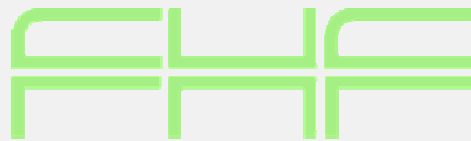


# 1.2 System- und Softwareentwicklungsprozesse



Prof. Mario Jeckle

Fachhochschule Furtwangen

[mario@jeckle.de](mailto:mario@jeckle.de)

<http://www.jeckle.de>

# Vorgehensmodelle und UML

Um Systeme und Software sinnvoll entwickeln zu können wurden Vorgehensmodelle geschaffen, um Arbeit und Ergebnisse ...

- effizienter
  - effektiver
- ...zu gestalten

Vorgehensmodelle unterteilen sich in

- klassisch
- agil

UML dient als standardisierte Modellierungssprache.

# Vorgehensmodelle

Klassische Vorgehensmodelle = „schwergewichtige“ Prozessmodelle

- statisch
- dokumentenorientiert
- teuer
- gut geeignet für große Projekte

Agile Vorgehensmodelle = „leichtgewichtige“ Methoden

- flexibel
- kundenorientiert
- chaotisch
- gut geeignet für kleine Projekte und Teams

# Klassische Vorgehensmodelle

Entstehung seit 60er Jahren

- Rational Unified Process (RUP)
- Spiralmodell
- Springbrunnenmodell
- V-Modell
- Wasserfallmodell
- ...

# Das Wasserfallmodell

- Entwickelt aus dem *stage wise model* von Benington
- Weiterentwickelt mit Rückkopplungsschleifen durch Winston Royce
- Benannt und in heutige Form gebracht von Barry Boehm

Das Wasserfallmodell ist

- sequentiell
- stufenweise
- dokumentenorientiert

# Das Wasserfallmodell

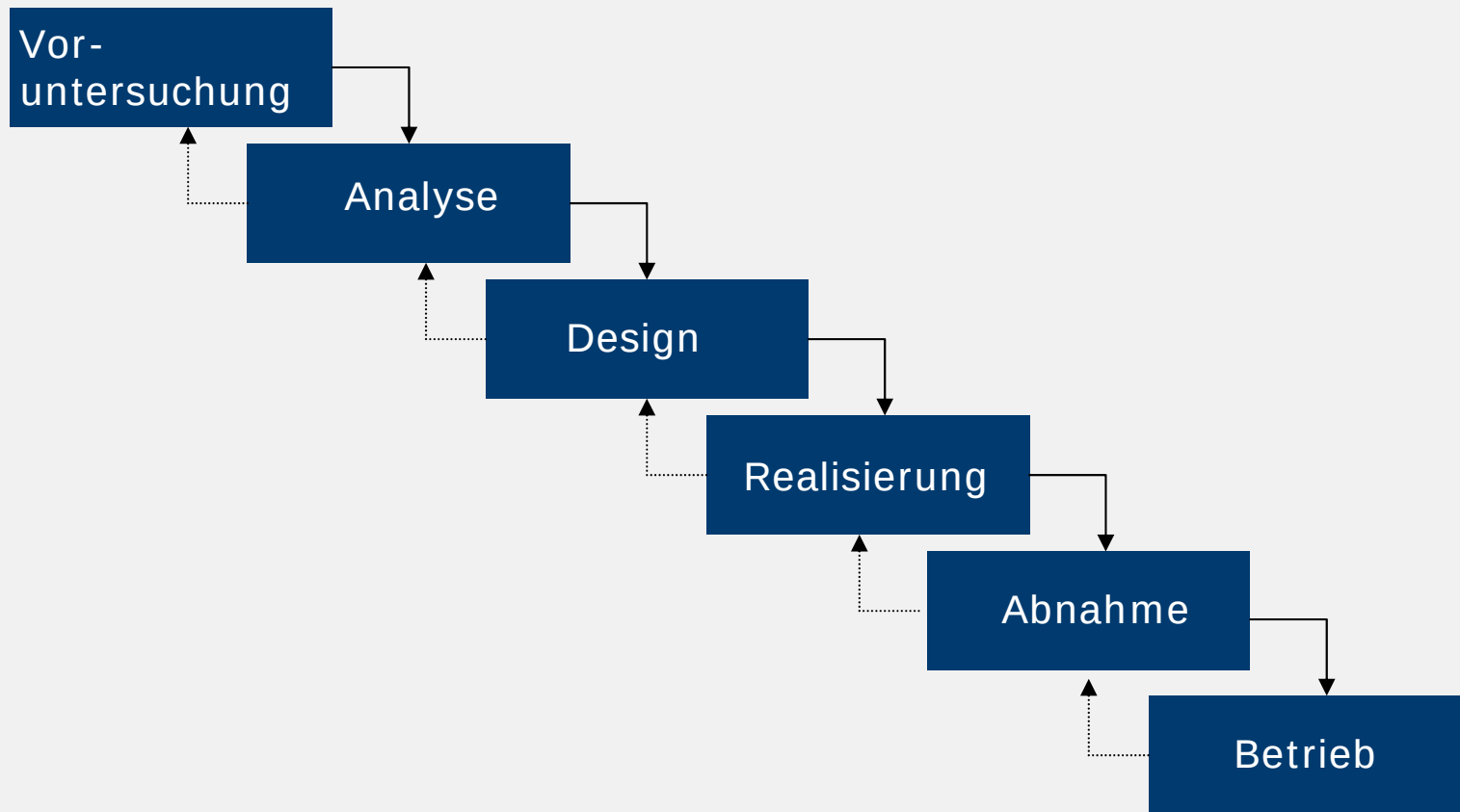
Die Einteilung des Modells erfolgt in Phasen, welche

- In richtiger Reihenfolge
- In vollständiger Breite
- Nur strikt nacheinander sequentiell bearbeitet werden können
- Mit Meilensteinen abgeschlossen werden

Eine neue Phase kann nur nach vollständigem Abschluss der vorherigen begonnen werden!

# Das Wasserfallmodell

Ein Überblick:



# Das Springbrunnenmodell

## Allgemeines:

- Entwickelt von Brian Henderson-Sellers
- Auch als *fountain model* bekannt
- Besteht aus Aktivitäten, die einer bestimmten, nicht vorgegebenen Reihenfolge unterliegen

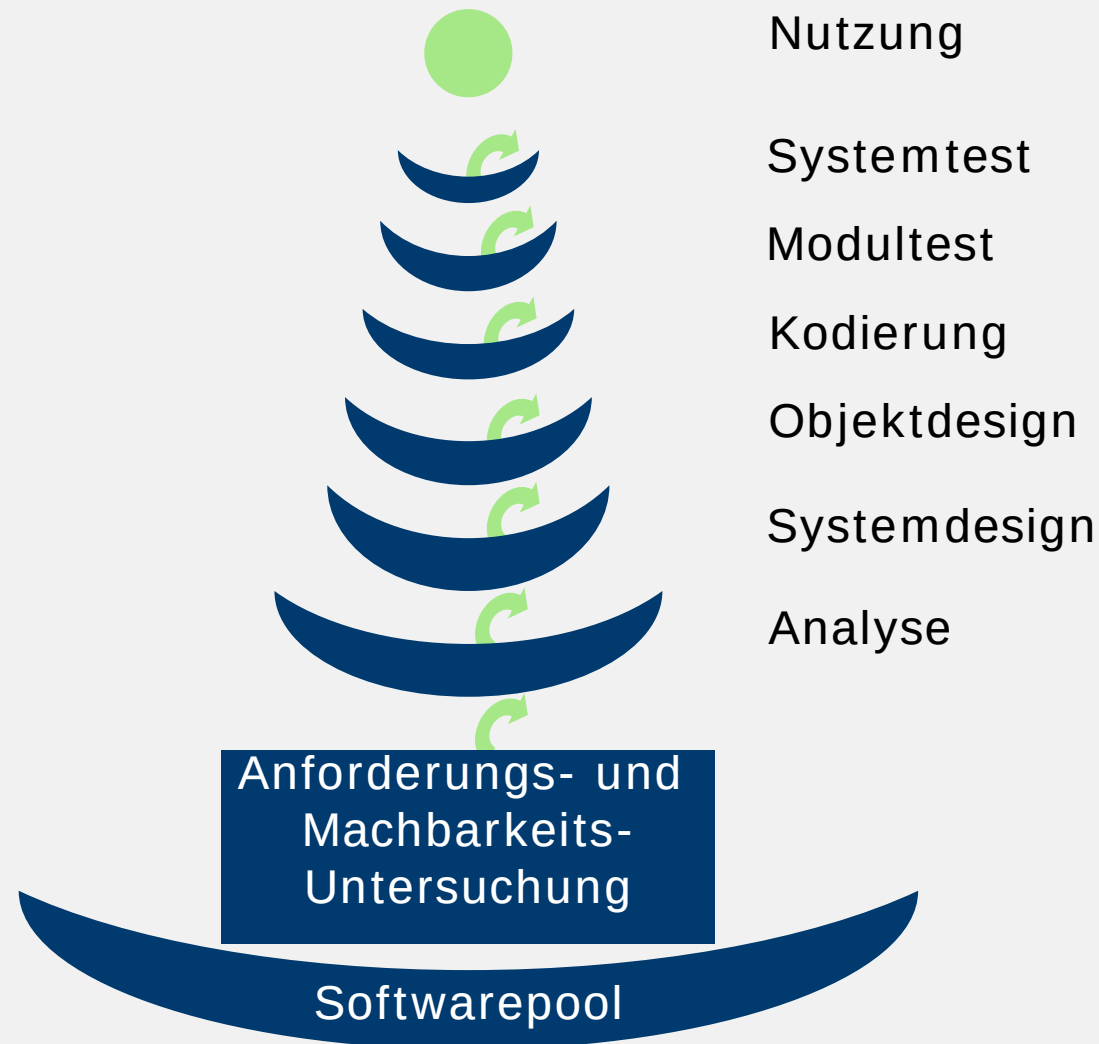
Ergebnisse der Entwicklung mit diesem Modell sind ähnlich des Wasserflusses in einem Springbrunnen:

- Das nach oben gepumpte Wasser symbolisiert Reihenfolge und Verlauf von Aktivitäten
- Die verschiedenen Ebenen repräsentieren Ergebnisebenen bzw. das Vollenden einer Teilaufgabe
- Zielerreichung wird von der Spitze der Fontäne dargestellt



# Das Springbrunnenmodell

Ein Überblick:



# Das Spiralmodell

## Allgemeines:

- Entwickelt von Barry Boehm
- Aufbauend auf dem Wasserfallmodell (WM)
- Eingeteilt in Zyklen, die wiederholt zu durchlaufen sind
- Zyklen entsprechen den Phasen des WM
- Dynamischere Anpassungsmöglichkeiten als bei WM

## Aufgaben der Zyklen:

- Klärung der Ziele, Alternativen und Randbedingungen.
- Gegebenfalls Einsatz von Prototyping zur Klärung der Ziele, Vorstellungen, Machbarkeit.
- Ablauf geeigneter Schritte, wie im Wasserfall-Modell.
- Review der abgelaufenen Phase und Planung der folgenden.

# Das Spiralmodell

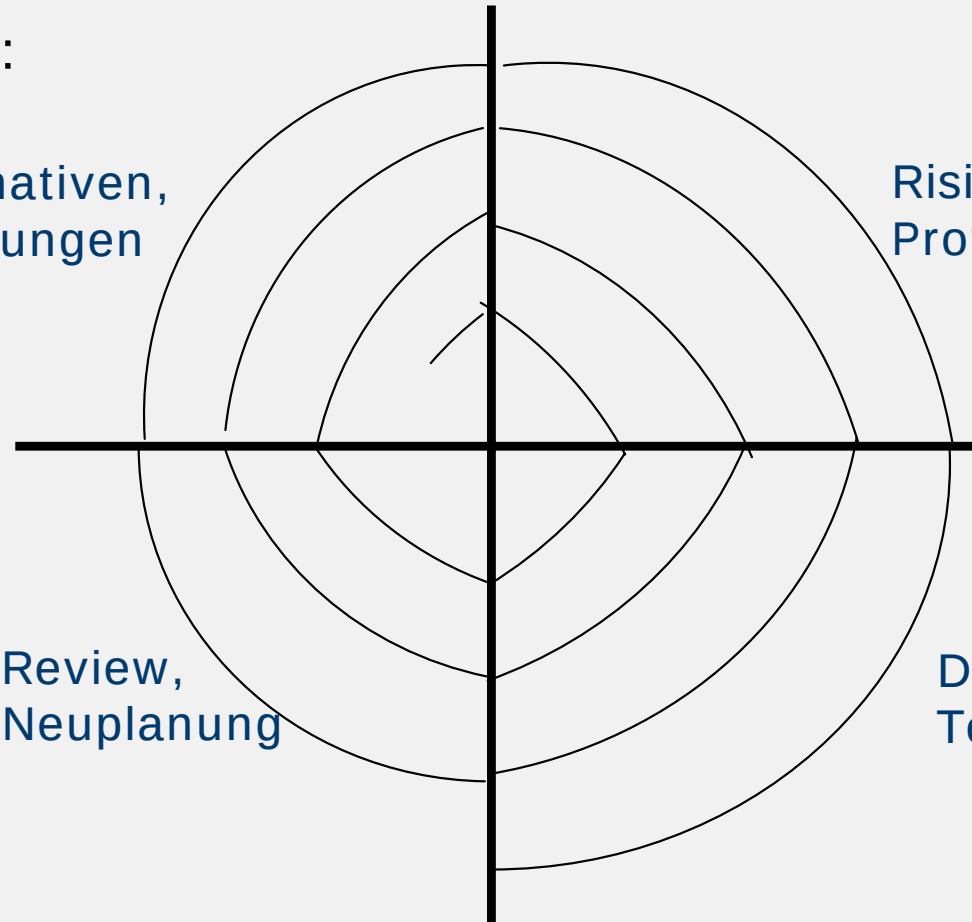
Ein Überblick:

Ziele, Alternativen,  
Randbedingungen

Risikoanalyse,  
Prototyping

Review,  
Neuplanung

Design,  
Test



# Das V-Modell

- Entwickelt im Auftrag des Bundesministeriums für Verteidigung (BMVg)
- Entwickelt in Zusammenarbeit von Bundesamt für Wehrtechnik und Beschaffung (BWB) und der Industrieanlagen- Betriebsgesellschaft mbH (IABG) Ottobrunn
- 1992 erstmaliger Einsatz im Bundesministerium des Inneren (BMI) für die Bundesverwaltung
- Seit 1996 verbindlicher Standard für alle IT-Systeme des Bundes
- Heute auch in vielen Wirtschaftsunternehmen Standard

# Das V-Modell

Das V-Modell wird in drei Teile aufgegliedert:

*Regelungsteil:*

verbindliche Regelungen für durchzuführende Arbeitsschritte

*Behördenspezifische Ergänzung:*

Angaben zur Anwendung des V-Modells; diesen Teil gibt es für den Bereich der Bundeswehr und für die zivile Bundesverwaltung

*Handbuchsammlung:*

enthält Handbücher zu speziellen IT-Themen (z.B. Sicherheit)

# Das V-Modell

Durch ständige Weiterentwicklung und Finanzierung durch BMVg und BMI ist das V-Modell ein *lebendiger Standard*

Dieser Standard umfasst drei Ebenen:

- *Vorgehensmodell*:  
legt Tätigkeiten, Ergebnisse und Inhalte der Ergebnisse fest
- *Methodenzuordnung*:  
legt die Methoden und Darstellungsmittel für die oben genannten Tätigkeiten und Ergebnisse fest
- *Funktionalen Werkzeuganforderungen*:  
legt die funktionalen Eigenschaften der benötigten Werkzeuge fest

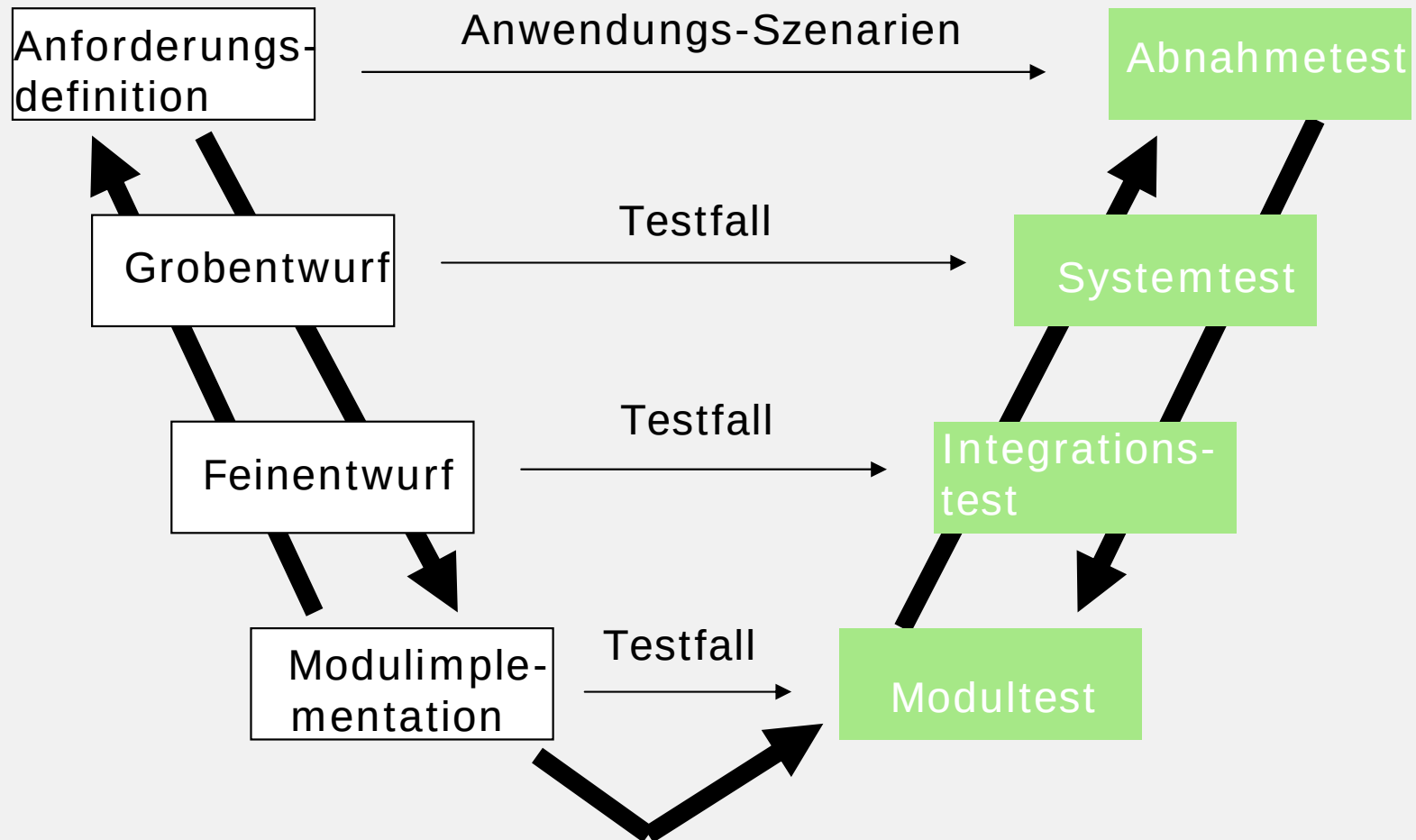
# Das V-Modell

Auf allen Ebenen werden Regelungen in Untermodelle eingeteilt:

- Projektmanagement (PM):  
plant, kontrolliert und informiert SE, QS und KM
- Systemerstellung (SE):  
erstellt System bzw. Software
- Qualitätssicherung (QS):  
gibt Qualitätsanforderungen vor und überprüft Ergebnisse auf Einhaltung
- Konfigurationsmanagement (KM):  
verwaltet erzeugte Produkte

# Das V-Modell

Ein Überblick:





# Rational Unified Process

## Übersicht:

- Entwickelt 1998 durch die Firma *Rational*
- Anwendungsfall-zentrierter Zugang zur Softwareentwicklung
- Rahmenwerk für den gesamten Entwicklungsprozess
- Definiert und steuert Aktivitäten, Messkriterien, Ergebnisse
- Sprache des RUP ist die UML
  - Erstellt Artefakte als Grundlage des Systems
  - Dokumentiert das Projekt

# Rational Unified Process

Zusammensetzung aus „Best Practices“

- Iterative Softwareentwicklung
- Anforderungsmanagement
- Verwendung Komponenten-basierter Architekturen
- Visuelle Software-Modellierung
- Prüfung der Softwarequalität
- Kontrolliertes Änderungsmanagement

... und sonstigen Eigenschaften

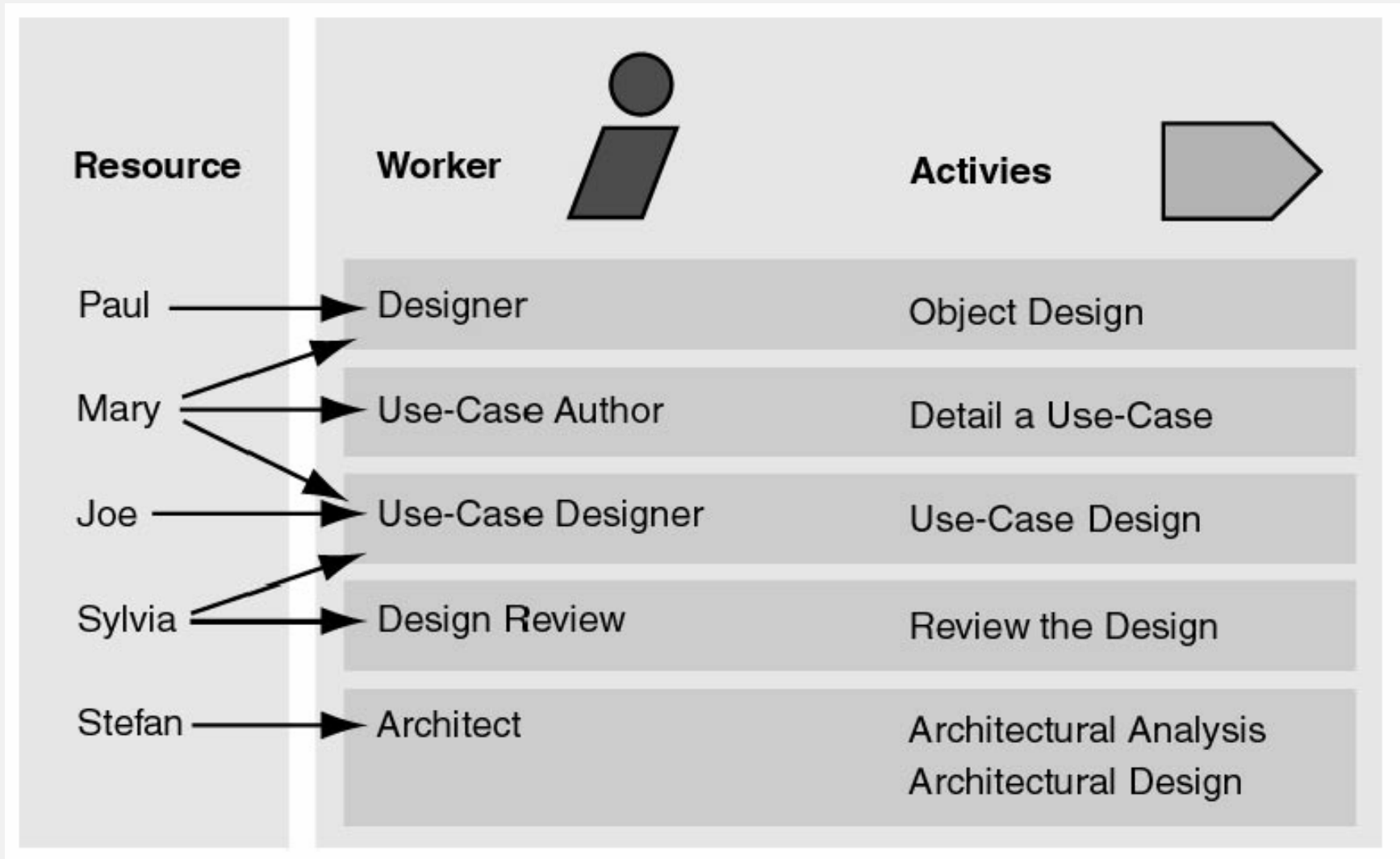
- Anwendungsfall-gesteuerte Softwareentwicklung
- Konfiguration des Prozesses
- Werkzeug-Unterstützung

# Rational Unified Process

Prozesse: Statische Struktur: Wer macht wann wie was?

- Wer** Worker; bearbeitet eine Reihe von zusammenhängenden Aktionen
- Wann** Workflow; gibt die Reihenfolge an, in der bestimmte Aktivitäten aufeinander folgen
- Wie** Aktivitäten; Teil der Tätigkeit des Workers, bestehen aus den Schritten *thinking*, *performing* und *reviewing* und sind beendet, wenn neues Artefakt entstanden ist
- Was** Artefakte; Informationseinheiten, die während des Projektes entstehen und gebraucht werden- sind sowohl Ein- als auch Ausgabe von Aktivitäten

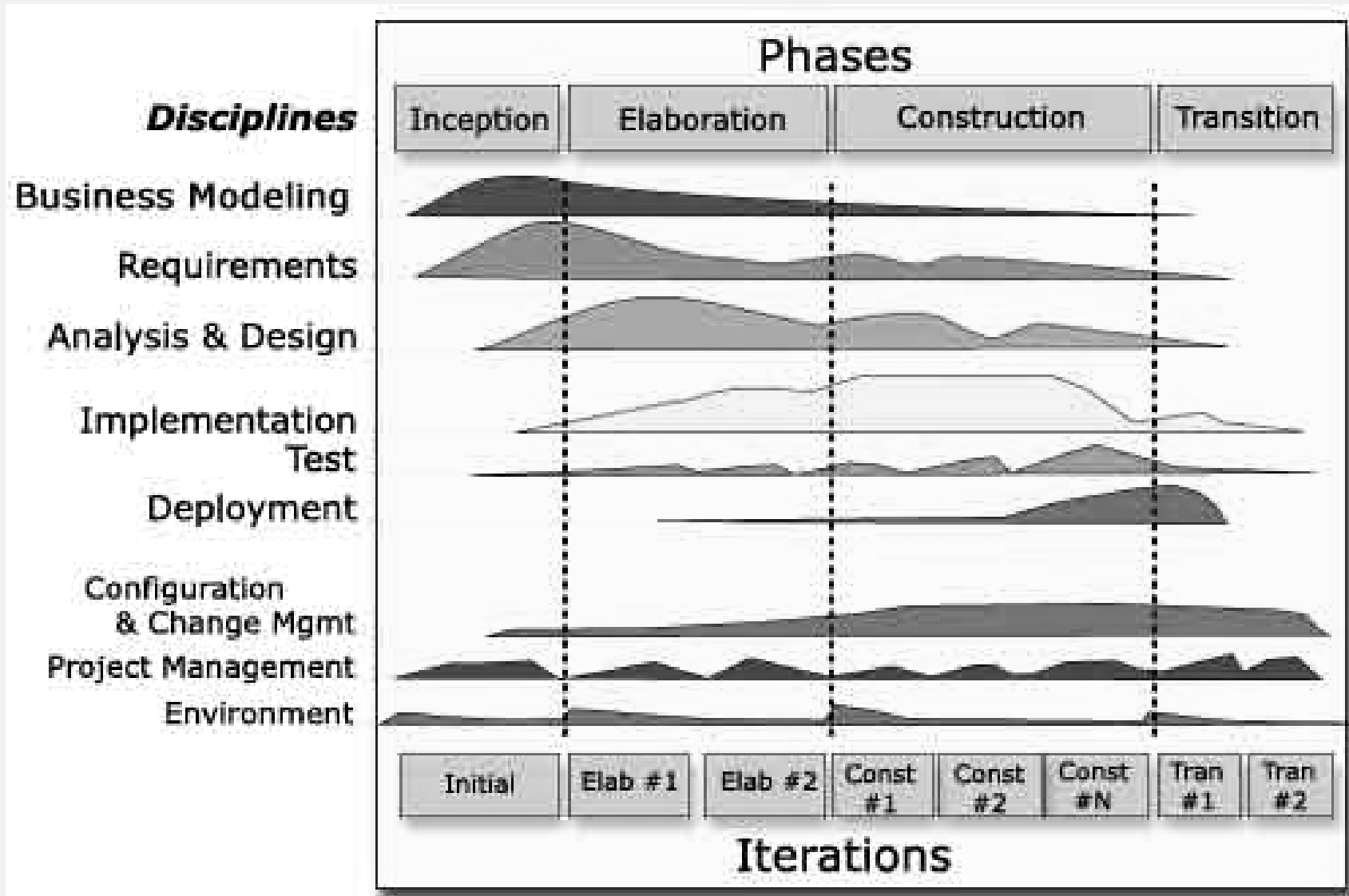
# Rational Unified Process: Personen und Worker



# Rational Unified Process: Kern Workflows

- *Sechs Core Workflows*
  - Business Modelling Workflow
  - Requirement Workflow
  - Analysis & Design Workflow
  - Implementation Workflow
  - Test Workflow
  - Deployment Workflow
- *Drei Supporting Workflow*
  - Project Management Workflow
  - Configuration and Change Management Workflow
  - Environment Workflow

# Rational Unified Process: Dynamische Struktur



# Rational Unified Process

## Iterativ-Inkrementeller Prozess

Lange Projektlaufzeiten und schnelle Veränderungen der Anforderungen in Projekten führen zu ...

- Aufgabenteilung in kleine Schritte (*Iterationen*)
- deren Ergebnisse (*Inkremente*) in das Produkt einfließen

Grund: bessere Erfolgsaussichten des Projektes

# Rational Unified Process

## Architekturzentrierter Prozess

Der Benutzer erhält eine Gesamtansicht der Architektur, unterteilt in fünf Sichten:

- Logische Sicht (Sicht des Endanwenders)
- Implementierungssicht (Sicht des Programmierers)
- Prozess- Sicht (Sicht des System-Integrators)
- Verteilungssicht (Sicht des System-Ingenieurs)
- Use-Case Sicht (Sicht des Analysten/Testers)



# Rational Unified Process: Ein architekturzentrierter Prozess

## Logische Sicht

- Funktionalität
- Endanwender

## Implementierungs-Sicht

- Software-Management
- Programmierer

## Use-Case Sicht

- Verhalten
- Analysten/Tester

## Prozess Sicht

- Performanz/  
Skalierbarkeit
- Systemintegrator

## Verteilungs Sicht

- Installation,
- Kommunikation
- System Ingenieur

# Rational Unified Process

## Anwendungsfall gesteuerter Prozess

- Anwendungsfall = Sequenz von Aktionen des Systems
- Beschreibt alle funktionalen Anforderungen an das System

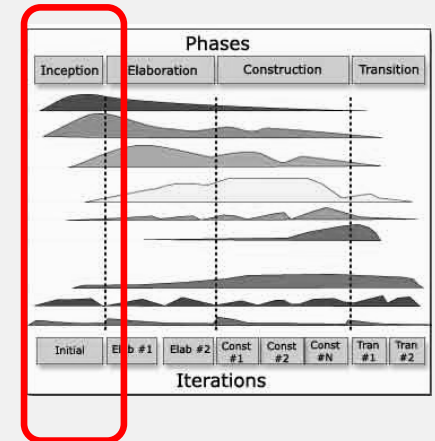
### Vorteile:

- intuitive Anforderungsgenerierung an das System möglich
- Steuerung der iterativen Entwicklung
- Unterstützung des Architekturentwurfs

# Phasen des Rational Unified Process

## I. Inception Phase-Konzeptualisierung

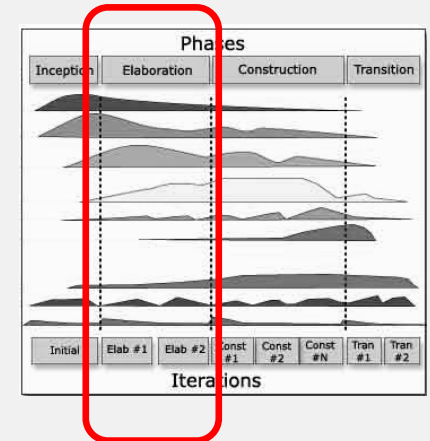
- Spezifikation der Endproduktvision
- Spezifikation der wesentlichen Geschäftsvorfälle
- Definition des Umfang des Projekts
- Kosten und Risiken vorhersagen
- Abschluß: Life Cycle Objective Milestone



# Phasen des Rational Unified Process

## II. Elaboration Phase -- Der Entwurf

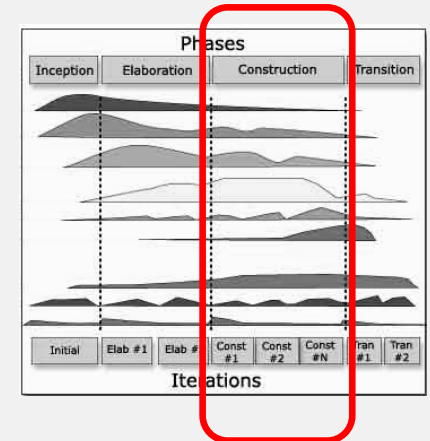
- Spezifikation der Produkteigenschaften
- Design der Architektur
- Planung der notwendigen Aktivitäten und Ressourcen
- Abschluß: Life Cycle Architecture Milestone



# Phasen des Rational Unified Process

## III. Construction Phase – Die Implementierung

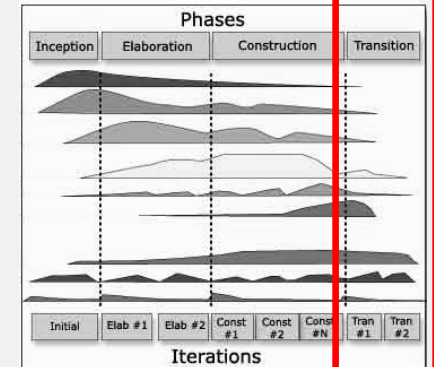
- Erstellung des Produkts
- Entwicklung der Architektur
- Ergebnis: fertiges Produkt
- Abschluß: Initial Operational Capability Milestone



# Phasen des Rational Unified Process

## IV. Transition Phase – Die Produktübergabe

- Freigabe des Produkts an die Benutzer
- Überprüfung des Qualitätslevels
- Auslieferung, Training, Einsatzunterstützung, Wartung
- Abschluß: Release Milestone



# Agile Methoden

Entwicklung agiler Methoden:

- 1975 beschreiben Victor Basili und Joe Turner iterative, inkrementelle Entwicklungsmöglichkeiten
- 1986 Barry Boehm veröffentlicht das *Spiralmodell*  
Erste Versuche mit der Methode *Scrum*
- 1994 Erste Versuch mit der Methode *DSDM*
- 1998 *Chaos-Report* der Standish-Group
- 1999 veröffentlicht Kent Beck sein *Extreme Programming (XP)*
- 2001 Gründung der *Agile Alliance*  
Veröffentlichung des *Agile Manifesto*

# Agile Methoden

Einige der bisher entstandenen Methoden:

- Adaptive Software Development (ASD)
- Crystal
- Dynamic System Development Method (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Lean Development (LD)
- Pragmatic Programming
- Scrum



# Extreme Programming (XP)

1999 entwickelt von Kent Beck als

*„eine leichte, effiziente, risikoarme, flexible, kalkulierbare,  
exakte und vergnügliche Art und Weise der  
Softwareentwicklung“*

# Extreme Programming (XP)

XP wird auch als *Puzzle-Programmierung* bezeichnet:

- Programmierung findet in kleinen Teilen statt
- Einzelne Code-Teile machen keinen Sinn
- Erst das zusammengesetzte Ganze funktioniert

Kernansprüche:

- Kunde ist Teil des Projektteams
- Konsequente Orientierung des Projektes an Kundenanforderungen
- Kurze Release-Zeiten
- Iteratives Vorgehen
- Integriertes Testen

# Extreme Programming (XP)

XP kann in vier Bereiche eingeteilt werden:

- |           |                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------|
| Planen:   | Ausarbeitung der Anforderungen durch Team und Kunde in <i>Stories</i><br>Einteilen des Projektes in Iterationen |
| Kodieren: | Code wird nach vereinbarten Standards entwickelt<br>Der Code wird immer in Paar-Programmierung erzeugt          |

# Extreme Programming (XP)

- Modellieren: Einfache Strukturen mit einheitlicher Systemmetapher  
*Refactoring* soviel und sooft wie möglich
- Testen: Jedes Codefragment wird getestet  
Ergebnisse von Akzeptanztests werden veröffentlicht

# www.xprogramming.com



## Extreme Programming Project

