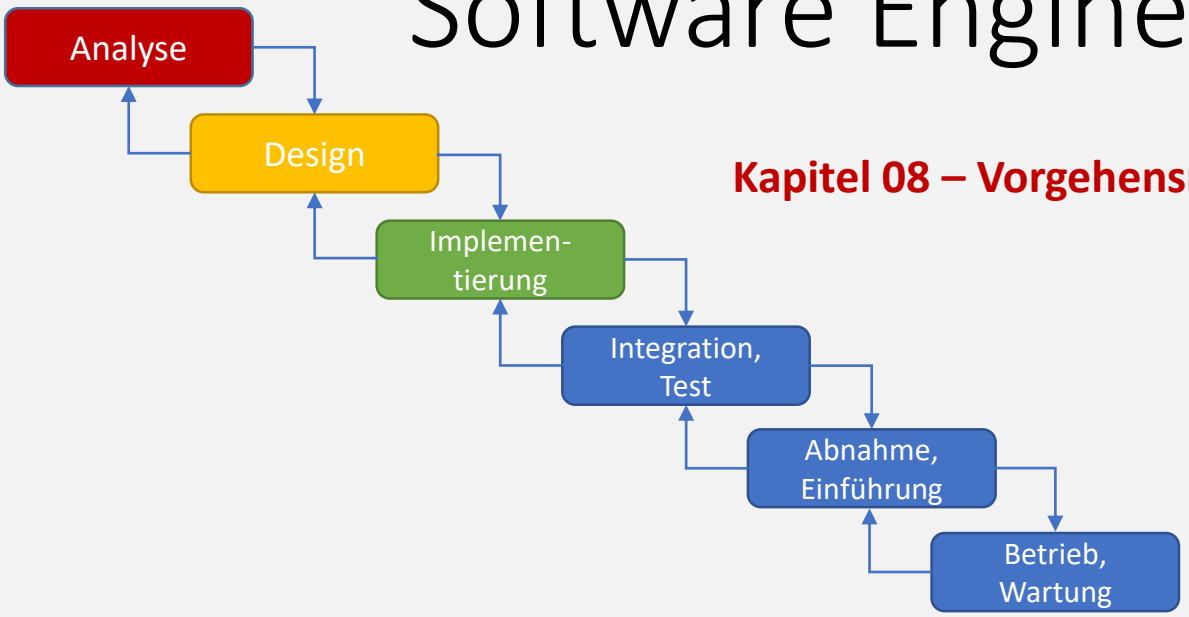
Software Engineering



Kapitel 08 – Vorgehensmodelle

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Gliederung

1. Was sind Vorgehensmodelle
2. Phasen der Software-Entwicklung
3. Vorgehensmodelle im Detail
    1. Wasserfall-Modell
    2. V-Modell
    3. Prototypen
    4. Iterative Modelle
    5. Inkrementelle Entwicklung
    6. Iterativ-Inkrementelle Entwicklung (Spiralmodell)
    7. Agile Methoden (Scrum)

Was sind Vorgehensmodelle?

Was sind Vorgehensmodelle und wozu benötigen wir sie?

Einsicht:

- Nicht die Gesamt-Vorgehensweise in jedem Projekt neu erfinden
- Sondern auf vorhandene Erfahrungen bauen

Vorgehensmodelle im SE:

- Strukturierung des Projektes:
  - Trennung der zeitlich abgegrenzten Phasen und darin enthaltenen Aktivitäten

Nutzen von Vorgehensmodellen:

- Kommunikation: gemeinsames Verständnis der Aufgaben und Verantwortlichkeiten
- Vollständigkeit: um nichts Wesentliches zu übersehen
- Vorhersage/Planung: über das Projektergebnis
- Basis für Projektkontrolle, Analyse und Harmonisierung der Erwartungen
- Erfahrungsaufbau durch Lernen aus Projektdurchführungen

Generelle Prinzipien von Vorgehensmodellen

- **Planung und Koordination**
  - Risikominimierung: alle Beteiligten können im Voraus erkennen, was wann getan werden muss
- **Korrektur und kontinuierliche Verbesserung**
  - Prozess so gestalten, dass unvermeidlich auftretende Fehler gut ausgeglichen werden können
- **Iteration**
  - Risikominimierung: Projekt bringt in definierten Abständen einsetzbare Versionen des Softwareproduktes hervor

Ziel von Vorgehensmodellen

Softwareentwicklungsprozess wird transparenter und somit:

- planbar
- nachvollziehbar
- kontrollierbar
- lehrbar

Auswirkung auf das Software-Produkt:

- höhere Qualität
- effizientere Produktion
- bessere Wartbarkeit
- und dadurch:
  - schnellere Fehlerbehebung
  - erhöhte Änderungsfreundlichkeit

Wichtigste Entscheidungen für SW-Projekte

- Projektziele
- Zeit- und Budgetplanung
- Projektorgansiation
- Verwendetes Vorgehensmodell
- Verwendete Technologie, Werkzeuge und Methoden
- Team-Mitglieder
- ...

Phasen der Softwareentwicklung

Eine erste Idee zur Strukturierung von Vorgehensmodellen

Definition von Phasen, d.h. zeitlich begrenzten Aktivitäten

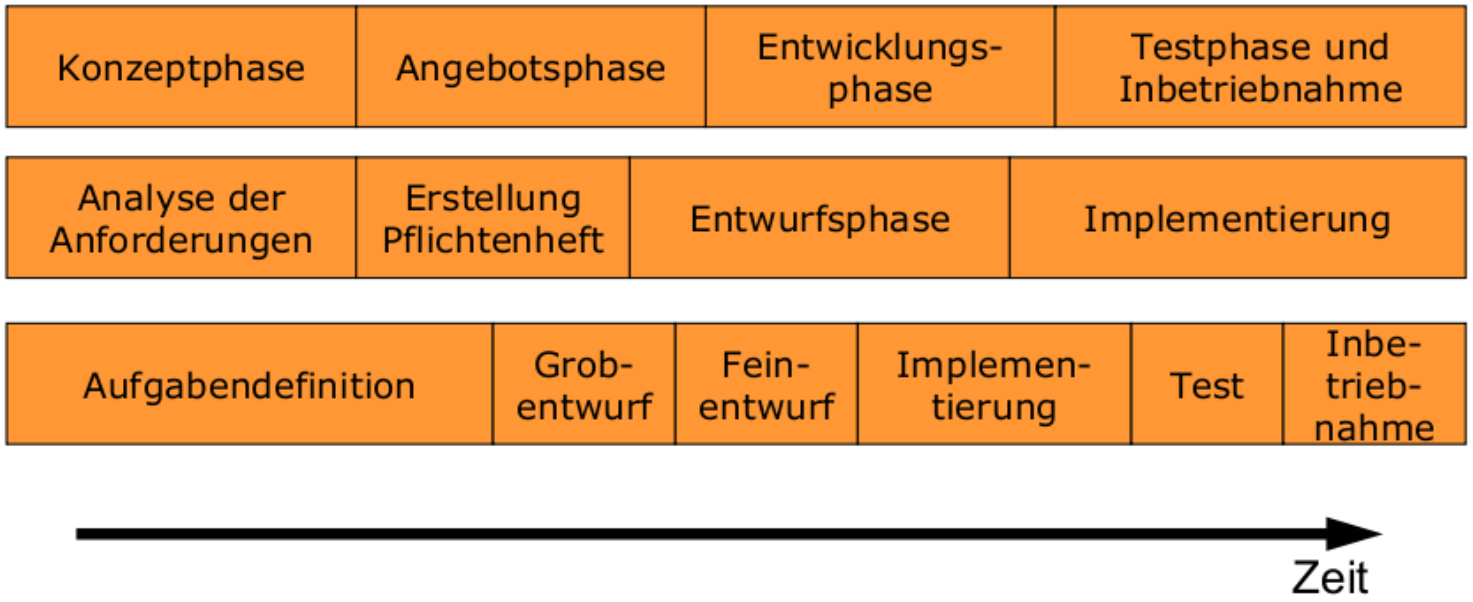
mit einer speziellen Aufgabe,

die von Mitarbeitern mit geeigneten Rollen bearbeitet werden,

um basierend auf vorgegebenen Artefakten neue, definierte Artefakte zu produzieren.

- Wird zu jedem Zeitpunkt nur genau eine Phase durchlaufen, so sprechen wir von einem Phasenmodell

Phasenmodelle



- Phasenmodelle entsprechen i.d.R. nicht der Realität

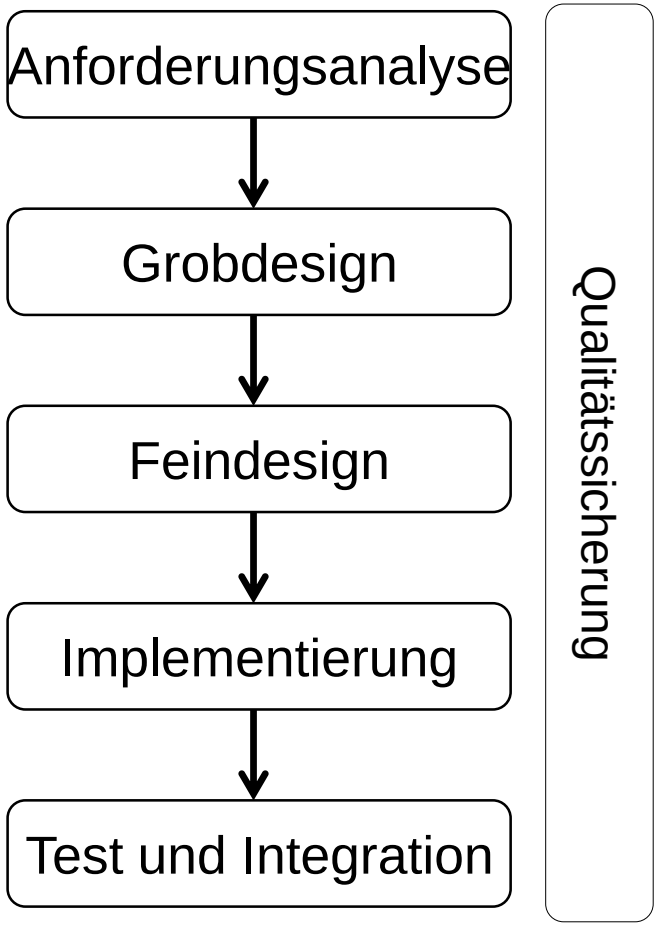
Aktivitäten in der Softwareentwicklung

Wir haben die Hauptaktivitäten bereits kennengelernt:

- Analyse
- Entwurf
- Implementierung
- Test (Validierung)
- Deployment (Installation, evtl. Schulung)
- Evolution (vor allem Wartung)
- … (Versionsmanagement, Reviews, etc.)

Die Phasen der Software-Entwicklung

- 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

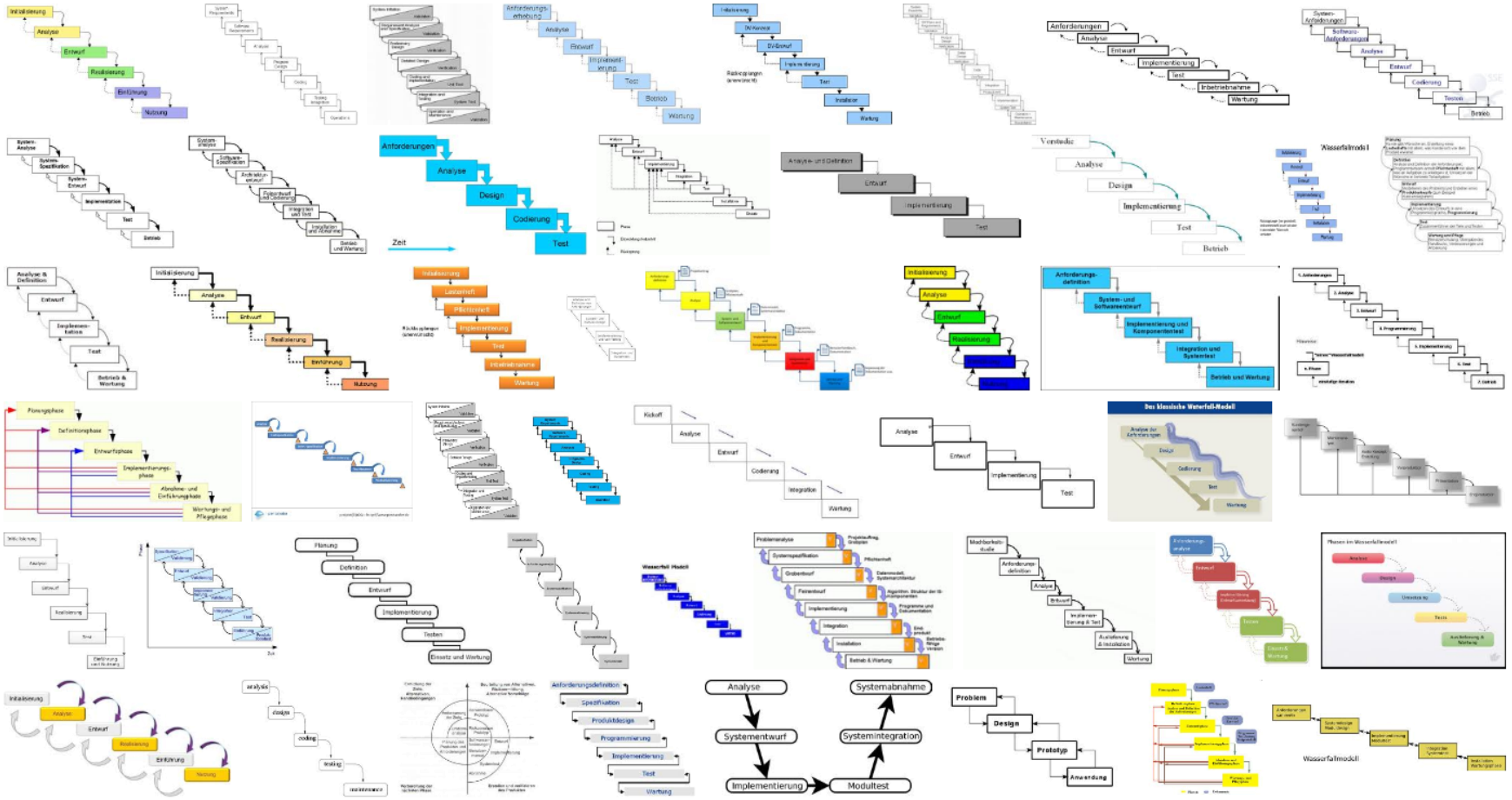


Vorgehensmodelle im Detail

Die bekanntesten Vorgehensmodelle

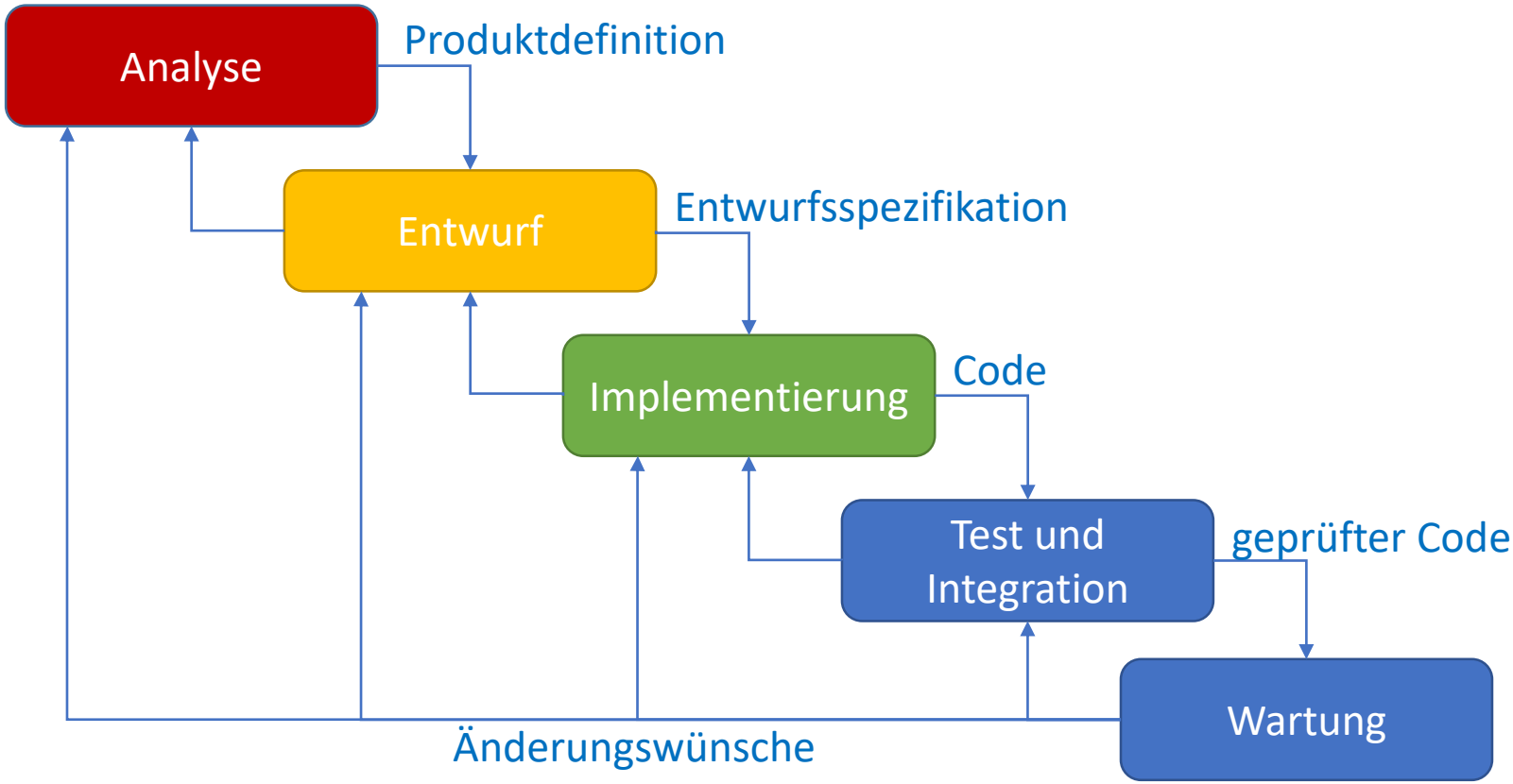
- Wasserfall-Modell
- V-Modell
- Prototypen
- Iterative/inkrementelle Modelle (Spiralmodell)
- Agile Methoden (Scrum)

Wasserfallmodell – weite Verbreitung



[Quelle: Google]

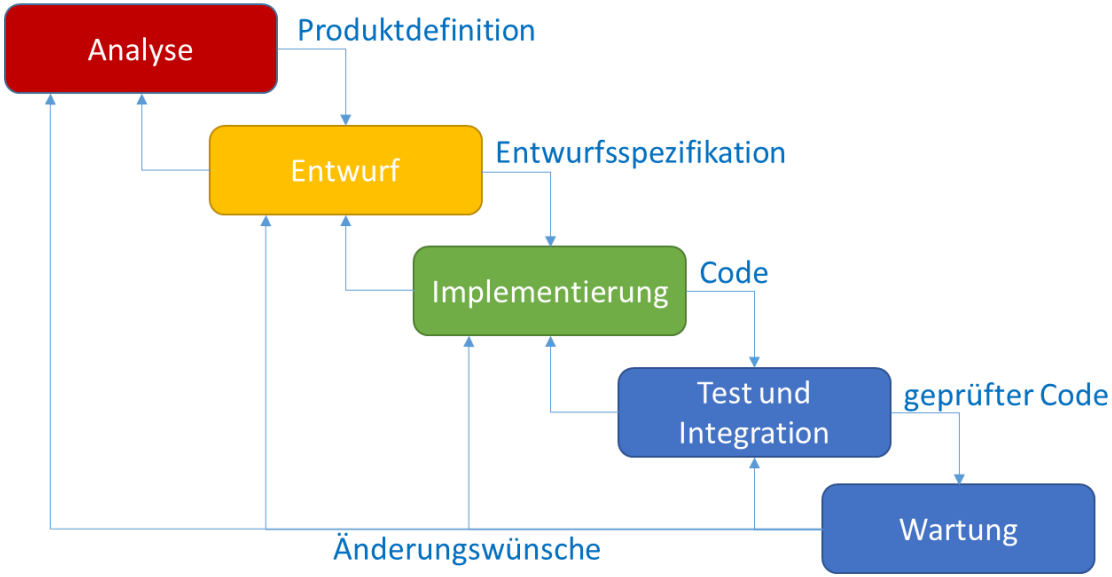
Wasserfall-Modell: Übersicht



Wasserfall-Modell nach W. Royce (1970), Erweiterung nach B. Boehm (1981)

Wasserfall-Modell: Eigenschaften

- Alle Phasen werden sequentiell durchgeführt
- Nächste Phase beginnt erst, wenn vorhergehende abgeschlossen ist
  (d.h.: Ergebnis der vorhergehenden Phase liegt vor)
- Rücksprung in vorherige Phase ist möglich (Erweiterung nach Boehm)

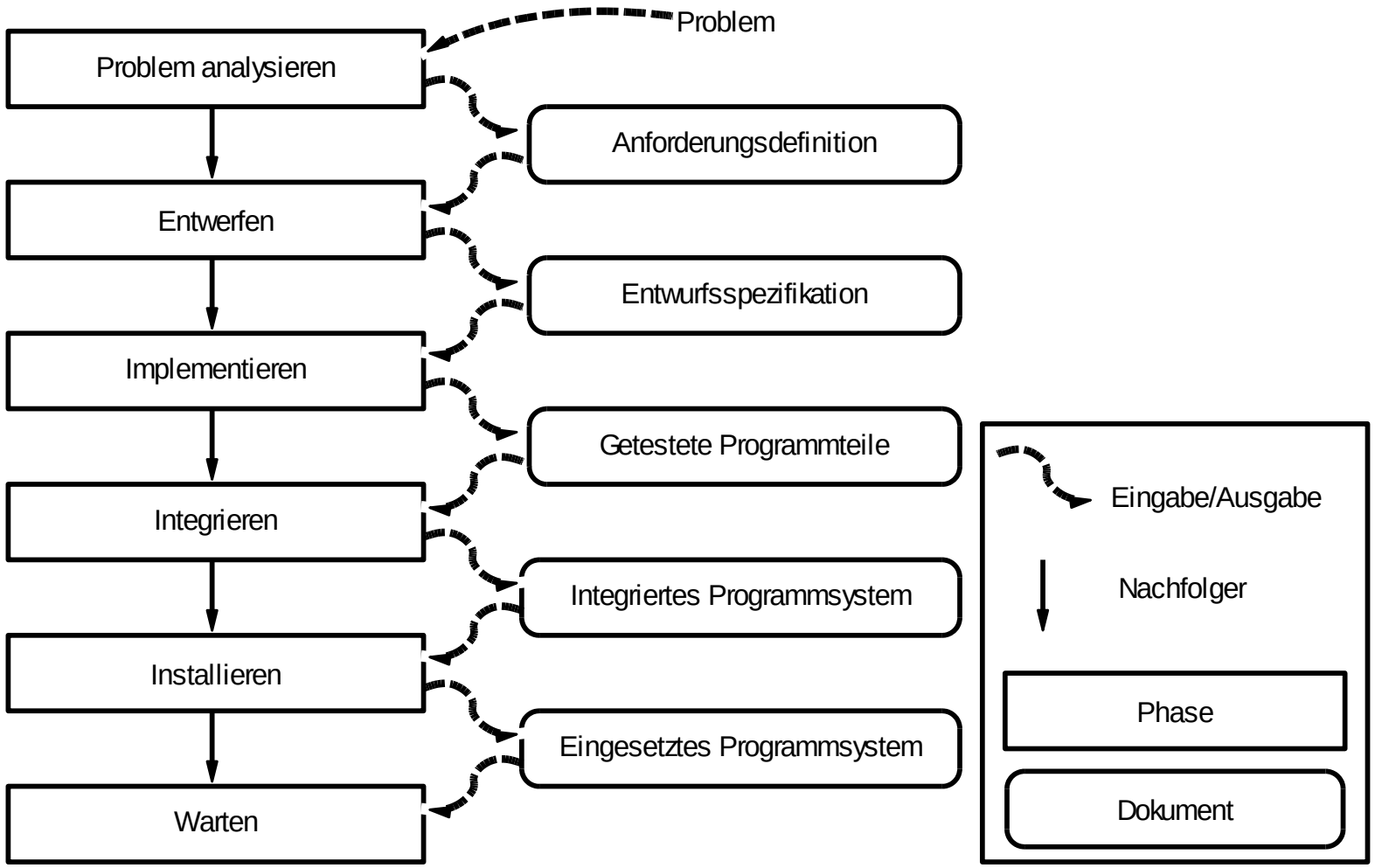


Wasserfallmodell: Diskussion

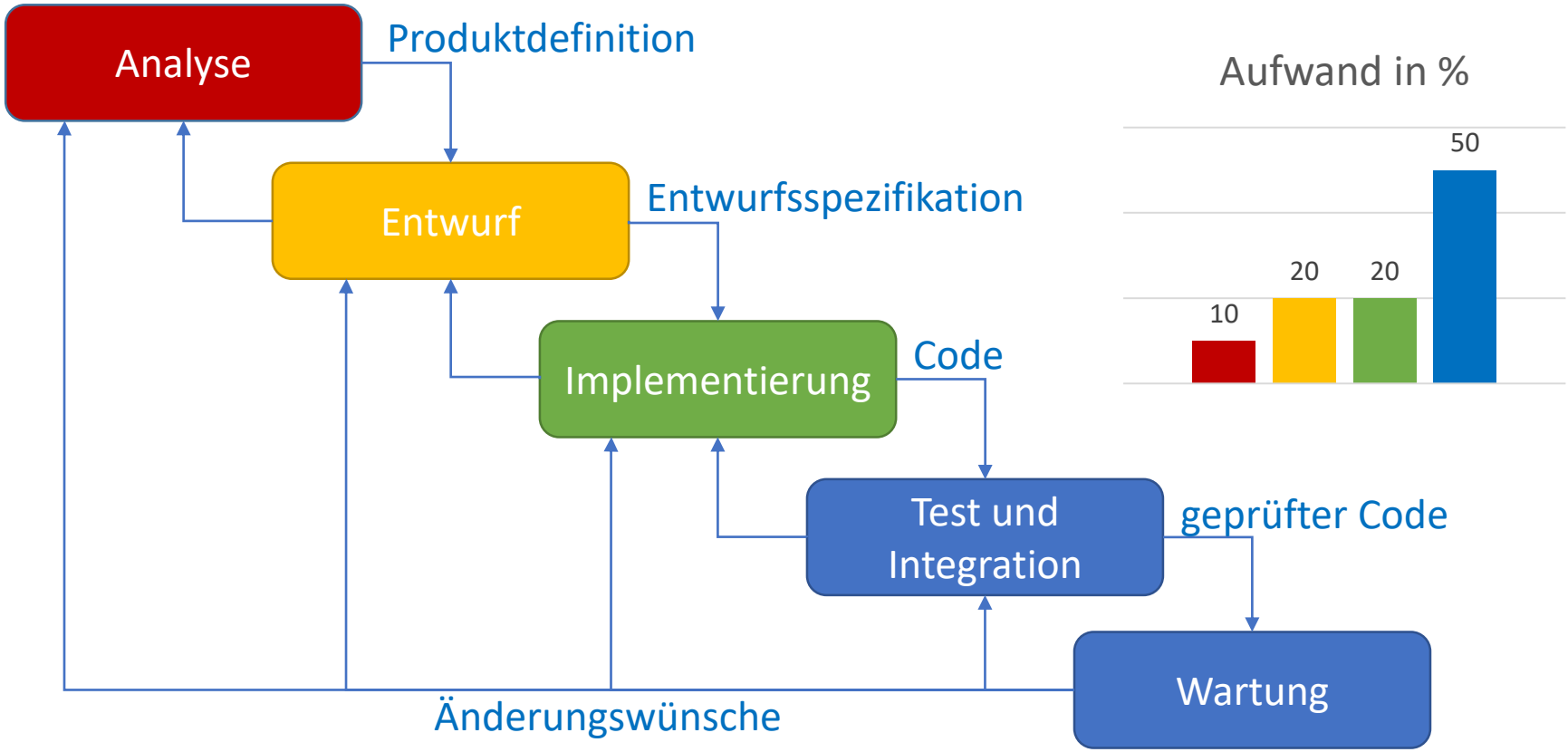
Was sind Vor- und Nachteile (Wasserfallmodell nach Royce):

- Vorteile:
  - Einfach zu verstehen
  - Einfach zu managen
  - Einfach zu überwachen (definierte Phasenübergänge)
- Nachteile:
  - Anforderungen müssen zu 100% feststehen
  - Auftraggeber nur in der ersten Phase eingebunden
  - Entwicklungsrisiken werden spät erkannt
  - Änderungen aufwendiger
  - Verzögerungen durch sequentielles Arbeiten
  - Nichteinhalten der Projektplanung führt zu Vernachlässigen später Phasen (Testen)
  - Testen nur am Ende des Entwicklungszyklus

Wasserfallmodell: Dokumentsicht



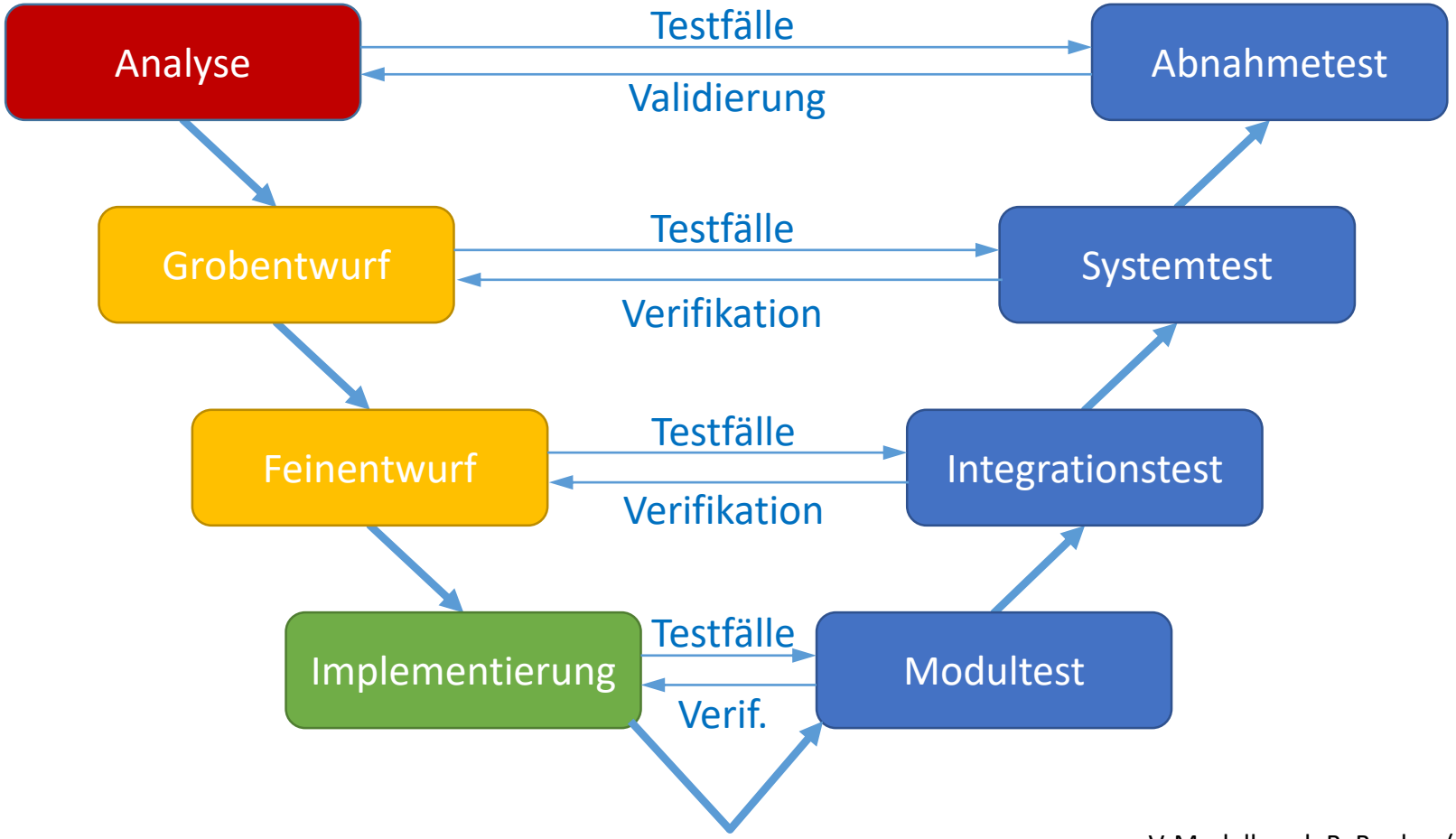
Wasserfall-Modell: Aufwandsverteilung



Die bekanntesten Vorgehensmodelle

- Wasserfall-Modell
- V-Modell
- Prototypen
- Iterative/inkrementelle Modelle (Spiralmodell)
- Agile Methoden (Scrum)

V-Modell: Übersicht



V-Modell nach B. Boehm (1979)

V-Modell: Eigenschaften

- Erweiterung des Waserfall-Modells
- Sehr umfangreiches Modell

- Vorteile:
  - Zu jeder Entwicklungsphase (links) existiert eine Qualitätssicherungsphase (rechts)
  - Qualitätssicherung also in Prozess integriert (Validierung und Verifikation)
  - Kann für konkrete Projekte angepasst werden (tailoring)
  - Organisationsneutral: setzt keine speziellen Strukturen beim Anwender voraus
- Nachteile:
  - Muss für konkrete Projekte angepasst werden (tailoring) → Mitarbeiterschulung
  - Viel Bürokratie (Dokumentation)

- Verbindliches Vorgehensmodell für Bundeswehr und Behörden
  (V-Modell XT = Extreme Tailoring)

Die bekanntesten Vorgehensmodelle

- Wasserfall-Modell
- V-Modell
- Prototypen
- Iterative/inkrementelle Modelle (Spiralmodell)
- Agile Methoden (Scrum)

Prototyping

Definition *Prototyp*:

- Vorabversion (von Teilen) des zu entwicklenden Systems

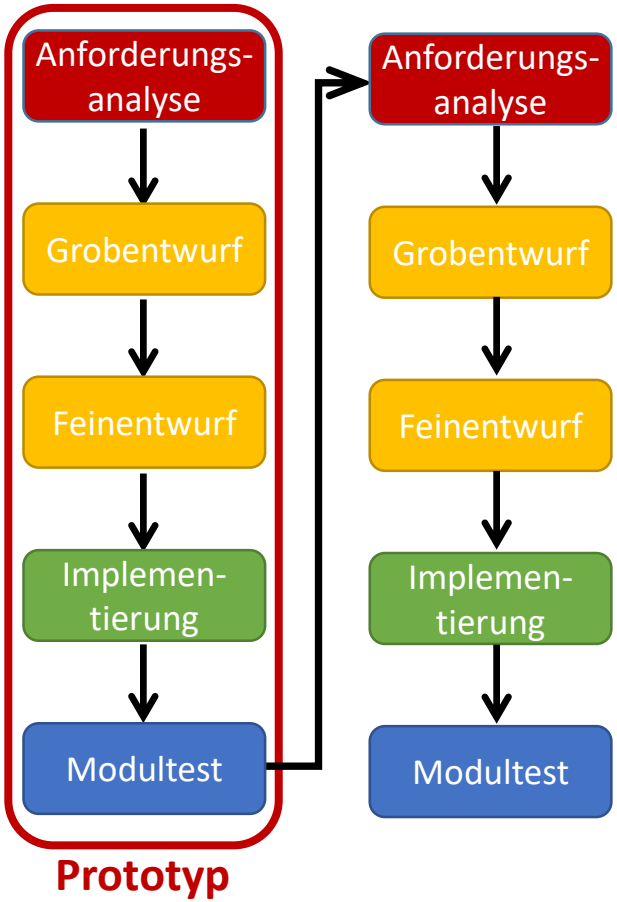
Ziel:

- Über Anforderungen klar werden
- Schnell zu ersten Ergebnissen kommen
- Frühes Feedback
- Grundsätzliche Machbarkeit relevanter Teile prüfen
- Probleme und Änderungswünsche früh erkennen
- Weniger Aufwand bei Behebung (der erkannten Probleme) als nach Fertigstellung
- Experimentieren und über das Projekt und die Requirements lernen

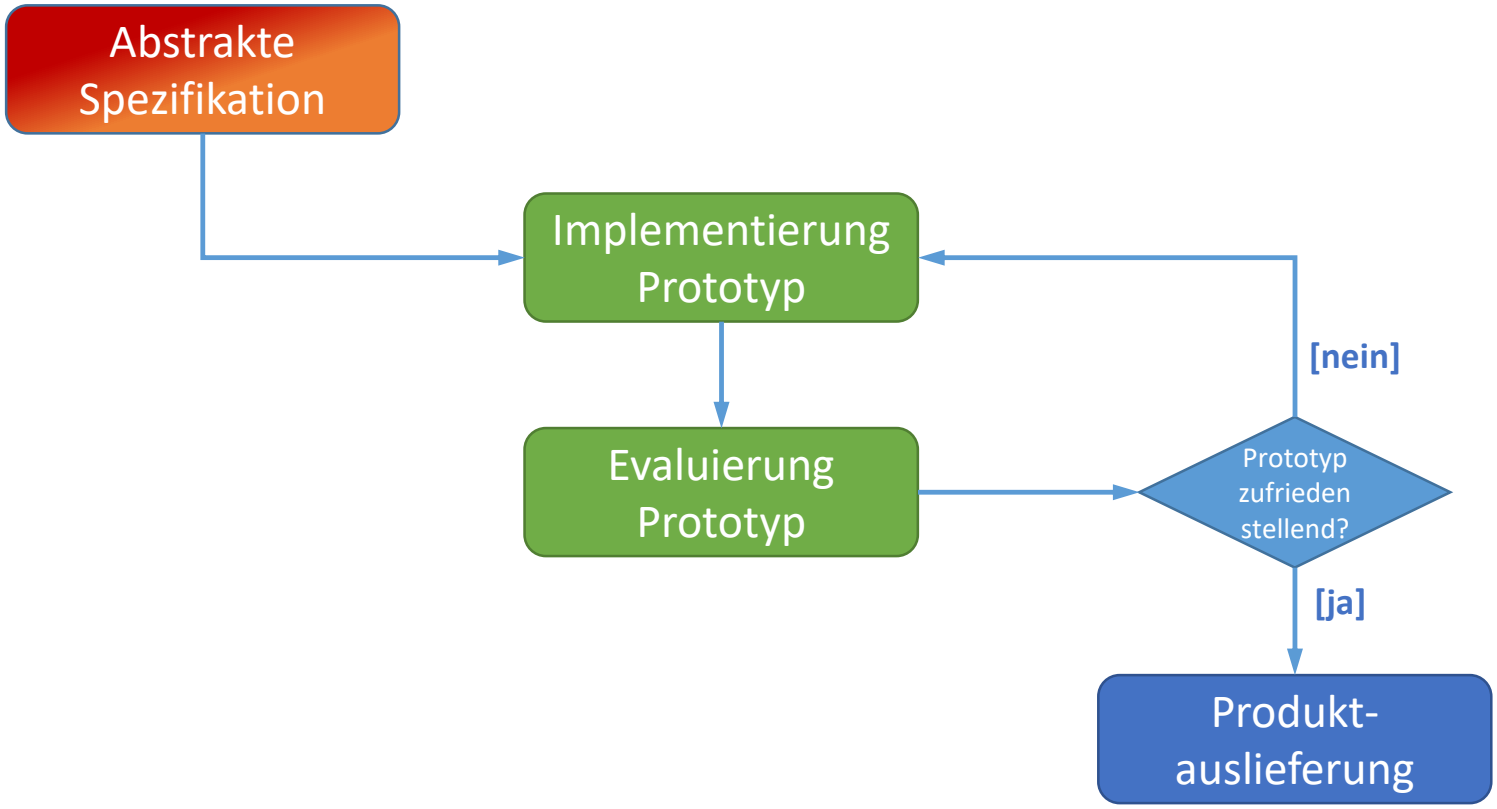
→ **Risikominimierung!!**

Arten von Prototyping (Auszug):

- Evolutionäres Prototyping
- Wegwerf-Prototyping



Evolutionäres Prototyping



Evolutionäres Prototyping

- **zeichnet sich aus durch:**
  - Prototyp wird in Iterationen immer weiter verfeinert
  - In jeder Iteration: gelerntes Wissen aus den vorhergehenden Iterationen nutzen, um aktuelle Prototypversion zu verbessern
  - Prototyp wird dabei stets lauffähig gehalten
  - Prototyp entwickelt sich schließlich zum finalen Produkt
- Vorteile:
  - Prototyp als zentrales Kommunikationsobjekt zwischen Kunde und Entwicklung
  - Erhöhte Akzeptanz beim Kunden, da er in Entwicklungsprozss integriert ist
  - Risikominimierung einer Fehlentwicklung
  - Beschleunigte Einführung
- Nachteile:
  - Schwierig Datum für Produktfertigstellung vorherzusagen
  - Anwendung in großen Projekten gefährlich, da sich Schnittstellen zu anderen Teilprojekten ändern können

Wegwerf-Prototyping (throw-away prototyping)

- **zeichnet sich aus durch:**
  - Prototyp wird erzeugt
  - Prototyp wird mit Kunden diskutiert
  - Nach Übereinkunft über seine Korrektheit weggeworfen
  - Wissen aus Erstellung und Diskussion über Prototyp fließt nun in Erstellung des finalen Systems ein
- Vorteile:
  - Lernen aus Prototyp
- Nachteile:
  - Eingang des Wegwerfprototypen in Produkt (aus Zeitmangel)
  - Zusätzlicher Entwicklungsaufwand (Kosten!)

Zusammenfassung: Prototyping

- **Entwicklung eines Prototyps nützlich, wenn:**
  - Anforderungen noch nicht hinreichend klar sind
  - Anforderungen anders nicht geklärt werden können
  - Benutzerschnittstellen
- **Erfolgsfaktoren:**
  - Direkter Kontakt Anwender ⇔ Entwickler
  - Dokumentation / Auswertung
- Vorteile von Prototyping:
  - Integration in andere Vorgehensmodelle (z.B. Erweiterung Wasserfall-Modell) mögl.
  - Reduzierung des Entwicklungsrisikos
- Nachteile von Prototyping:
  - Evtl. höherer Entwicklungsaufwand
  - Beschränkungen und Grenzen des Prototypen oft nicht bekannt
  - Anforderungen müssen nahezu 100%-ig sein
  - Prototyp (illegal) in die Entwicklung übernehmen

Die bekanntesten Vorgehensmodelle

- Wasserfall-Modell
- V-Modell
- Prototypen
- Iterative/inkrementelle Modelle (Spiralmodell)
- Agile Methoden (Scrum)

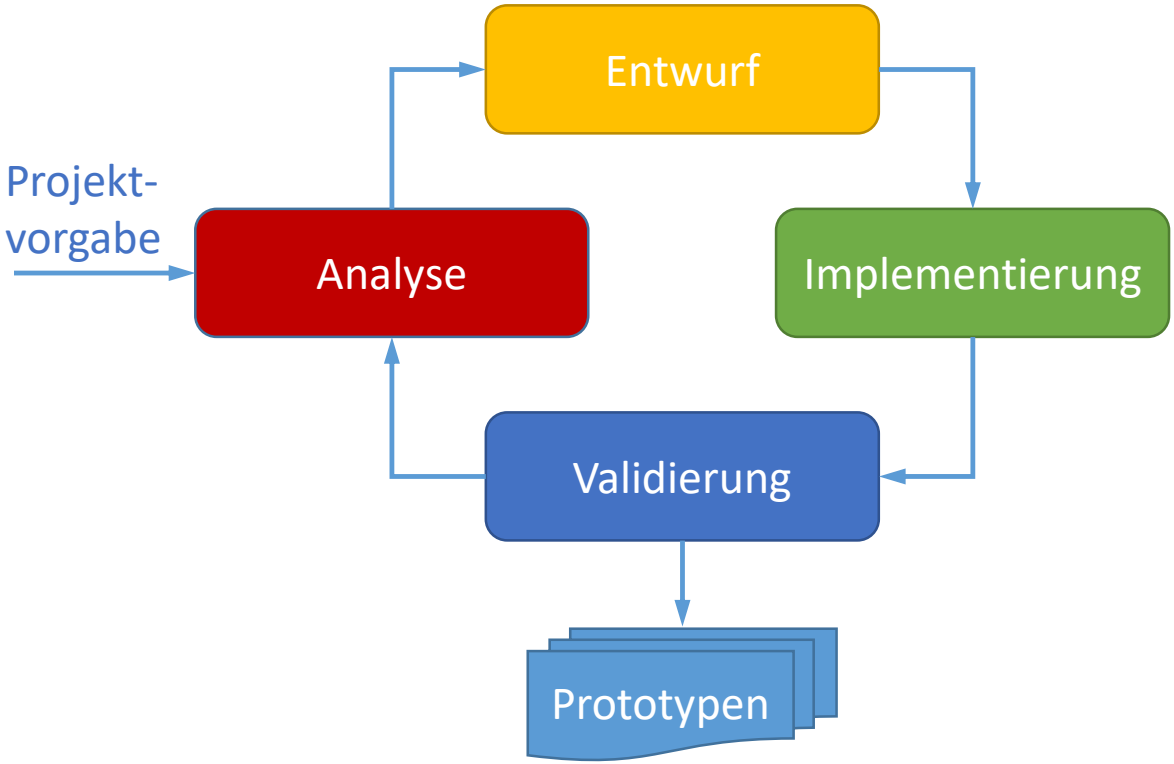
Nachteile von Wasserfall- und V-Modell

Welche Nachteile könnten die vorherigen Entwicklungsmodelle haben?

1. Alle Requirements müssen (nahezu) von Beginn an feststehen
2. Alle Testaktivitäten geschehen am Ende des Entwicklungsprozesses
3. Aus der Praxis:
   - Genannte Modelle legen viel Wert auf Dokumentation und Modelle
   - Üblicherweise werden diese aber in der Praxis während der Wartungsphase aus Zeitmangel nur unzureichend angepasst
   - Probleme entstehen, wenn auf dieser Basis dann neue Software-Version erstellt werden soll
4. Fehlende Rückkopplung zwischen Auftraggeber, Entwickler und Anwender

Iterative Modelle

- Der Entwicklungsprozess besteht aus Folge von Zyklen (Iterationen)
- Am Ende jeder Iteration: neue (ausführbare) Version des Produktes
- Neue Version verbessert bzw. erweitert Funkionsumfang der letzten



Iterative Entwicklung

Merkmale:

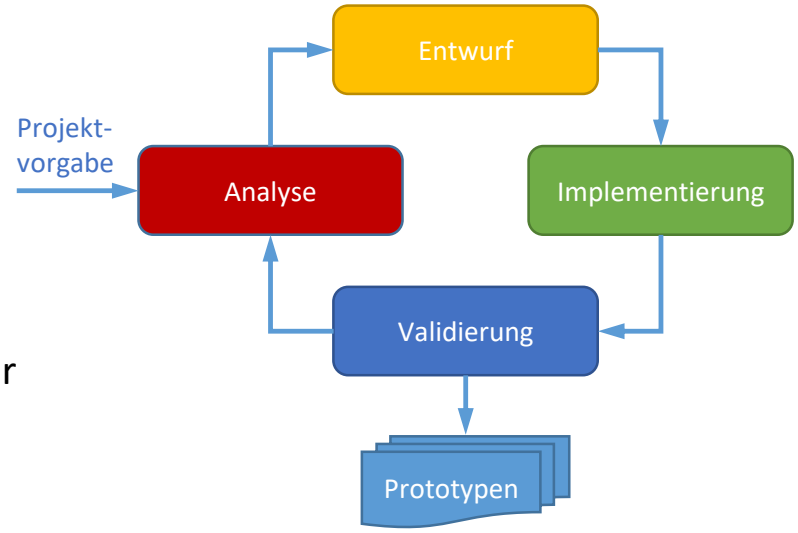
- Erweiterung der Prototypidee;
- Software wird in Iterationen entwickelt
- In jeder Iteration wird das System weiter verfeinert
- In ersten Iterationen Schwerpunkt auf Analyse und Machbarkeit; später auf Realisierung

Vorteile:

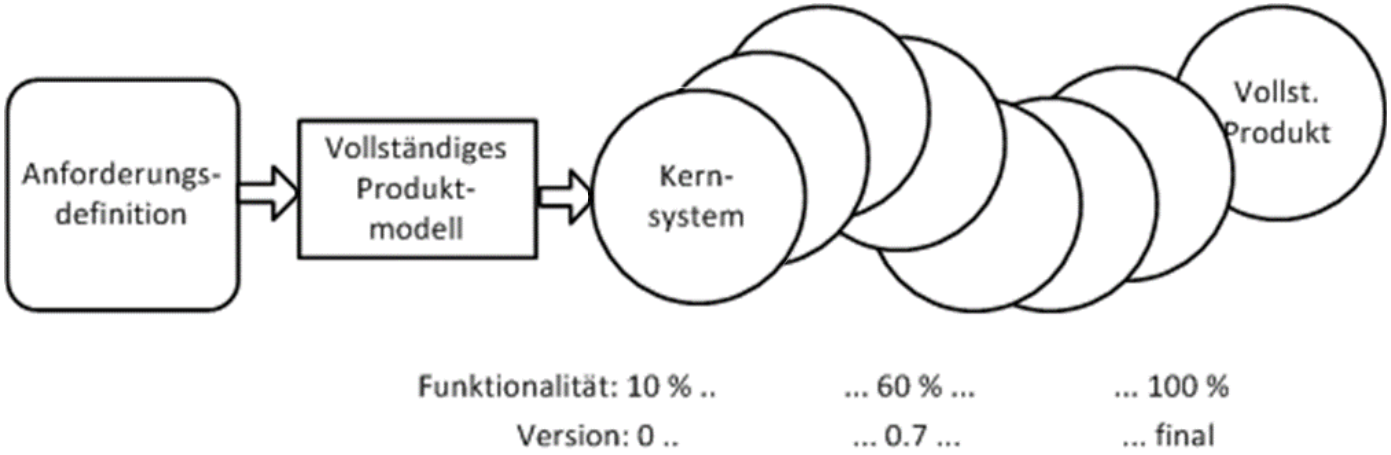
- Dynamische Reaktion auf Risiken
- Teilergebnisse mit Kunden diskutierbar

Nachteile:

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



Inkrementelle Entwicklung

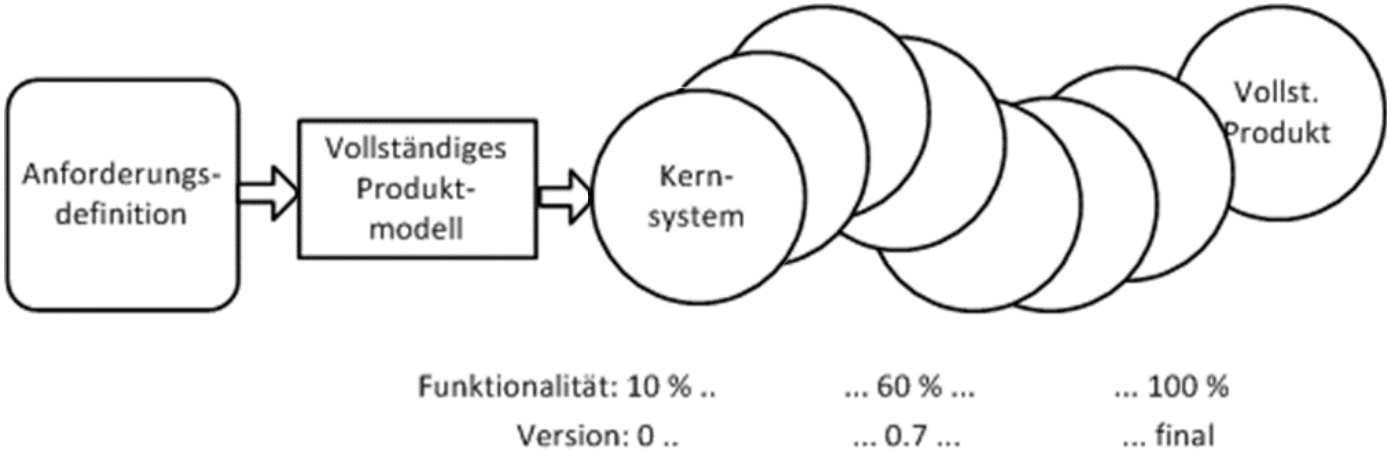


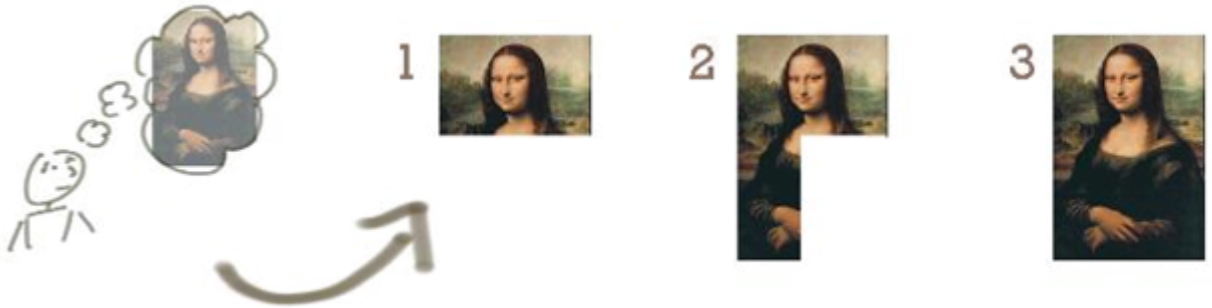
Anforderungen vollständig erfassen und modellieren

Auf Kernanforderungen des Auftraggebers konzentrieren

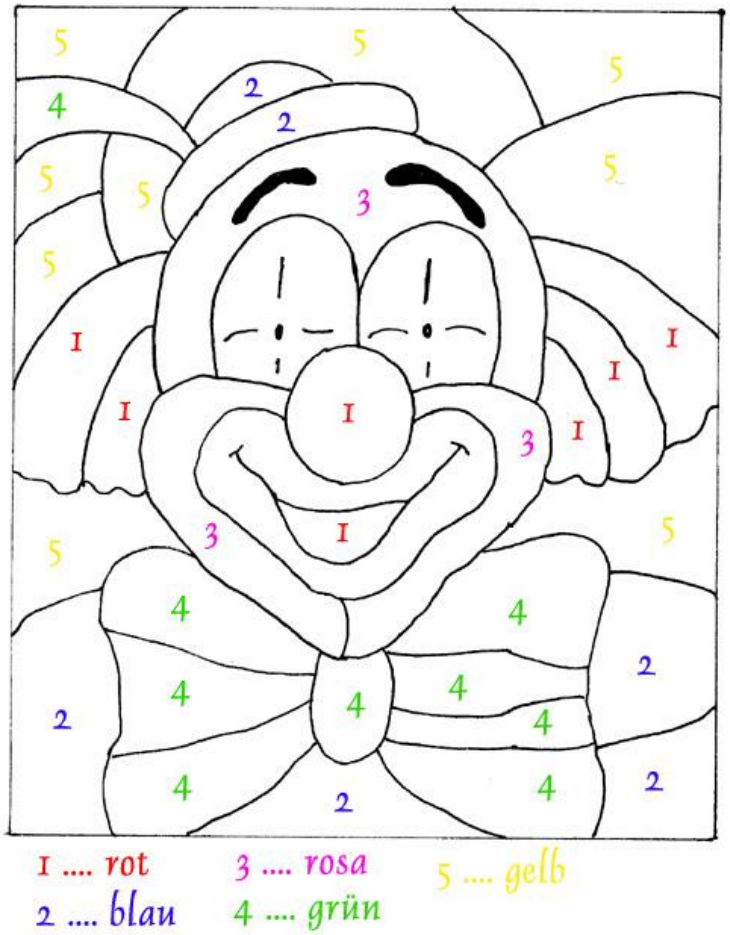
- Zunächst nur Kernsystem entwickeln (Null-Version)
- Kernsystem ausliefern
- Anwender sammeln Erfahrungen und äußern Wünsche

Inkrementelle Entwicklung





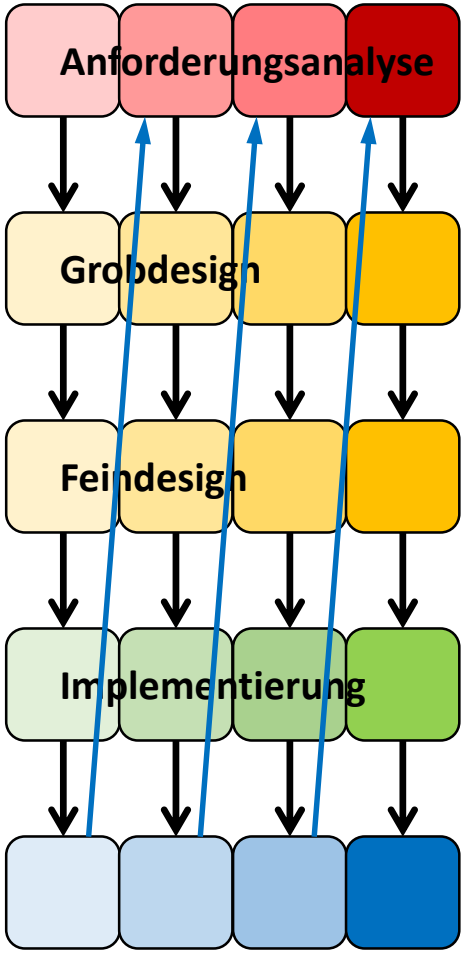
Iterativ oder Inkrementell?



Unterschiede in iterativen Modellen Inkrementell vs. Evolutionär

- Inkrementell:
  - Anforderungsanalyse und Konzeption nur zu Beginn der Entwicklung
  - Jede Iteration erzeugt weiteres Stück der Lösung
  - Schneller zu ersten Ergebnissen
  - **ABER:** anfällig bei Anforderungsänderungen
- Evolutionär:
  - Anforderungsanalyse und Konzeption in jeder Iteration
- Beide Verfahren:
  - Wichtig ist enge Zusammenarbeit von Auftraggeber und -nehmer

Iterativ-Inkrementelle Entwicklung (State of the Art)



Merkmale:

- 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

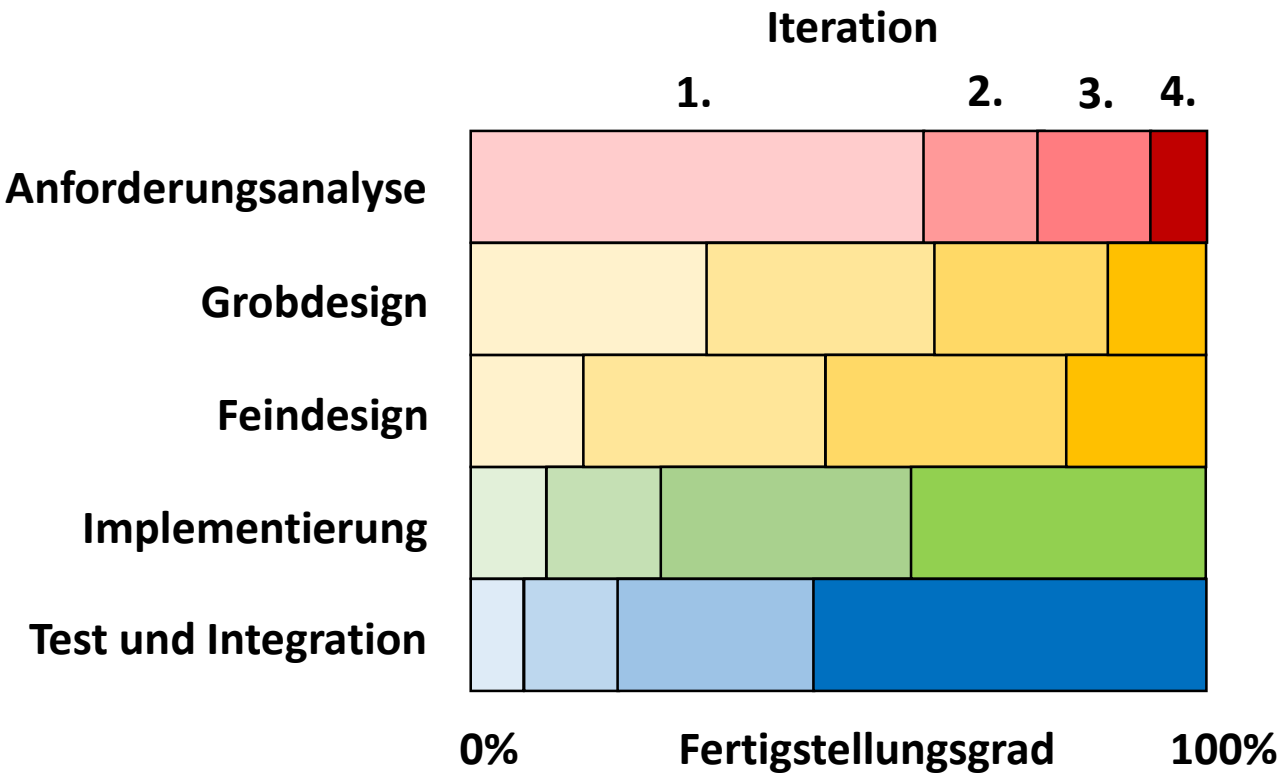
Nachteile:

- Siehe „iterativ“ (etwas verstärkt)

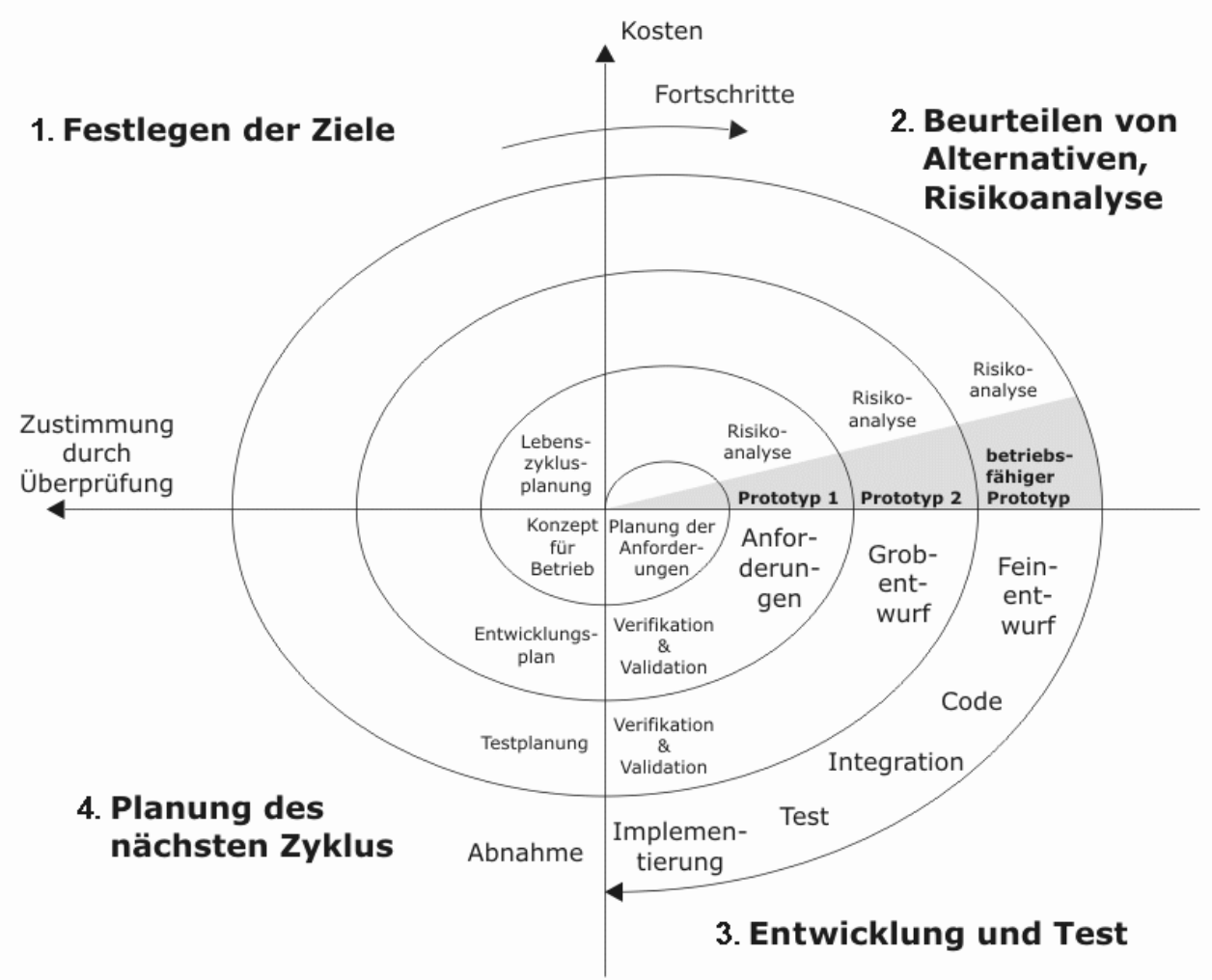
Optimierung/Anpassung:

- Anforderungsanalyse am Anfang intensiver durchführen

Fertigstellung mit Iterationen



Spiralmodell (B. Boehm, 1988)



Spiralmodell (B. Boehm, 1988)

- Eines der ersten Entwicklungsmodelle, die die Probleme von Wasserfall- und V-Modell überwanden
- Iteratives (inkrementelles) Entwicklungsmodell
- Phasen werden in einer Spirale durchlaufen
- Unterstützt iterative Entwicklung von Anforderungen und Prototypen
- Am Ende jeder Iteration: Planung der folgenden Iteration

Die bekanntesten Vorgehensmodelle

- Wasserfall-Modell
- V-Modell
- Prototypen
- Iterative/inkrementelle Modelle (Spiralmodell)
- Agile Methoden (Scrum)

Kritik an klassischen Vorgehensmodellen

Nachteile klassischer Modelle:

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

Alternativer Ansatz:

- „Menschen machen Projekte erfolgreich, traue den Menschen“

→ Agile Prozesse (urspr. Name: leichtgewichtige Prozesse)

Agile Entwicklungsmethoden

- **Agile Manifesto (K. Beck et al., 2001)**
  - Beinhaltet 12 Grundprinzipien agiler Softwareentwicklung, u.a.:
    - **Individuen und Interaktion:** wichtiger als Prozesse und Werkzeuge
    - **Funktionierende Software:** wichtiger als ausführliche Dokumentation
    - **Zusammenarbeit mit Kunden:** wichtiger als Vertragsverhandlungen
    - **Reagieren auf Veränderung:** wichtiger als Befolgen eines Plans
  - In >55 Sprachen übersetzt
- **Agile Methoden:**
  - Extreme Programming (XP)
  - SCRUM

[Quelle: http://agilemanifesto.org/]

Arten von agilen Prozessen (1)

Generelles Credo:

- Methoden lernen von einander
- Es gibt nicht die eine agile Methode

Variante 1: Konkrete Prozessbeschreibung

- Vorschlagen konkreter Verfahren für verschiedene Software-Entwicklungsphasen
- Dokumentieren der Abhängigkeiten der Verfahren (Wer “A” macht, muss auch “B” machen) → Möglichkeit zur individuellen Optimierung
- Beispiele:
  - eXtreme Programming (XP) (u.a. Kent Beck, Ward Cunningham)
  - Dynamic Systems Development Method (Konsortium)

Arten von agilen Prozessen (1)

Generelles Credo:

- Methoden lernen von einander
- Es gibt nicht die eine agile Methode

Variante 2: Beschreibung auf Metaprozessebene

- Grundregeln zur Projektorganisation
- Wenige Hinweise zur konkreten Umsetzung
- Vorgehensweisen in Projekten werden vom Team festgelegt
- Beispiele:
  - Scrum (u.a. Ken Schwaber, Jeff Sutherland)
  - Crystal Methodenfamilie (Alistair Cockburn)

Scrum als populäres Beispiel

A **scrum** (short for **scrummage**) is a method of restarting play in rugby that involves players packing closely together with their heads down and attempting to gain possession of the ball

[Quelle: Wikipedia]

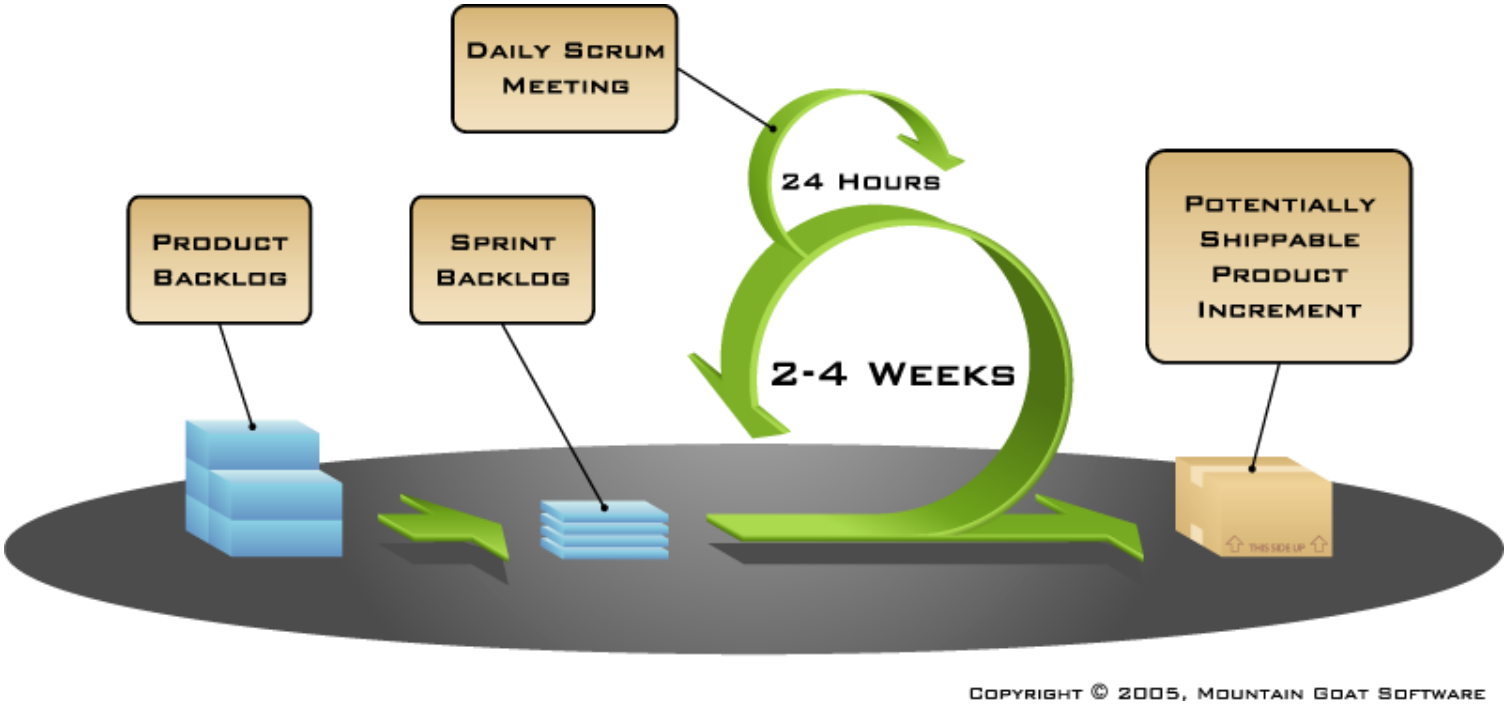
Scrum: Motivation

- 60% aller IT-Projekte sind nicht im Plan
- Falsche Annahmen:
  - Anforderungen sind klar
  - Anforderungen sind stabil
  - Entwicklungsprozess ist vorhersehbar

- Ansatz von Scrum: empirisch, inkrementell, iterativ
- Prinzipien von Scrum:
  - Zerlegung
  - Transparenz
  - Überprüfung
  - Anpassung

Scrum: Charakteristika
Einführung in Scrum

- **Idee: in kurzen Zyklen releasefähige Software ausliefern**
- Produkt schreitet in Abschnitten von monatlichen Sprints fort
- Anforderungen sind als Listeneinträge im Product-Backlog festgehalten



Scrum: Rahmen
Einführung in Scrum

Rollen

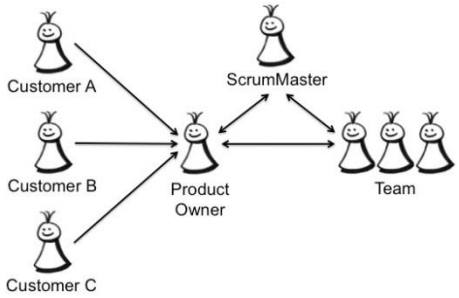
- Product Owner
- Scrum Master
- Entwicklungsteam

Meetings

- Sprint-Planung
- Sprint-Review
- Sprint-Retrospektive
- Tägliches Scrum-Meeting

Artefakte

- Vision
- Product-Backlog
- Sprint-Backlog
- Burndown-Diagramm

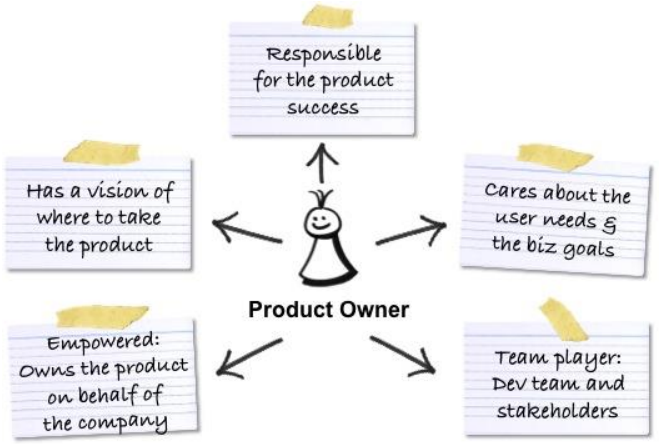


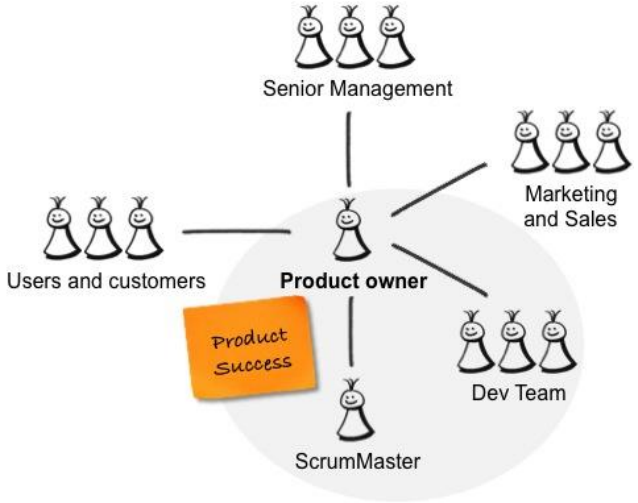
[Bildquelle: romanpichler.com]

Scrum Rollen: Product Owner

Product Owner:

- Ist verantwortlich für:
  - Die Wertmaximierung des Produkts, sowie
  - Die Arbeit des Entwicklungsteams
- Ist die einzige Person, die für das Management des Product-Backlogs verantwortlich ist



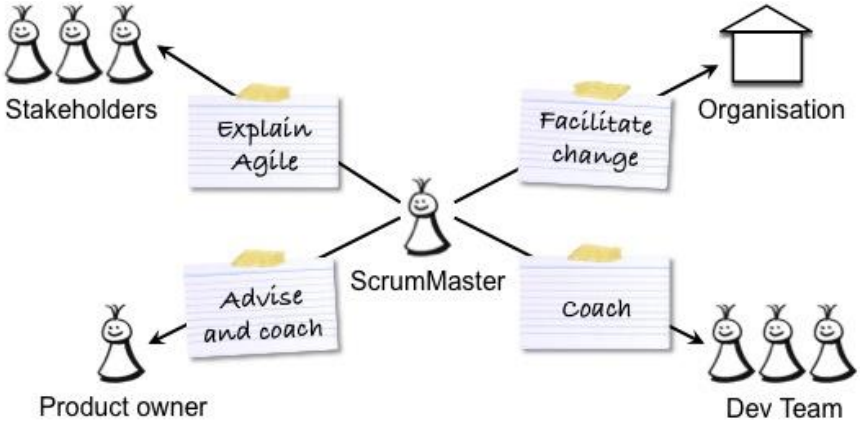


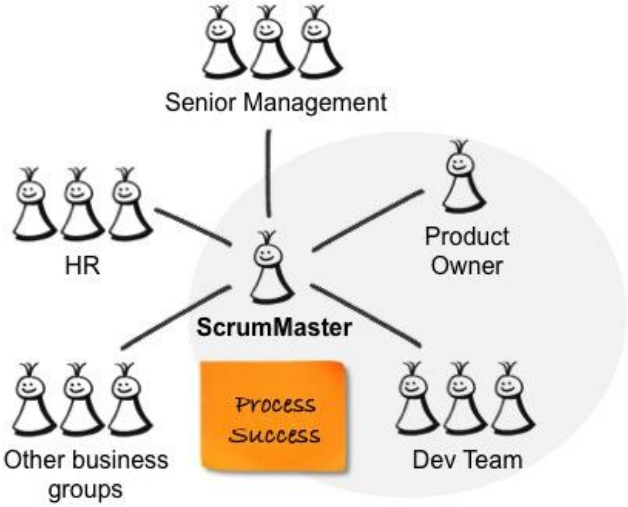
[Bildquelle: romanpichler.com]

Scrum Rollen: Scrum Master

Scrum Master:

- Für Verständnis und Durchführung von Scrum verantwortlich
- Sorgt dafür, dass Team die Theorie, Praktiken und Regeln von Scrum einhält





[Bildquelle: romanpichler.com]

Scrum Rollen: Team

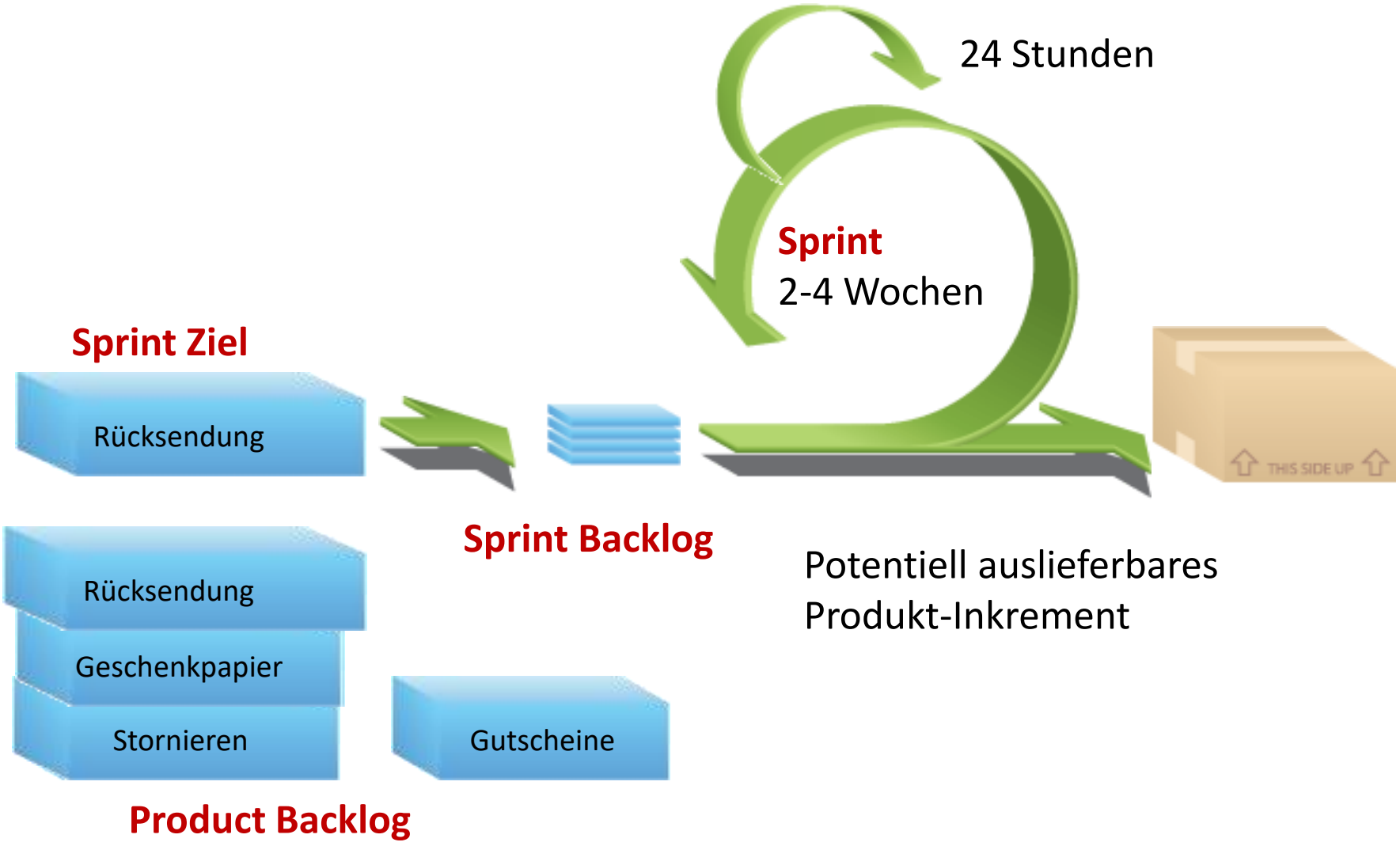
Entwicklungsteam:

- Profis, die am Ende eines jeden Sprints ein fertiges Inkrement übergeben, welches potentiell auslieferbar ist
- Interdisziplinär
- Selbstorganisierend
- Niemand (auch nicht Scrum Master) schreibt Team vor, wie es aus Product-Backlog auslieferbare Funktionalität machen soll
- Nur Mitglieder erstellen das Produkt-Inkrement

Scrum: Meetings

- Sprint Planning:
  - Was ist in Produkt-Inkrement enthalten (kommender Sprint)?
  - Wie wird für Lieferung des Produkt-Inkrements erforderliche Arbeit erreicht?
- Daily Scrum: jedes Teammitglied beantwortet drei Fragen:
  - Was habe ich gestern gemacht?
  - Was werde ich heute tun?
  - Was hindert mich bei meiner Arbeit?
- Sprint Review: (Scrum Team und Stakeholder)
  - Inkrement überprüfen und ggf. Product-Backlog anpassen
- Sprint Retrospektive:
  - Team überprüft sich selbst
  - Erstellt Verbesserungsplan für kommenden Sprint

Scrum: Beispiel
Entwicklung eines Webshops



Scrum: Die Sprints
Einführung in Scrum

- Scrum-Projekte schreiten in Serien von Sprints voran
- Typische Sprintdauer: 2 – 4 Wochen (bzw. nicht länger als ein Kalendermonat)
- Konstante Dauer führt zu einem besseren Rhythmus
- Das Produkt wird während des Sprints
  - entworfen
  - kodiert und
  - getestet

- **Keine Änderungen während des Sprints (die Sprint-Ziel gefährden könnten)**
  - Sprintdauer ist vorab festgelegt
  - Sprintdauer hängt davon ab, wie lange Veränderungen vom Sprint ferngehalten werden können

Scrum on Youtube with Jeff Sutherland

https://www.youtube.com/watch?v=oheekef7oJk

Versuch einer Beurteilung von agilen Methoden

- **Agile Methoden:**
  - Haben viele Innovationen in verstaubte Entwicklungsprozesse gebracht
  - Einsetzbarkeit stark von technischen und menschlichen Fähigkeiten abhängig
  - Große Erfolge möglich, wenn “Dream-Team” gefunden wurde
  - Agiles Manifest interessant als Ergänzung für alle SW-Entwicklungsprozesse

- **Generell:** Das gewählte Vorgehensmodell muss zum Projekt und den Menschen im Projekt passen

Diskussion:
Wann sind welche Modelle angemessen?

Strikte, stark planende Modelle:

- Wenn wohldefiniertes Resultat in definierter Zeit erreicht werden muss
- Wenn sehr große (insbesondere verteilte) Projektgruppen koordiniert werden müssen

→ Sind die Pläne und Dokumente zur Koordination unverzichtbar

Agile Modelle:

- Wenn Projekte komplex sind
- Hohe Unsicherheit über die Anforderungen besteht
- Hohe Unsicherheit über technische Realisierung

Zusammenfassung: Vorgehensmodelle (1)

Hauptunterschied zwischen Vorgehensmodellen: Wie viel Planung?

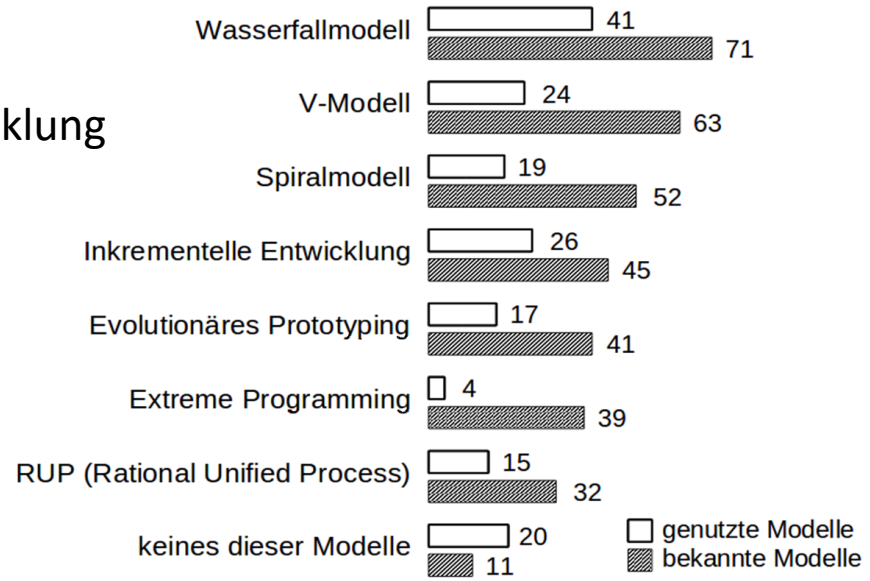
- Wichtigste Dimension zur Unterscheidung: wie präzise/strikt/weit voraus wird geplant?

- Viel: Wasserfallmodell
  → Möglichst präzise und strikt, für das gesamte Projekt im Voraus
- Mittel: Iterative Modelle
  → Präzise, wo möglich
  → Nicht strikt (nötige Veränderungen werden akzeptiert)
  → Nur für wenige Iterationen im Voraus
- Wenig: Agile Prozesse
  → Nur so viel Planung wie unbedingt nötig
  → Lieber Ziele als Pläne (→ um flexibel zu bleiben)

Zusammenfassung: Vorgehensmodelle (2)

Wir haben kennengelernt:

- Wasserfall-Modell
- V-Modell
- Prototyping
- Iterative/Inkrementelle Softwareentwicklung
  - Spiralmodell
- Agile Softwareentwicklung
  - Scrum



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

Literatur

- I. Sommerville: Software Engineering, 9. Auflage, Pearson, 2012
- H. Balzert, Lehrbuch der Softwaretechnik, 3. Auflage, Spektrum Akademischer Verlag, 2009
- K. Beck: Extreme Programming Explained, 2. Auflage, Addison Wesley, 2004

- Kolumne “Agile Entwicklung”: http://www.coldewey.com/publikationen/Kolumne.html