DHBW
Duale Hochschule
Baden-Württemberg

Skript zur Vorlesung

Software Engineering I

Vorgehensmodelle

Dozenten:
Markus Rentschler
Christian Holder

Definition: Entwicklungsprozess

- Zur erfolgreichen Durchführung eines Projekts wird ein **Software-Entwicklungsprozess** angewendet.

- Darunter versteht man ein **Vorgehensmodell**, also die Beschreibung einer koordinierten Vorgehensweise bei der Abwicklung eines Vorhabens.

- Das Vorgehensmodell legt eine Reihe von **Aktivitäten** sowie deren **Input** und **Output** (Artefakte) fest. Weiterhin erfolgt eine feste Zuordnung von **Rollen**, welche die jeweilige Aktivität ausüben.

- Die Rollen, und somit die Verantwortung für die Aktivitäten, werden den einzelnen Mitgliedern des **Projektteams** zugeordnet.

Definition: Rollen, Aktivitäten, Artefakte

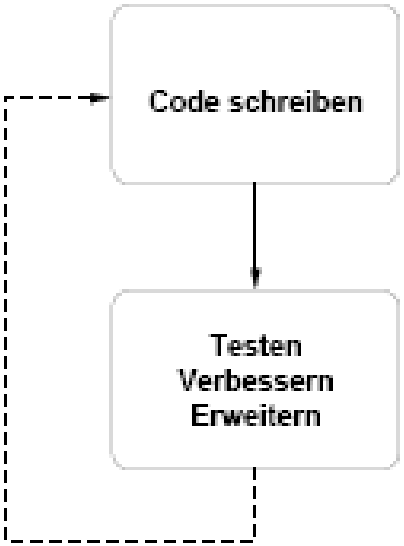
Prozesse werden mittels folgender Hauptkonzepte beschrieben:

- **Rolle:**
  - Aufgabe einer Person in einer bestimmten Situation
  - z.B. Analytiker, Entwickler, Tester, Projektleiter usw.
  - dieselbe Person hat in einem Projekt oft mehrere Rollen

- **Aktivität:**
  - Zielgerichtetes Handeln in einem Projekt
  - z.B. Anforderungsermittlung, Architekturentwurf, Modulentwurf, Codierung, Modultest etc.

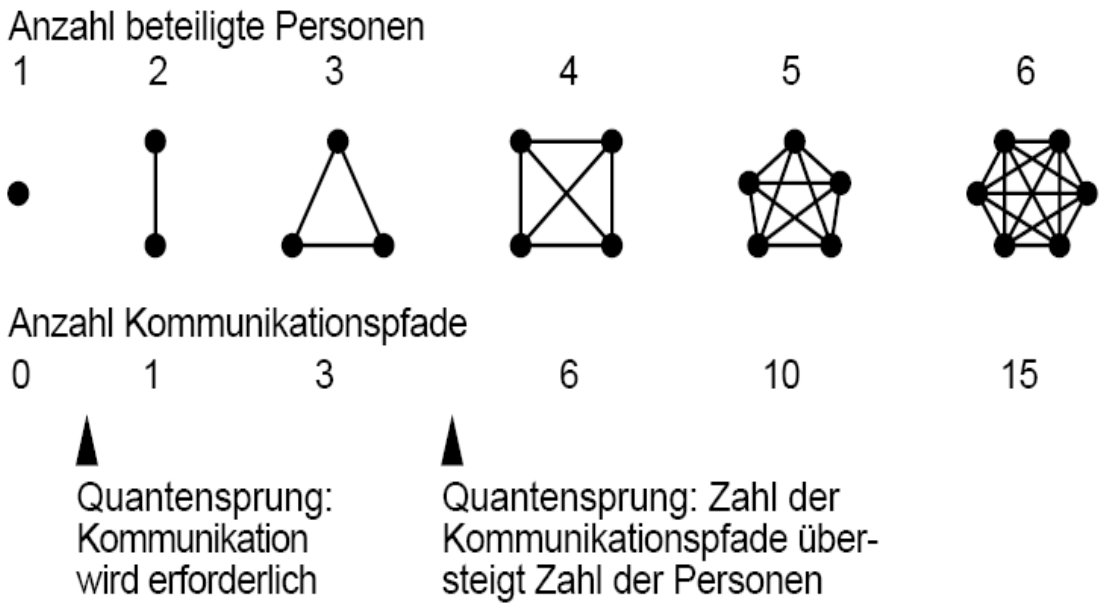
- **Artefakt:**
  - Arbeitsergebnis einer Aktivität
  - z.B. Pflichtenheft, UML-Klassendiagramm, Programmcode, Testfall, Testbericht

„Code-and-Fix“ Vorgehensmodell



- Das wohl einfachste Modell ist die „Code-and-Fix“-Vorgehensweise, bei der Entwickler ohne ein festgelegtes Prozessmodell Funktionen implementieren.
- Anschließend werden die so entstandenen Komponenten ad-hoc „getestet“ und gegebenenfalls verbessert (= Debugging).
- Für kleine Projekte ist dieser Ansatz durchaus angemessen, versagt aber bei größeren Software-Projekten völlig.
- **Die Verwendung dieses Modells bei größeren Systemen war ursächlich für die Softwarekrise.**

Problem: Kommunikation in Projekten



Anforderungen an Vorgehensmodelle

- Für die Entwicklung größerer Systeme müssen daher **geeignete Vorgehensmodelle** definiert werden, die ein **strukturiertes**, **arbeitsteiliges** Vorgehen erzwingen.

- Diese Modelle müssen den Entwicklungsprozess in an der Realität orientierte, sinnvolle Phasen unterteilen.

- Für die Übergänge zwischen den Phasen müssen Dokumentationsstandards vereinbart werden, um die arbeitsteilige Kommunikation zu unterstützen.

→ Geeignete Phasen müssen identifiziert werden !

Definition: Phasen im Software-Lebenszyklus

- Software unterliegt (wie andere Produkte in ähnlicher Form) auch einem allgemeinen **Lebenszyklus**:
  - **Systementwicklung**
    Entwicklung der Software
  - **Systemeinführung**
    Vorbereitungen zur Nutzung der Software
  - **Wachstum**
    Verbreitung der Nutzung der Software
  - **Reife (Wartung)**
    Vornahme von Verbesserungen an der Software, Beseitigung von Fehlern
  - **Ablösung (Migration)**
    Schrittweiser Übergang zu einem neuen Software-Produkt oder zu einer neuen Version des Produkts

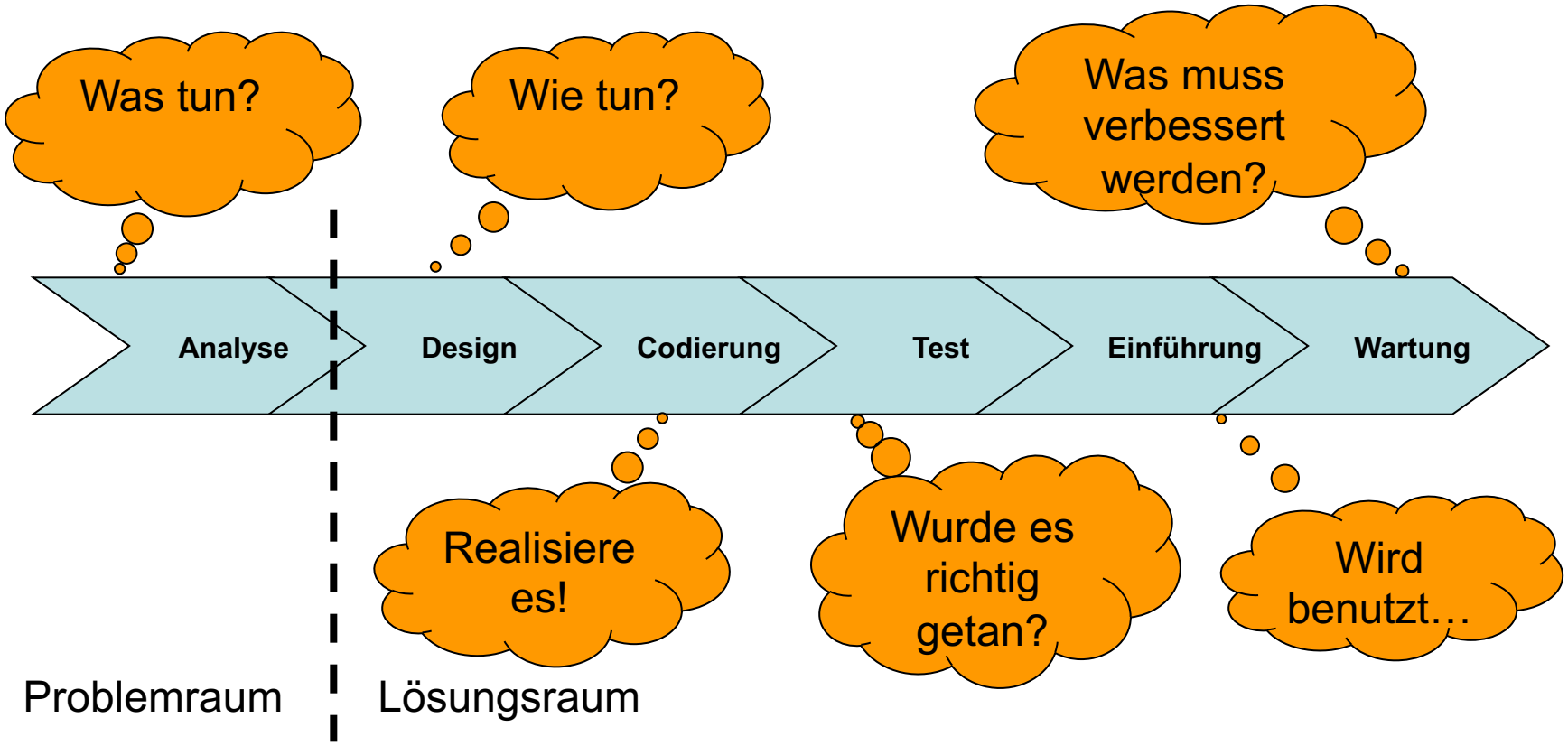
Phasen im Entwicklungsprozess

- Um ein Software-Projekt erfolgreich durchführen zu können, sollte ein **Software-Entwicklungsprozess** angewendet werden, der die Phasen des **Software-Lebenszyklus** angemessen behandelt.

- Das in der Vorlesung behandelte Themengebiet beschränkt sich im Wesentlichen auf die Phasen **Systementwicklung** und **Systemeinführung**, also von der Problemspezifikation bis zur Abnahme des daraufhin entwickelten Produkts durch den Kunden.

→ Weitere Unterteilung dieser Phasen ist notwendig, um ein passendes Vorgehensmodell zu definieren!

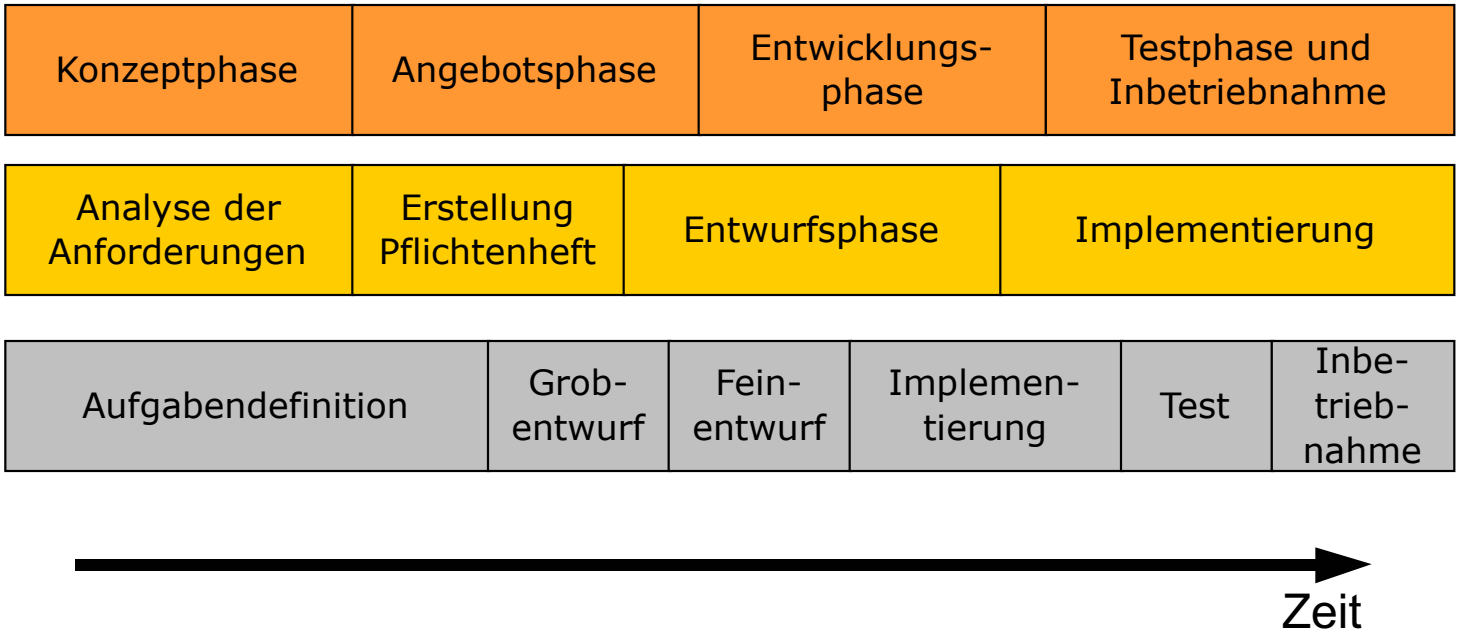
Entwicklungsphasen: Aktivitäten



Phasen im Entwicklungsprozess

- Diese **Aktivitäten** sind Teil jeder Softwareentwicklung:
  - **Analyse** ➔ definieren, **was** das System tun soll
  - **Design** ➔ definieren, **wie** das System realisiert werden soll
  - **Implementierung** ➔ Realisierung des Systems
  - **Validierung** ➔ prüfen, ob das System die Anforderungen erfüllt
  - **Einführung** ➔ das System beim Kunden in Betrieb nehmen
  - **Wartung** ➔ weiterentwickeln des Systems nach geänderten Kundenanforderungen
- Da diese **Aktivitäten** normalerweise sequentiell bearbeitet werden, können dafür auch **Phasen** definiert werden.
- Tatsächlich enthalten **alle** Prozessmodelle typischerweise diese **Aktivitäten** und daraus abgeleitete **Phasen**!

Phasenmodelle



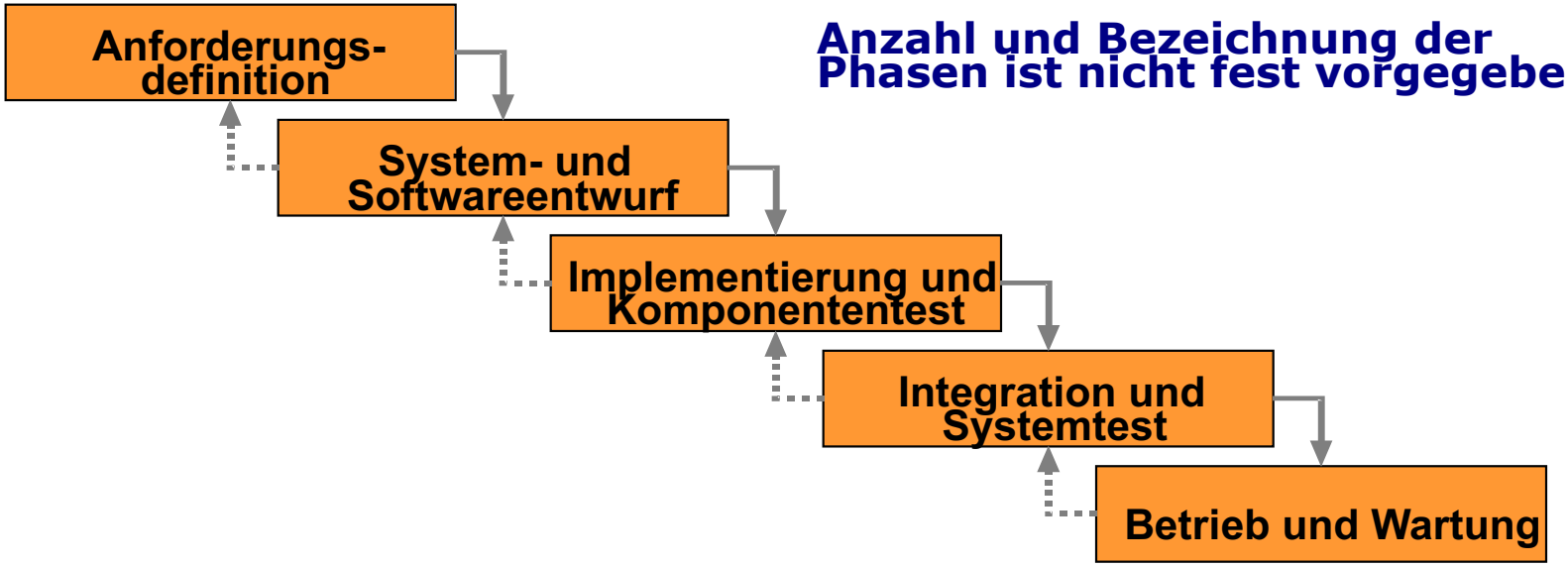
- Streng sequenziell, keine Iterationen
- Zeitanteil der einzelnen Phasen fest

Anzahl und Bezeichnung der Phasen ist nicht fest vorgegeben

Reine Phasenmodelle entsprechen nicht der Realität!

Wasserfallmodell

- Das Wasserfallmodell ist ein bekannter Ansatz zur Strukturierung des Entwicklungsprozesses nach dem Phasenprinzip.
- Der Name drückt aus, dass man sich wie bei einem mehrstufigen Wasserfall von der Planungsphase zur Wartungsphase bewegt.
- Weitere Unterteilung der Phasen ist möglich (z.B. in Implementierungsphase und Integrationsphase)
- Grundlegendes Modell mit vielen Varianten wie Anzahl der Phasen, Überlappung, Rücksprungmöglichkeiten etc.

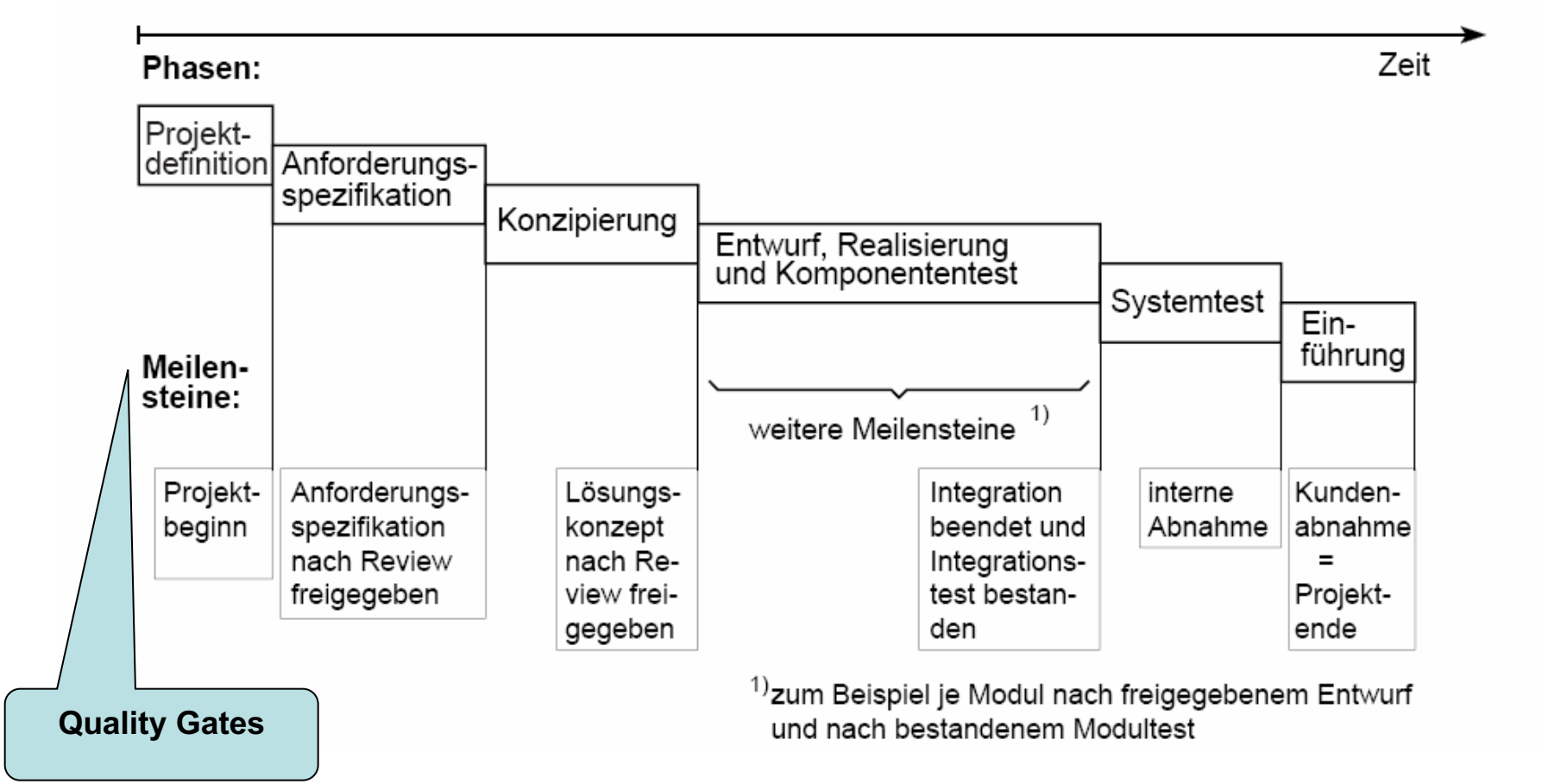


Phasenmodelle: Nachteile

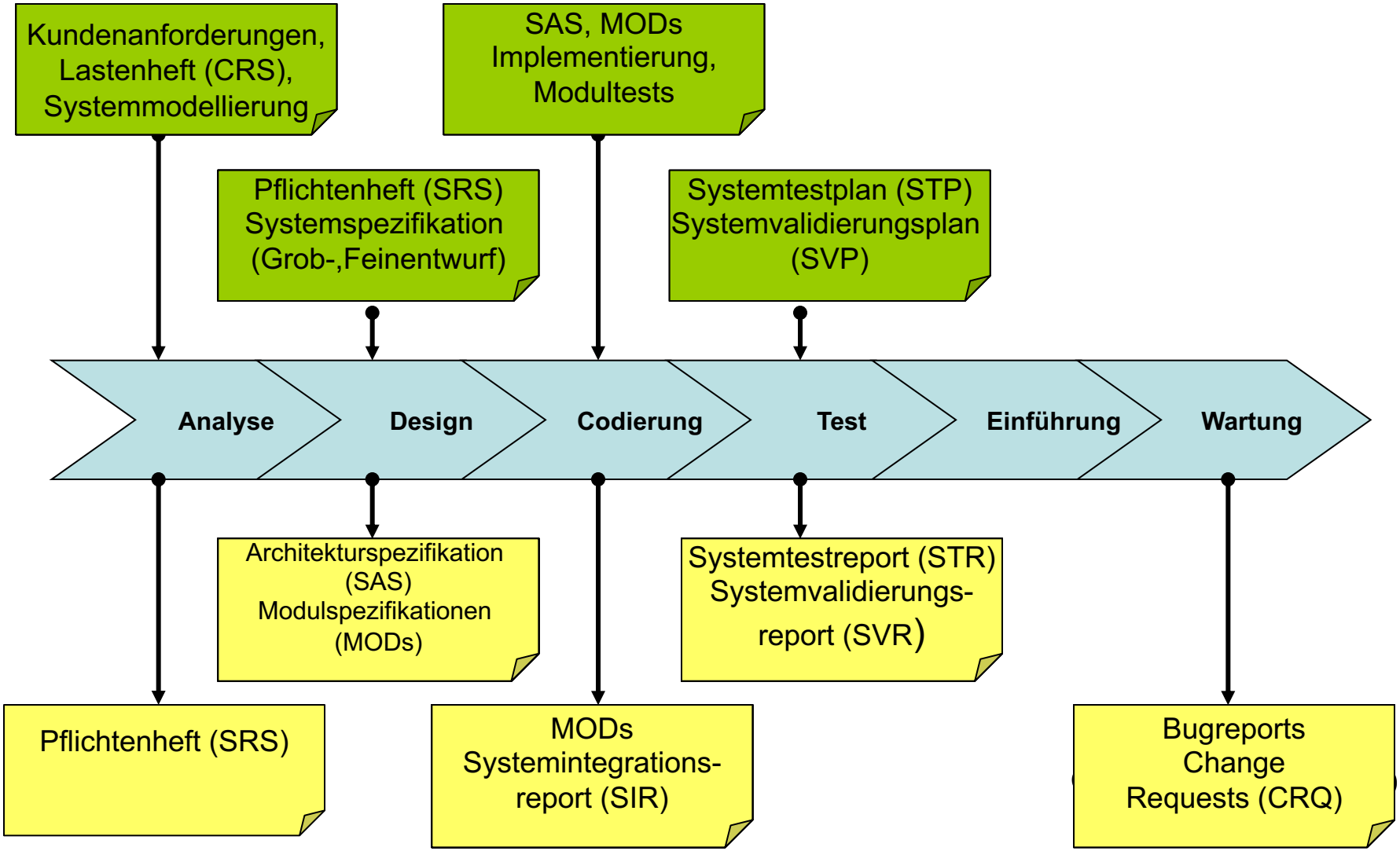
- Die Annahme, dass zu Beginn eine abgeschlossene und korrekte Anforderungsdefinition existiert, entspricht meist nicht der Realität.
- Auftraggeber ist nur in der ersten Phase (Anforderungsdefinition) mit eingebunden.
- Lauffähige Version des Systems liegt erst am Ende der Entwicklung vor (Big-Bang).
- Testen ist nur am Ende des Entwicklungszyklus vorgesehen.
- Nichteinhalten der Projektdauer führt oft zu Abstrichen in den späten Phasen (vor allem beim Testen)

Phasenmodelle Vorteile

- Gut geeignet zur Projektführung

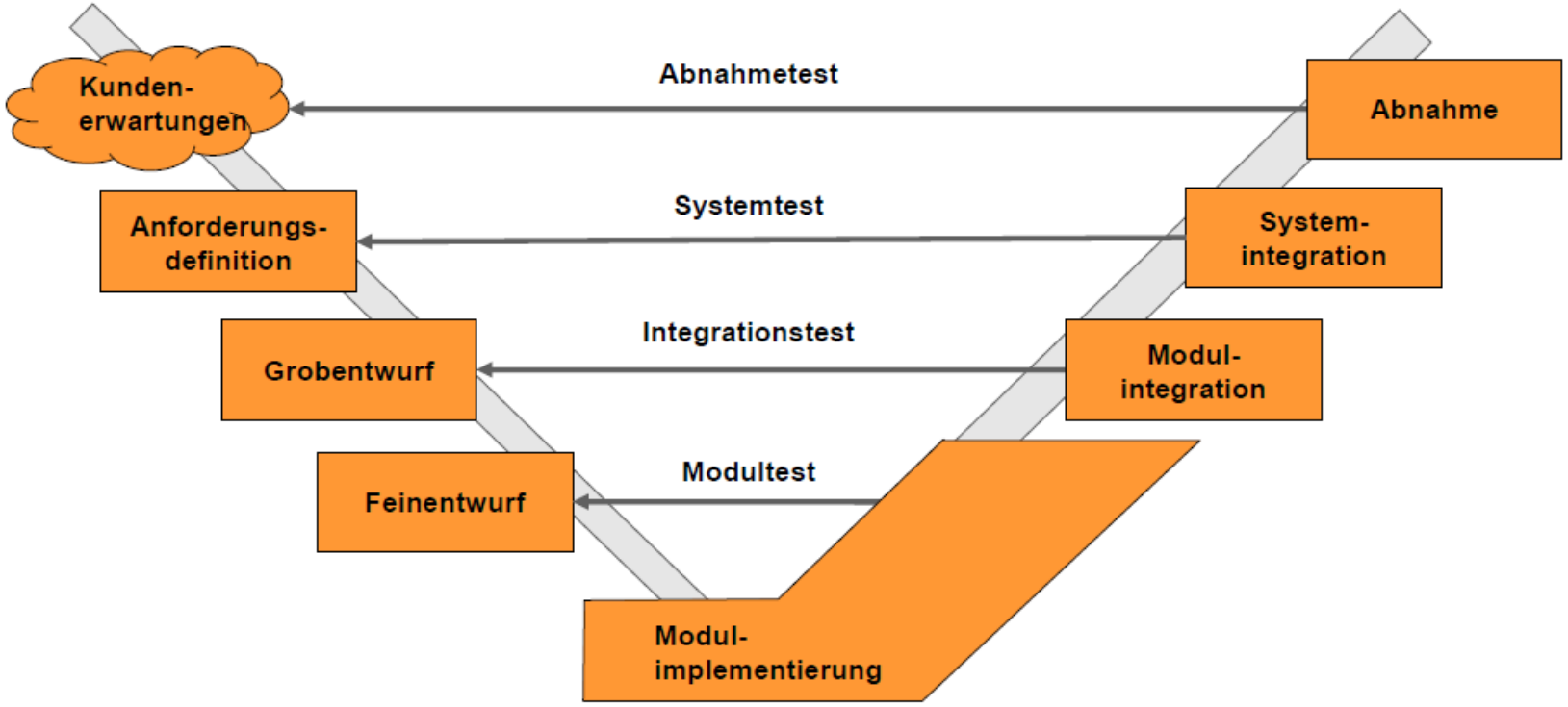


Phasenmodell: Inputs, Outputs



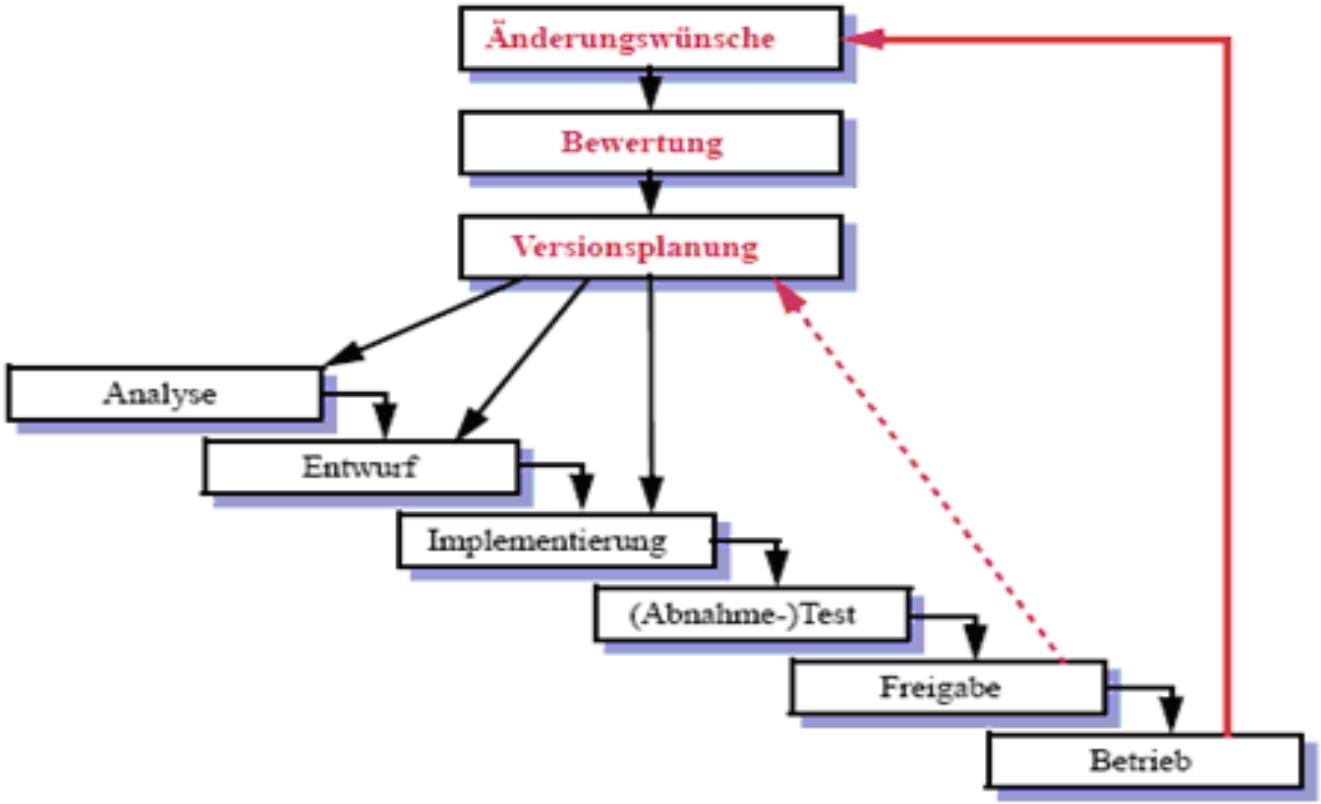
V-Modell

- Weiterentwicklung des ergebnisorientierten Phasenmodells.
- **QS-orientiert**: Es hebt die Validierung der Phasen durch Testphasen auf jeder Abstraktionsebene hervor.
- Weit verbreitet in der Serienproduktentwicklung.
- Vorgeschrieben in der Entwicklung sicherheitsgerichteter Systeme

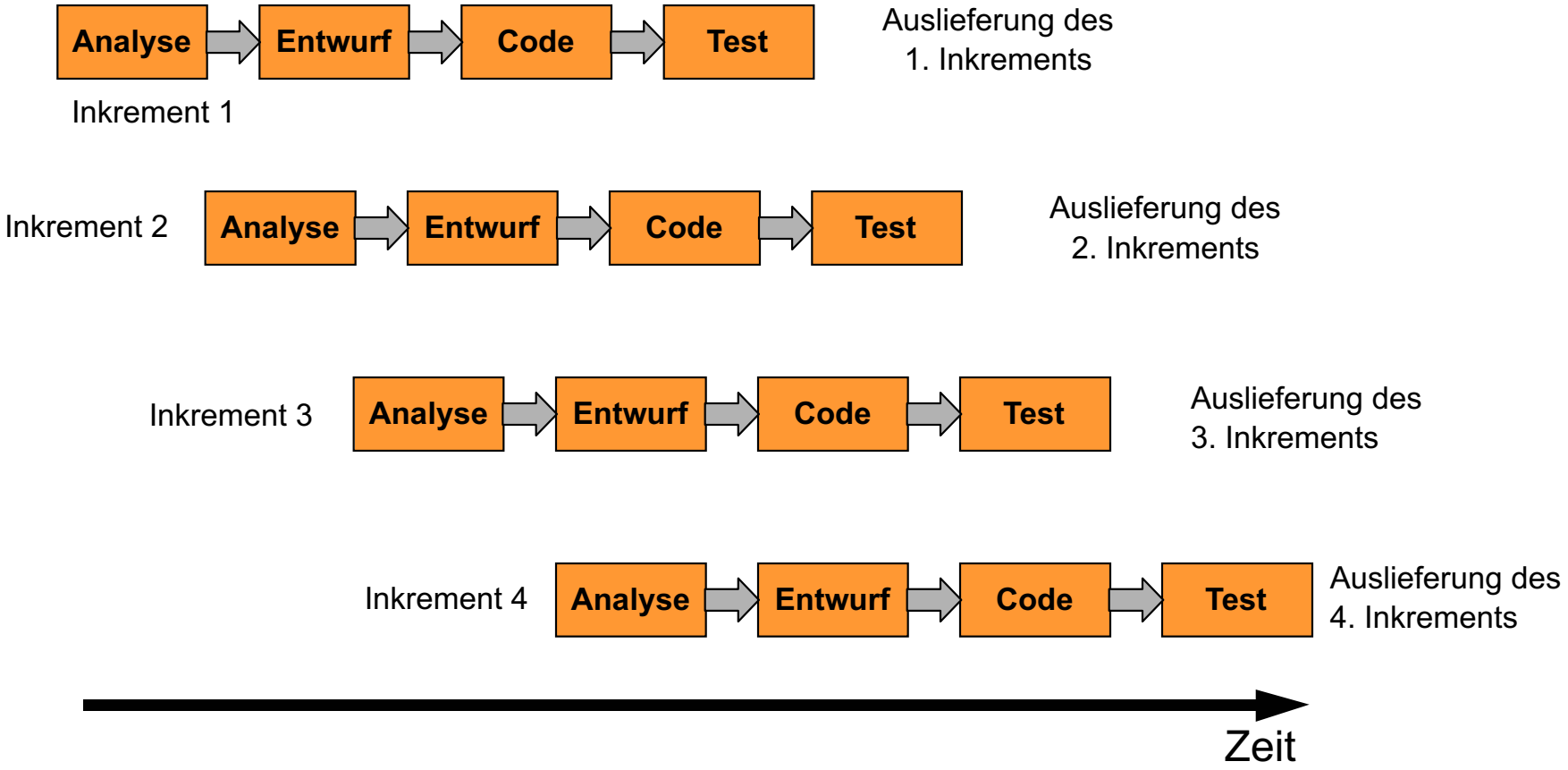


Inkrementelles Metamodell

- Darstellung eines übergeordneten Modells zur Durchführung von Software-Inkrementen (Versionen).
- Änderungswünsche können natürlich auch Fehlermeldungen sein!



Parallelisierte inkrementelle Entwicklung



Inkrementelle Entwicklung: Vor-/Nachteile

Vorteile der inkrementellen Entwicklung

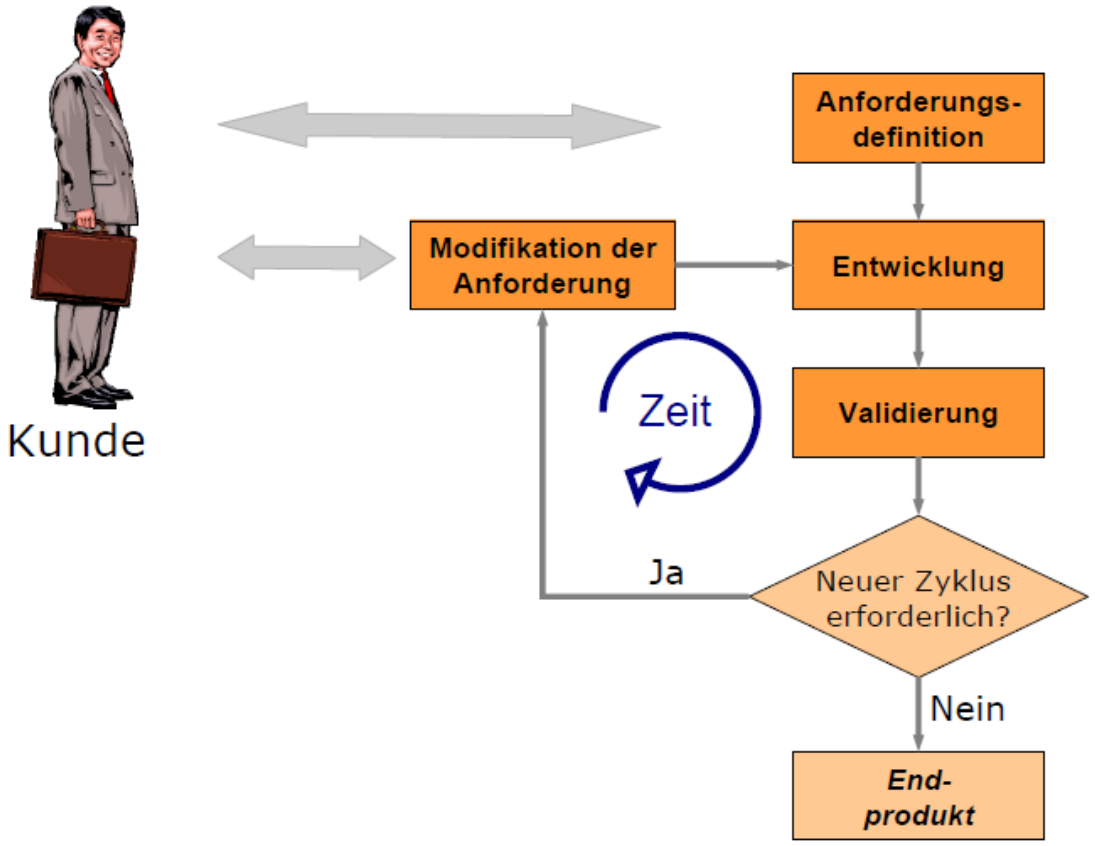
- Kundenwünsche können in jedem Inkrement erfüllt und ausgeliefert werden, somit ist Systemfunktionalität früher verfügbar.
- Frühe Inkremente agieren als Prototyp und helfen Anforderungen für spätere Inkremente zu eruieren.
- Geringeres Risiko für das Scheitern des Gesamtprojektes.

Nachteile der inkrementellen Entwicklung

- Abbildung der Kundenbedürfnisse (Anforderungen) auf einzelne Inkremente ist nicht trivial
- Ermittlung der Grundfunktionen für das erste Inkrement ist schwierig

Grundfunktionen sind Funktionen, die später von Teilsystemen gebraucht werden. → Gute Systemarchitektur ist wichtig!

Evolutionäre iterative Modelle



Evolutionäre Modelle: Eigenschaften

- Softwareentwicklungsprozess ist nicht linear, sondern Folge von **Entwicklungszyklen** (Iterationen)
- Entwicklung verläuft über viele kleine Versionen, bis ein angemessenes Produkt entsteht
- Verwandt mit inkrementellem Modell

- Evolutionäre Entwicklung beginnt typischerweise mit einem Prototyp:
  - **Evolutionäres Prototyping**
    - Anforderungen des Kunden werden nach und nach umgesetzt
    - Entwicklung beginnt bei den eindeutigen Anforderungen
  - **"Wegwerf"-Prototyping (Throw-Away)**
    - Anforderungen des Kunden sollen verstanden werden
    - Entwicklung beginnt bei den unklaren Anforderungen

Evolutionäre Modelle: Vor-/Nachteile

- **Vorteile der evolutionären Modelle**
  - Flexibilität ("Agilität") während des gesamten Entwicklungsprozesses
  - Akzeptanz des Kunden kann durch fortlaufende Einbeziehung seiner Wünsche verbessert werden
  - Kleine, inkrementelle Teilschritte können die Messung des Projektfortschritts erleichtern

- **Nachteile der evolutionären Modelle**
  - Schlechte Planbarkeit der Projektumfänge
  - Kosteneffiziente Dokumentation schwierig
  - Stetige Veränderungen führen u.U. zu schlecht strukturierten Systemen
  - Schlechte Eignung für große Systeme mit langer Lebensdauer

Evolutionäres Modell: e**X**treme **P**rogramming (**XP**)

- Flexibles („agiles“) Vorgehensmodell
- Basiert auf der Entwicklung und Auslieferung sehr kleiner Inkremente (Teilsysteme)
- Geeignet für Projekte mit max. 10-15 Entwicklern

- Wesentliche Richtlinien in XP:
  - ***Develop for today***
    Konzentration auf aktuelle Probleme
  - ***Do the simplest thing that could possible work***
    - Verwendung des einfachsten Entwurfs (Simple Design)
  - ***Test Driven Development***
    - Vor der Implementierung Tests statt Spezifikationen erstellen
    - Kunde entwickelt Testfälle für eine Story (functional test)
    - Entwickler schreiben automatisierte Tests für ihre Klassen (unit tests)

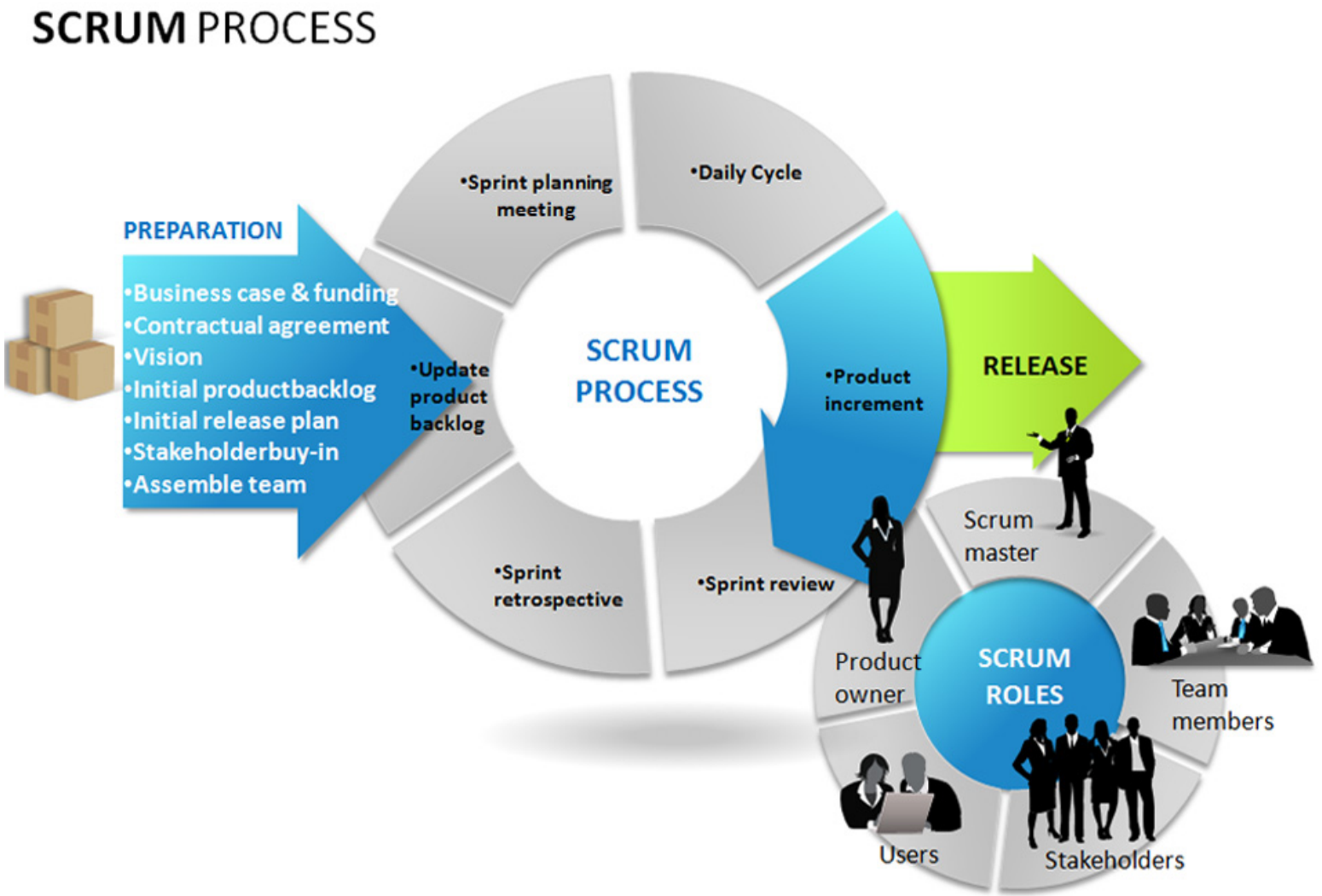
➔ Ist schon wieder etwas aus der Mode gekommen…

Evolutionäres Modell: **SCRUM**

- Inkrementelle Vorgehensweise mit dem Ziel eines releasefähigen Produkt am Ende jedes Entwicklungsabschnitts
- Organisation der Entwicklungsabschnitte und Meetings in vordefinierten Zeitabschnitten (Time-Boxes)

- Viele neue Terminologien wurden eingeführt , z.B.:
  - ***Product Owner*** (stellt fachliche Anforderungen und priorisiert)
  - ***Scrum Master*** (managt den Prozess und beseitigt Hindernisse)
  - ***Team Member*** (entwickelt das Produkt)
  - ***Stakeholder*** (Beobachter und Ratgeber)
  - ***Product Backlog*** (Anforderungsliste für das Gesamtprodukt)
  - ***Sprint*** (Entwicklungsabschnitt)
  - ***Sprint Backlog*** (Verwaltung der "**Tasks**" für den Sprint-Zyklus)
  - ***Daily Scrum Meeting*** (Tägliches 15min Projektmeeting)
  - ***Sprint Review Meeting*** (Präsentation des Produkts am Ende jedes Sprints)
  - …

Evolutionäres Modell: **SCRUM**



Zusammenfassung Iterative Prozessmodelle

- **Inkrementelles Modell:**
  - Baue das Gesamtsystem schrittweise mit anfänglich klaren Anforderungen.
  - In jedem Schritt werden nur neue Teile hinzugebaut, es wird jedoch (theoretisch) nie etwas Existierendes verändert.
  - Folge in jeder Iteration (Inkrement) einem Phasenmodell

- **Evolutionäres ("Agiles") Modell:**
  - Baue das Gesamtsystem schrittweise mit anfänglich unklaren Anforderungen.
  - In jedem Schritt werden neue Teile hinzugebaut und wo nötig auch existierende verändert.
  - Am Ende jeder Iteration wird neu geplant.

- **Prototypenmodell:**
  - Baue anfangs ein (Teil)System "zum Wegwerfen", um unklare Anforderungen besser zu verstehen.
  - Dann folge einem Phasenmodell.

Überblick: Vorgehensmodelle

- **Phasenmodelle**
  - Plangetrieben. Getrennte und eindeutige Phasen für Spezifikation und Entwicklung.

- **Iterative Modelle**
  - Spezifikation, Entwicklung und Validierung erfolgen überlappend. Kann plangetrieben oder evolutionär (agil) sein.

Es gibt keine richtigen oder falschen Modelle.
In der Realität werden die meisten großen Systeme nach einem Prozess entwickelt, der Elemente aus allen diesen Modellen enthält!

Planungshorizont der Prozessmodelle

Wichtiges Unterscheidungsmerkmal für Prozessmodelle:

- Wie präzise / strikt / weit voraus wird geplant?

- **Sehr:** Phasenmodelle
  - möglichst präzise und strikt, für das gesamte Projekt im Voraus

- **Mittel:** Inkrementelle Modelle
  - präzise wo möglich
  - nicht strikt (nötige Veränderungen werden akzeptiert)
  - nur für wenige Iterationen im Voraus

- **Wenig:** Agile Modelle
  - Nur so viel Planung wie unbedingt nötig
  - Lieber Ziele als Pläne (wegen Flexibilität)

Auswahl des Prozessmodells

- **Plangetriebene Modelle:**
  - Wenn exaktes Resultat in definierter Zeit erreicht werden muss
  - Wenn sehr große (insbesondere verteilte) Projektgruppen koordiniert werden müssen
    - Dann sind Pläne und Dokumente zur Koordination unverzichtbar
  - Parallele Entwicklung von HW und SW (z.B. Automotive)
  - Verteilte Entwicklung (z.B. über mehrere Firmen, Standorte)
  - Wiederverwendung ist geplant/gewünscht

- **Agile Modelle:**
  - Wenn hohe Unsicherheit über die Anforderungen besteht
  - Wenn Änderungen von außen häufig sind
    - Anforderungen, Zeitplan, Budget, Qualitätsziele etc.
  - Projekt hat keinen Zeitplan (z.B. Hobbyprojekt)
  - Arbeit mit unausgereifter und unbeherrschter Technologie

Zusammenfassung

- **Softwareentwicklungsprozesse** bestehen aus Aktivitäten, die bei der Entwicklung von Softwaresystemen vorkommen und in einem Vorgehensmodell dargestellt werden.

- Allgemeine **Aktivitäten** sind Anforderungsanalyse und -spezifikation, Entwurf und Implementierung, Validierung und Weiterentwicklung.

- **Phasenmodelle** arbeiten streng sequentiell nach einem am Anfang vorgegebene Plan

- **Iterative Vorgehensmodelle** beschreiben den Softwareentwicklungsprozess als Kreislauf von Aktivitäten

- Für eine **inkrementelle Entwicklung** muss im Initialprojekt ein Grobkonzept und die Definition der Systemarchitektur für den gesamten Projektumfang festgelegt werden.