

PROJEKT ENGINEERING

Ablauforganisation

Herwig Mayr

Fakultät für Informatik, Kommunikation und Medien
Fachhochschule OÖ, Hagenberg

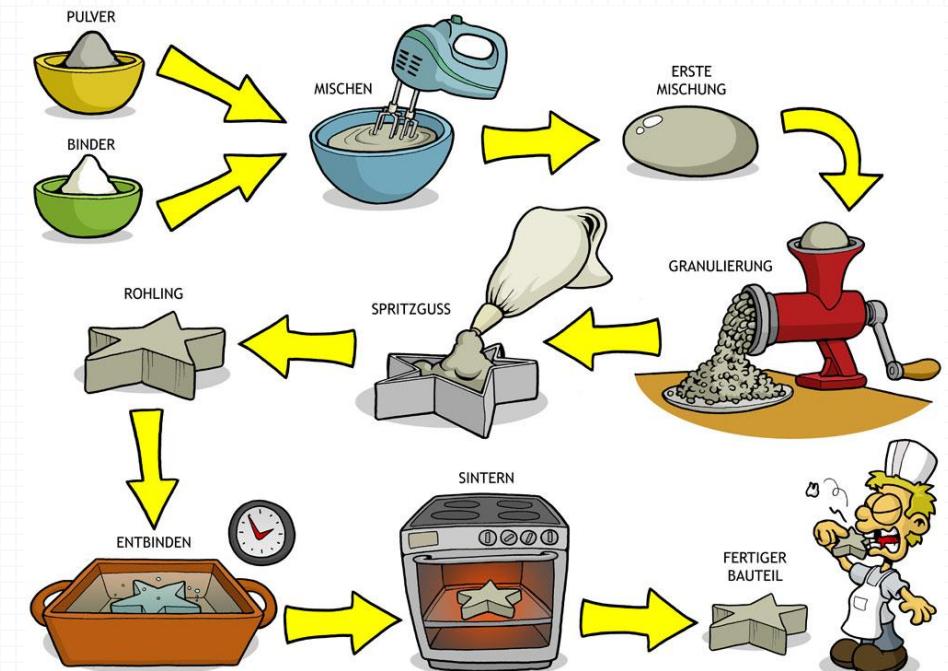
Prozess

Der Prozess in der Ablauforganisation

Ablauforganisation = Aneinanderreihung systeminterner Elemente (Arbeitsabläufe) zur Zielerreichung

Zweck:

- Arbeitsvorgänge organisieren
- Verantwortlichkeiten formalisieren
- Rationalisierung ermöglichen
- Standardisierung anstreben

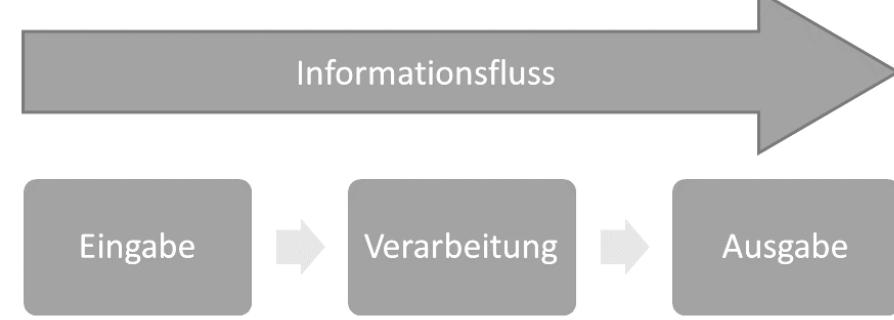


(Bild: www.stefanstrasser.at)

Prozessbegriff

Prozessdefinitionen (Beispiele):

- kontinuierliche Entwicklung mit vielen Änderungen [Webster's]
- Menge von Tätigkeiten, die Eingaben in Ausgaben transformieren [ISO 90003]
- Sequenz von Schritten zur Zweckerfüllung [IEEE]
- Methoden, Praktiken und Vorschriften zur Softwareentwicklung [SEI]



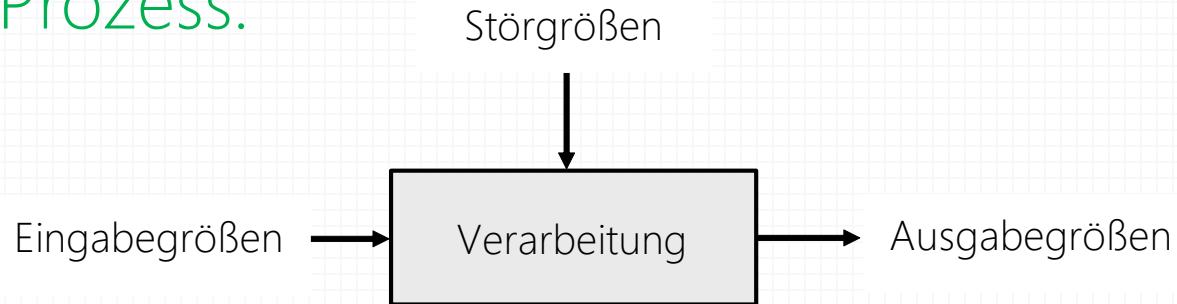
„EVA – Prinzip“



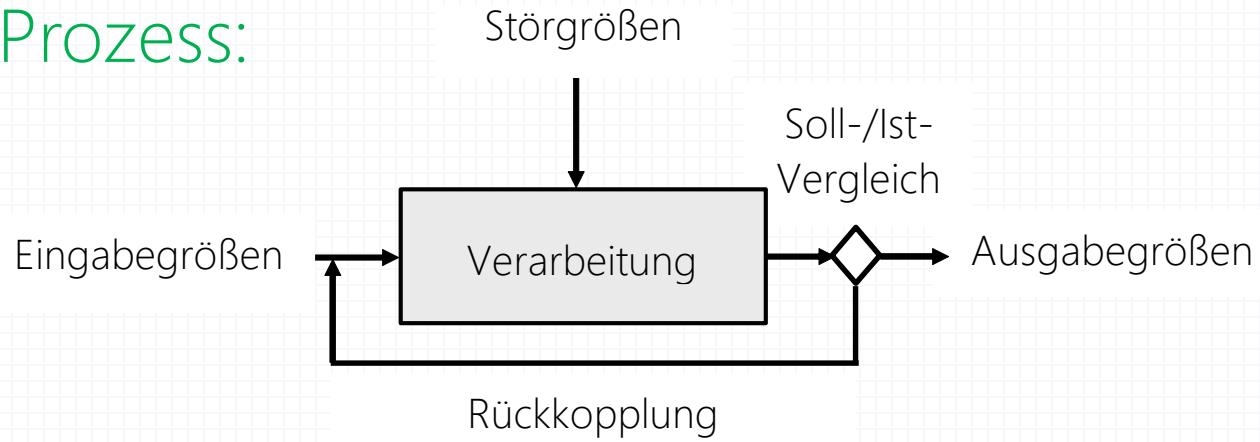
(Bild: www.joelle.de)

Gesteuerter vs. geregelter Prozess

Gesteuerter Prozess:



Geregelter Prozess:



-> In der Softwareentwicklung benötigt man **geregelte** Prozesse!

Beispiel/Übung: Wo in diesem Hörsaal sind gesteuerte Prozesse, wo geregelte Prozesse?

Gesteuerte Prozesse:



Geregelte Prozesse:

Definierter vs. empirischer Prozess

Definierter Prozess:

- Aktivitäten vorab bekannt und verstanden
- Ergebnis vorhersehbar
- Prozess wiederholbar
- Beginn und Ende vorab festlegbar

Technik: z.B. schrittweise Verfeinerung

Empirischer Prozess:

- komplex und unvorhersehbar
- nicht vollständig verstanden
- nicht durchgehend definierbar

Technik: z.B. iterativ-inkrementelles Adaptieren

Auch empirische Prozesse können erfolgreich geregelt werden, selbst wenn sie nicht vollständig verstanden werden!

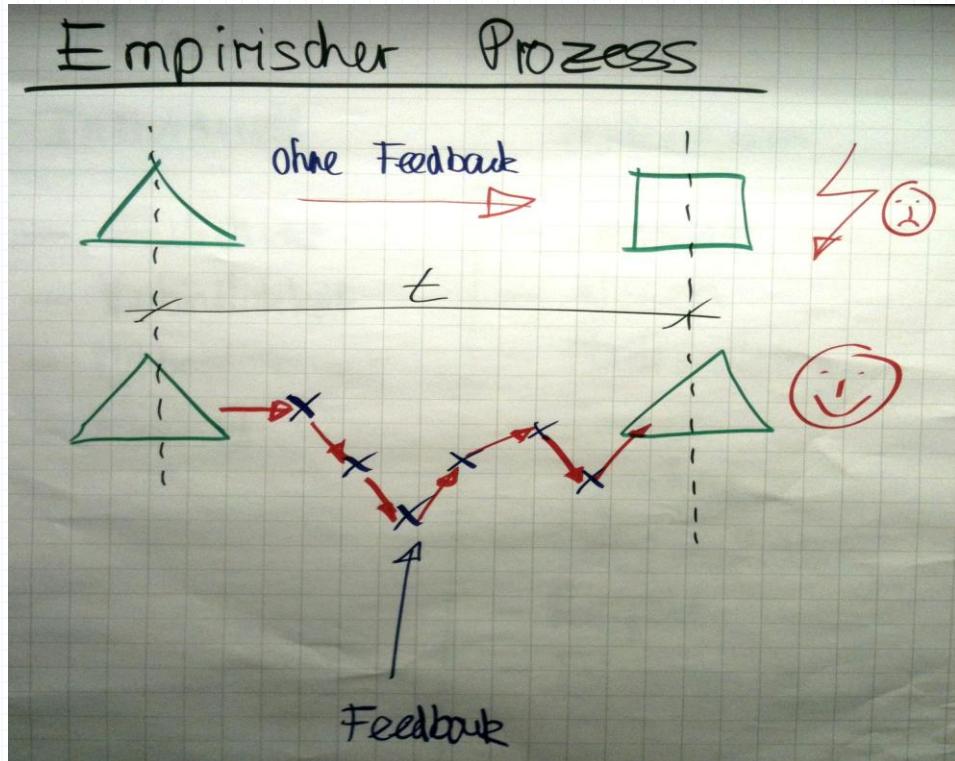
Beispiel/Übung: Wo werden definierte, wo empirische Prozesse verwendet?

Definierte Prozesse:

Empirische Prozesse:



Empirischer Prozess



(Bild: www.pro4-it-solutions.de)

Erfolgreiche Regelung durch:

- häufige Inspektionen
- laufende Anpassung
- Trennung der Streuung der Eingabe von Störfaktoren

Prozesse in der Softwareentwicklung werden im 21. Jahrhundert als empirische Prozesse gesehen!

Empirischer Softwareentwicklungsprozess

Paradigmenwechsel:

- kontinuierlich planen
- frühzeitig realisieren, sobald genügend Anforderungen bekannt; „the art of the possible“ [A. Cockburn]
- keine Stabilität erwarten
- auf Änderungen von Anforderungen oder Rahmenbedingungen rasch reagieren

Rückkopplung durch Bewertung der Zwischenprodukte durch den Kunden ist entscheidend für den weiteren Projektverlauf (**emergentes Verhalten**) sowie die Ausrichtung des Produkts (**Value-added Software Development**).

Entwicklung der Prozessbedeutung

Verschiebung der **Bedeutung des Prozesses** im Lauf der Zeit:

- **normative Bedeutung** (ab ~1985):
Es gibt einen „besten“ Prozess.
- **transformierende Bedeutung** (ab ~1995):
Prozesse brauchen konkrete Umsetzungsrichtlinien.
- **wertgenerierende Bedeutung** (ab ~2005):
Der Prozess wird durch den Wert des (Teil-)Produkts für den Kunden gesteuert.

Kernaufgaben der Ablauforganisation

Aufgaben:

- Prozesse in Gang setzen
- Prozesse analysieren
- Prozesse verbessern



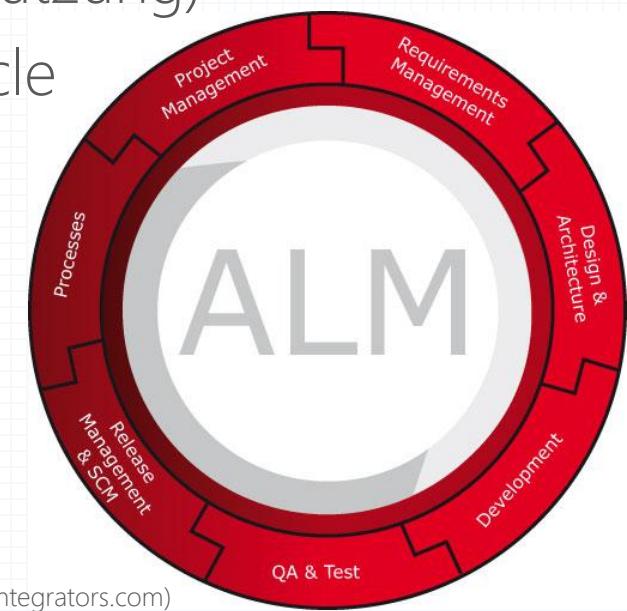
(Bild: www.apollon-hochschule.de)

Prozesse sind produktbezogen (aus Projektsicht statisch)
oder prozessbezogen (aus Projektsicht dynamisch).

Prozesse vs. Lebenszyklen (I)

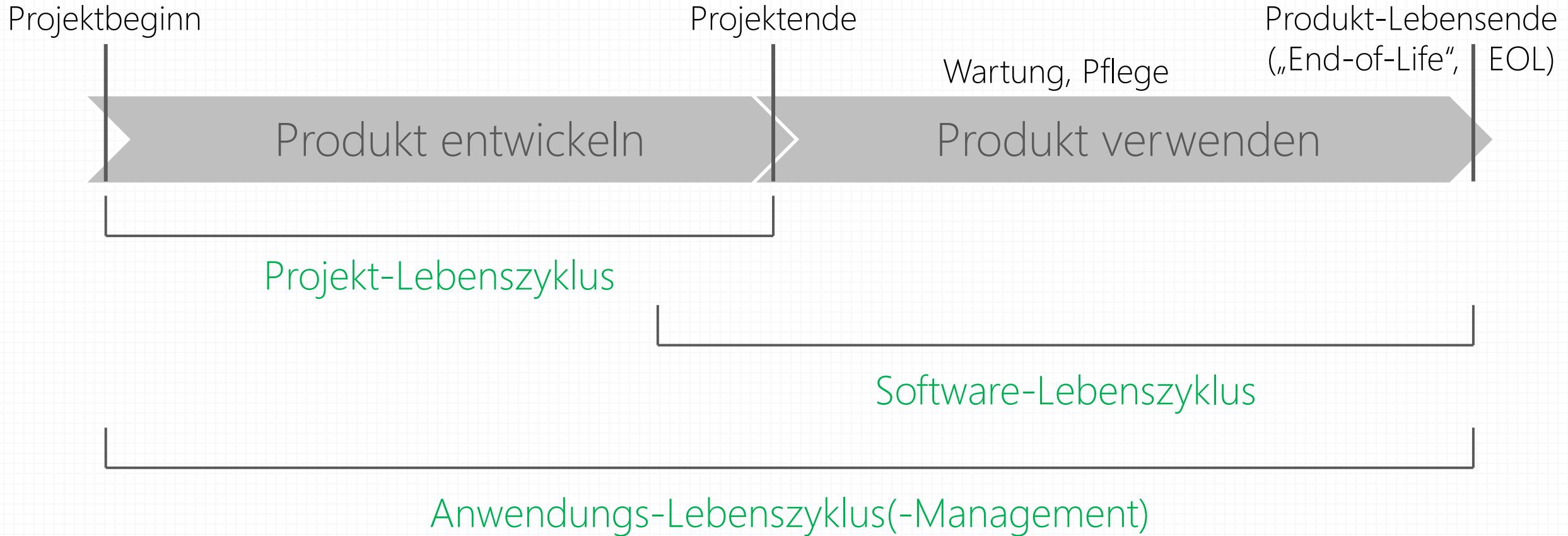
„Prozesse im Großen“

- **Projekt-Lebenszyklus**: Ordnung der zeitlichen Abfolge der Aktivitäten (von der Entwicklung bis zum Einsatz)
- **Software-Lebenszyklus**: Ordnung der zeitlichen Abfolge der Aktivitäten (von der Entwicklung über den Einsatz bis zum Ende der Benutzung)
- **Anwendungs-Lebenszyklus-Management** (Application Lifecycle Management; ALM): Moderne Sicht – Trennung vor/nach Projektende wird unscharf (kontinuierliche Entwicklung und Freigabe während des gesamten Lebenszyklus des Produkts)



(Bild: redmondintegrators.com)

Prozesse vs. Lebenszyklen (II)

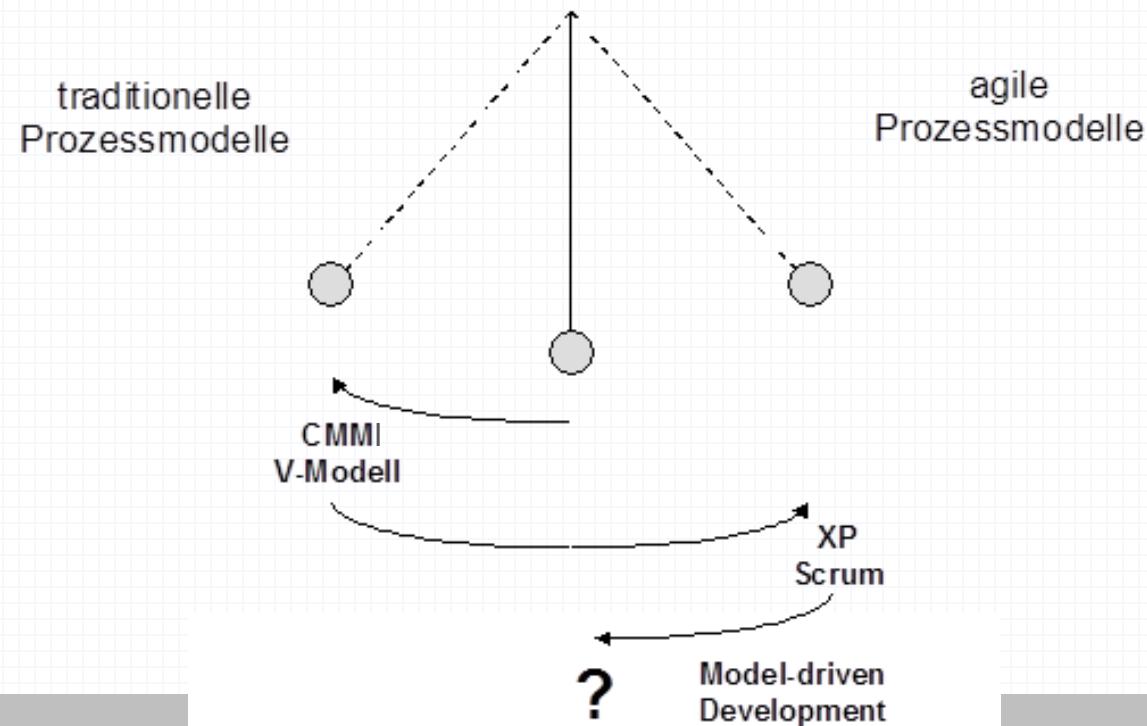


Prozessmodelle und Vorgehensmodelle (I)

ALT: Vorgehensmodell = Dokumentation der Ablauforganisation
-> wurde Ende der Neunziger Jahre immer umfangreicher

„Gegenbewegung“: Agile Softwareentwicklung

Suche nach der „goldenen Mitte“:



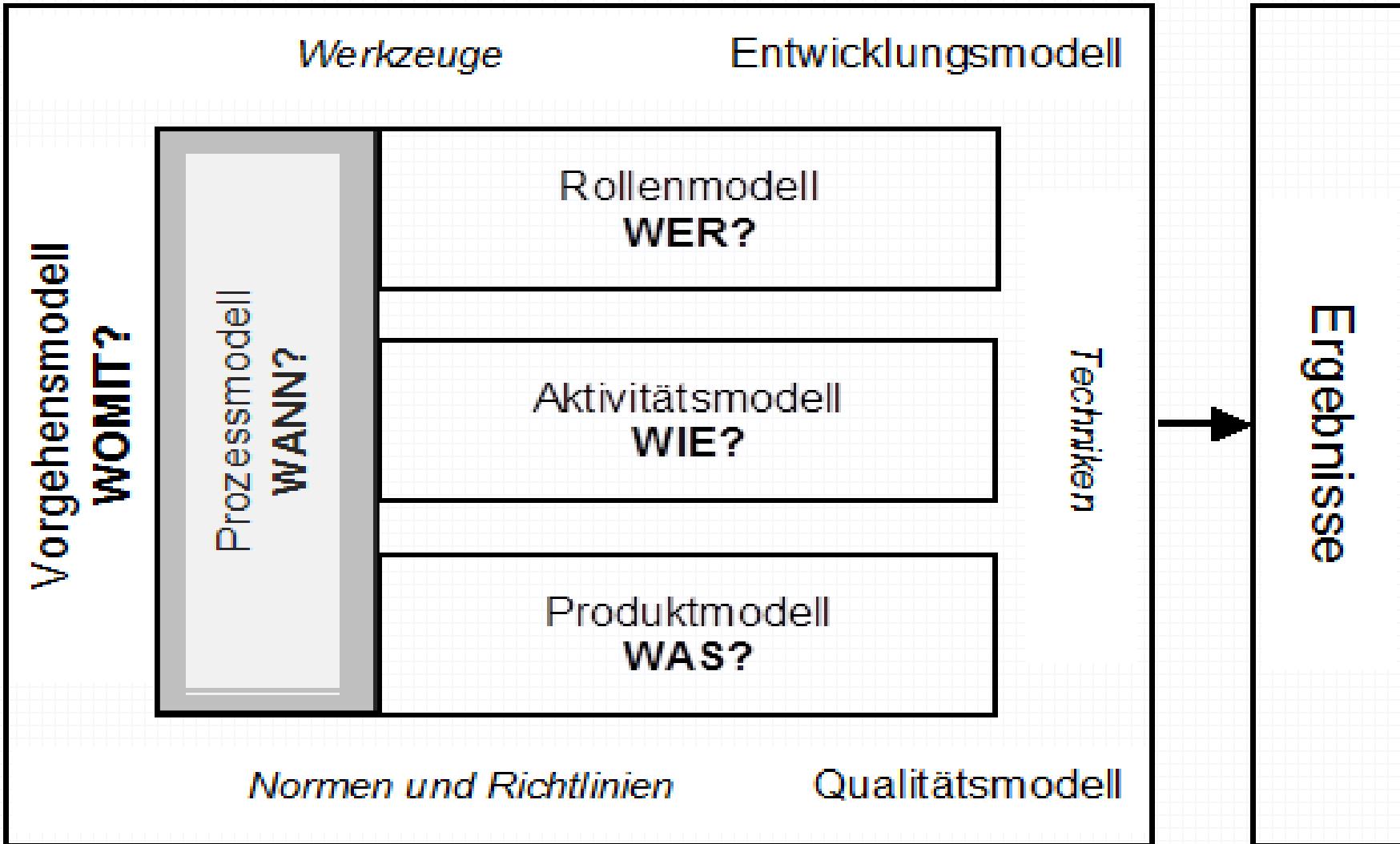
Prozessmodelle und Vorgehensmodelle (II)

NEU: **Vorgehensmodell** = systematische Gliederung einer Lösung

- Projektmanager versuchen zunehmend, aus Prozesserfahrungen zu lernen.
- **Prozessmuster (Patterns)** sind bewährte Praktiken, die induktiv aus Prozesserfahrungen erarbeitet wurden.
- Es gibt auch Sammlungen von Negativbeispielen (**Anti-Patterns**).

Beispiel/Übung: Welche Modelle für Vorgehen in der Softwareentwicklung kennen Sie?

Komponenten eines modellbasierten Vorgehens

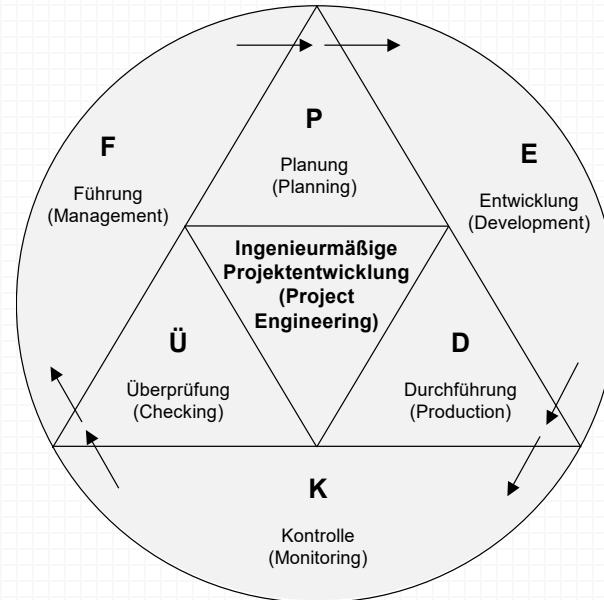


Prozessorientiertes Vorgehen

Prozessmodell unterteilt Vorgehen in überschaubare Abschnitte,

- schrittweise Planung,
- Durchführung und
- Überprüfung,

mit Rückkopplungen.



Prozessorientiertes Vorgehen = Vorgehen unter Beachtung eines expliziten Prozessmodells.

Vorgehensmethoden

Vorgehensmethoden

Unterscheide:

- Vorgehensprozess (engineering process); Umsetzung durch
- Vorgehensmethoden (engineering practices)

Oft Erfolg nur durch Kombination verschiedener Methoden!

Vorgehensmethoden – Sequenzielles Vorgehen

Phasenmodell

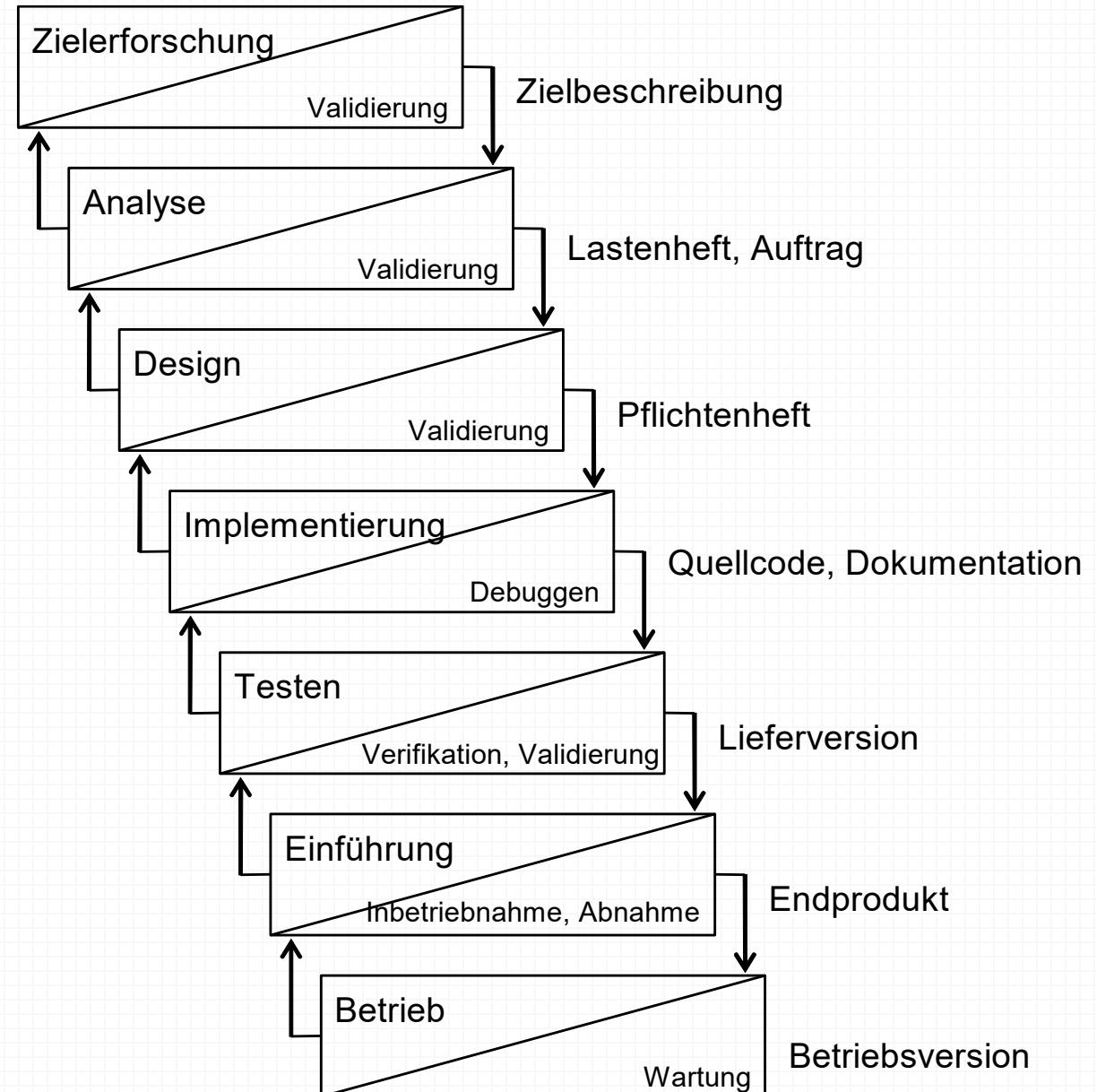


abgeleitet von der Organisation der Entwicklung
allgemeiner technischer Systeme -> veraltet

Vorgehensmethoden – Inkrementelles Vorgehen (I)

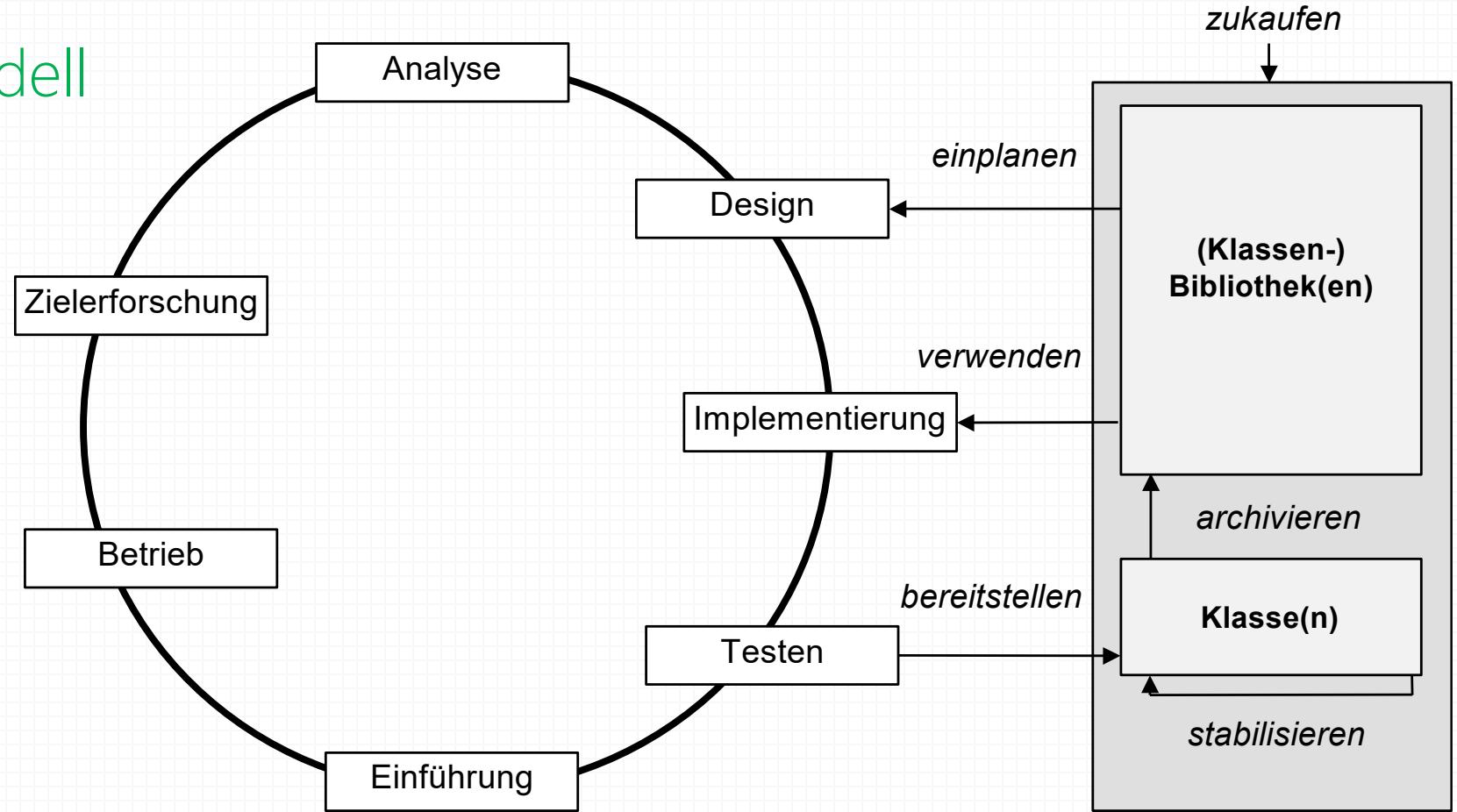
Wasserfallmodell

kaum Flexibilität,
Änderungswünsche teuer



Vorgehensmethoden – Inkrementelles Vorgehen (II)

Objektorientiertes Modell



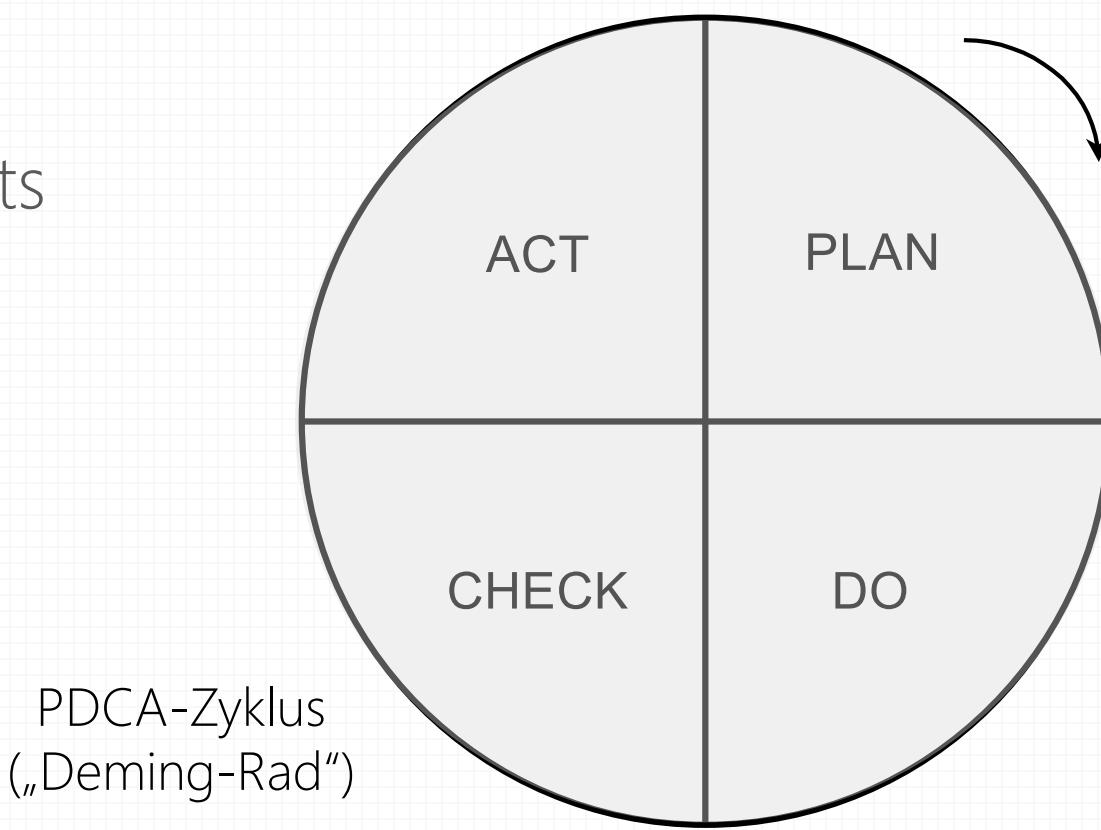
„Tool-Smithing“,
Wiederverwendung setzt Wiederverwendbarkeit voraus!

Beispiel/Übung: Wo kommt inkrementelles Vorgehen in der Praxis vor?

Vorgehensmethoden – Iteratives Vorgehen (I)

Grundidee: kontinuierliche Verbesserung (PDCA-Zyklus) [Deming]
(iterativ = wiederholt!)

Erstellung eines neuen Produkts
in jeder Iteration,
aber Aufbauen auf
erarbeitetem Wissen!



Beispiel/Übung: Wo kommt iteratives Vorgehen in der Praxis vor?

Vorgehensmethoden – Iteratives Vorgehen (II)

Motivation: Fehlerbehebungskosten steigen im Laufe der Projektentwicklung überproportional an.

Beispiel: Fehlerbehebungskosten bei Siemens PSE Österreich (ca. 2005)

Bereich	Analyse	Design	Codieren	Debuggen	Testen	Betrieb
Relative Anzahl der entstandenen Fehler	10%	40%	50%	-	-	-
Relative Anzahl der erkannten Fehler	3%	5%	7%	25%	50%	10%
Kosten pro Fehlerkorrektur (€)	250	250	250	1000	3000	12500

Vorgehensmethoden – Iteratives Vorgehen (III)

Exploratives Prototyping

Entwicklung so früh wie möglich – rasches Feedback vom Kunden,
danach neue Iteration

Prototypen und Prototyping:

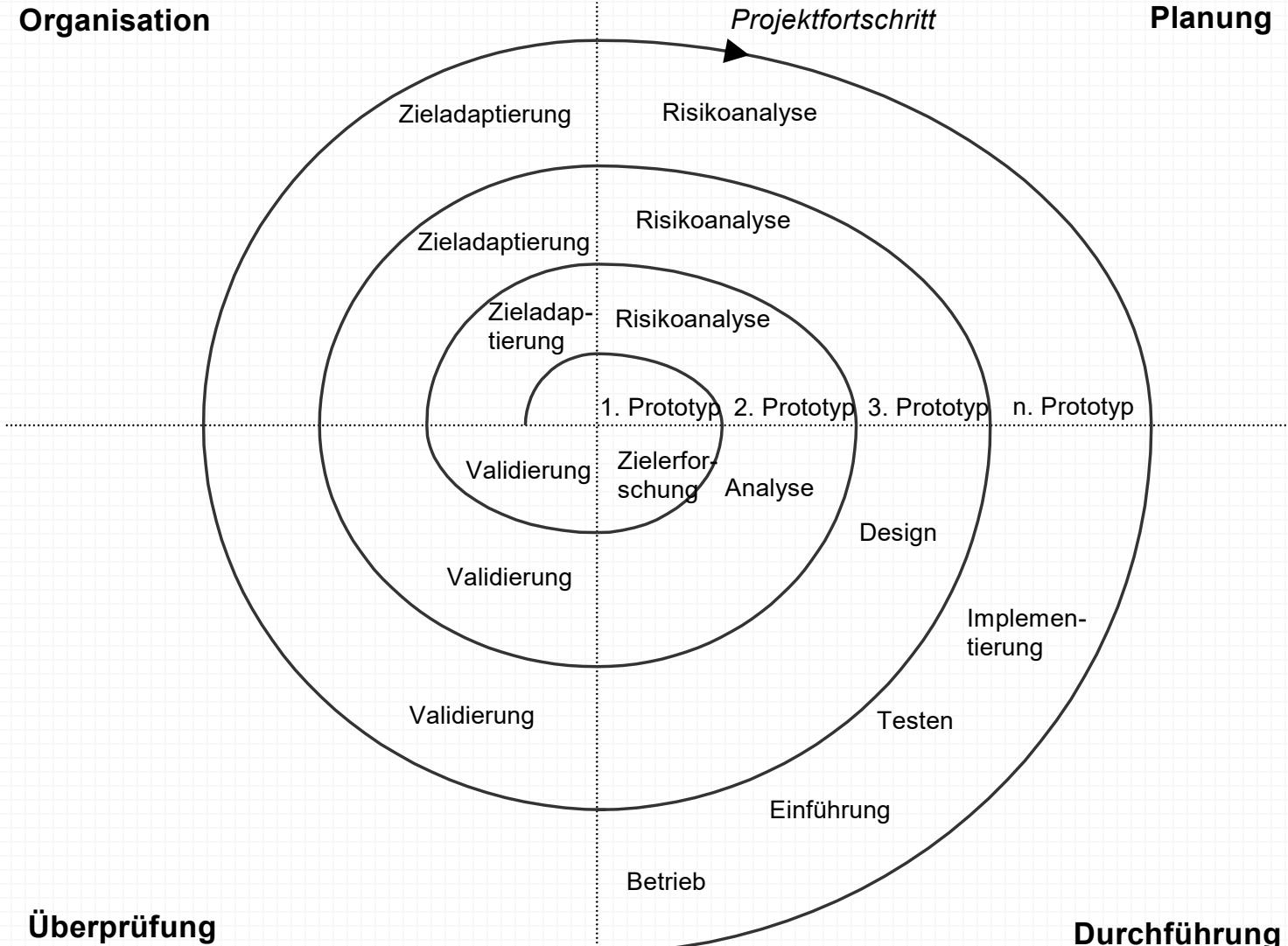
- Prototyp = (günstiges) ausführbares Modell eines Produkts
- Prototyping = Arbeiten zu Herstellung von Prototypen

Arten des explorativen Prototyping:

- Revelationäres Prototyping (eher Kunden-orientiert)
- Experimentelles Prototyping (eher Entwickler-orientiert)

Vorgehensmethoden – Iteratives Vorgehen (IV)

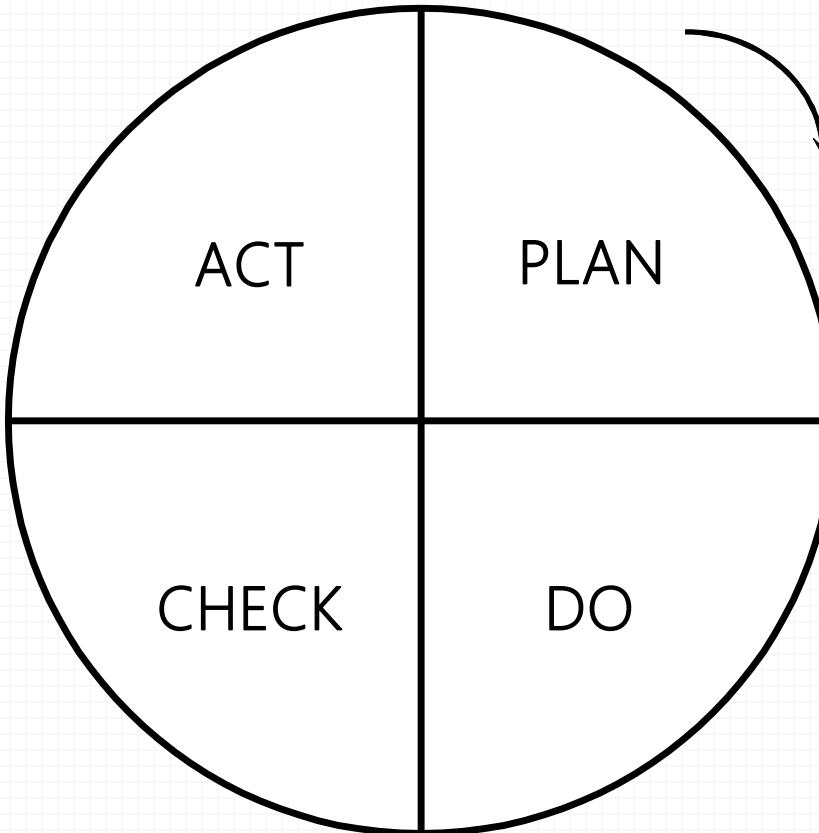
Spiralmodell



Verbindung von Prototyping
mit dem PDCA-Zyklus

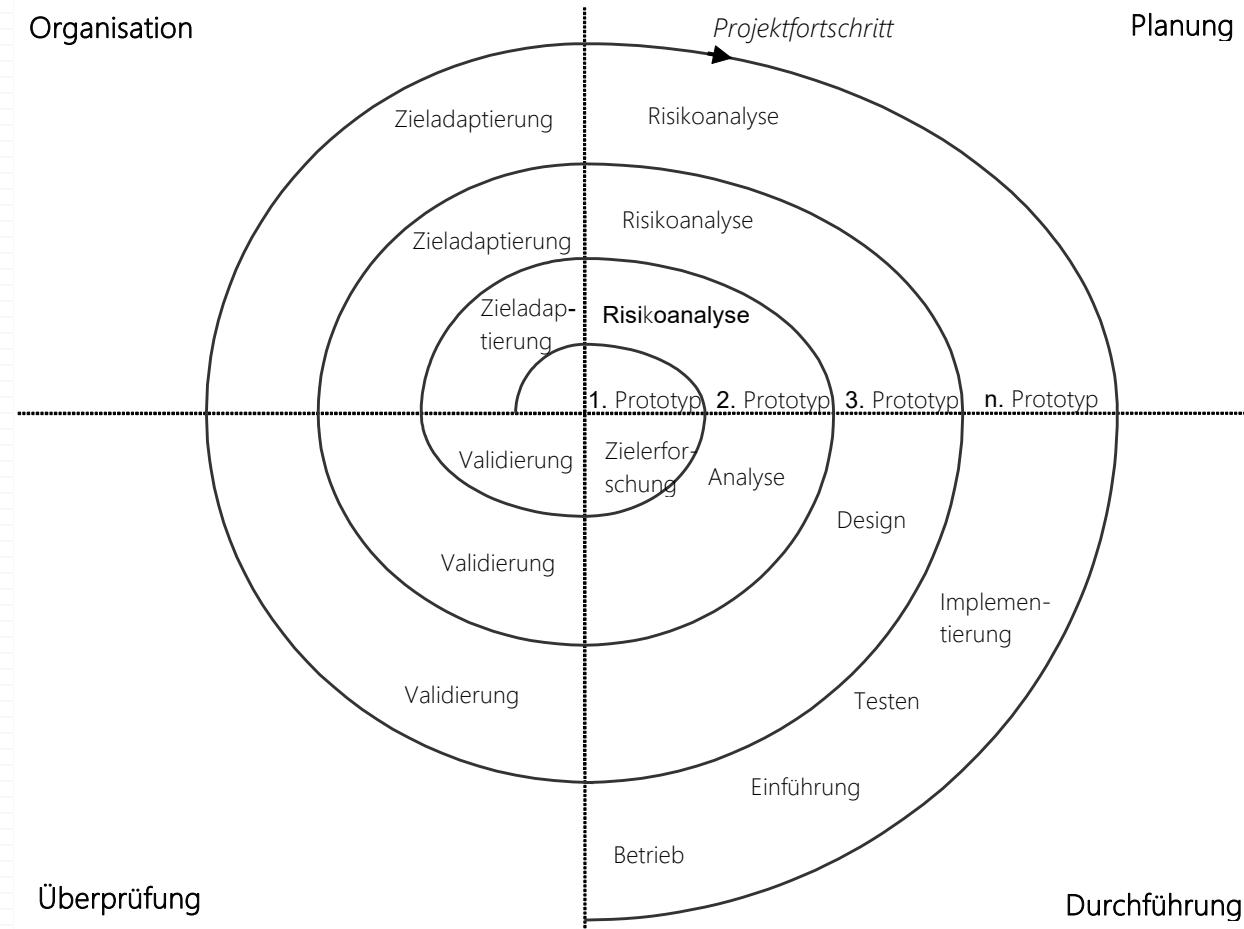
Spiralmodell (I): Grundlage Kaizen

Idee der kontinuierlichen Verbesserung [Deming]



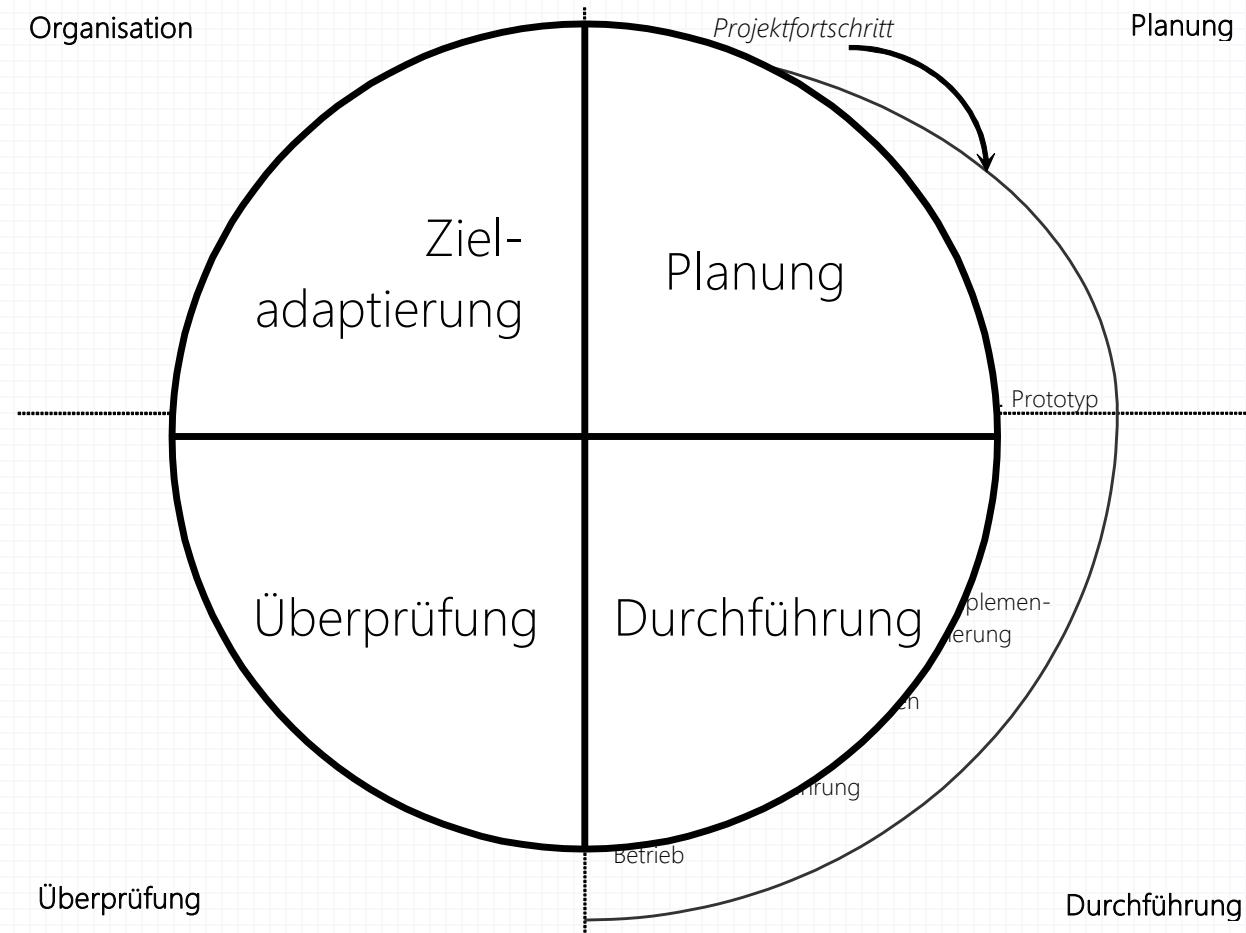
Spiralmodell (II): (rein) revolutionäres Prototyping

Einbringen von Prototyping-Konzepten [Boehm]



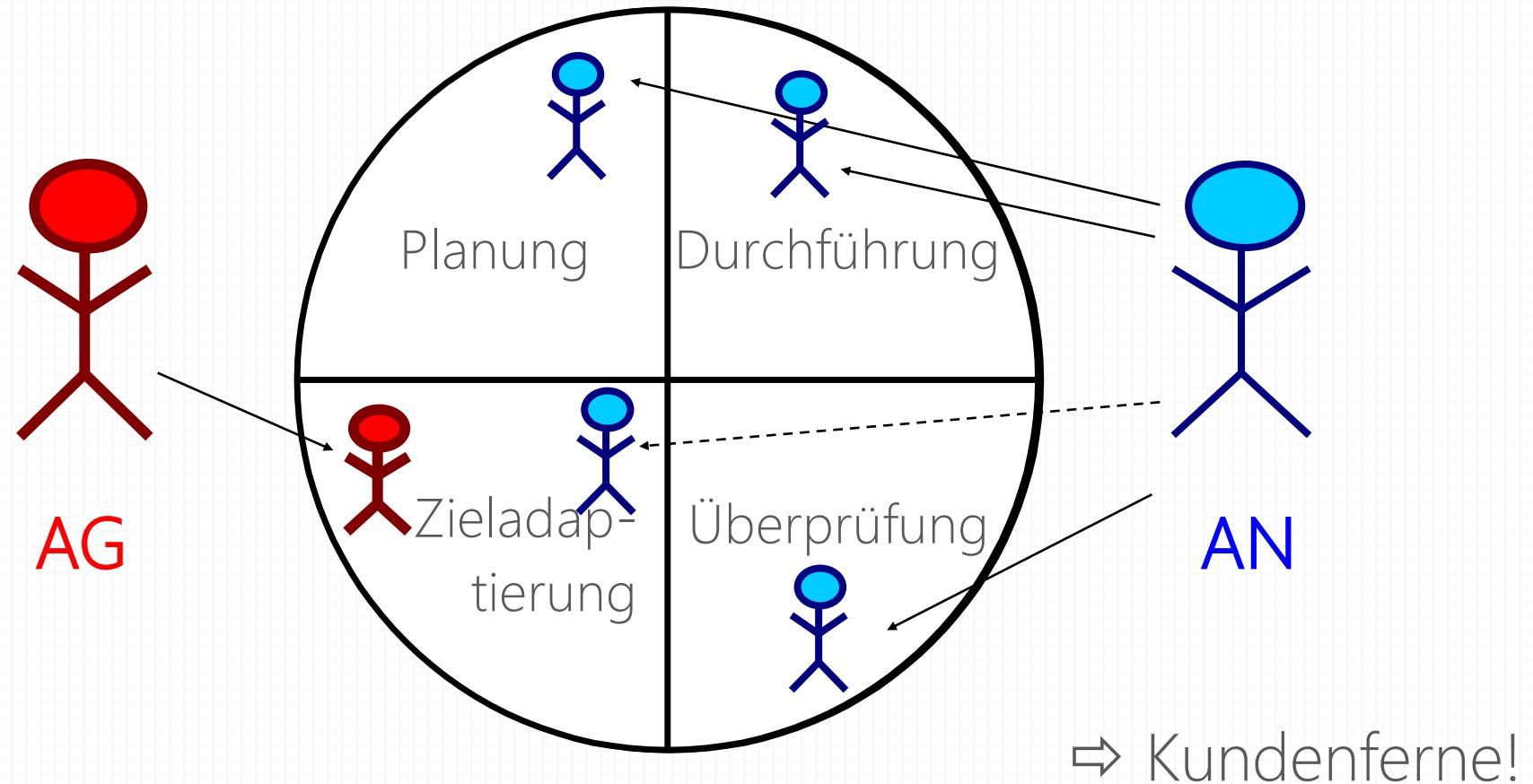
Spiralmodell (III): (rein) revolutionäres Prototyping

Einbringen von Prototyping-Konzepten [Boehm]



Spiralmodell (IV): (rein) revolutionäres Prototyping

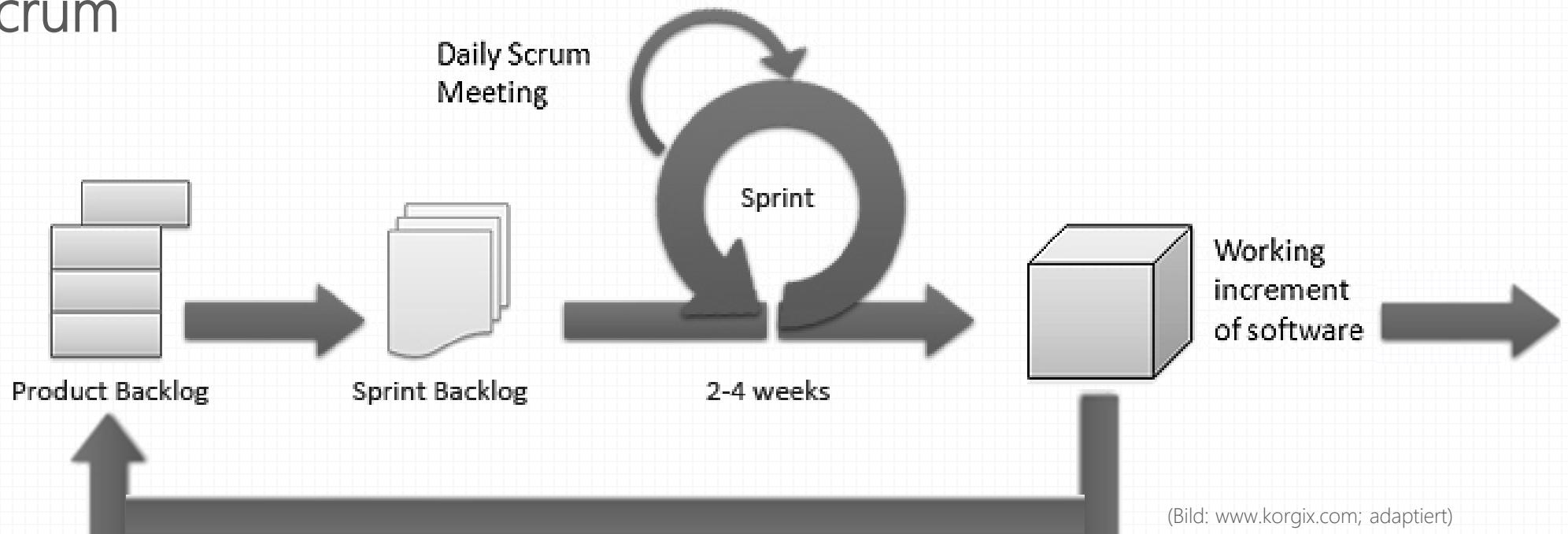
„klassisches“ Spiralmodell



Vorgehensmethoden – Iterativ-inkrementelles Vorgehen (I)

Grundidee: **Evolutionäres Vorgehen** – Entwicklung in Zyklen, jeweils auf den Ergebnissen des vorherigen Zyklus aufbauend

Beispiel: Scrum



Vorgehensmethoden – Iterativ-inkrementelles Vorgehen (II)

Evolutionäres Prototyping

Prototypen werden inkrementell weiterentwickelt;
funktioniert gut bei objektorientiertem Denken

Probleme:

- rasch Illusion eines „fertigen“ Produkts
- Fortschrittsmessung schwer möglich (z.B. Meilensteine)
- Gefahr der Lieferung von Prototypen statt Produkten



Vorgehensmethoden – Iterativ-inkrementelles Vorgehen (III)

Schablonenmodell

Erkennen und Anwenden von

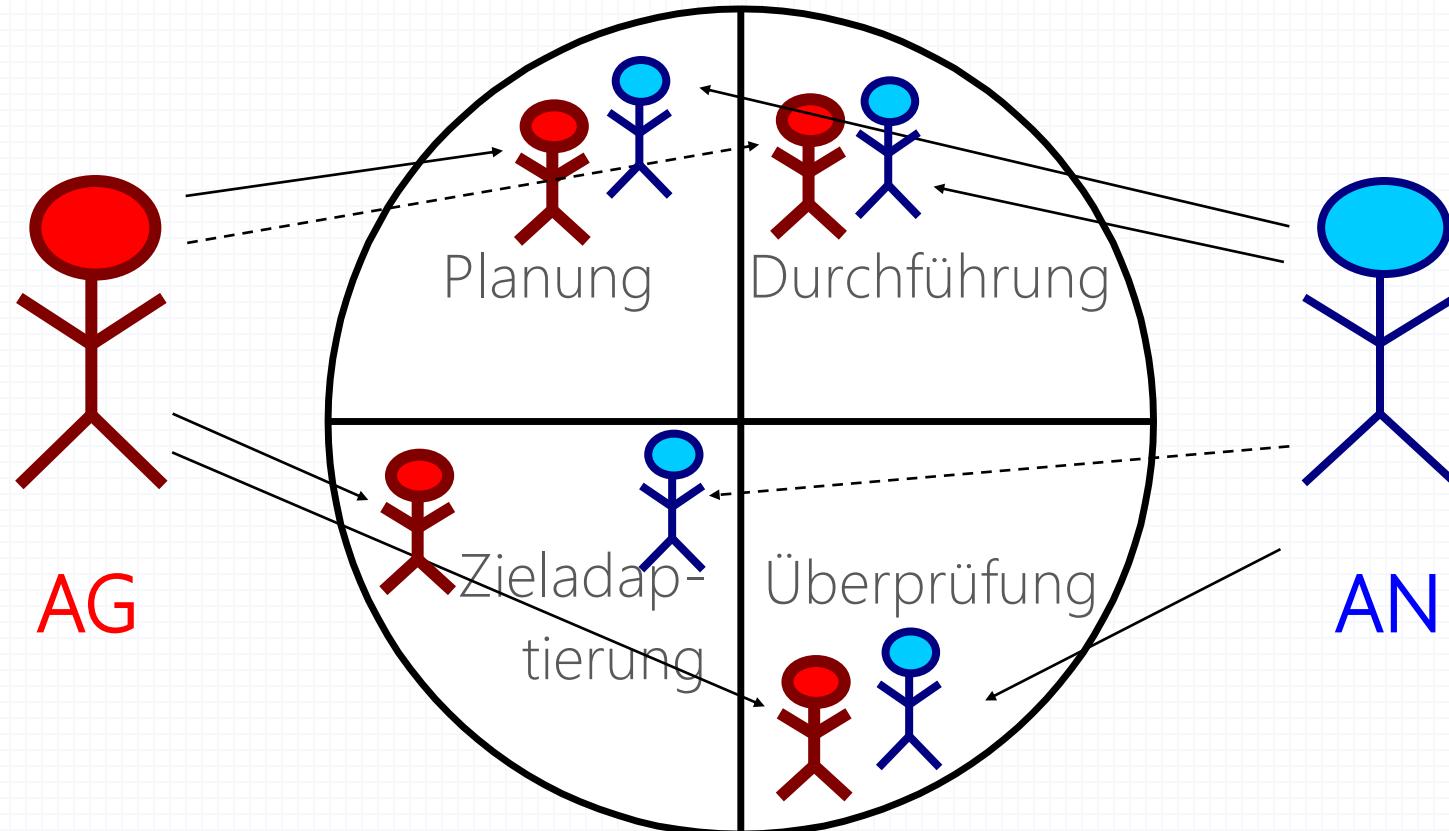
- Denkmustern (**patterns**) und
- Problemlöseschablonen (**templates**)

Klassifizieren eines neuen Problems durch
Pattern Matching auf einen Schablonenkatalog



Iterativ-inkrementelles Vorgehen: 1. Kunden-Integration

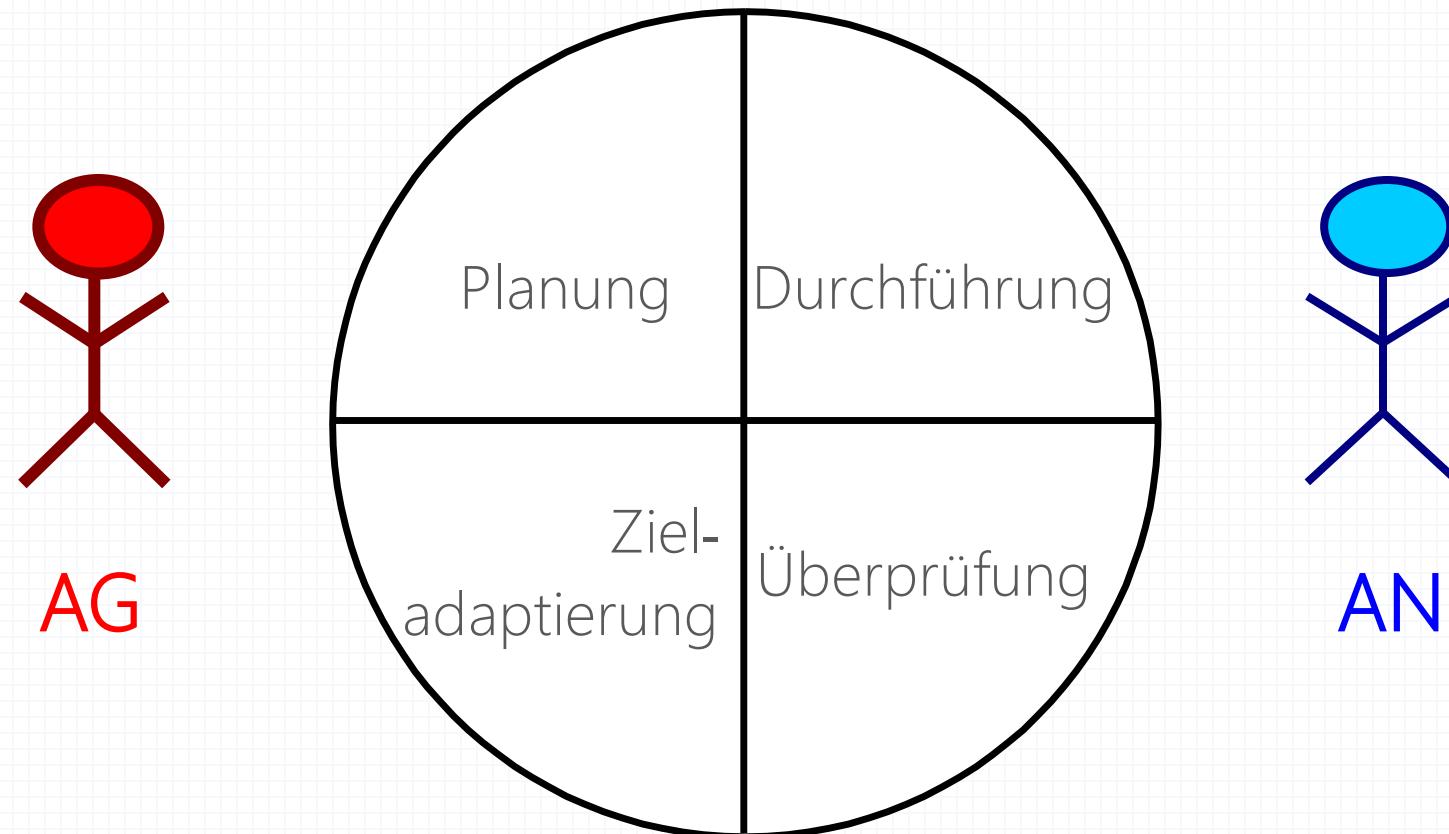
„agil“ (XP-Hype)



⇒ „zu große“
Kundennähe,
Einmischung!

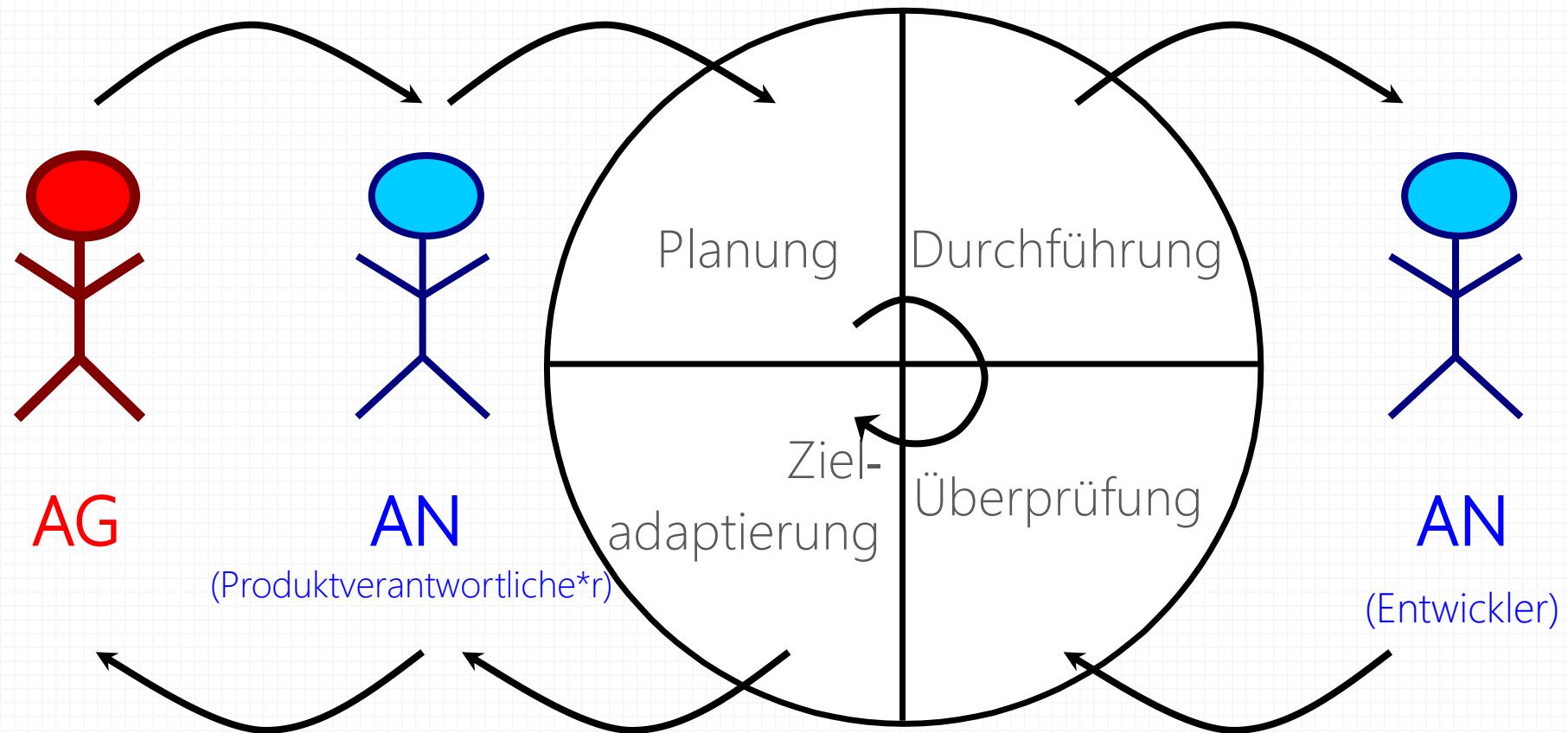
Iterativ-inkrementelles Vorgehen: 2. Pufferfunktion

„ideale“ Kundennähe



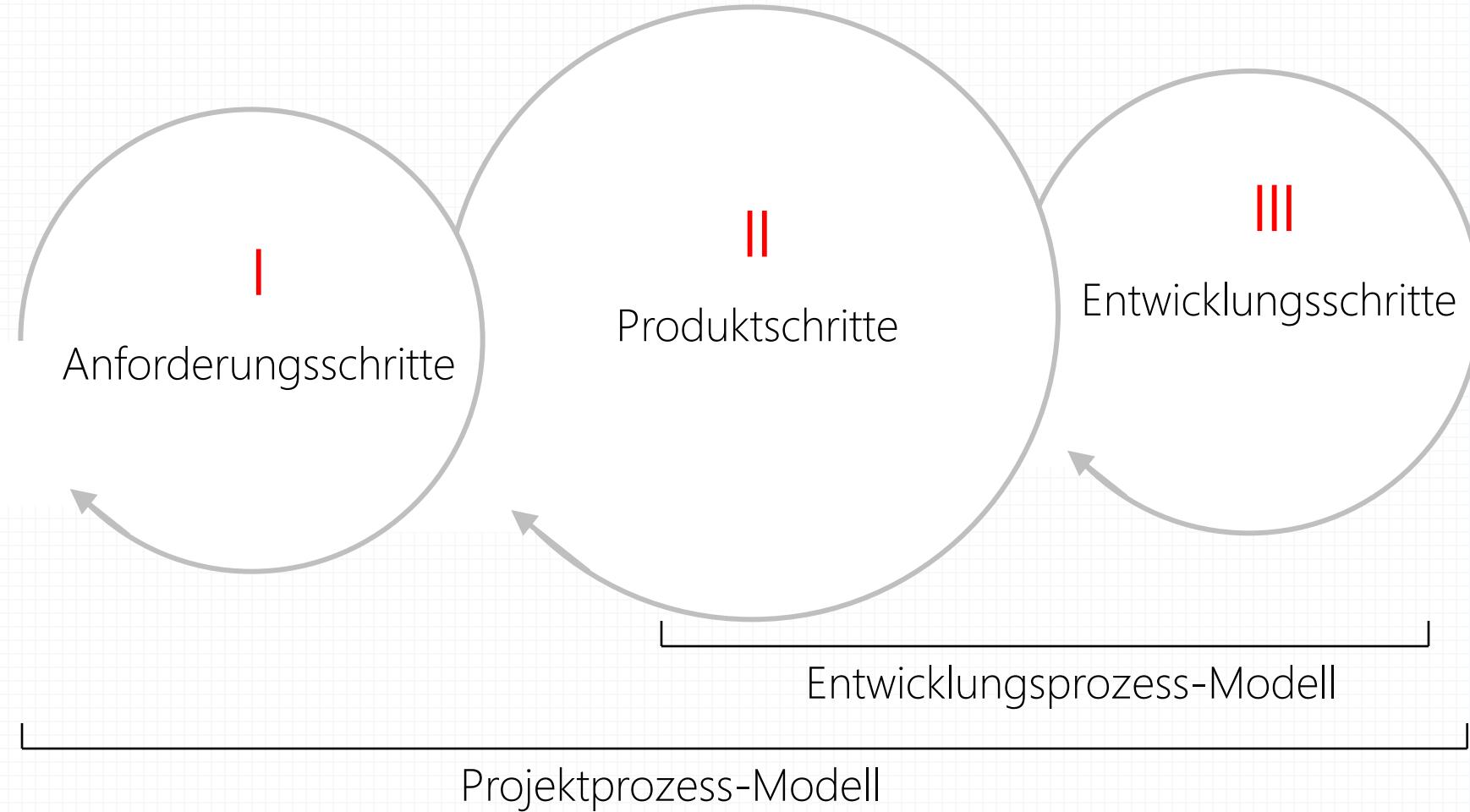
Iterativ-inkrementelles Vorgehen: 3. Produktverantwortliche*r

„ideale“ Kundennähe



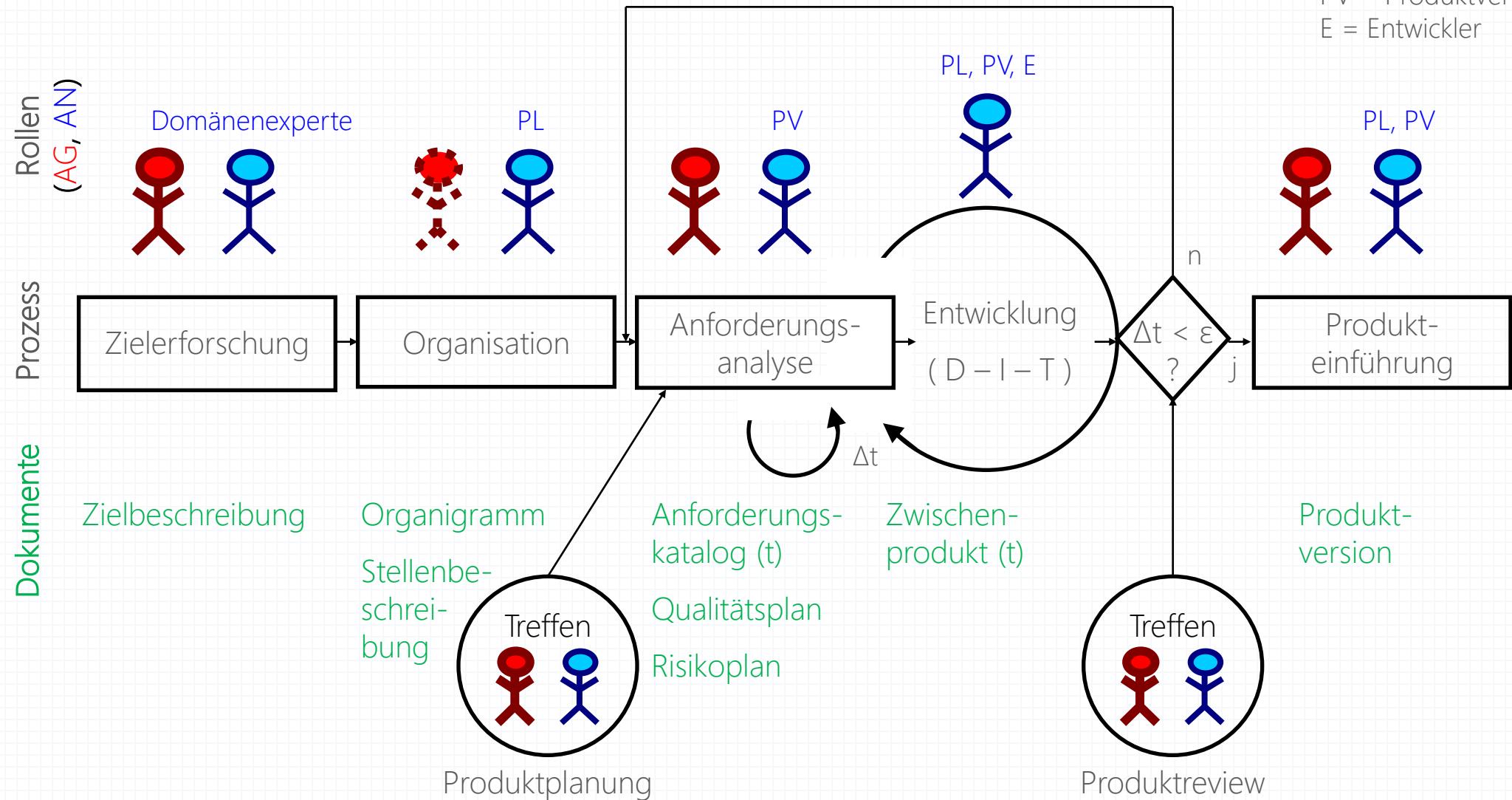
Iterativ-inkrementeller Entwicklungsprozess

dreifach rückgekoppelter Prozess:

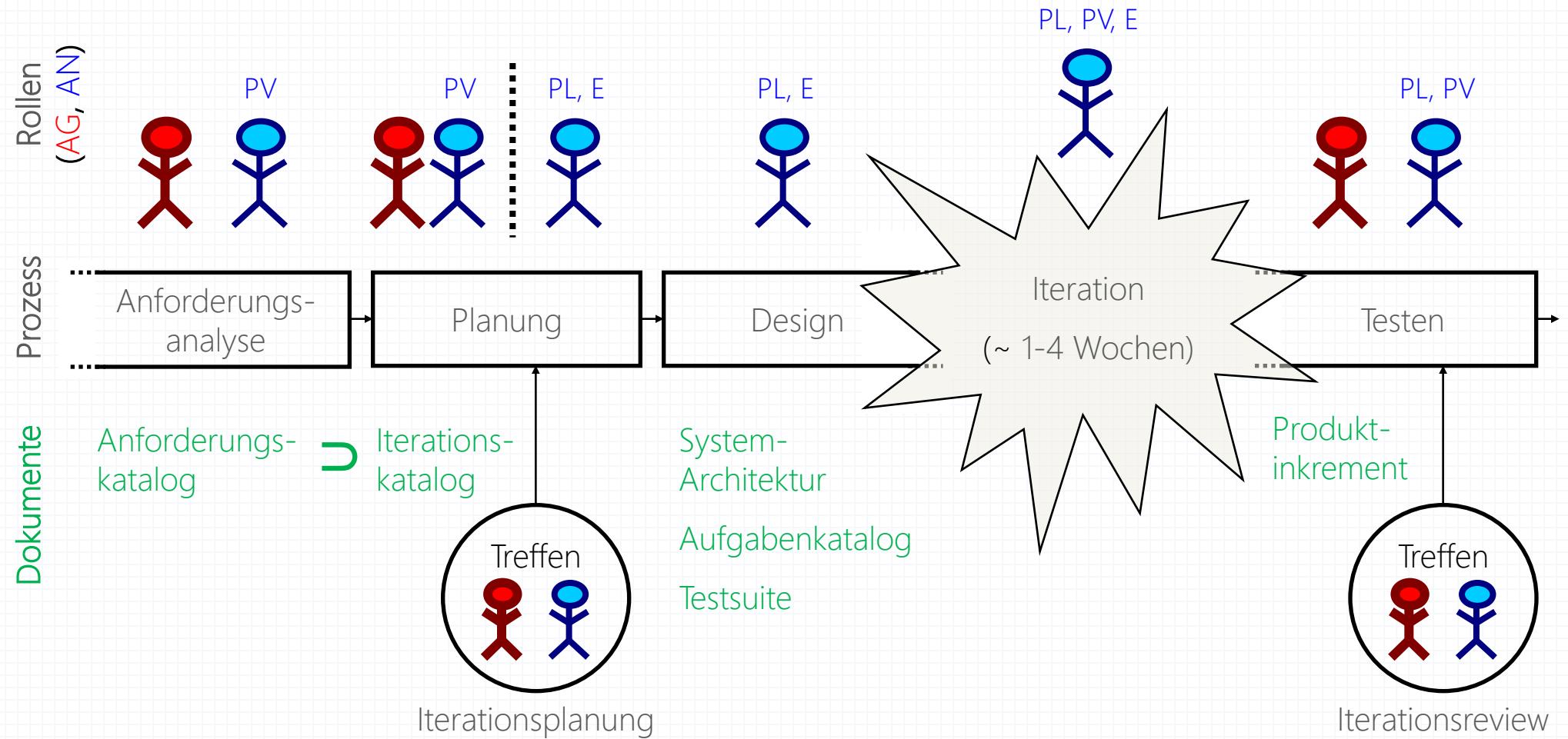


Einbettung in den Entwicklungszyklus (I)

AG = Auftraggeber
 AN = Auftragnehmer
 PL = Projektleiter
 PV = Produktverantwortlicher
 E = Entwickler

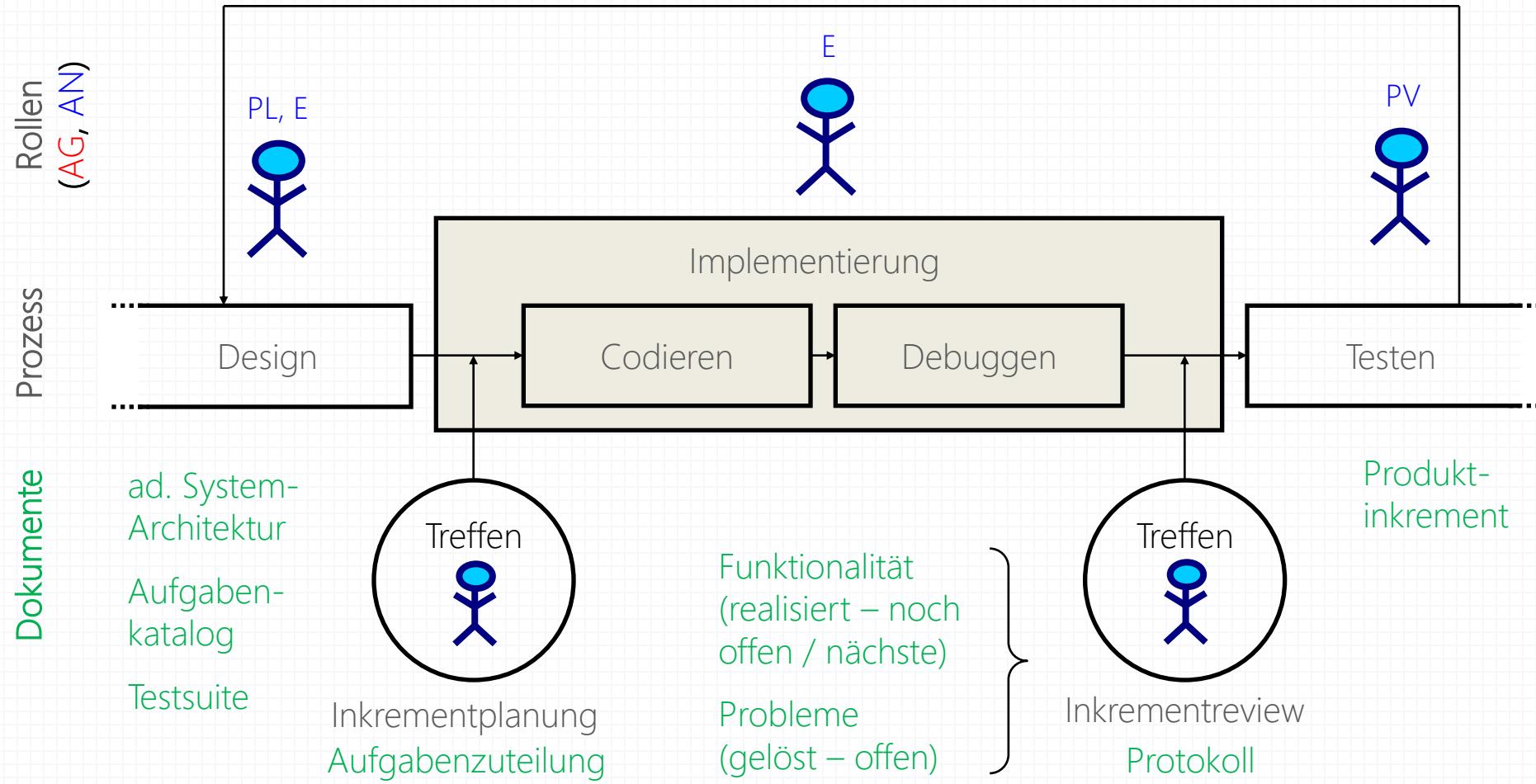


Einbettung in den Entwicklungszyklus (II)



Einbettung in den Entwicklungszyklus (III)

Inkrement (z.B. 1 Tag)



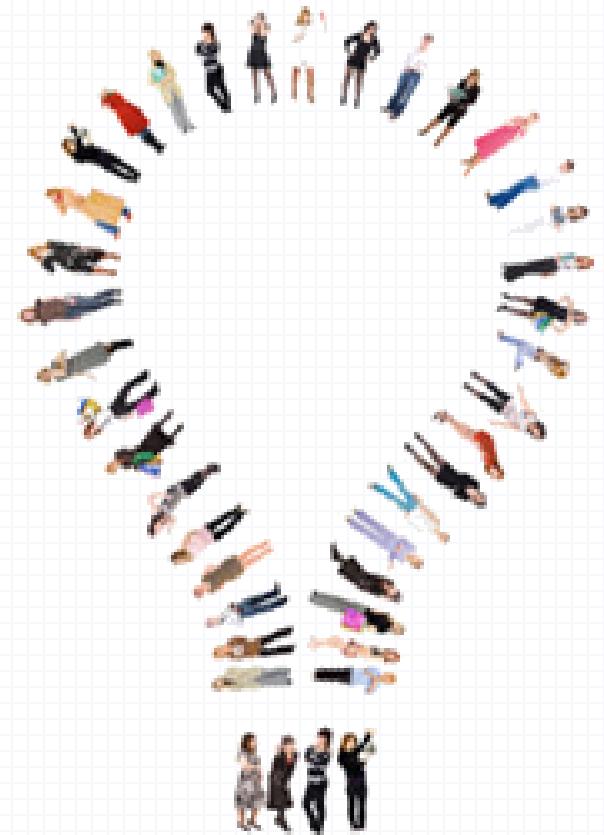
Moderne („Agile“) Softwareentwicklung

Positive Charakteristika:

- + enge Kundeneinbindung
- + iterativ-inkrementelle Entwicklung
- + effiziente Werkzeugnutzung

Negative Charakteristika:

- weniger erfahrene Entwickler (heterogene Teams)
- Aversion gegenüber Management-Information
- kaum Prozessverständnis, geschweige denn -verbesserung



Agile Softwareentwicklung: Anlass

Software-Prozessmodelle wurden in den Neunziger Jahren immer umfangreicher!

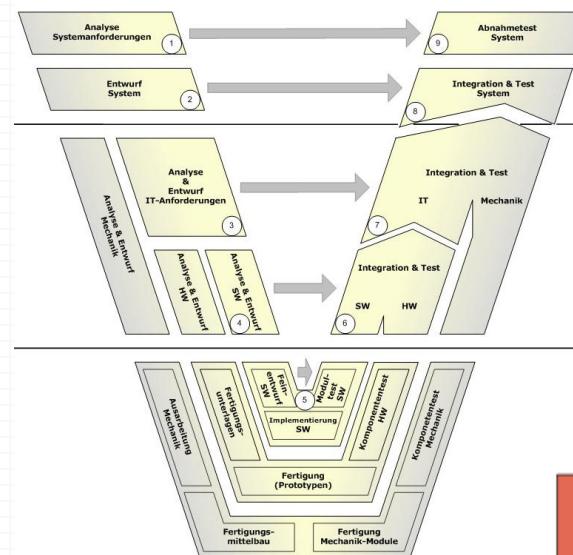


(Grafik: <http://oeag.test.oeag.at>)

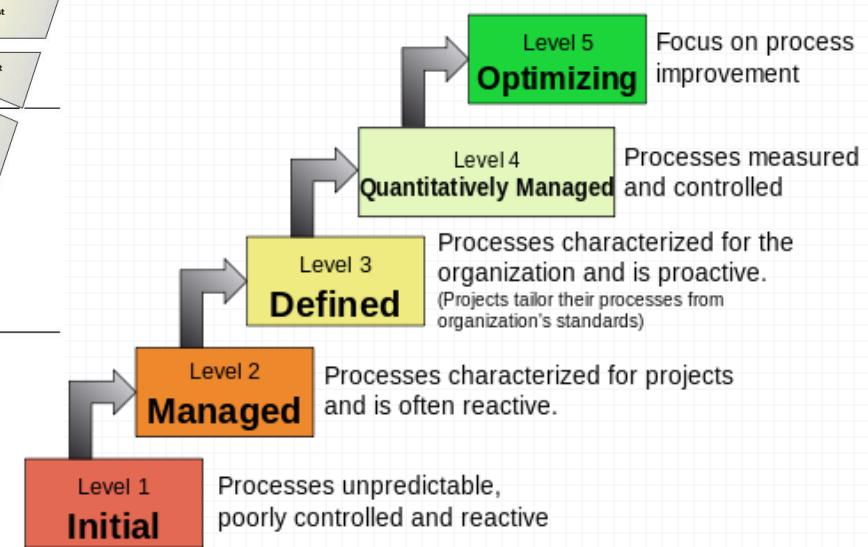
⇒ „Dokumentationsmodelle“

z.B.

- in Deutschland „V-Modell“
- in USA „SW-CMM“



(Grafik: TU München)



(Grafik: Wikipedia)

Agile Softwareentwicklung: Auslöser

Agiles Manifest (nach Beck *et al.*, 2001):

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over *processes and tools*,
Working software over *comprehensive documentation*,
Customer collaboration over *contract negotiation*,
Responding to change over *following a plan*.

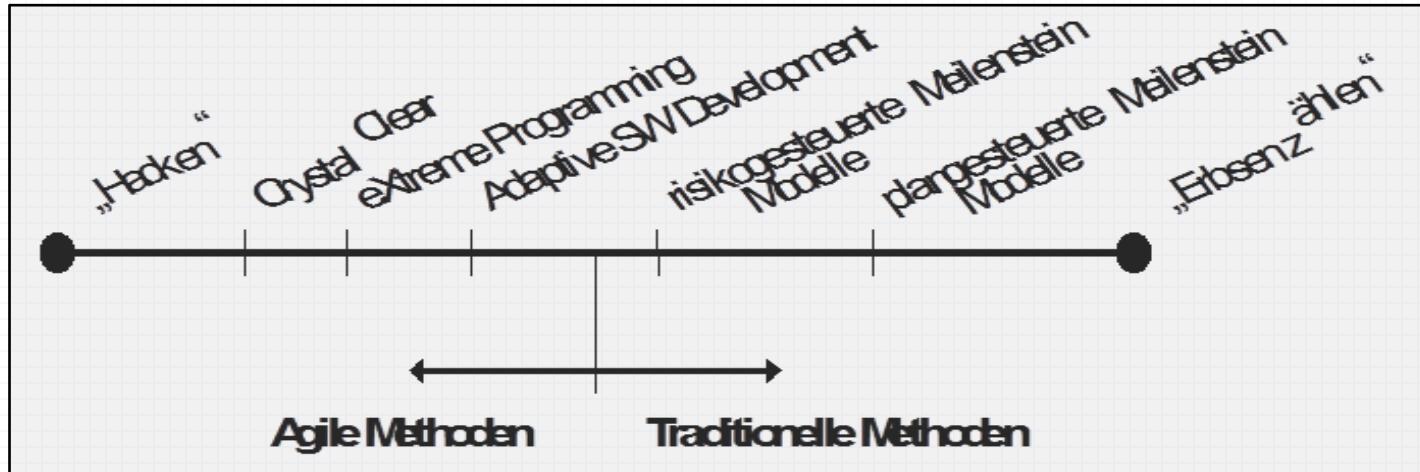
That is, while there is value in the items on the right, we value the items on the left more.



(Grafik: Snowbird, sandy.utah.gov)

Agile Vorgehensmethoden

Bandbreite des Methodenspektrums:



Paradigmenwechsel durch agile Methoden:

heute: 3-6 monate planen ist top. Früher Jahre geplant

- einheitlicher Entwicklungsprozess nicht definierbar
- Projekte nicht langfristig detailliert planbar
- Projektkontrolle ist ohne Entwicklerkooperation unmöglich

Vorteile agiler Vorgehensmethoden

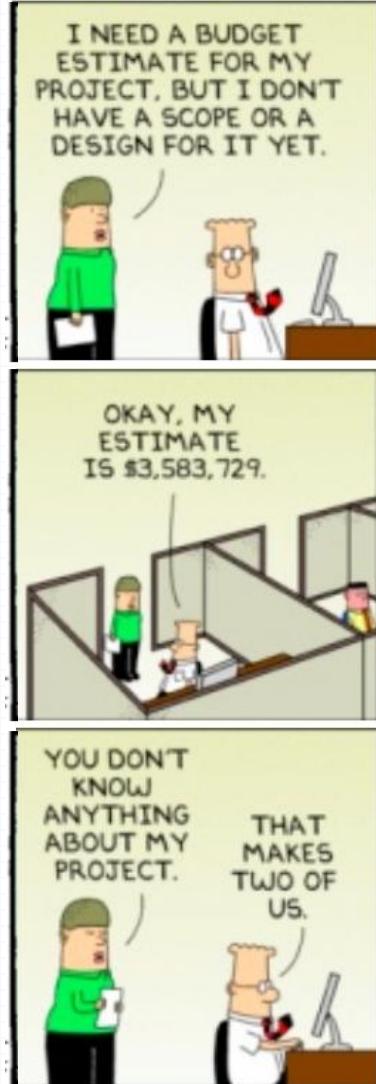
- + iterativ-inkrementelle Entwicklung
- + kurze Releasezyklen
- + überprüfbare Qualität
- + intensive Kundeneinbindung
- + intensive Kommunikation im Team
- + Anwendung von Programmierstandards
- + einfaches Design
- + laufende (Code-)Reviews
- + testgesteuerte Entwicklung
- + kontinuierliche Integration



(Bild: www.dilbert.com)

Nachteile agiler Vorgehensmethoden

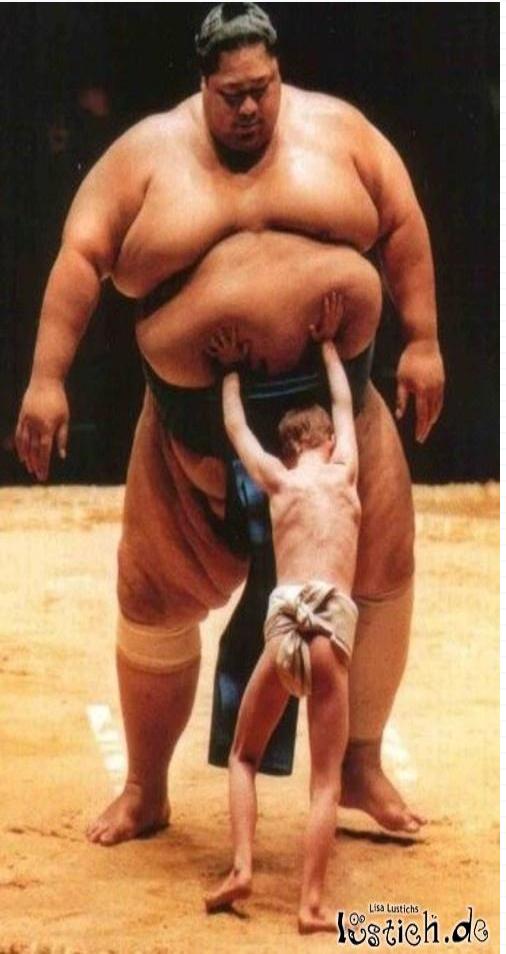
- nur hoch qualifizierte Mitarbeiter brauchbar
- schlechte Skalierbarkeit
- mangelnde Transparenz für das Management
- keine Migrationsmöglichkeit aus bestehenden Prozessen
- fehlender Qualitätsplan
- kontinuierliche Anforderungsänderungen
- keine Design-Reviews
- stark inkrementelles Design, fehlende Dokumentation dazu
- Code-Zentriertheit statt Design-Zentriertheit
- fehlende Quellen für Systemtests



(Bild: www.dilbert.com)

Größtes Problem: **fehlendes „Big-Picture“-Design (Architektur!)**

Folgen des Agilen Manifests (I)



(Bild: www.marconetz.de)

Prozesskrieg

traditionell

schwergewichtig

rigoros

stabil



modern

leichtgewichtig

agil

fragil

„If you want to start a religious or software war,
issue an edict or a manifesto“ (K. Orr, Cutter 2003)

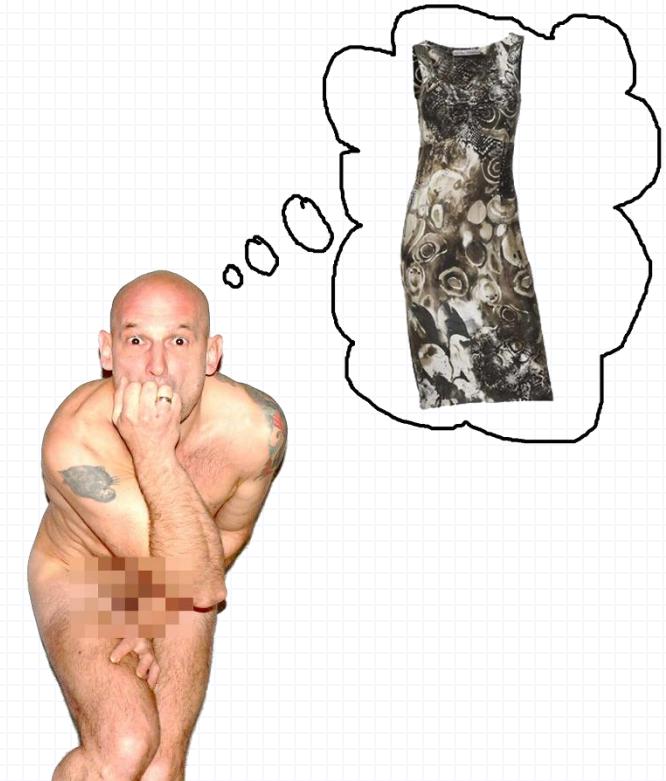
Folgen des Agilen Manifests (II)

„Des Kaisers neue Kleider“

⇒ „Dokumentationsmodelle“ werden zu skalierbaren (Dokumentations-)Modellen.

z.B.

- in Deutschland vom „V-Modell“ zum „V-Modell XT“ (2005)
- in USA vom „SW-CMM“ zum „CMMI“ (2002)



(Bild: <http://freie-zeit.eu/2012/07/24/des-kaisers-neue-kleider/>)

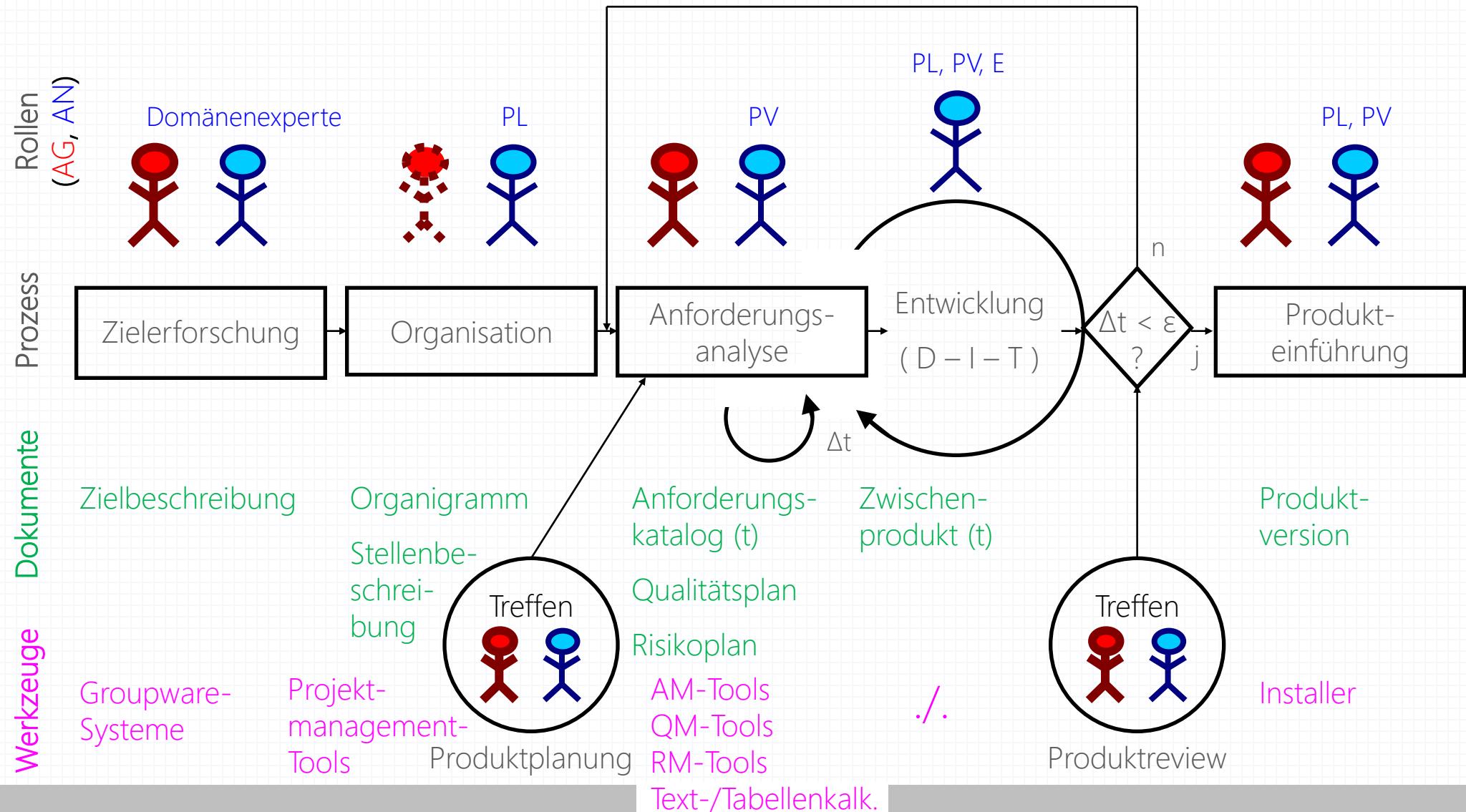
Dokumente & Werkzeuge in agilen Prozessmodellen

Erkenntnisse:

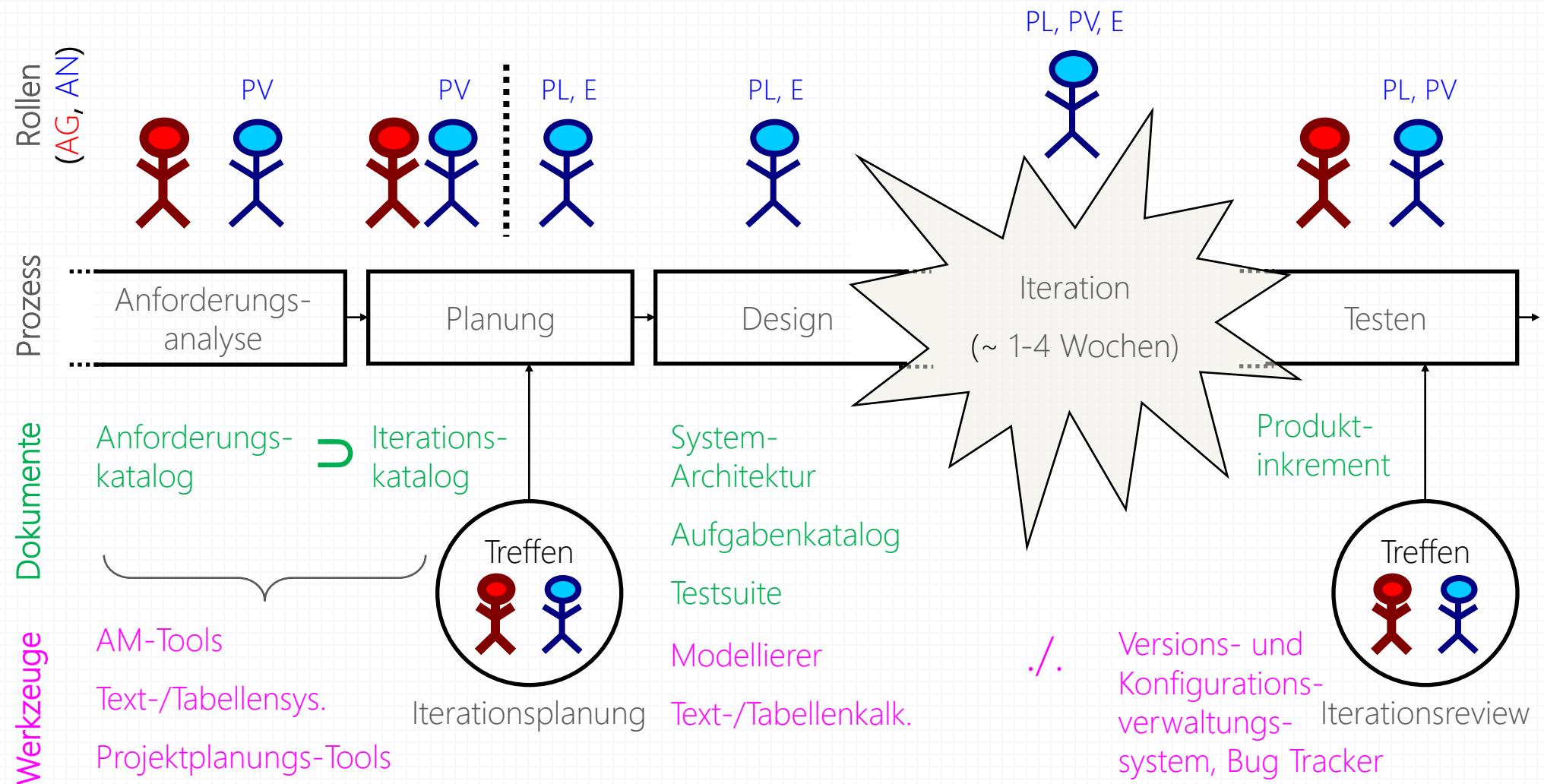
- Auch agile Methoden erzeugen eine **Anzahl** von Dokumenten.
- Dokumente müssen **kompakt** und mit **wenig** Aufwand zu erstellen sein.
- Agile Methoden benötigen häufig **mehr** Werkzeugunterstützung als traditionelle Methoden.
- Werkzeuge müssen **einfach** erlernbar und **rasch** handhabbar sein.

Ein gutes Werkzeug soll den Prozess unterstützen,
nicht vorgeben!

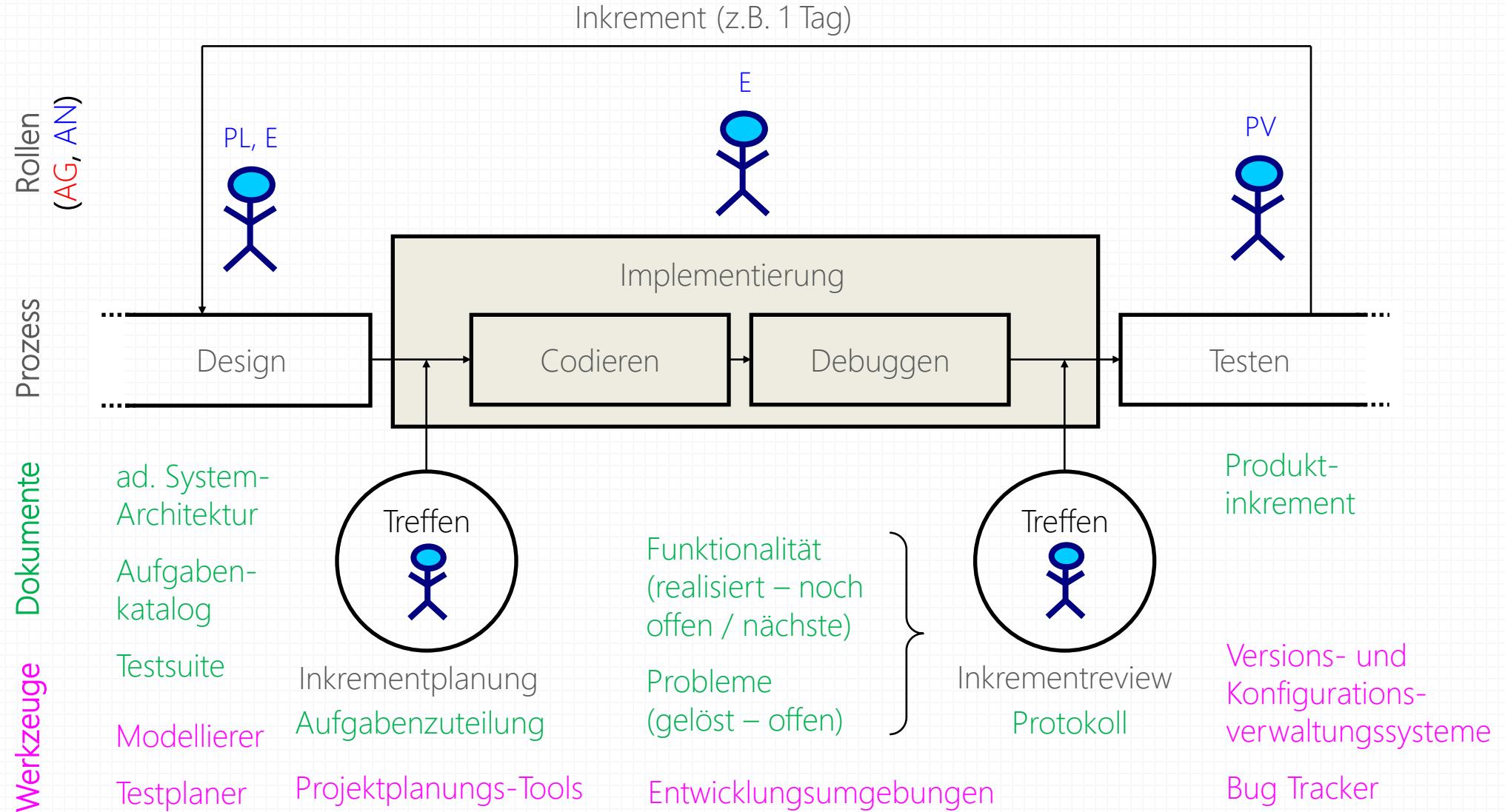
Werkzeugunterstützung im agilen Projekt Engineering-Zyklus (I)



Werkzeugunterstützung im agilen Projekt Engineering-Zyklus (II)



Werkzeugunterstützung im agilen Projekt Engineering-Zyklus (III)



Prozessoptimierung

Prozessoptimierung

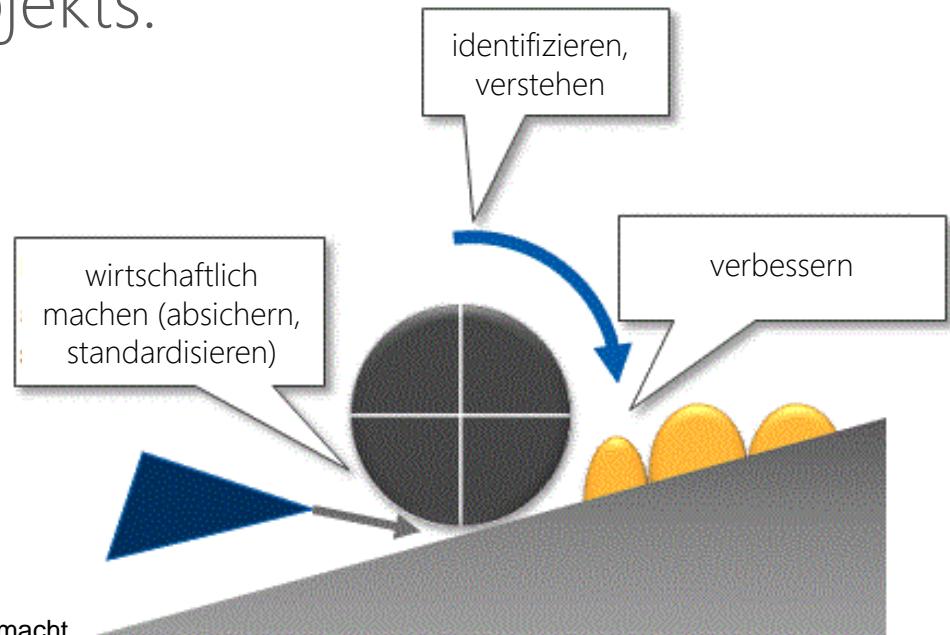
Motivation: Definierte (Teil-)Prozesse müssen laufend angepasst und verbessert werden, damit sich der Aufwand amortisiert!

Vor allem bei agilen Methoden Änderungen nicht nur von Projekt zu Projekt, sondern auch während eines Projekts.

Vier wichtige Schritte:

1. Prozess identifizieren,
2. Prozess verstehen,
3. Prozess wirtschaftlich machen,
4. Prozess verbessern.

wichtig ist die Kosten-Nutzen-"Abrechnung"; Entscheidungen treffen, ob man etwas nicht macht



(Bild: www.ingenieurbuero-wilkens.de; adaptiert)

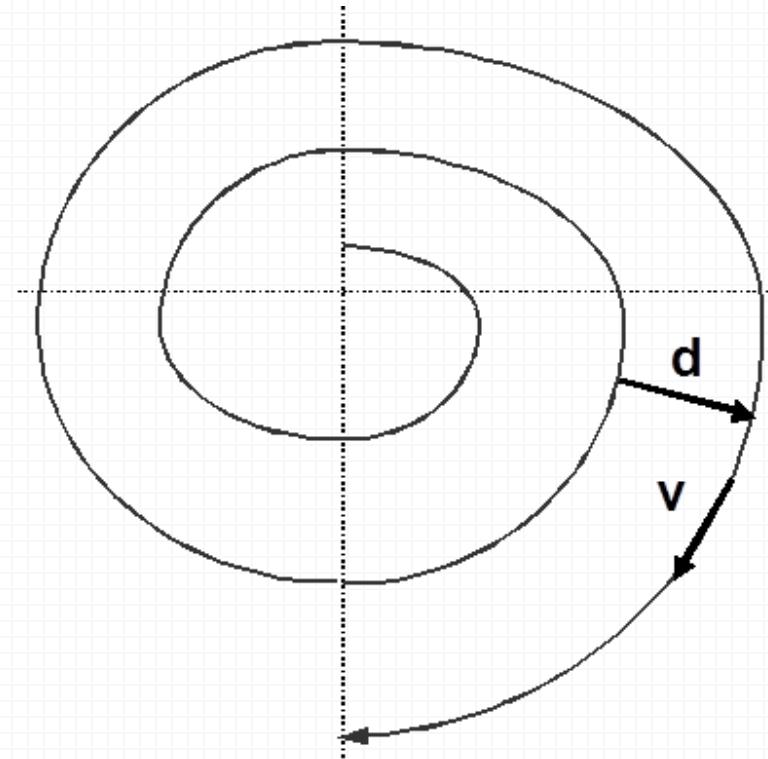
Prozessidentifikation

Bedeutung des Messens

Zwei Aspekte:

- Messen des **Projektfortschritts**
- Messen des Grades der Zielerreichung

agil: Kundenwünsche ändern sich dauernd



*traditionelle Methoden:
agile Methoden:*

**v klein, d groß
d klein, v groß**

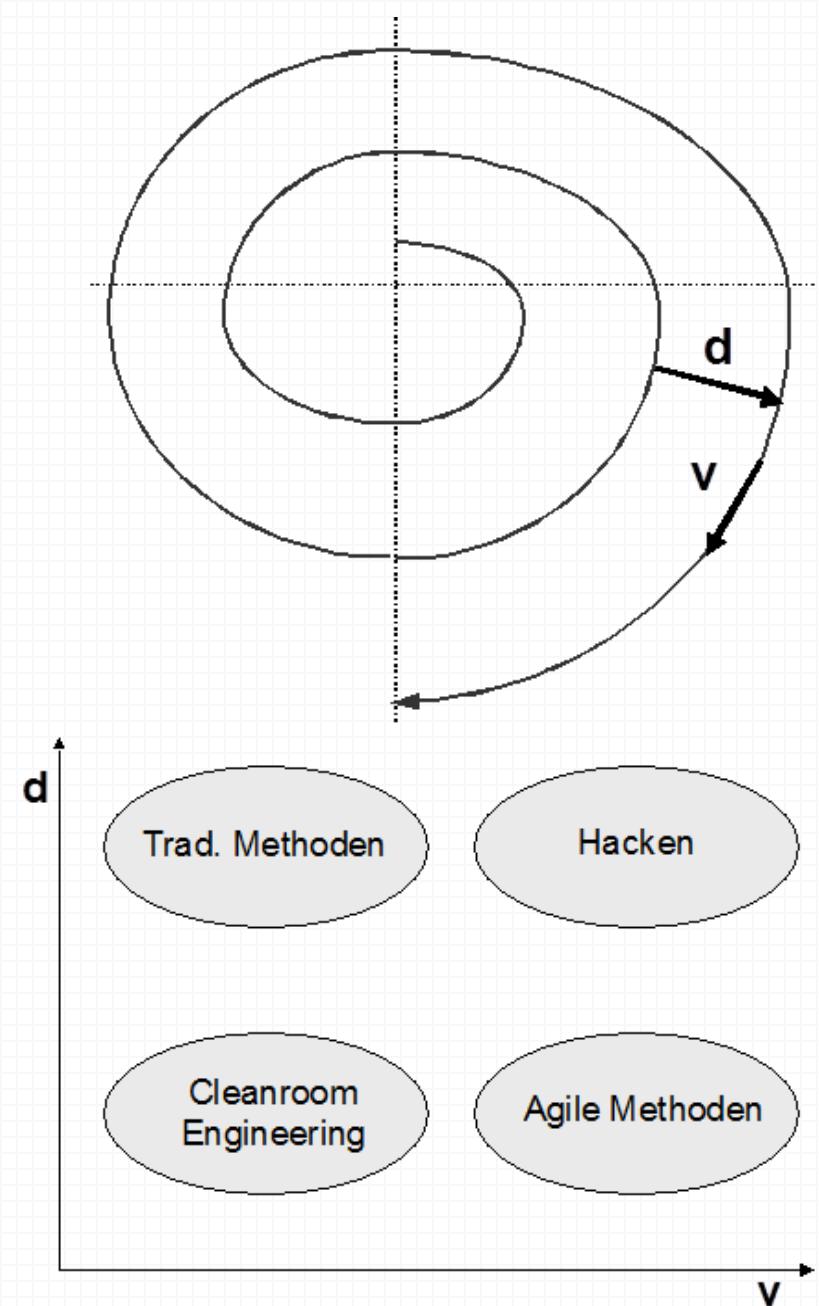
Gerade bei agilem Vorgehen Anvisieren eines beweglichen Ziels – Fortbewegung in raschen, kurzen Schritten nötig!

Prozessverständnis

Regelmäßiges Feedback durch
iterativ-inkrementelles Vorgehen:

- kurze Iterationszyklen
- häufiges Feedback vom Auftraggeber
- genauere Ermittlung des Projektstatus
- rasche Adaptierbarkeit

Durch laufende Validierung auch klare
Abgrenzung vom Hackertum!



Prozessökonomie: Traditionelles Vorgehen

Traditionelle Vorgehensmethoden:

- Vorhersehbarkeit
- Wiederholbarkeit
- Optimierbarkeit
- Streben nach vollständigen Anforderungen
- Plan- und Modell-fokussiert

Risiken:

- Anforderungen ändern sich.
- Aktualisierung der Pläne und Modelle ist sehr aufwändig.



(Bild: hdwallpapersinn.com)

Prozessökonomie: Agiles Vorgehen

Agile Vorgehensmethoden:

(AgileAlliance.org, adaptiert durch J. Coldewey)



Agiles Manifest

„*Kollaboration* statt *Formalismus*,
*kurze Inkreme*nten statt *jahrelanger Meditation*,
Flexibilität statt *Planungsorgien*,
Einbindung des Kunden statt *Absicherung*.“

- Reduktion von Plänen und Modellen auf das Mindestmaß
- (ursprünglich) keine eigene explizite Designphase
- (ursprünglich) keine Prozessverbesserung

Prozessverbesserung

Gemeinsamkeiten aktueller Vorgehensmethoden:

- Projektziel ist **Vision**, die im Auge behalten werden muss.
- **Kunde** ist laufend eingebunden.
- Realisierung wird nach Festlegung minimaler **Mindestanforderungen** begonnen.
- **Testfälle** werden bereits mit der Anforderung erstellt (vor der Implementierung!).
- Entwicklung erfolgt **iterativ-inkrementell** mit kurzen Zyklen.
- Laufende **Risikobehandlung** ist wichtig.
- Nur für Auftraggeber oder Auftragnehmer **essenzielle Dokumente** werden erstellt.
- Anforderungsänderungen sind alltäglich und eingeplant.
- **Pläne** werden laufend geprüft und **adaptiert**.
- Design ist **einfach**, aber leicht erweiterbar.
- Tests werden möglichst **automatisiert** durchgeführt.



(Bild: hundeschule-pfotenteam.de)

PROJEKT ENGINEERING

Ablauforganisation

Herwig Mayr

Fakultät für Informatik, Kommunikation und Medien
Fachhochschule OÖ, Hagenberg