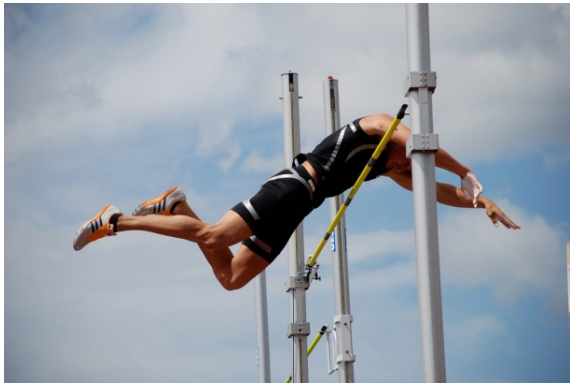
Prozesse

- Einführung
- Prozess-Paradigmen
  - Sequenziell
  - Iterativ
  - Notation
- Scrum
- Weitere Prozesse im Vergleich

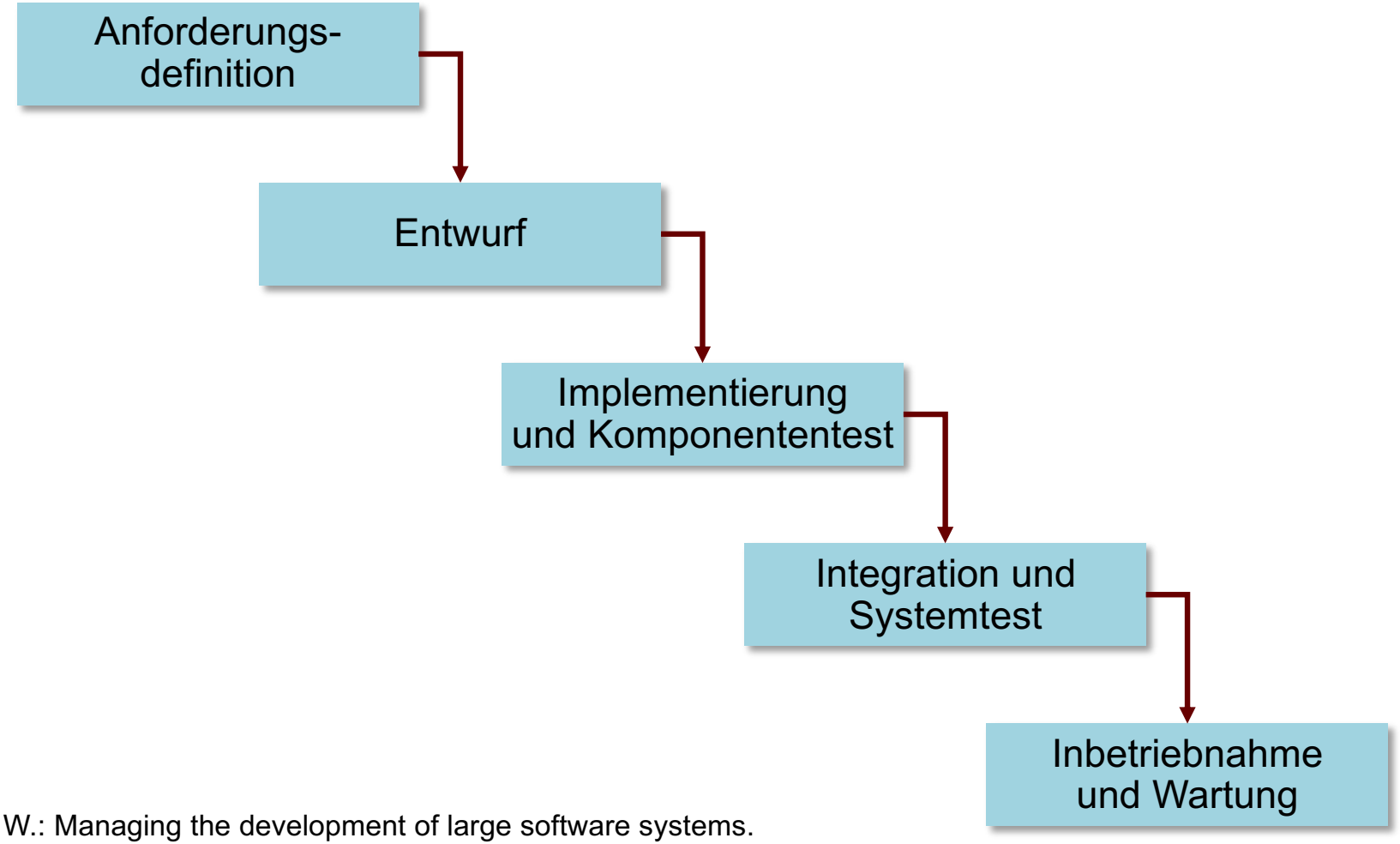
Ziele

- Sie kennen grundlegende Ansätze von Software-Entwicklungsprozessen
- Sie kennen das Basisvokabular für die Beschreibung von Prozessen



- Wir nutzen diese Kenntnisse, um Entwicklungstechniken in ihren Kontext (je nach Prozess) einzuordnen
  - Programmieren, PP, TDD, Refactoring
  - (Fein-)Design, Architektur-Entwurf
  - Testen
  - Inspizieren
  - ...

Ein einfacher Software-Prozess



Royce, W. W.: Managing the development of large software systems.
IEEE Westcon, Los Angeles, 1970

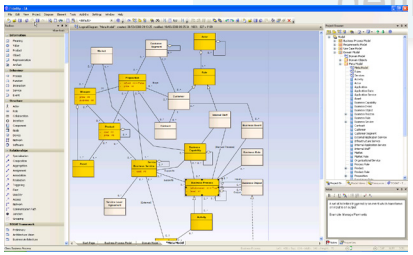
Ein Software-Prozess definiert...

Wer?	Verantwortung, Durchführung
Was?	Aufgabe, Ergebnis
Warum?	Begründung, Ziel
Wann?	Zeitpunkt, Reihenfolge
Wie?	Art und Weise, Methoden
Womit?	Arbeitsmittel, Werkzeuge
Wonach?	Standard, Normen
Wofür?	Zielgruppe, Anwendungsdomäne

Ein einfacher Software-Prozess

Software für Kontrollzentrum

Anforderungs-definition



Enterprise Architect

Architekt

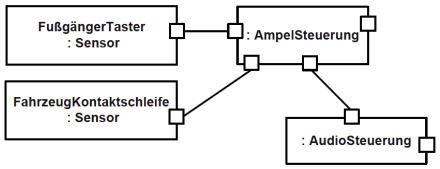
SW-Designer

Tester

Entwurf

ISO/IEC 12207

ECSS-E-40



OOA/OOM

Implementierung und Komponententest

Aerospace

Integration und Systemtest

Inbetriebnahme und Wartung

HOCHSCHULE DER WIRTSCHAFT FÜR MANAGEMENT
UNIVERSITY OF APPLIED MANAGEMENT STUDIES

Aufgaben des SE-Prozesses 1/2

- **Anforderungsdefinition** — Specification
  - Verstehen der Wünsche des Kunden
  - Definition der Funktionen, Randbedingungen (Kundensicht ←→ Entwicklersicht)
- **Entwurf** — Design and Implementation
  - Entwicklung der Architektur
  - Definition der Komponenten
- **Implementierung und Komponententest**
  - Entwicklung / Test der Komponenten
- **Integration und Systemtest** — Validation
  - Sicherstellen, dass die Software tut, was der Kunde will

Aufgaben des SE-Prozesses 2/2

- Inbetriebnahme und Wartung — Maintenance
  - Installation / Schulung beim Kunden
  - Wartung der Software
    - Weiterentwicklung
      - wegen neuen Kundenwünschen oder
      - geänderten Randbedingungen
    - Fehlerbehebung

Bei der SW-Entwicklung sind *immer alle* Aufgaben *erforderlich*
… die Prozesse unterscheiden sich lediglich in der Reihenfolge

Nutzen von Prozessmodellen 1/2

- Unterstützung bei Kommunikation
  - „Software wird in Teams entwickelt“:
    Strukturierung des Ablaufs
  - Leichtere Einarbeitung von neuen Mitarbeitern
- Verteilung der Arbeit
  - Koordinierte Arbeit an großen Aufgaben
- „Standard“-Aufgaben
  - Standard-Vorgehen bei immer wiederkehrenden Aufgaben
  - Hilfestellung bei täglicher Arbeit
  - Verständnis der Arbeit(sergebnisse) Anderer

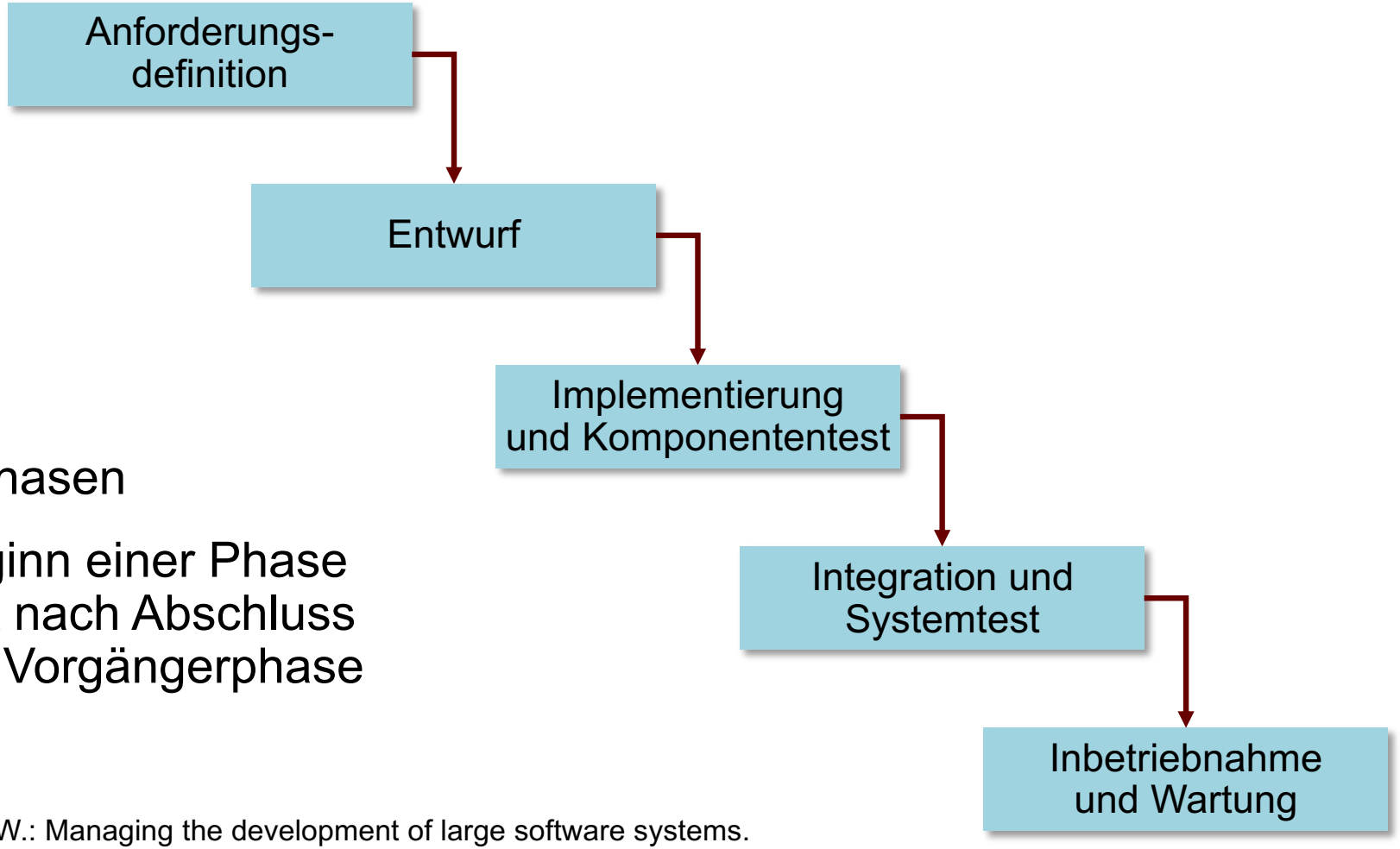
Nutzen von Prozessmodellen 2/2

- Erfahrungsaufbereitung
  - Lernen aus Erfahrung
- Projektkontrolle, -überwachung
  - Wissen über Aufgaben und Aufwand
    = Grundlage für Planung und Kontrolle
- Unterstützung durch Software-Entwicklungsumgebungen
  - „Software wird Tool-unterstützt erstellt“
  - Unterstützen von Standard-Aufgaben mit (Standard-)Tools

Inhalt

1. Einführung
2. Prozessparadigmen
   - Sequenziell
   - Iterative Prozesse
   - Notation
3. Scrum
4. Weitere Prozesse im Vergleich

Wasserfallmodell

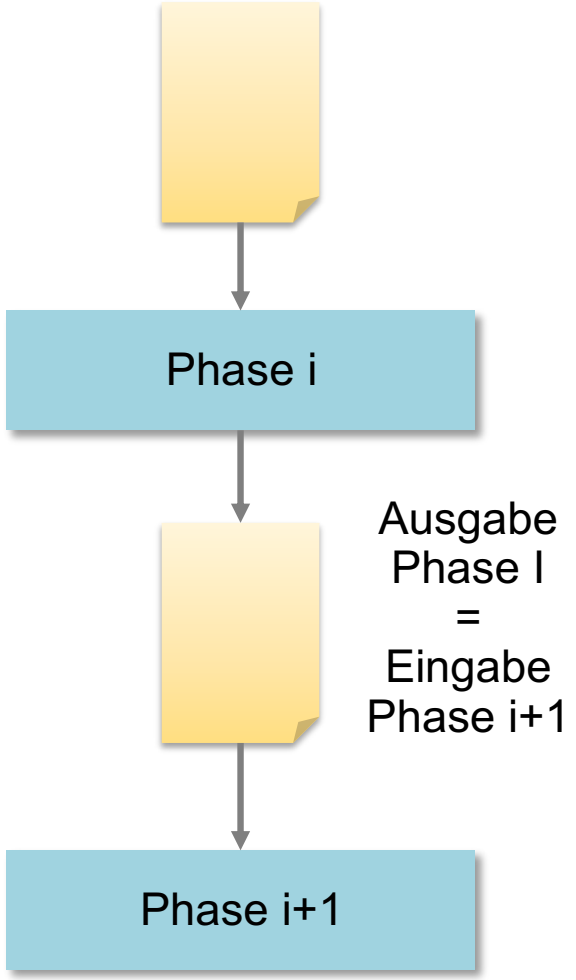


- 5 Phasen
- Beginn einer Phase erst nach Abschluss der Vorgängerphase

Royce, W. W.: Managing the development of large software systems. IEEE Westcon, Los Angeles, 1970

Dokumentation

- Eingaben
  - Dokumente, die in einer Phase verwendet werden
- Ausgaben
  - Dokumente, die in einer Phase entwickelt werden
- Beim Wasserfall
  - Eingabe in Phase 1 (Anforderungsdefinition)
    - Kundenwünsche
  - Ausgabe aus Phase i = Eingabe in Phase i+1



Ausgabe der Anforderungsdefinition

- Zweck
  - Festlegen der Anforderungen
  - Gemeinsames Verständnis von Kunde (Auftraggeber) und Entwicklungsteam (Auftragnehmer)
- Inhalt
  - Was soll das zu entwickelnde System können?
    Wie soll es das durchführen?
- Beispiele
  - “Das System muss ein Produkt mit Bild, Details und Preis anzeigen (Angaben wie im bisherigen Papierkatalog)”
    “Das System muss für einen angemeldeten Käufer einen Warenkorb führen”
    “Das System muss 24/7 zur Verfügung stehen: pro Woche sind max. 30 Min. Ausfallzeit (für Wartung, Fehlerbehebung etc.) erlaubt.”

Ausgabe des Entwurfs

- Zweck
  - Festlegen der Software-Architektur und -Struktur
  - Gemeinsames Verständnis des Progammaufbaus durch die (möglicherweise global verteilte) Entwicklungsmannschaft
- Inhalt
  - Wie ist das zu entwickelnde System strukturiert?
  - Welche Schnittstellen gibt es nach außen und innerhalb?
- Beispiel

Ausgabe von Implementierung & Komponententest

- Zweck
  - Bereitstellen einer Menge von funktionsfähigen Komponenten
- Inhalt
  - Dokumentierter Code
  - Getestete, ausführbare Komponenten
- Beispiel
  - Bibliotheken
    - a.dll, b.a, c.dylib, d.jar
  - Skripte
    - A.sh, b.pl, c.py

Ausgabe von Integration und Systemtest

- Zweck
  - Bereitstellen eines beim Kunden lauffähigen Systems
- Inhalt
  - Ausführbares System
  - Installations- und Benutzungsanleitung
- Beispiel
  - Installierbare Anwendung in Form
    - eines Installationsprogrammes (a.exe)
    - eines Softwarepakets für einen Paketmanager (b.deb, c.rpm)

Ausgabe von Inbetriebnahme und Wartung

- Keine Ausgabe…
  - ... am Ende der Wartungsphase wird die Software ausgemustert

Diskussion

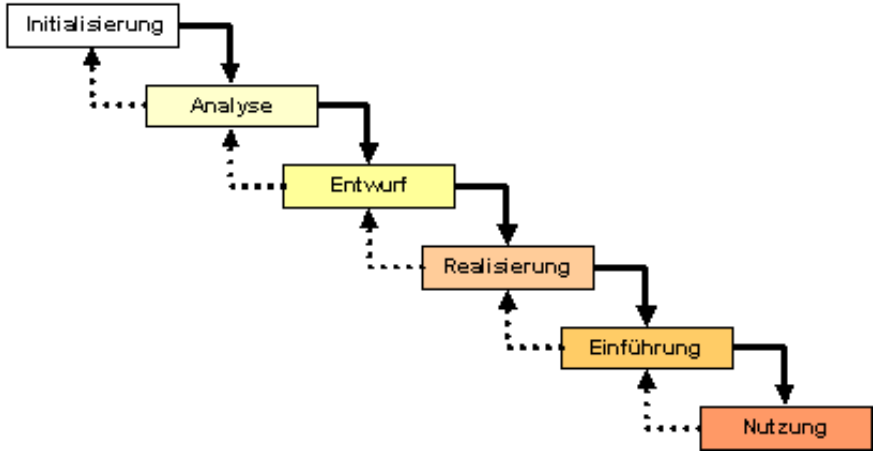
- Nennen Sie jeweils drei Vorteile und drei Nachteile des Wasserfallmodells

Arto Teräs
http://ajt.iki.fi/travel/debconf5/page2.html

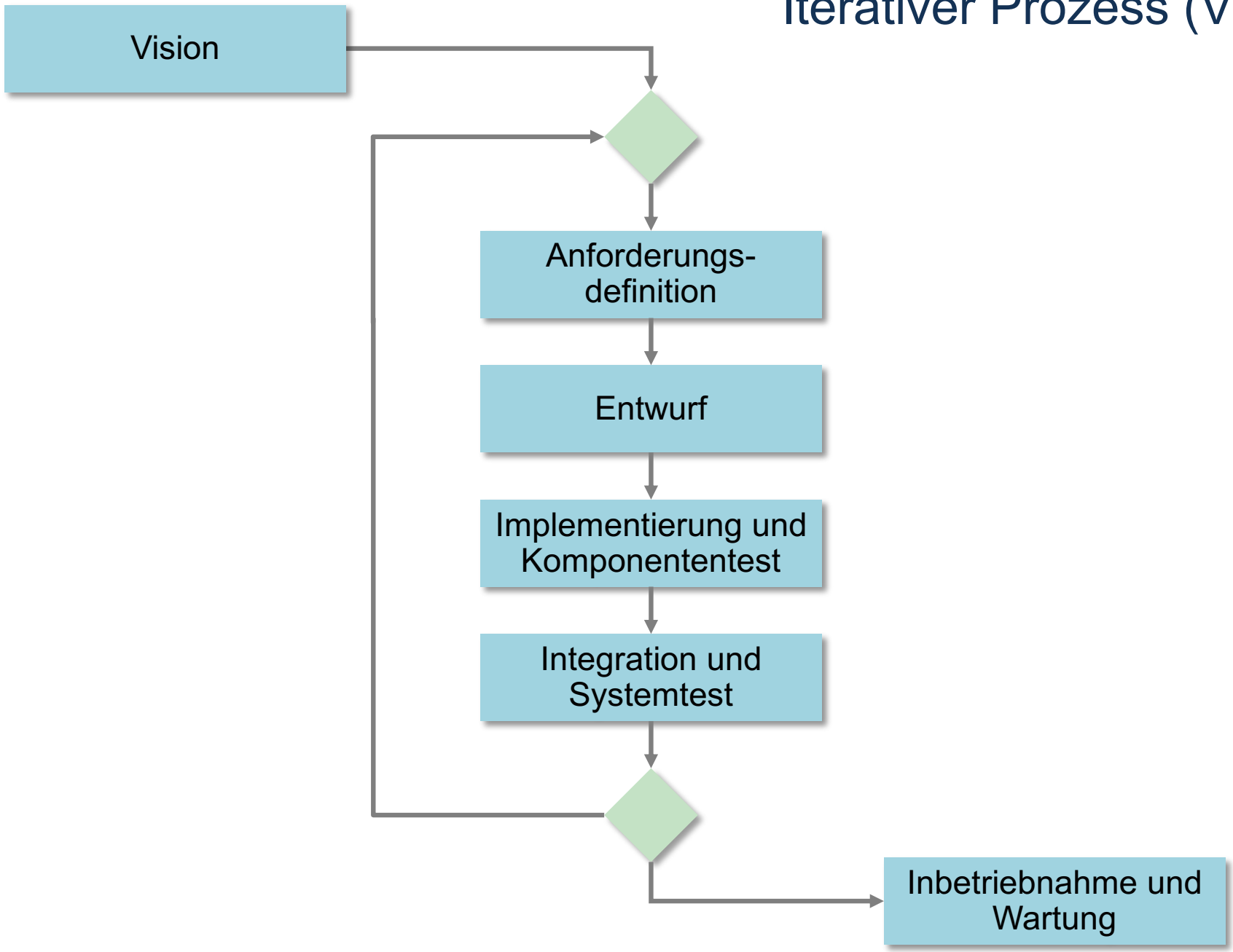
Anwendung des Wasserfallmodells

- bei gut verstandenen („klaren“) Anforderungen
- falls keine/wenig Änderungen der Anforderungen erwartet werden

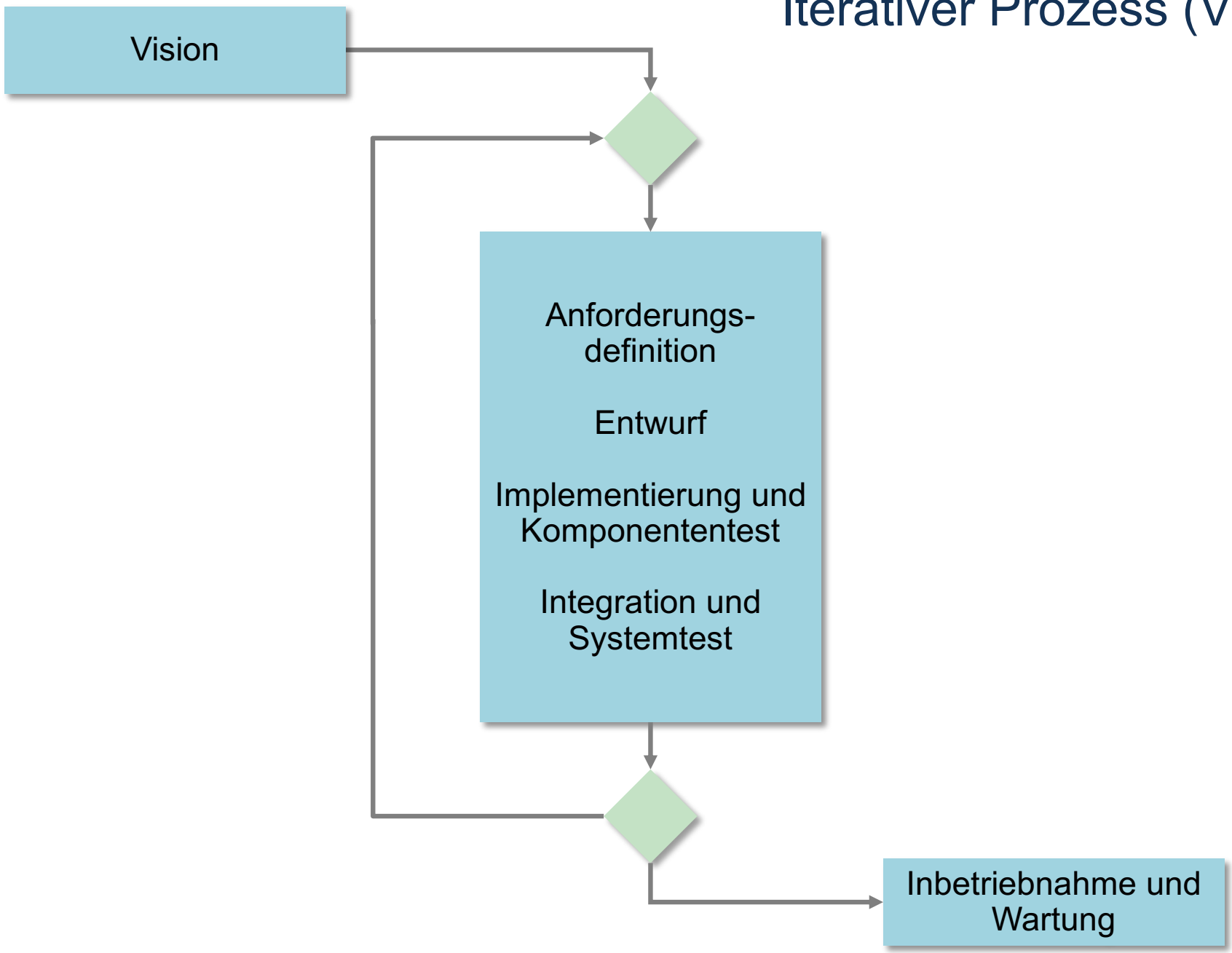
- Varianten
  - z.B. „mit Rücksprung“



Iterativer Prozess (V1)



Iterativer Prozess (V2)



Diskussion

- Nennen Sie jeweils drei Vorteile und drei Nachteile des iterativen Prozessmodells

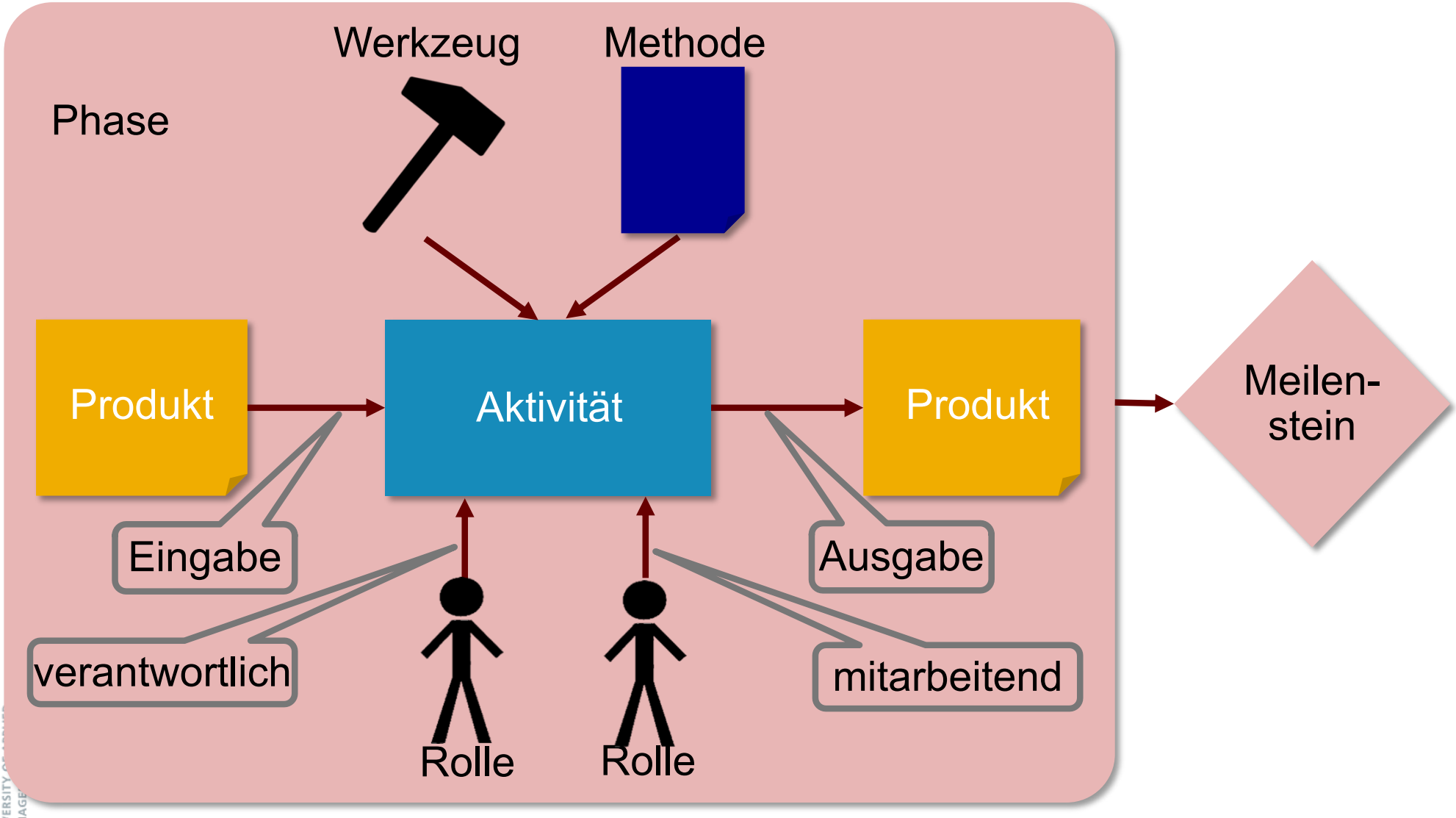
Arto Teräs
http://ajt.iki.fi/travel/debconf5/page2.html

Anwendung iterativer Modelle

- Anwendung
  - Projekte mit offenen Fragen (Technologie, Domäne, …)
  - Projekte mit unklaren / sich ändernden Anforderungen
  - Komplexe (bzgl. Technik, Domäne, …) Projekte

...also bei den meisten Projekten

Rollen, Werkzeuge, Produkte, Phasen, Meilenstein



Produkt

- Ein *Produkt (Artefakt)* ist (niedergelegte) Information, die erstellt, modifiziert oder verwendet wird

- Endprodukt
  - Wird an den Nutzer / Kunden geliefert
- Zwischenprodukt
  - Wird für andere Entwicklungsschritte verwendet

- Bsp: Entwurfspezifikation, Klassendiagramm, Code

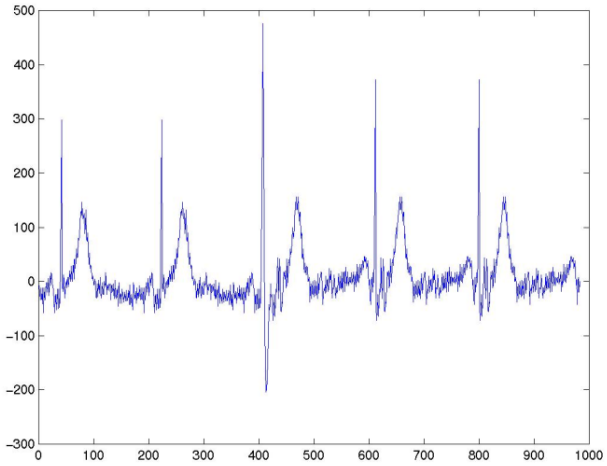
Rolle

- Eine *Rolle* fasst Fähigkeiten / Verantwortlichkeiten zusammen
- Zur Durchführung einer Aktivität sind i.d.R. bestimmte Fähigkeiten erforderlich, d.h. eine bestimmte Rolle kann diese Tätigkeiten durchführen
- Eine (jede) Person, die diese Fähigkeiten hat, kann eine Rolle wahrnehmen
- Eine Rolle kann von mehreren Personen ausgeübt werden
- Eine Person kann mehrere Rollen innehaben
- Bsp: Entwickler, Tester, Konfigurator

Aktivitäten

- Eine *Aktivität* ist ein Arbeitsschritt zum Erreichen eines Zieles

- Typischerweise
  - werden Produkte verwendet
  - werden Produkte erzeugt / geändert
  - führen Rollen den Schritt aus
  - werden Methoden eingesetzt



- Hierarchische Struktur möglich
  - (große) Aktivität besteht aus (kleineren) Aktivitäten

- Bsp: Entwerfen der Klassen, Reviewen der Klassen

Werkzeuge / Methoden / Phase / Meilenstein

- Werkzeuge
  - Bsp.: Eclipse, JUnit, Git
- Methoden
  - Bsp.
    - Strukturiertes Interview (für Anforderungsdefinition)
    - OOD (Objektorientiertes Design -> Entwurf)
    - Blackbox-Test (für Integration und Systemtest)
- Phase
  - Gruppierung von Aktivitäten
- Meilenstein
  - Überprüfbares (Zwischen-)Ergebnis
  - Bsp.: “Anforderungsdokument vom Kunden unterzeichnet”

Inhalt

1. Einführung
2. Prozessparadigmen
3. Scrum
4. Weitere Prozesse im Vergleich

Scrum

Scrum und Rugby

„Both are adaptive, quick, self-organizing, and have few rests“

Ken Schwaber

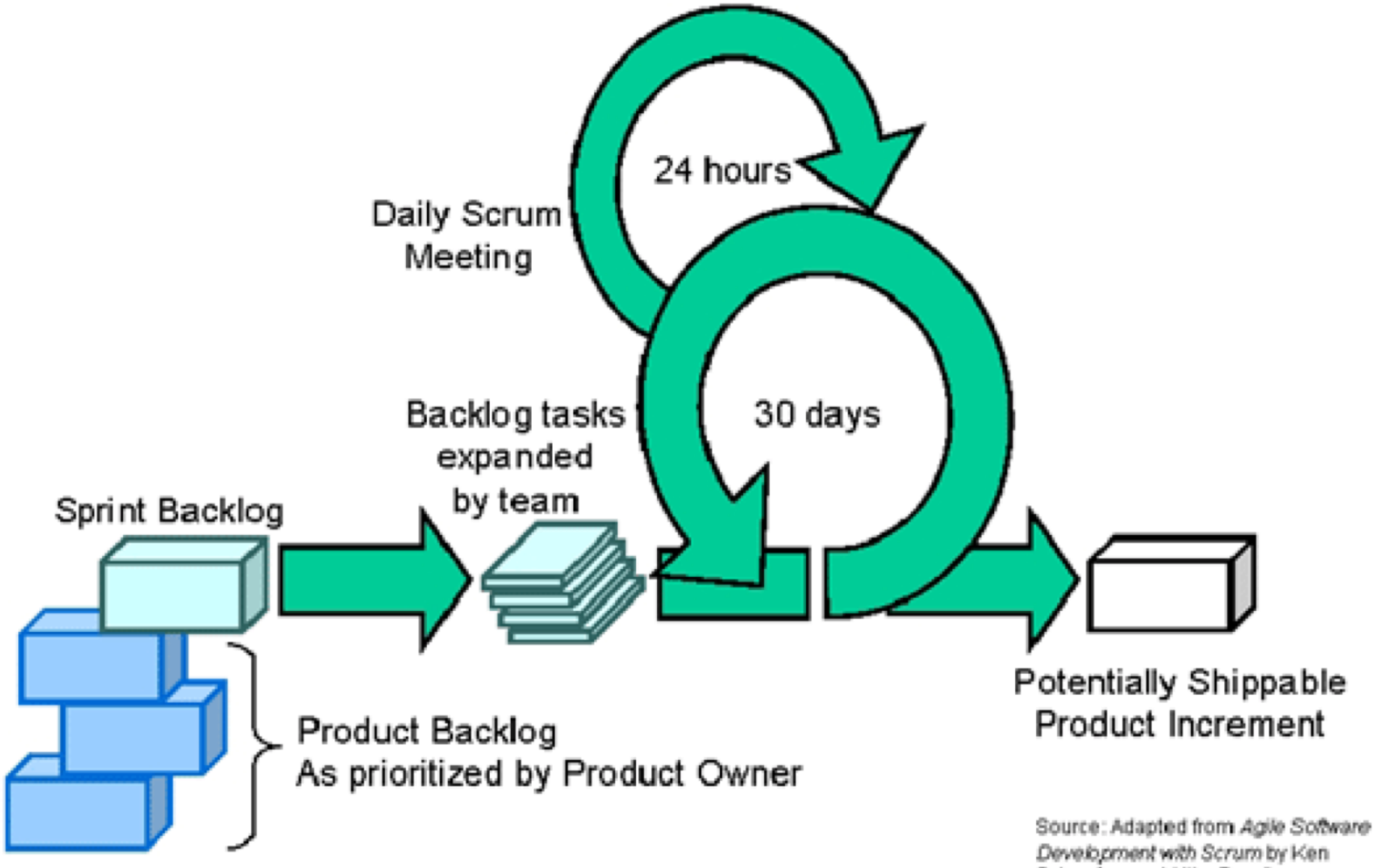
Annahme von Scrum

„You can‘t predict or definitely plan
what you will deliver,
when you will deliver it, and
what the quality and cost will be“

Scrum: Vokabular

- Die Rollen
  - Team
  - Product Owner
  - Scrum Master
- Kernprodukte
  - Product Backlog
  - Sprint Goal / Sprint Backlog
- Tätigkeiten
  - Sprint
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

Scrum: Vorgehen



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Rollen

- Product Owner
  - Priorisiert Anforderungen
  - Wählt Anforderungen aus dem Product Backlog in das Sprint Backlog
  - Gibt als einziger Ziele vor
- Scrum Master
  - Sorgt dafür, dass das Team ungestört arbeiten kann
  - Unterstützt den Scrum-Prozess (in Projekt und Organisation)
- Team
  - Setzt Anforderungen in Produkt um
  - Übernimmt alle anfallende Aufgaben
    - keine fachliche oder organisatorische Hierarchie
  - Teamgröße: 5 +/- 2

Kein Projektleiter --
das Team organisiert sich

Kernprodukte

- Product Backlog
  - Priorisierte Anforderungen (*User Stories, Epics*)
    die möglicherweise im Produkt realisiert werden können
- Sprint Goal
  - Wesentliches Ergebnis (aus Sicht des Product Owners) für den Sprint
- Sprint Backlog
  - Aufgaben (*Tasks*), die notwendig sind, um die für einen Sprint ausgewählten Anforderungen (User Stories) zu realisieren
  - Eine User Story sollte in max. 1 Woche umsetzbar sein

Das war alles?

- Scrum ist ein Managementprozess
  - …der gut zu Software, nicht aber zu Brücken passt

- Scrum lässt sich gut verbinden mit
  - Kanban
    - “Begrenze den Durchfluss!“
      (wenige Arbeitspakete gleichzeitig bearbeiten)
    - „Visualisiere!“
  - XP (eXtreme Programming)
    - Verwenden von Praktiken wie
      - Pair Programming
      - Test Driven Development
      - Continous Integration

Diskussion

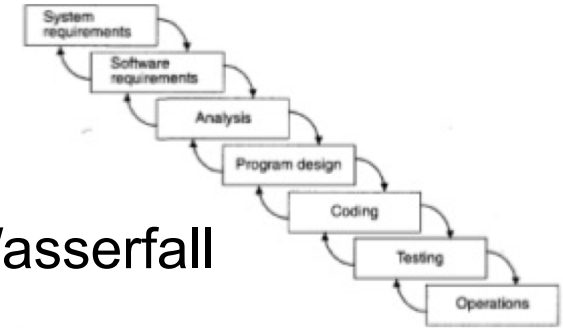
- Überlegen Sie jeweils drei Vorteile und drei Nachteile von Scrum

Arto Teräs
http://ajt.iki.fi/travel/debconf5/page2.html

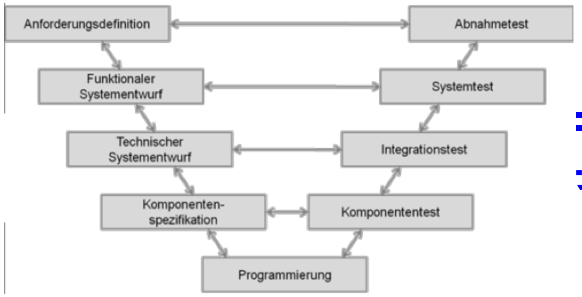
Inhalt

1. Einführung
2. Prozessparadigmen
3. Scrum
4. Weitere Prozesse im Vergleich

Beispiele für SE-Prozesse



Wasserfall



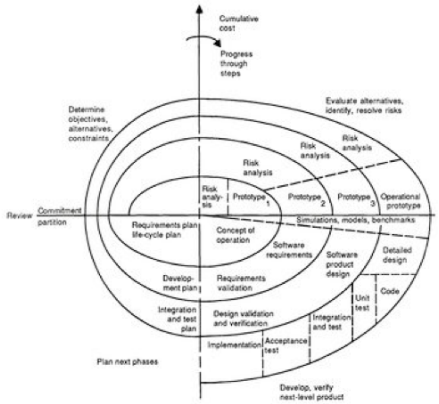
V-Modell

sequentiell

iterativ

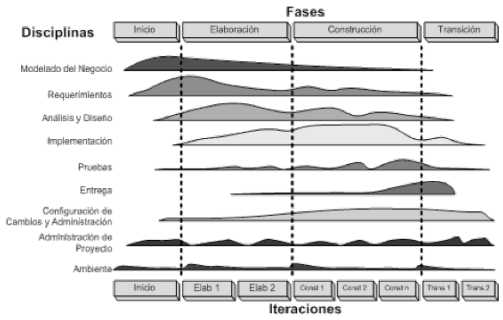
plangetrieben

agil

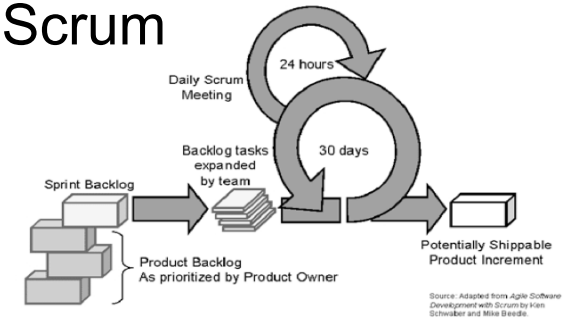


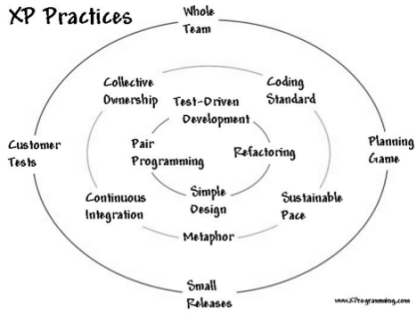
Spiralmodell

RUP
Unified Process



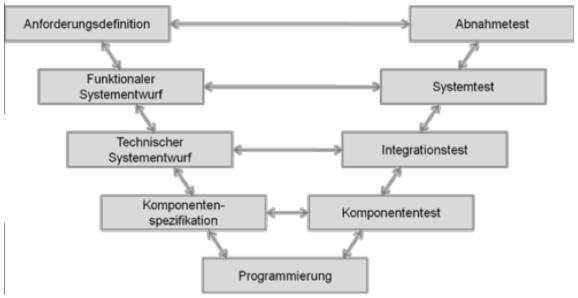
Scrum



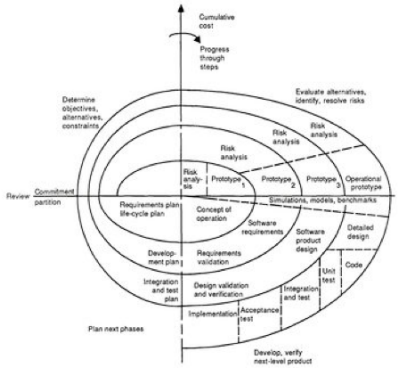


Extreme Programming

V-Modell / Spiralmodell

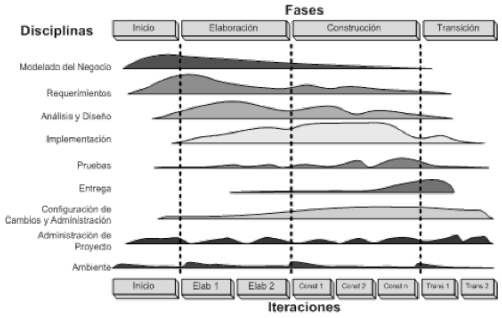


- V-Modell
  - Sequentiell (wie Wasserfall)
  - Wesentliche Idee
    - jeder Tätigkeit/Phase explizit eine Testtätigkeit / -phase zuordnen
  - Verbesserte / erweiterte Version: V-Modell XT

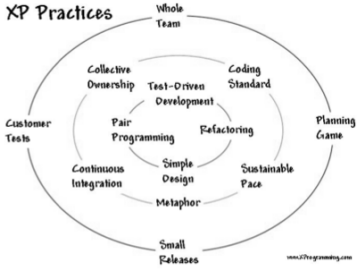


- Spiralmodell
  - Sequentiell oder iterativ
  - Wesentliche Idee
    - Vor jeder Phase wird eine Risikoabschätzung (rechter oberer Quadrant) gemacht und es werden mit Hilfe von Prototypen offene Punkte geklärt
    - Die “eigentliche” Projektarbeit findet im rechten unteren Quadrant statt. Die Tätigkeiten können dort sequentiell oder iterativ eingebettet sein.

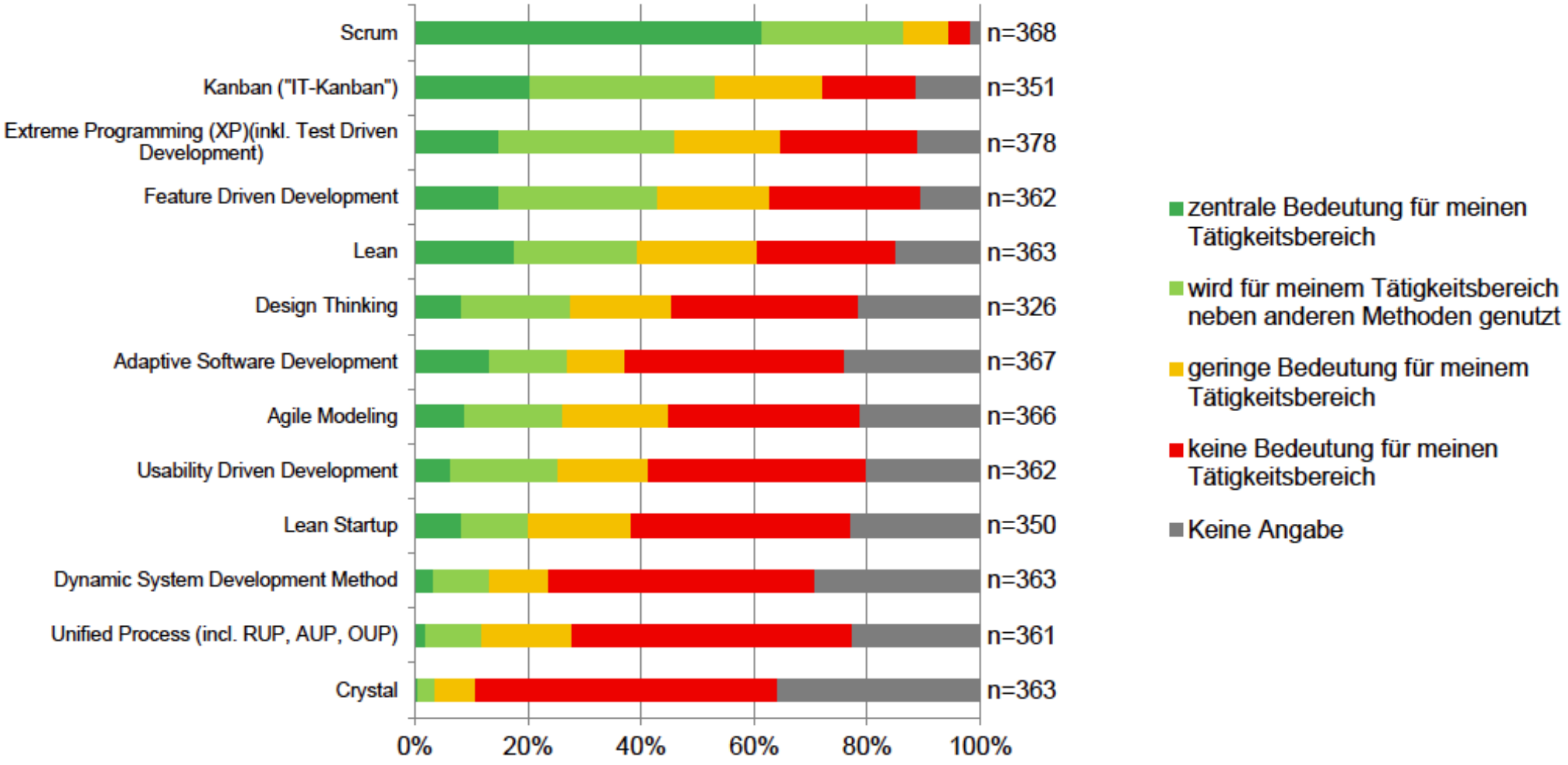
RUP / XP



- RUP – (Rational) Unified Process
  - Iterativ
  - Wesentliche Idee
    - Zu Beginn die Architektur festlegen
    - In jeder Iteration mit Anforderungsänderungen umgehen
    - Früh mit Testen beginnen
    - Grafische Modellierung der Software (UML)
    - Inkrementelle Erstellung der Dokumentation / des Produkts
- XP – eXtreme Programming
  - Iterativ, agil
  - Wesentliche Idee
    - Sammlung von *Praktiken* für erfolgreiche, agile Entwicklung
    - Bsp.: Pair Programming, TDD, Small Releases



Studie an der HS Koblenz



www.status-quo-agile.de | © BPM-Labor, Prof. Dr. A. Komus

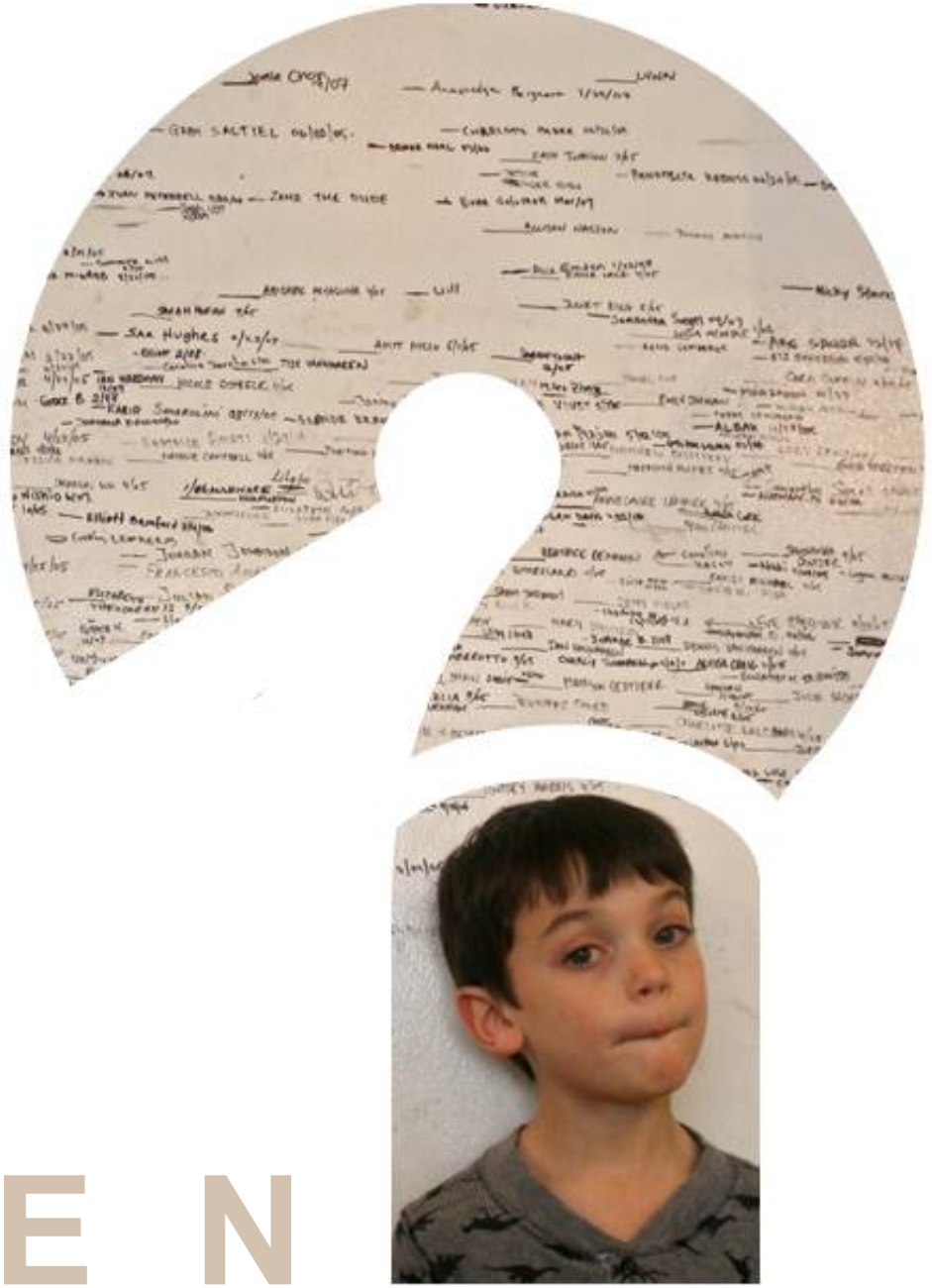
Plangetrieben vs. Agil (sehr komprimiert)

	Plangetrieben	Agil
Planung	+ welche Funktion + welche Kosten + welcher Termin	+ welches Team + wie viele Sprints (Dauer)
Kontrolle	+ Überwachung Kosten, Termin	+ impl. Funktion abnehmen + neue Funktionen festlegen
Methoden	+ explizite Dokumentation	+ Kommunikation
Scope	+ „groß“	+ „klein“

Fragerunde

Bei einem plangetriebenen Prozess weiß der Auftraggeber, wann er was zu erwarten hat (bzw. was für wann versprochen wurde).
Bei einem agilen Prozess weiß der Auftraggeber nicht, welches Produkt er am Ende des Projektes erwarten kann.

- Warum sollte ein Auftraggeber diese Ungewissheit auf sich nehmen?
- Was könnte man tun, um sie zu mildern?



F R A G E N