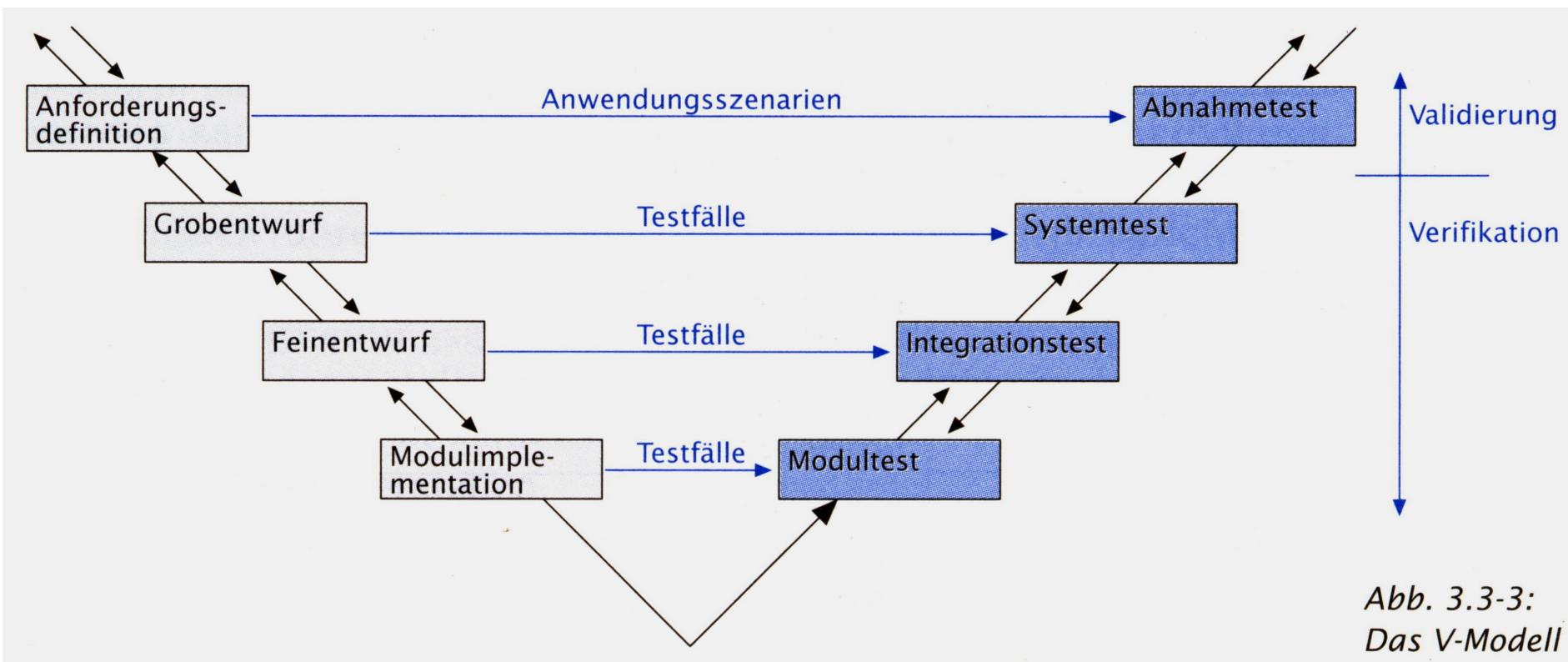
Vorteile:

- Flexibles Modell
- In periodischen Intervallen Überprüfung und u.U. erneute Festlegung des Ablaufs in Abhängigkeit von den Risiken
- Integration anderer Prozessmodelle
- Ein Prozessmodell wird nicht für die gesamte Entwicklung festgelegt
- Wiederverwendung wird unterstützt (durch Alternativen-betrachtung)

Nachteile:

- Hoher Management-aufwand, da oft neue Entscheidungen
- Wissen über das Identifizieren und Managen von Risiken noch nicht weit verbreitet
- Eignung für kleinere und mittlere Projekte?

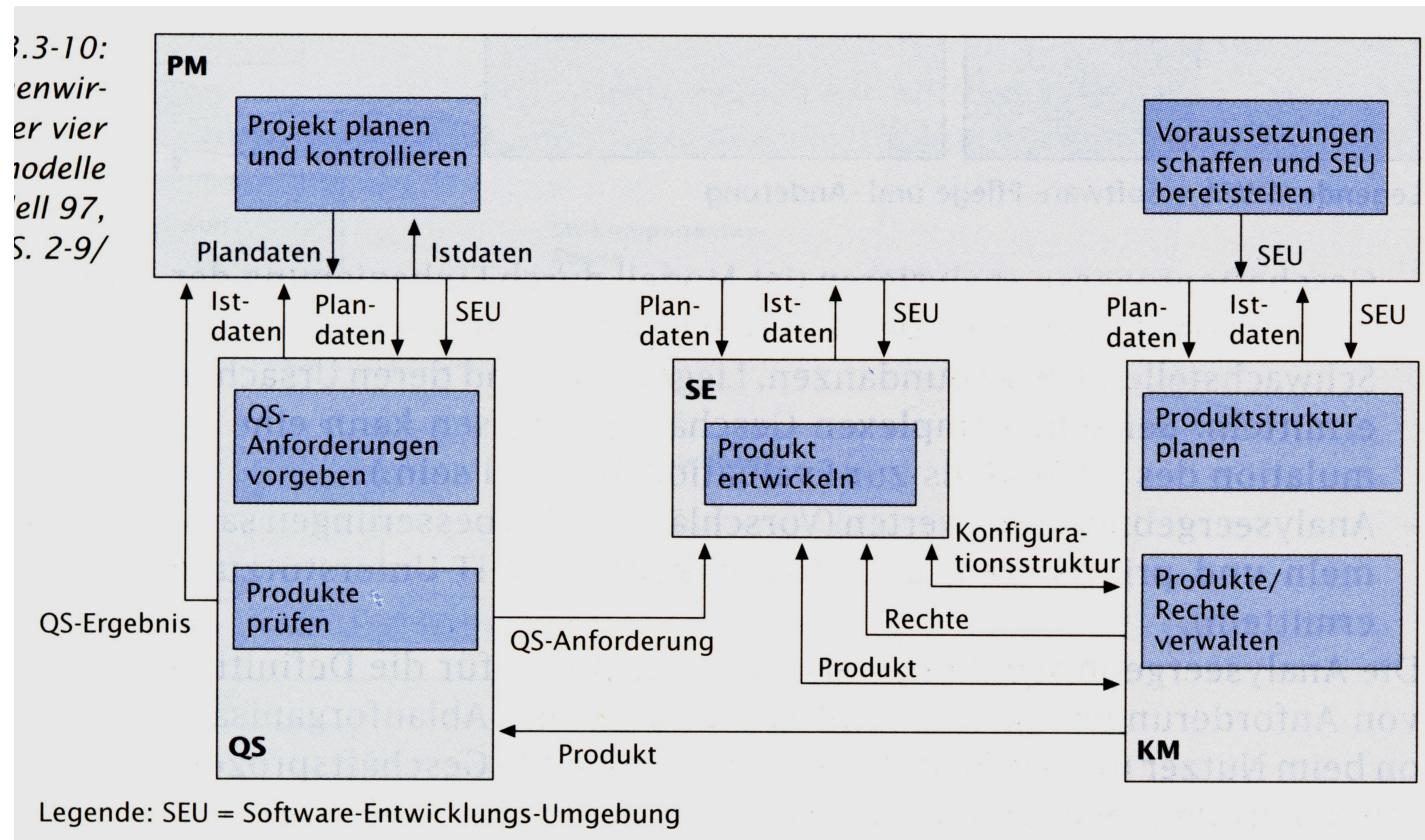
Das V-Modell



[Balz98]

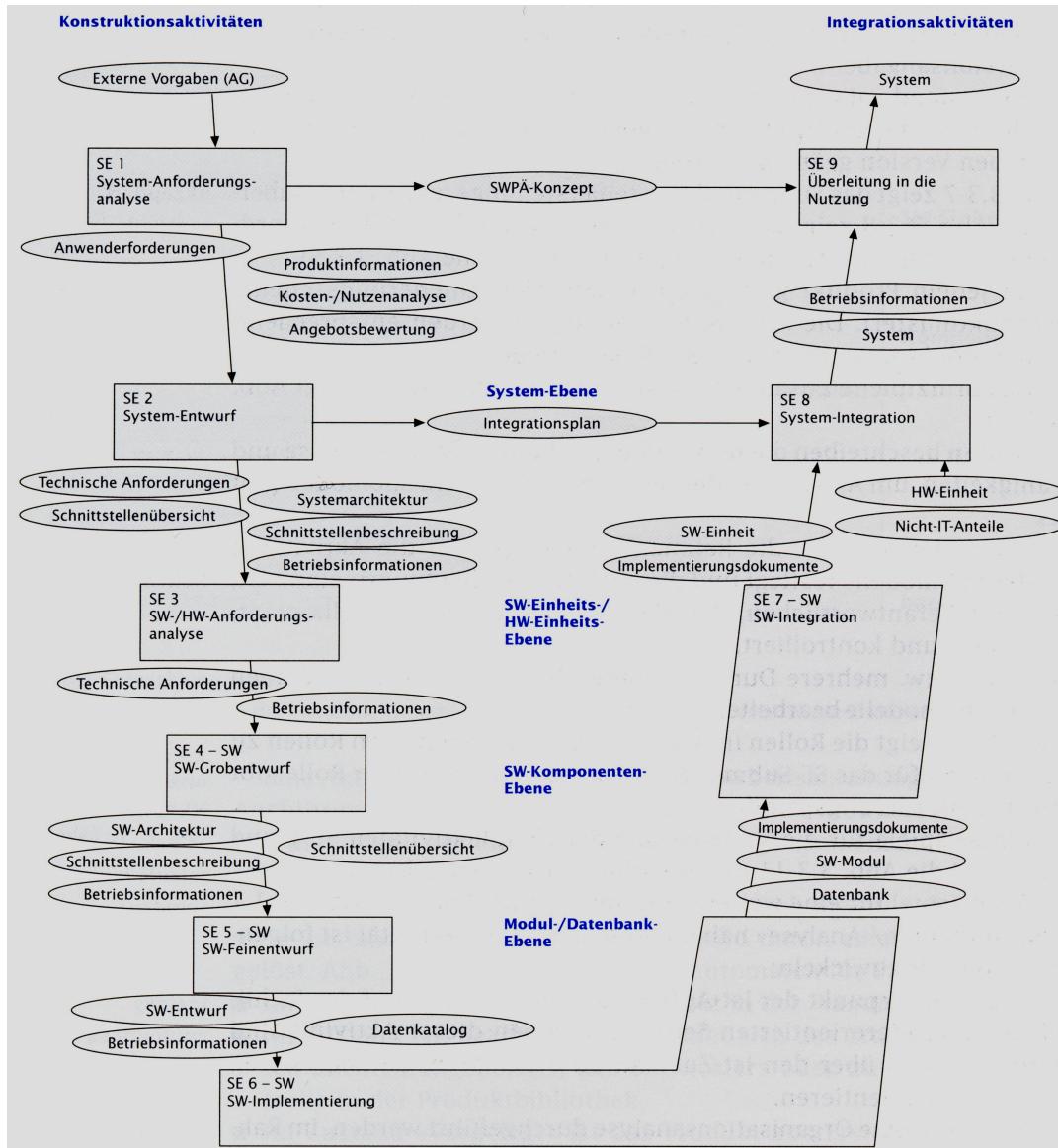
- Erweiterung des Wasserfallmodells
 - **Verifikation:** Übereinstimmung Produkt mit Spezifikation.
 - **Validierung:** Eianuna Produkt auf seinen Einsatzzweck.
- HTBLA Leonding
your IT-School*

Das V-Modell - Submodelle



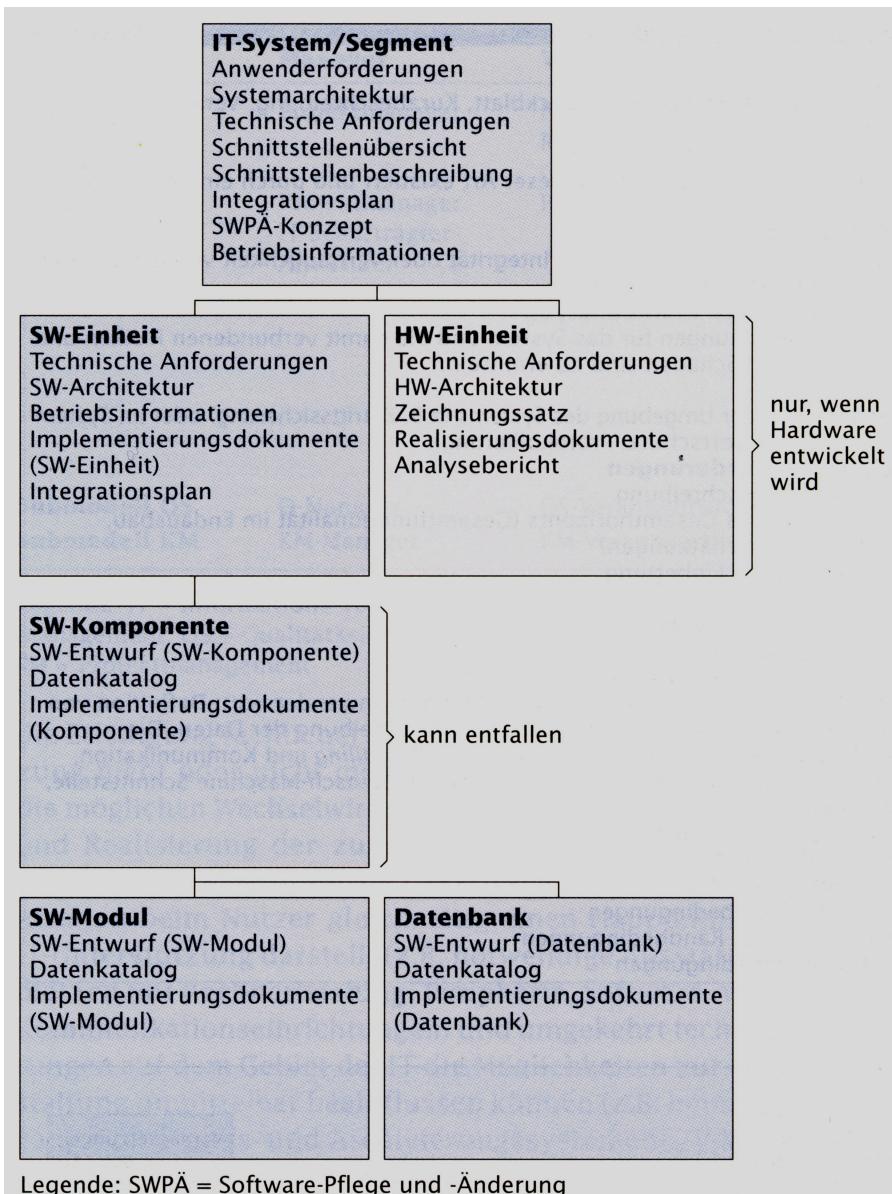
- SE - Systemerstellung
- QS - Qualitätssicherung
- PM - Projektmanagement
- KM - Konfigurationsmanagement

Das V-Modell



- Funktions-überblick Submodell Software-erstellung (SE)
- dokumenten-getrieben

Das V-Modell - Produktstruktur



Produktstruktur

- Produkte des Submodells Softwareerstellung (SE)

[Balz98]

Das V-Modell - Rollen

	Manager	Verantwortliche	Durchführende
Submodell PM	Projektmanager	Projektleiter Rechtsverantwortlicher Controller	Projektadministrator
Submodell SE	Projektmanager IT-Beauftragter Anwender	Projektleiter	Systemanalytiker Systemdesigner SW-Entwickler HW-Entwickler Technischer Autor SEU-Betreuer Datenadministrator IT-Sicherheitsbeauftragter Datenschutzbeauftragter Systembetreuer
Submodell QS	Q-Manager	QS-Verantwortlicher	Prüfer
Submodell KM	KM-Manager	KM-Verantwortlicher	KM-Administrator

Legende: IT = Informations-Technik, Q = Qualität, KM = Konfigurationsmanagement, QS = Qualitätssicherung, SEU = Software-Entwicklungs-Umgebung, PM = Projektmanagement

Tab. 3.3-1:
Rollen im V-Modell
/V-Modell 97,

Das V-Modell – Zuordnung Rolle zu Aktivität

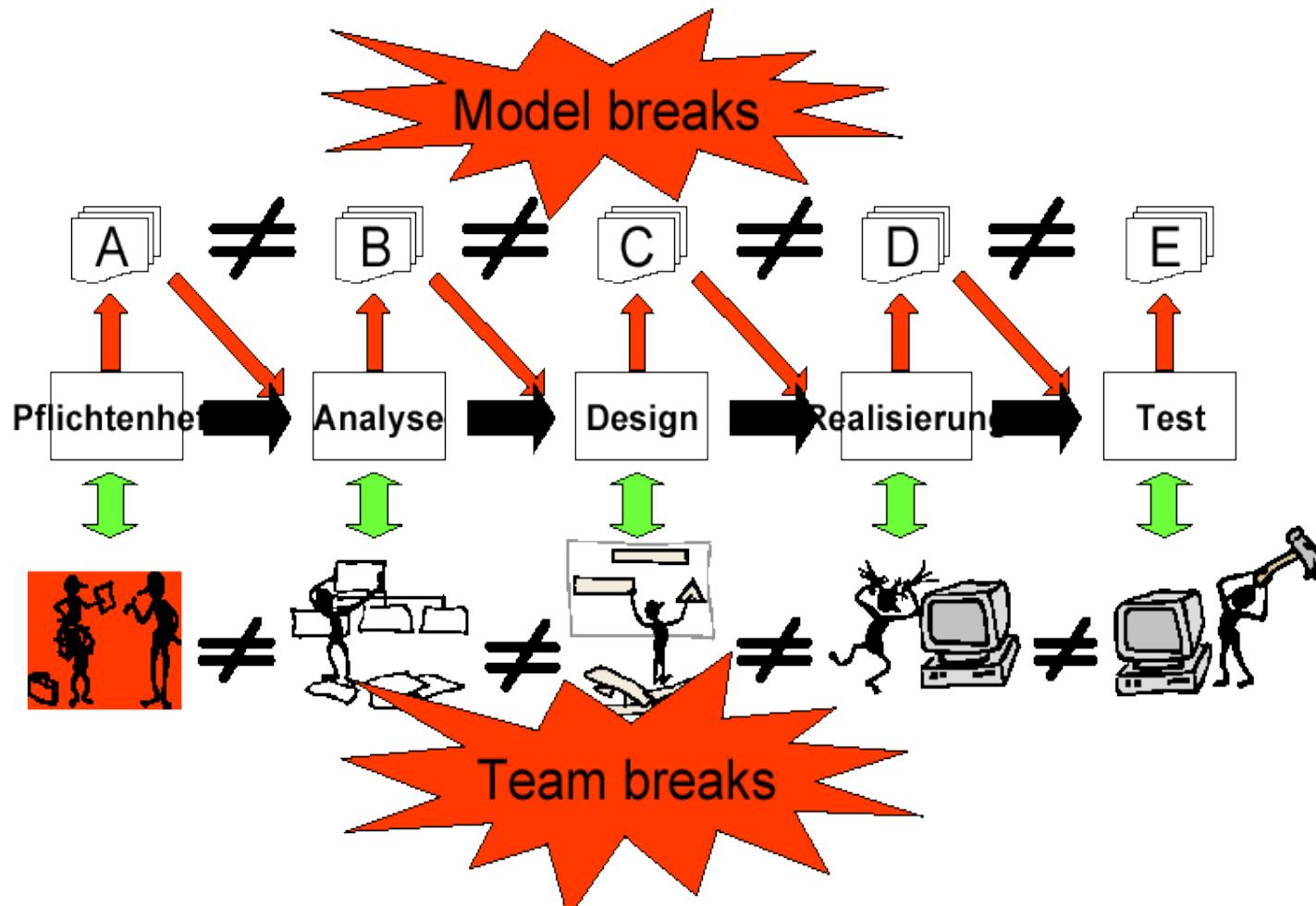
Aktivität	Rolle	Systemanalytiker	Systemdesigner	SW-Entwickler	HW-Entwickler	Technischer Autor	Systembetreuer	SEU-Betreuer	Datenadministrator	Datenschutzbeauftragter	IT-Sicherheitsbeauftragter	IT-Beauftragter	Anwender	Projektleiter
SE 1.1 Ist-Aufnahme/-Analyse durchführen	v	b												
SE 1.2 Anwendungssystem beschreiben	v													
SE 1.3 Kritikalität und Anforderungen an die Qualität definieren	v												m	
SE 1.4 Randbedingungen definieren	v												m	
SE 1.5 System fachlich strukturieren	v			m	m			m	b	m	m	m		
SE 1.6 Bedrohung und Risiko analysieren										v				
SE 1.7 Forderungscontrolling durchführen	m	m										v	m	
SE 1.8 Software-Pflege und Änderungs-Konzept erstellen		v										m	m	
SE 2.1 System technisch entwerfen	v		m								m		m	
SE 2.2 Wirksamkeitsanalyse durchführen										v				
SE 2.3 Realisierbarkeit untersuchen	m	v								m	m	m		
SE 2.4 Anwenderaforderungen zuordnen	v													
SE 2.5 Schnittstelle beschreiben	v													
SE 2.6 System-Integration spezifizieren	v		m											
SE 3.1 Allgemeine Anforderungen an die SW-/HW-Einheit definieren	v													
SE 3.2 Anforderungen an die externen Schnittstellen der SW-/HW-Einheit präzisieren	v													
SE 3.3 Anforderungen an die Funktionalität definieren	v													
SE 3.4 Anforderungen an die Qualität der SW-/HW-Einheit definieren	v													
SE 3.5 Anforderungen an Entwicklungs- und SWPA-Umgebung definieren	v		m		m									
SE 4.1-SW SW-Architektur entwerfen		v	m											
SE 4.2-SW SW-interne und externe Schnittstellen entwerfen		v												
SE 4.3-SW SW-Integration spezifizieren		v	m											
SE 5.1-SW SW-Komponente-/Modul/Datenbank beschreiben		v	m											
SE 5.2-SW Betriebsmittel- und Zeitbedarf analysieren		v												
SE 6.1-SW SW-Module codieren		v												
SE 6.2-SW Datenbank realisieren		v												
SE 6.3-SW Selbstprüfung des SW-Moduls/ der Datenbank durchführen		v												
SE 7.1-SW Zur SW-Komponente integrieren		v												
SE 7.2-SW Selbstprüfung der SW-Komponente durchführen		v												
SE 7.3-SW Zur SW-Einheit integrieren		v												
SE 7.4-SW Selbstprüfung der SW-Einheit durchführen		v												

Legende: v = verantwortlich m = mitwirkend b = beratend

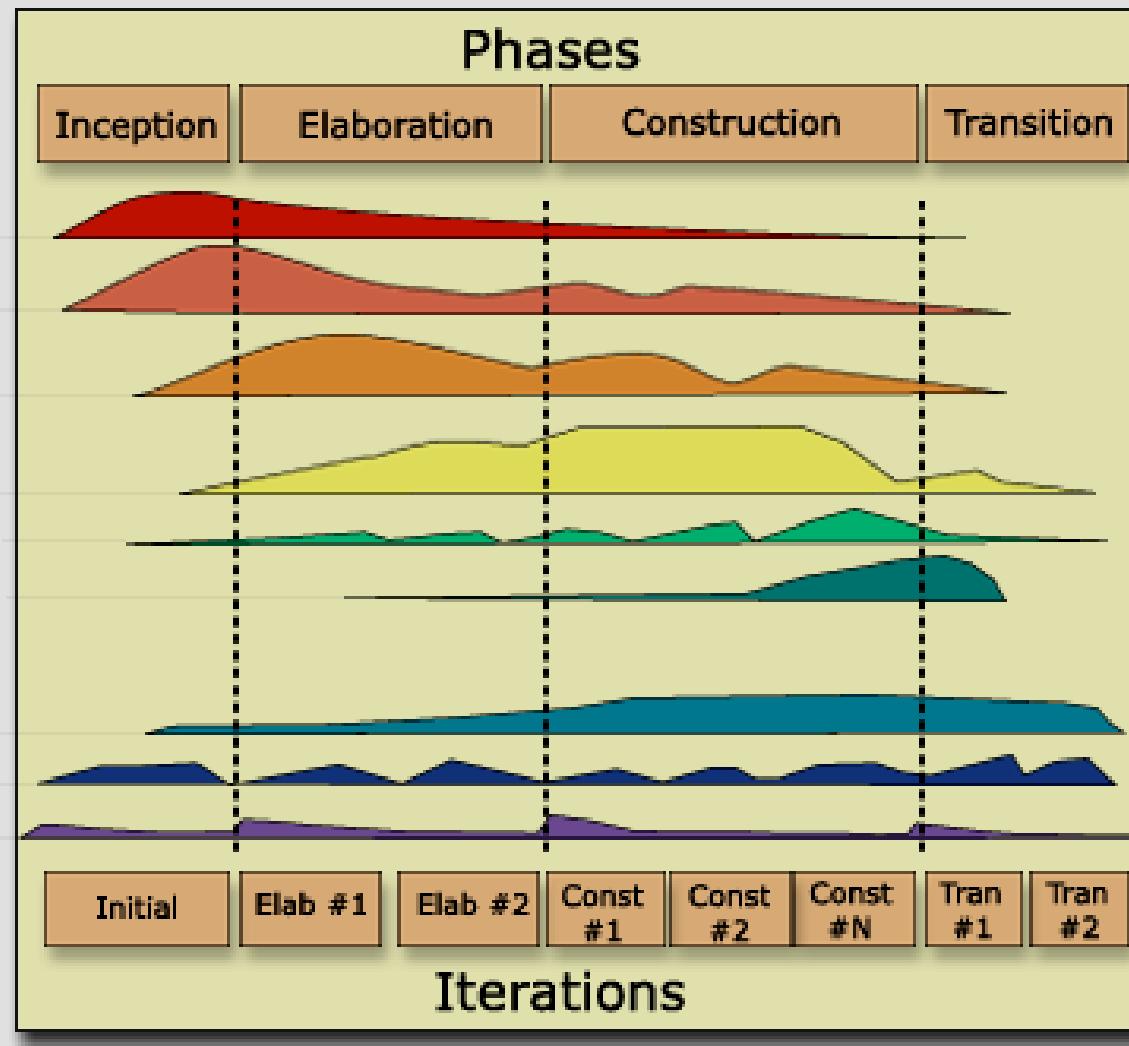
- Aktivitäten / Rollen-Matrix für das Submodell Softwareerstellung (SE)

[Balz98]

RUP (Rational Unified Process)

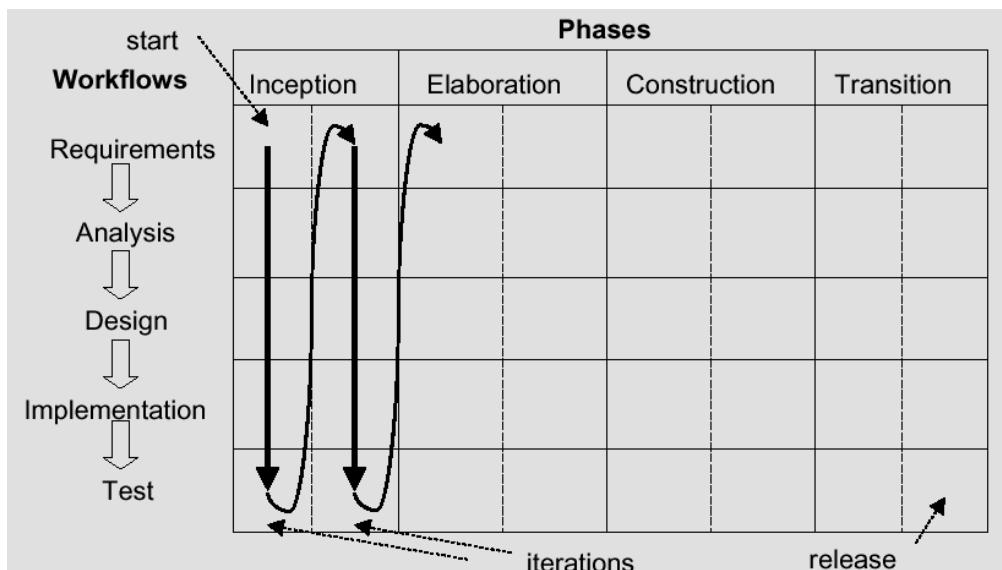
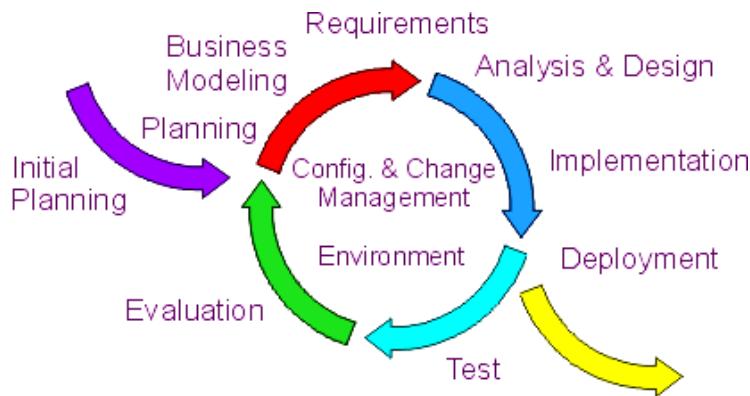


RUP - Aufbau



Phasen dienen dem Erreichen wohldefinierter Ziele.
Ein Entwurf entsprechend RUP durchläuft 4 Phasen.

RUP Rational Unified Process- Iterationen



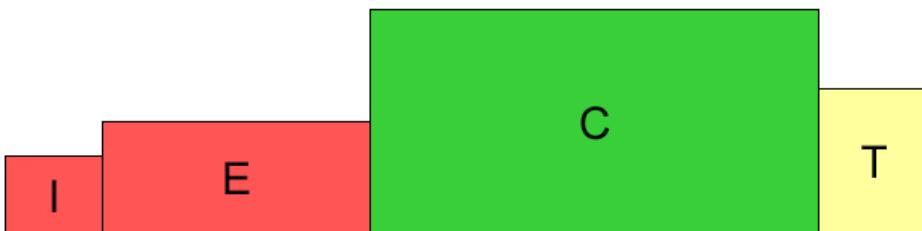
Iteration:

- Teilschritt in einer Phase
- Resultiert in einem Release (intern oder extern) einer Teilmenge des Gesamtproduktes
- Durchläuft alle Arbeitsabläufe (aber mit unterschiedlicher Gewichtung)

Entwicklungszyklus:

- Kompletter Durchlauf durch alle Phasen

RUP - Projekterfahrungen



Phase	Zeitbedarf	Budget
Inception	10%	5%
Elaboration	30%	20%
Construction	50%	65%
Transition	10%	10%

Anzahl Personen	Dauer einer Iteration
5	1-2 Wochen
15	1 Monat
45	6 Monate
100	12 Monate

Achtung: Dies sind Richtwerte, die vor dem Einsatz für Projekte einer Verifikation bedürfen

Extreme Programming (XP)

Common Principles and Practices

- "Code reviews" sind gut
- Testen ist gut
- Integrationstests sind wichtig
- Design ist wichtig
- Einfachheit ist gut
- Architektur ist wichtig
- Kurze Iterationen sind gut

... taken to extreme levels

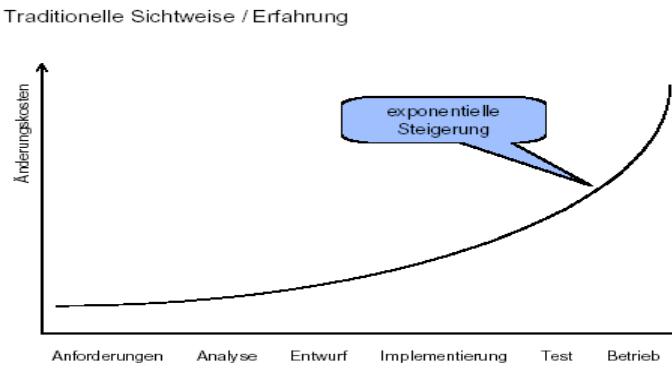
- ➔ ständige "Code reviews"!
 - ◆ "pair programming"
- ➔ ständiges testen!
 - ◆ Programmierer: Modultests
 - ◆ Kunden: Funktionstests
- ➔ mehrmals am Tag!
 - ◆ "continuous integration"
- ➔ ständiges (re)design!
 - ◆ "refactoring"
- ➔ alles so einfach wie möglich
 - ◆ bzw. so komplex wie unbedingt nötig
- ➔ ständige Architektur-Weiterentwicklung
 - ◆ "metaphor"
- ➔ extrem kurz: Minuten bis Stunden!
 - ◆ "planning game"

Extreme Programming (XP)

- Gegenüberstellung

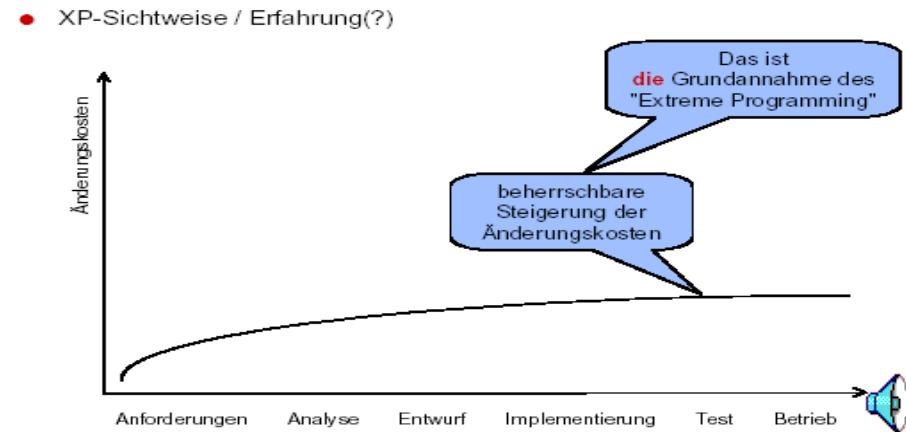
Traditionelle Methodiken

- Annahme
 - ◆ exponentielle Kostenexplosion
- Konsequenzen
 - ◆ mögliche Änderungen antizipieren
 - ◆ entsprechende Möglichkeiten einbauen
 - ◆ komplexere Software
 - ◆ höhere Anfangskosten
 - ◆ langsamerer Anfangsfortschritt
 - ◆ geringere Gesamtkosten!!!
 - ◆ geringere Gesamtzeit!!!



Extreme Programming

- asympotische Kostensteigerung
- Änderungen erst bedenken wenn gefordert
- nur das notwendigste implementieren
- ◆ einfache Software
- ◆ geringere Anfangskosten
- ◆ schnellerer Projektfortschritt
- ◆ geringere Gesamtkosten!!!
- ◆ geringere Gesamtzeit!!!



- Beurteilung

Randbedingungen und Grenzen

- XP skaliert nicht: Maximale Teamgröße ca. 10 Entwickler aufgrund der hohen Kommunikationsanforderungen
- Entscheidungsbefugter Kunde muss vor Ort verfügbar sein
- Bestehende Unternehmensstandards (CMM, ISO, HBSG, ...)
- Politik / Vertragsgestaltung (Festpreise, ...)
- Teamfähigkeit und Disziplin notwendig:
- Nichts für „Lonely Coders“ und Primadonnen

XP ist sinnvoll in Projekten

- ... wo der Kunde selbst nur vage Vorstellungen vom Endergebnis hat
- ... in sehr variablem Umfeld (→ neue Technologien)
- ... in denen eine hohe Update rate erwartet bzw. akzeptiert wird (z.B. bei eCommerce-Sites)
- ... bei denen schnell erste Teilergebnisse benötigt werden