

# **SOFTWARE ENGINEERING II**

**Vorlesung**

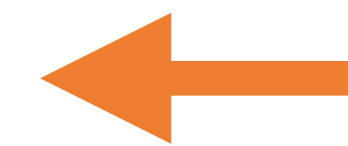
**SS 2021**

**Einführung  
Softwareentwicklungsprozesse I**



# INHALTE



<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# INHALTE

	<b>0</b>	<b>Einführung</b>	
	0.1	Definition Software Engineering	
	0.2	FAQs	
	<b>1</b>	<b>Software Entwicklungsprozesse</b>	
	1.1	Herausforderungen und Aktivitäten	
	1.2	Sequentielle Prozesse	
	1.3	Iterative Prozesse	
	1.4	Agile Prozesse	
	1.4.1	<i>Scrum</i>	
	1.4.2	<i>Kanban</i>	

# LITERATUR



**[1] Software Engineering  
Ian Sommerville  
10. Auflage**

# SOFTWARE ENGINEERING

Software engineering is an **engineering discipline** that is concerned with **all aspects of software production** from the early stages of system specification through to maintaining the system after it has gone into use. [1]

### ► Technische Disziplin

- **Bewusste** Selektion von **Theorien, Methoden, Werkzeugen** ■
- **neue Lösungen**, auch unter organisatorischen und finanziellen Einschränkungen

### ► Alle Aspekte

- technischen Aspekte
- Projektmanagement ■
- Entwicklung von Werkzeugen, Methoden und Theorien zur Unterstützung der professionellen Software Entwicklung.



# SOFTWARE ENGINEERING

Software engineering is an **engineering discipline** that is concerned with **all aspects of software production** from the early stages of system specification through to maintaining the system after it has gone into use. [1]

### ► Warum brauchen wir Software Engineering?

- Um zuverlässige und vertrauenswürdige Systeme wirtschaftlich und schnell herzustellen. ■

### ► Unterschied zur Informatik?

- Theorien und Methoden der SW Systeme vs. Probleme der SW Herstellung ■

### ► Unterschied von Software Projekten auf der Uni vs. im „echten“ Leben? ■

# HACKEN VS. SOFTWARE ENGINEERING

Personal Software	Industrial-strength Software
Developer is user	Client is user
Bugs are tolerable	<b>Bugs are not tolerated</b>
UI not important	UI important
No/Minor documentation	<b>Lots of documentation</b>
SW not in critical use	Supports business functions
Reliability/Robustness not crucial	Reliability/Robustness crucial
No investment	Heavy investments (5\$-25\$ per LOC)
Portability not so important	Portability is economic advantage

# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# FAQs

- 1. Was ist das Besondere an Software?**
- 2. Wozu braucht man Software Engineering?**
- 3. Was sind Merkmale guter Software?**
- 4. Was sind die Herausforderungen im Software Engineering?**



# 1. BESONDERHEITEN

## SW ...

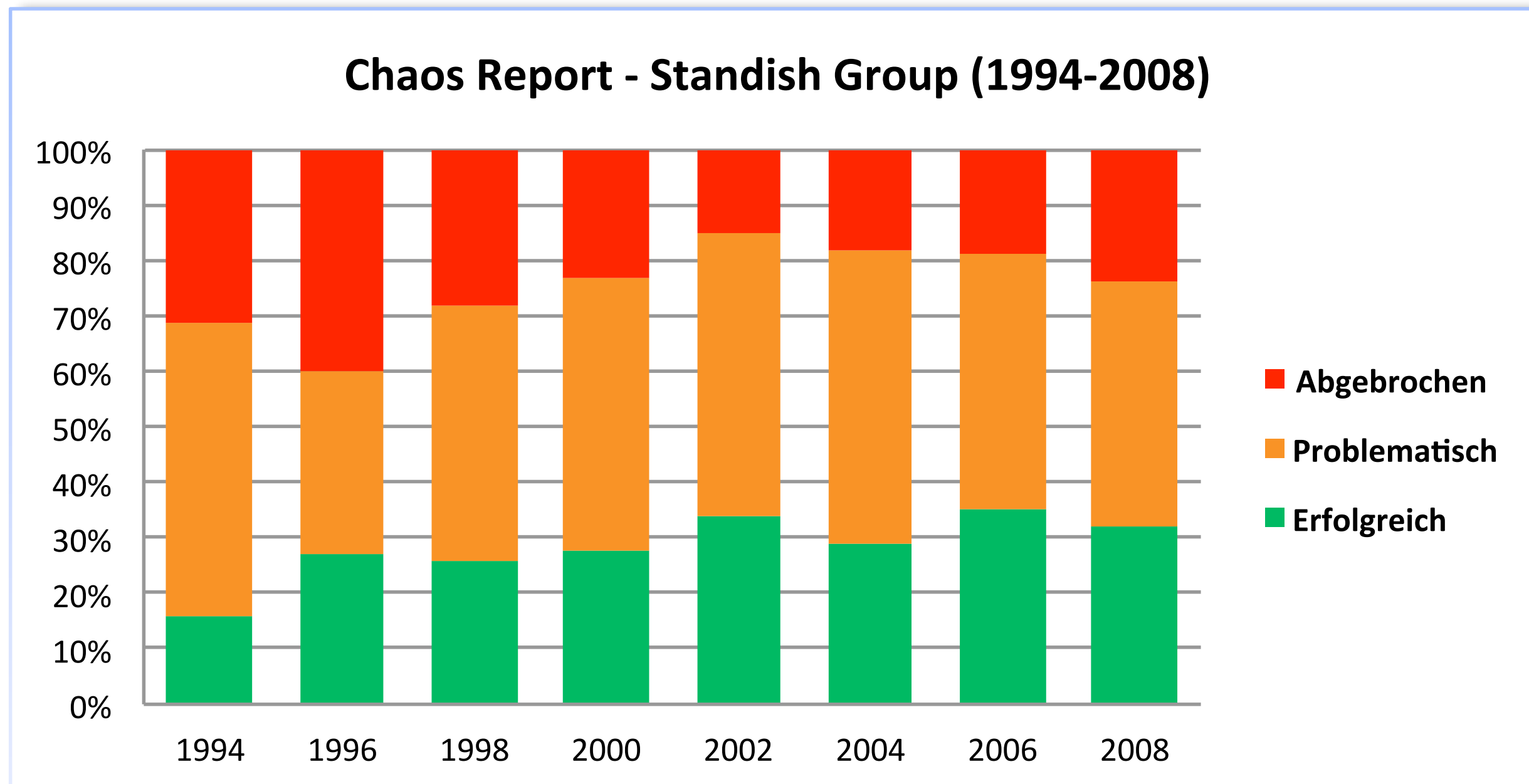
- ▶ ... ist **unsichtbar**
- ▶ ... verbraucht **keinen Platz**
- ▶ ... hat **keine physikalische Substanz**
- ▶ ... riecht nicht
- ▶ ... hat kein Aussehen

ABER

## SW ...

- ▶ ist **risikobehaftet**
- ▶ benötigt **Platz und Zeit** (Entwicklung)
- ▶ lebt länger, als man denkt
- ▶ kann **großen Schaden** verursachen

## 2. WOZU SOFTWARE ENGINEERING?



- ▶ Kaliforniens DMV. Erneuerung der Motorzulassung. Projektstart 1987. 1993 nach **\$45 Millionen** Dollar gestoppt.
- ▶ American Airlines. Autoverleih und Hotelreservierungssystem. 1994 nach **\$145 Millionen** Dollar gestoppt.

## 2. WOZU SOFTWARE ENGINEERING?



**Terac-25 Strahlentherapie, 1985 - 1987**



## 2. WOZU SOFTWARE ENGINEERING?



**Terac-25 Strahlentherapie, 1985 - 1987**



**Ariane 5, 1996**



## 2. WOZU SOFTWARE ENGINEERING?



**Terac-25 Strahlentherapie, 1985 - 1987**



**Ariane 5, 1996**



**Heathrow, 2018**



# 3. MERKMALE GUTER SOFTWARE

Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable. [1]

**1. Wartbarkeit**



**2. Sicherheit und  
Zuverlässigkeit**



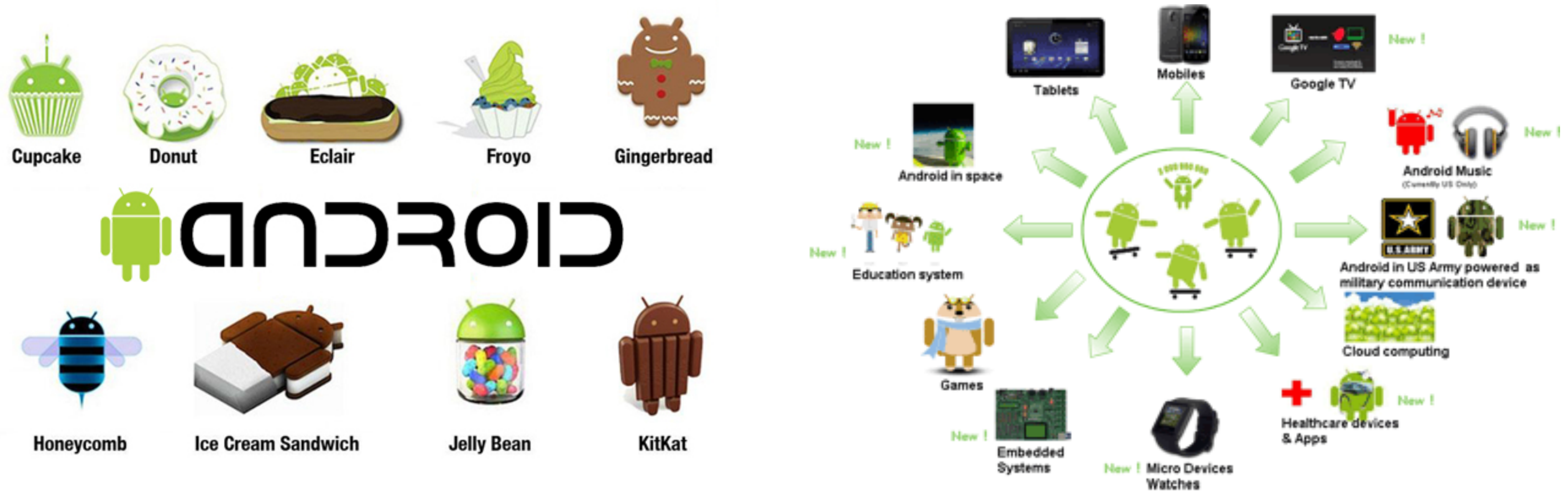
**3. Effizienz**



**4. Benutzerfreundlichkeit  
(Acceptability)**



## 4. HERAUSFORDERUNGEN IM SE (1)



# Heterogenität



## 4. HERAUSFORDERUNGEN IM SE (2)



**neue Technologien,  
wirtschaftlicher und sozialer Wandel**

## 4. HERAUSFORDERUNGEN IM SE (3)



# Sicherheit und Zuverlässigkeit



## ZUSAMMENFASSUNG

### Was sind die Merkmale guter Software?



1. Wartbarkeit



2. Effizienz



3. Sicherheit



4. Acceptability

### Was sind die Herausforderungen im Software Engineering?



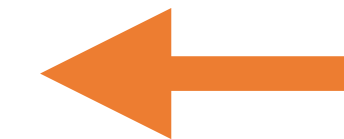
1. Heterogenität    2. neue Technologien wirtschaftl./soz. Wandel    3. Sicherheit und Zuverlässigkeit

### Wozu braucht man Software Engineering?

um **langlebige, qualitätsvolle SW Systeme effizient entwickeln** zu können

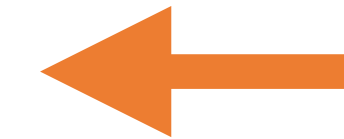
# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# HERAUSFORDERUNGEN (1/2) ■

## Unterschiedliche Dimensionen der Komplexität

- ▶ Eine Aufgabe **selbst zum ersten Mal** erledigen
- ▶ Eine Aufgabe **überhaupt zum ersten Mal** erledigen
- ▶ Wenn das **Problemfeld unklar** ist
- ▶ Wenn die **Problemlösung umfangreich** ist
- ▶ Wenn das **Problem facettenreich** ist
- ▶ Wenn die **Facetten des Problems ineinandergreifen** (und damit evtl. Teillösungen ineinandergreifen)

## Möglichkeiten, um komplexe Aufgaben zu lösen

- ▶ Einen Weg ausprobieren und sehen, ob er funktioniert
- ▶ Jemanden anderen eine Lösung finden lassen
- ▶ Planen, Ausführen, Überprüfen und Anpassen



# HERAUSFORDERUNGEN (2/2)

## Naives Grundmodell: Code & Fix

- ▶ Schreibe ein Programm
- ▶ Finde und behebe die Fehler im Programm

### Vorteile

- + Schnell lauffähiges Programm
- + Coding und spontanes Testen sind einfach

### Nachteile

- Projekt nicht planbar
- Anforderungen werden nicht erhoben
- Keine Basis für Tests
- Schwierige und teure Wartung
- Wie wird erworbenes Wissen transferiert?
- Skalierbarkeit?



SW-Prozesse (bzw. Vorgehensmodelle) helfen!





# AKTIVITÄTEN (1/2)

Ein SW-Prozess Modell ist eine abstrakte Repräsentation einer **Menge an Aktivitäten und Leistungen**, welche zur **Erzeugung eines SW-Produkts** führen.

### Wozu?

- ▶ **Überblick** behalten
- ▶ effektives und effizientes **Arbeiten im Team**
- ▶ gemeinsame Notation, konsistente Bedeutung





# AKTIVITÄTEN (2/2)

Ein SW-Prozess Modell ist eine abstrakte Repräsentation einer **Menge an Aktivitäten und Leistungen**, welche zur **Erzeugung eines SW-Produkts** führen.

### Aktivitäten:

- **SW Spezifizierung:** Festlegung der Funktionalität und Einschränkungen
- **SW Design und Implementierung:** Umsetzung der Architektur und Entwicklung des Systems
- **SW Validierung/Verifizierung:** Überprüfung auf Erfüllung der Anforderungen des Kunden und der korrekten Funktionalität
- **SW Evolution:** Anpassungen und Erweiterungen aufgrund von veränderten Kundenwünschen

## SOFTWARE ENTWICKLUNGSPROZESSE

### 1. Sequentielle Modelle

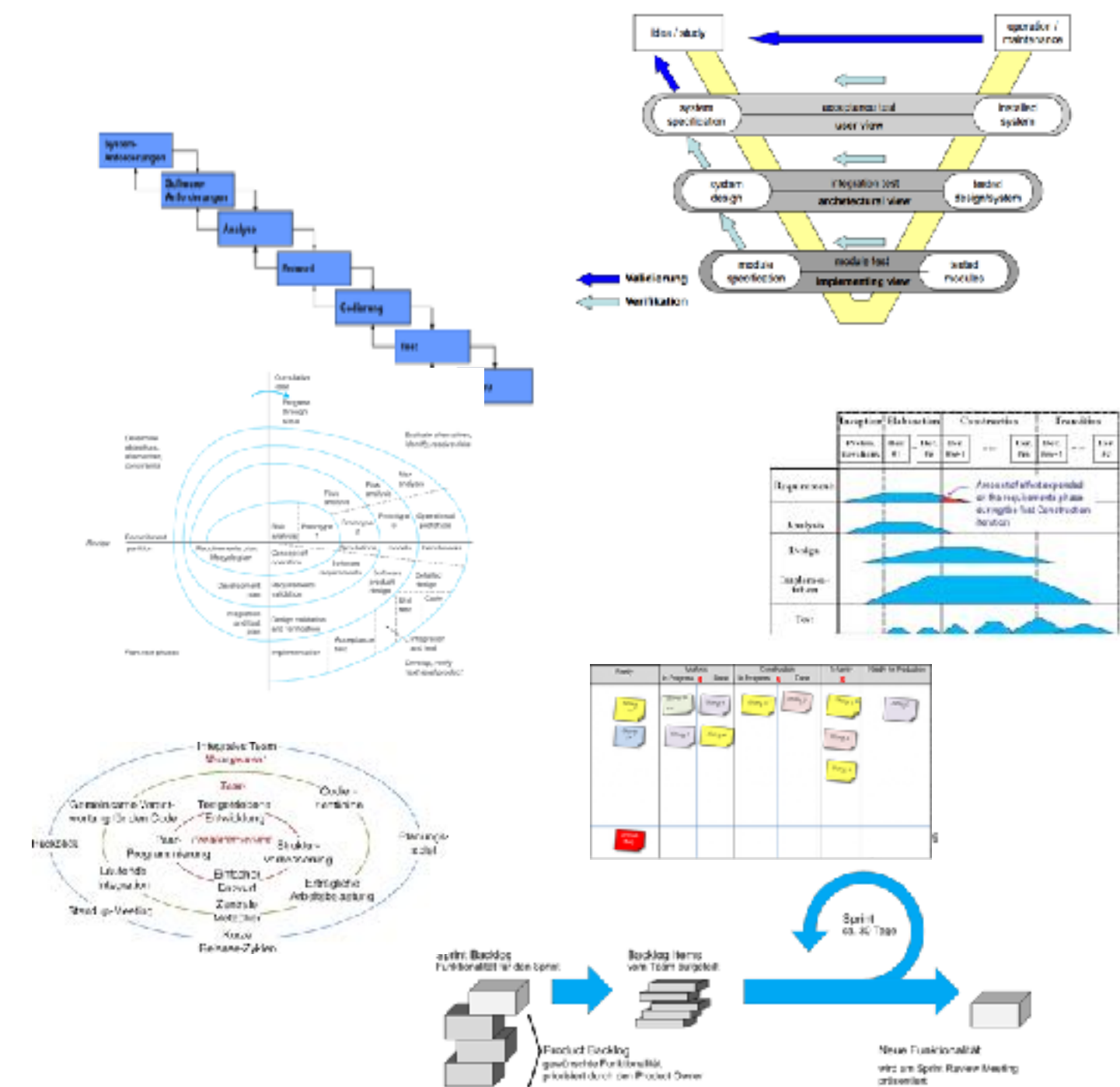
- Wasserfallmodell
- V-Modell

### 2. Iterative Modelle

- Spiralmodell
- (Rational Unified Process (RUP))

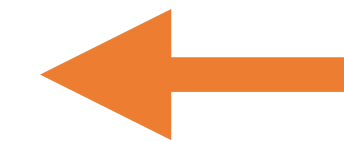
### 3. Agile Methoden

- XP
- Scrum → 10.03.2020
- Kanban → 02.04.2020
- Scaled Agile



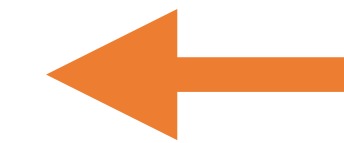
# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# INHALTE

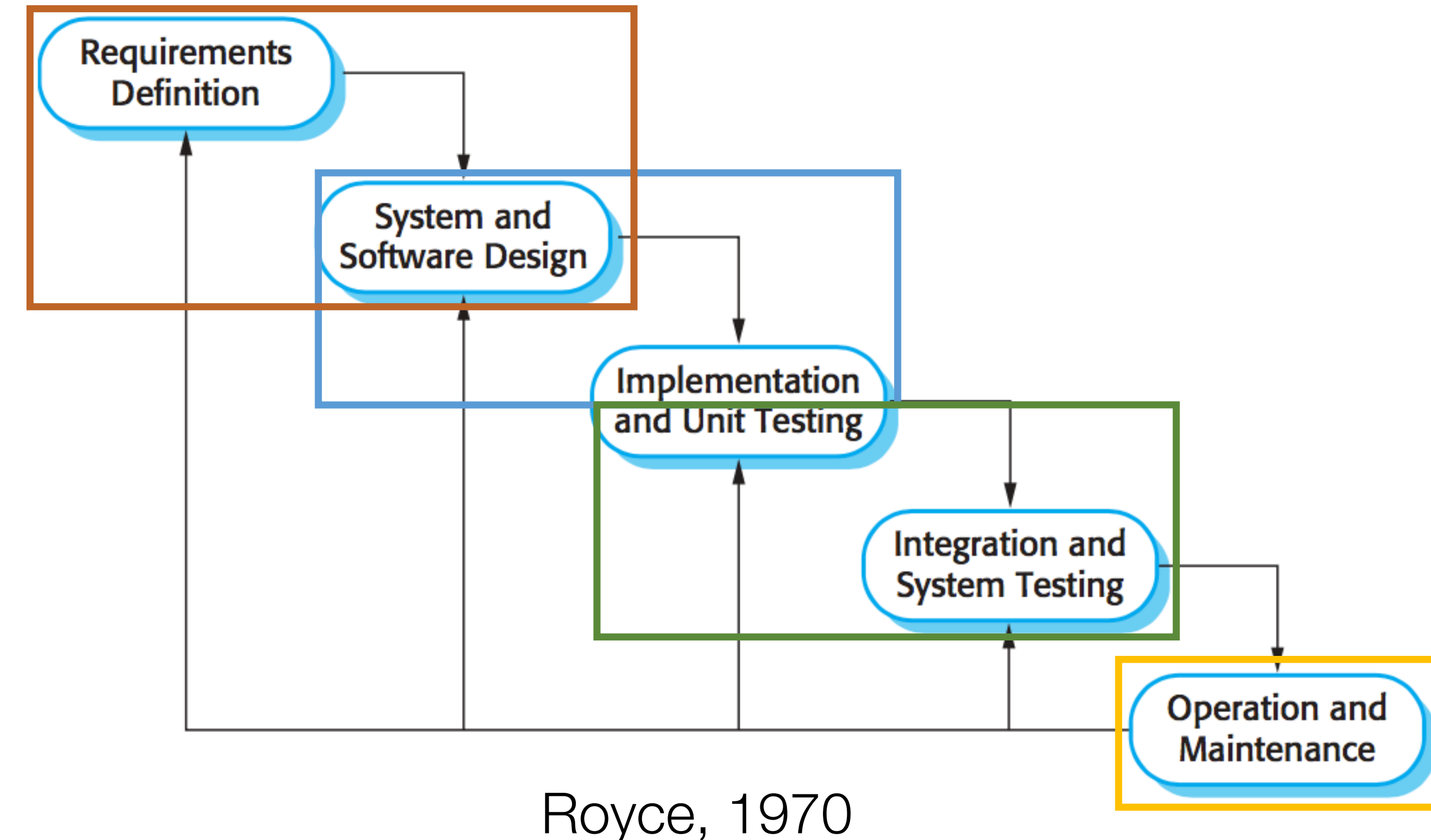
<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# DAS WASSERFALL MODELL (1)

## Charakteristika

- ▶ Top-Down
- ▶ festgelegte Reihenfolge der Aktivitäten
- ▶ eingeschränkte Rückkopplung
- ▶ **Zwischenprodukt** am Ende jeder Aktivität
- ▶ Benutzerbeteiligung nur zu Beginn
- ▶ Variante: formale Systementwicklung



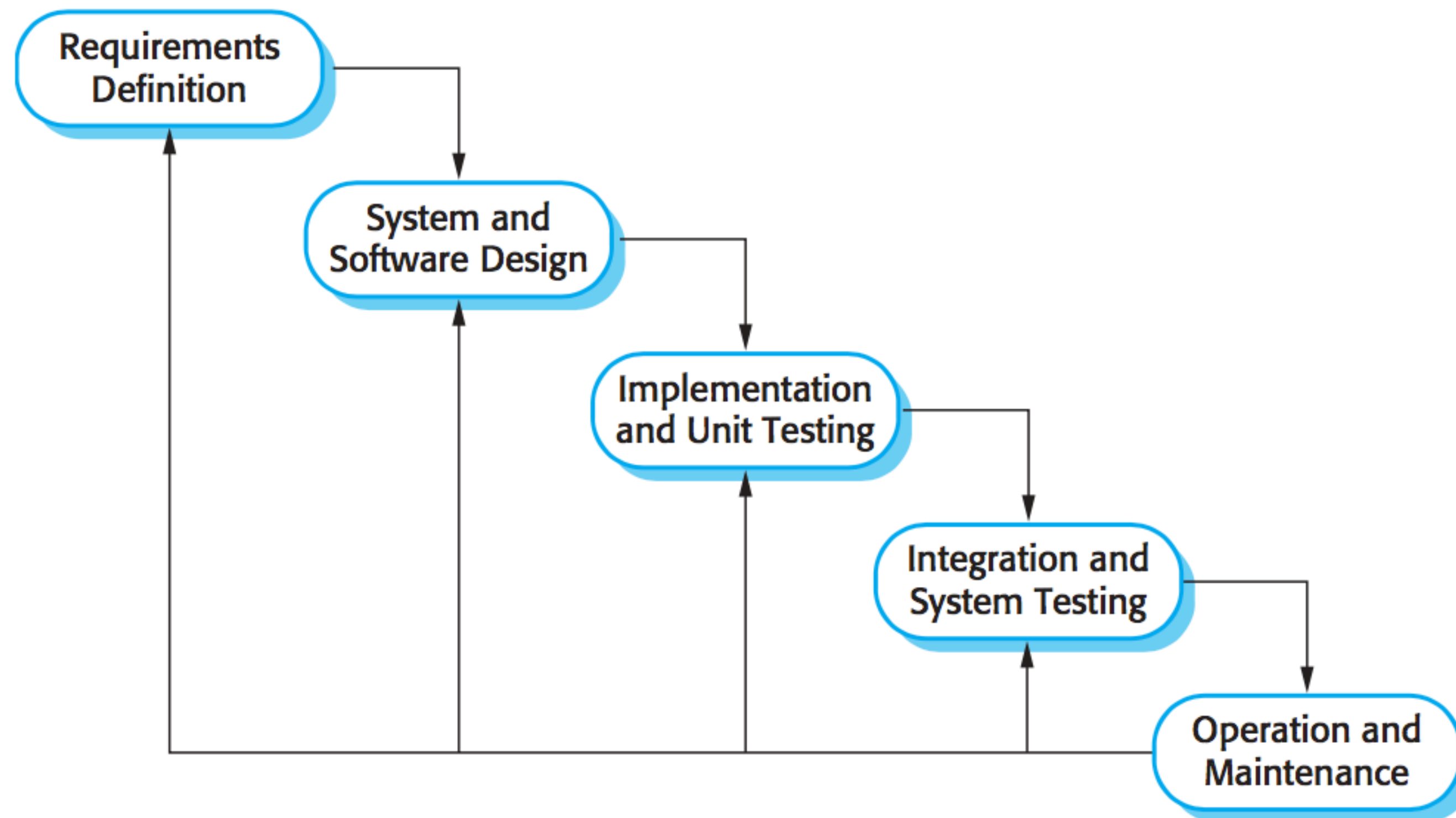
# DAS WASSERFALL MODELL (2)

## Vorteile

- + (theoretisch) gut planbar
- + geringer Managementaufwand

## Nachteile

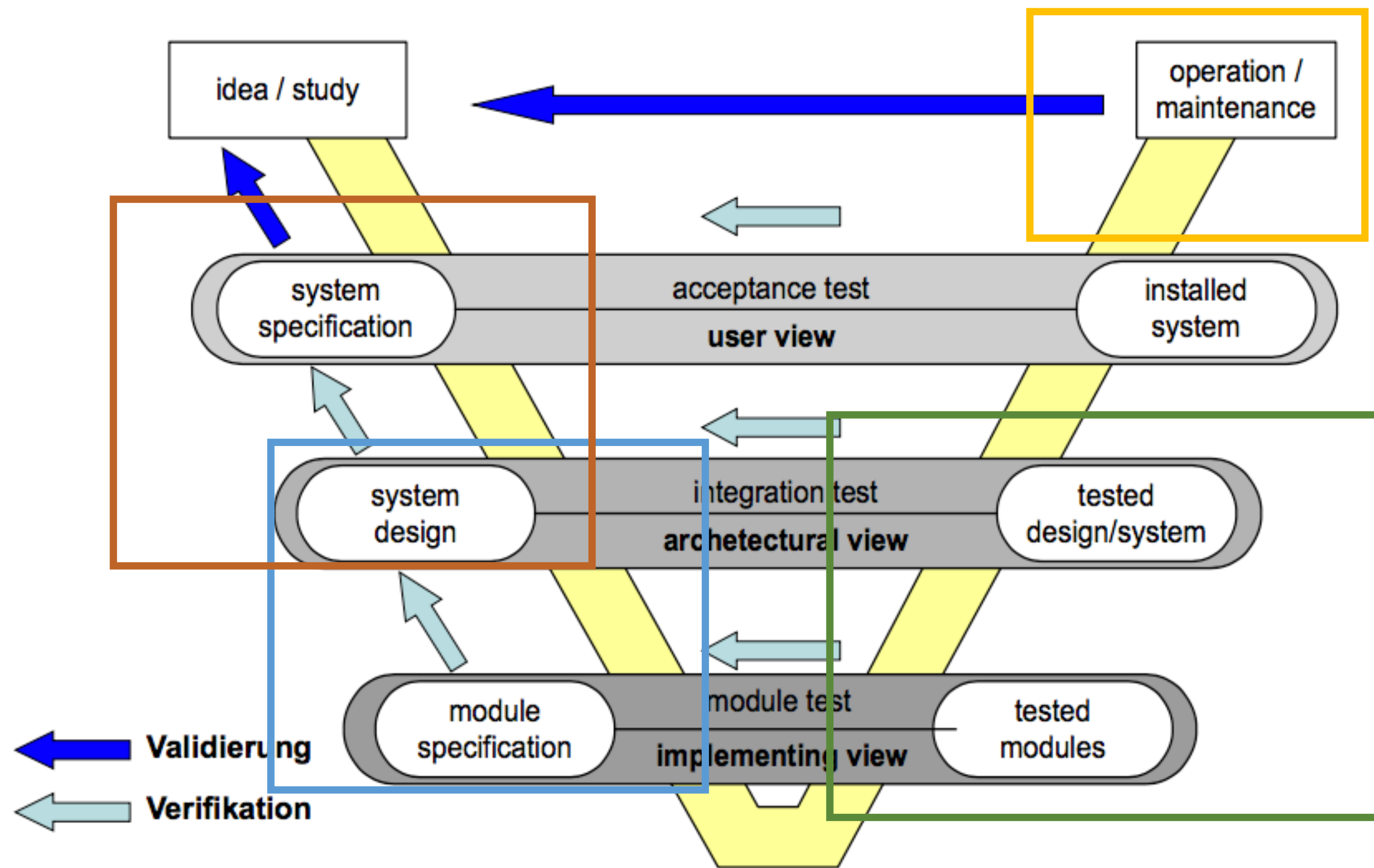
- Notwendige „Kurskorrekturen“ nicht frühzeitig erkennbar
- Sequentialität nicht immer nötig
- Gefahr, dass Dokumente wichtiger als das System werden
- Risikofaktoren werden u.U. zu wenig berücksichtigt



Royce, 1970



## DAS V-MODELL (1/3)



### Charakteristika

- ▶ Qualitätssicherung integriert
- ▶ Sehr umfangreiches Modell, wurde für große Projekte konzipiert
- ▶ Phasen, Aktivitäten, (Zwischen-Produkte)
- ▶ Verifikation und Validation
- ▶ Produkt muss mit den Anforderungen und den Kundenwünschen übereinstimmen

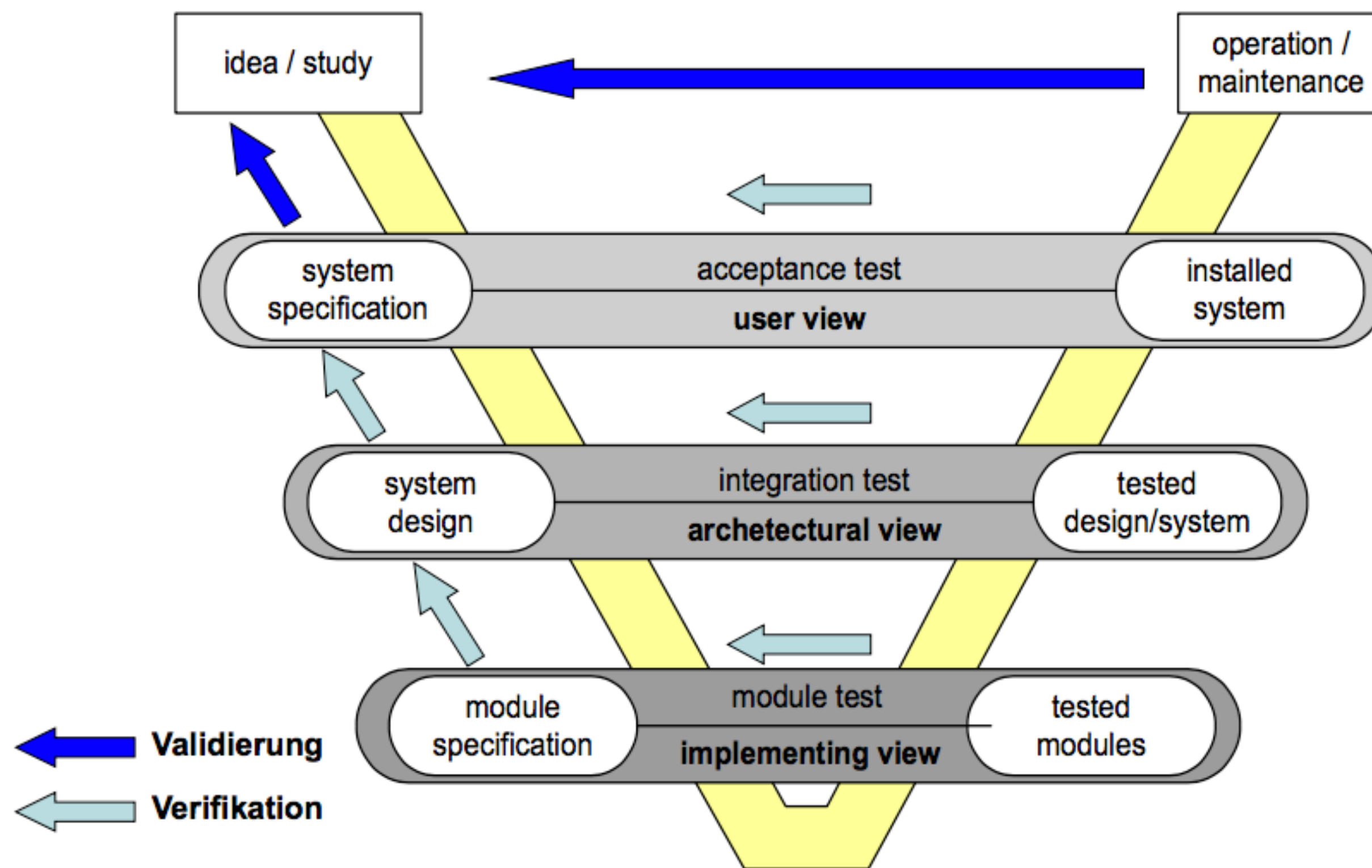
## DAS V-MODELL (2/3)

### Vorteile

- + Integrierte, detaillierte Beschreibung von Systemerstellung, Qualitätssicherung, Konfigurationsmanagement und Projektmanagement
- + Lässt sich gut anpassen und erweitern
- + Generisches Vorgehensmodell
- + Gut geeignet für große Projekte
- + Frei zugänglich und lizenzfrei

### Nachteile

- Software-Bürokratie bei kleinen & mittleren Projekten
- „Big Bang“ am Ende



[QS-Folien, TU Wien]



# DAS V-MODELL (1/3)

## Anwendung

- ▶ Gut anwendbar bei klarer, relativ fixer Funktionalität
- ▶ System- und Basissoftware, wie OS, DB, Web-Server
- ▶ Branchen-Software wie SAP R/3

## Vorteile

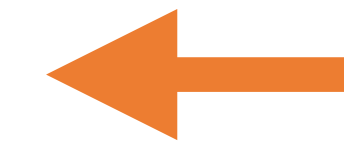
- + Einfach durchzuführen
- + Schränkt Freiheitsgrade stark ein, daher auch für sehr große Projekte anwendbar
- + Sehr effizient bei bekannten und konstanten Anforderungen

## Nachteile

- Risiken gesammelt am Schluss —> „Big Bang“
- Starr während des Ablaufs

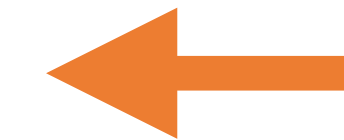
# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>

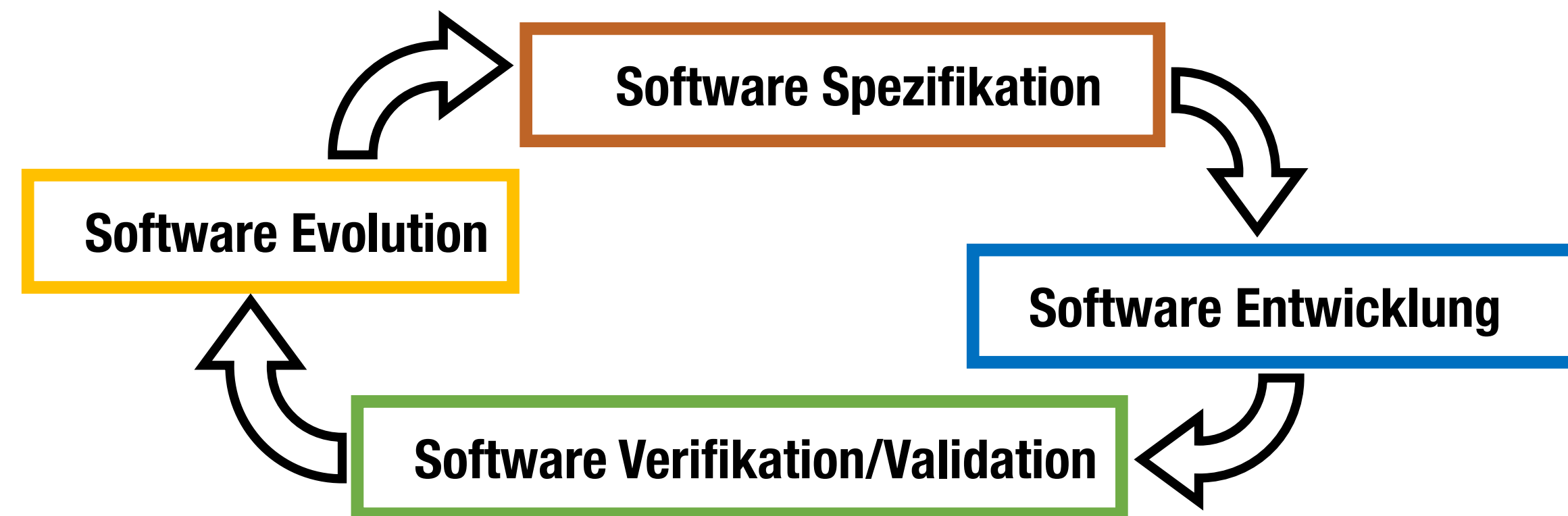


# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>

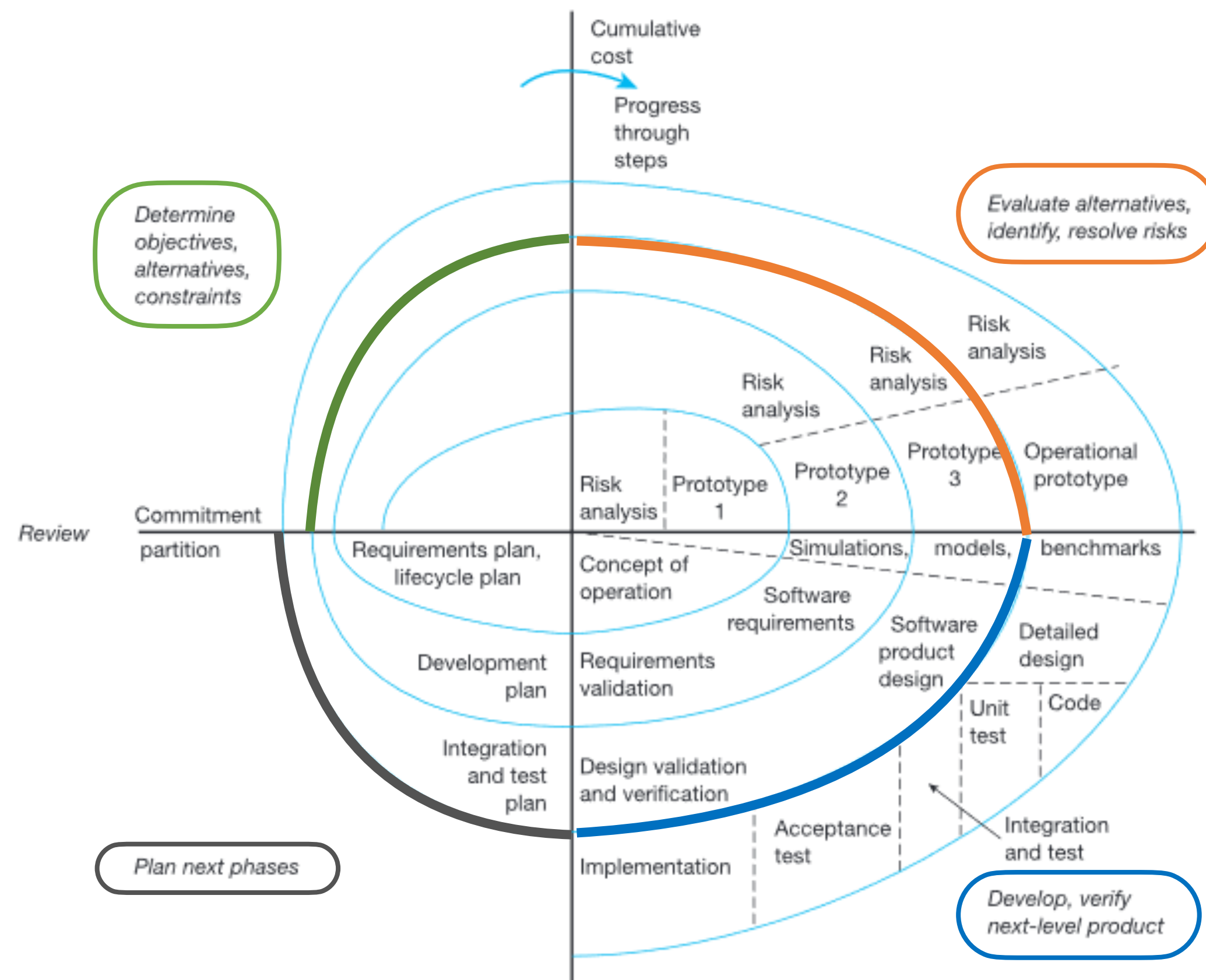


# ITERATIVE PROZESS MODELLE



➔ **Weiterentwicklung der sequentiellen Methoden  
unterstützen inkrementelle Entwicklung** ➔

# DAS SPIRAL MODELL (1/2)



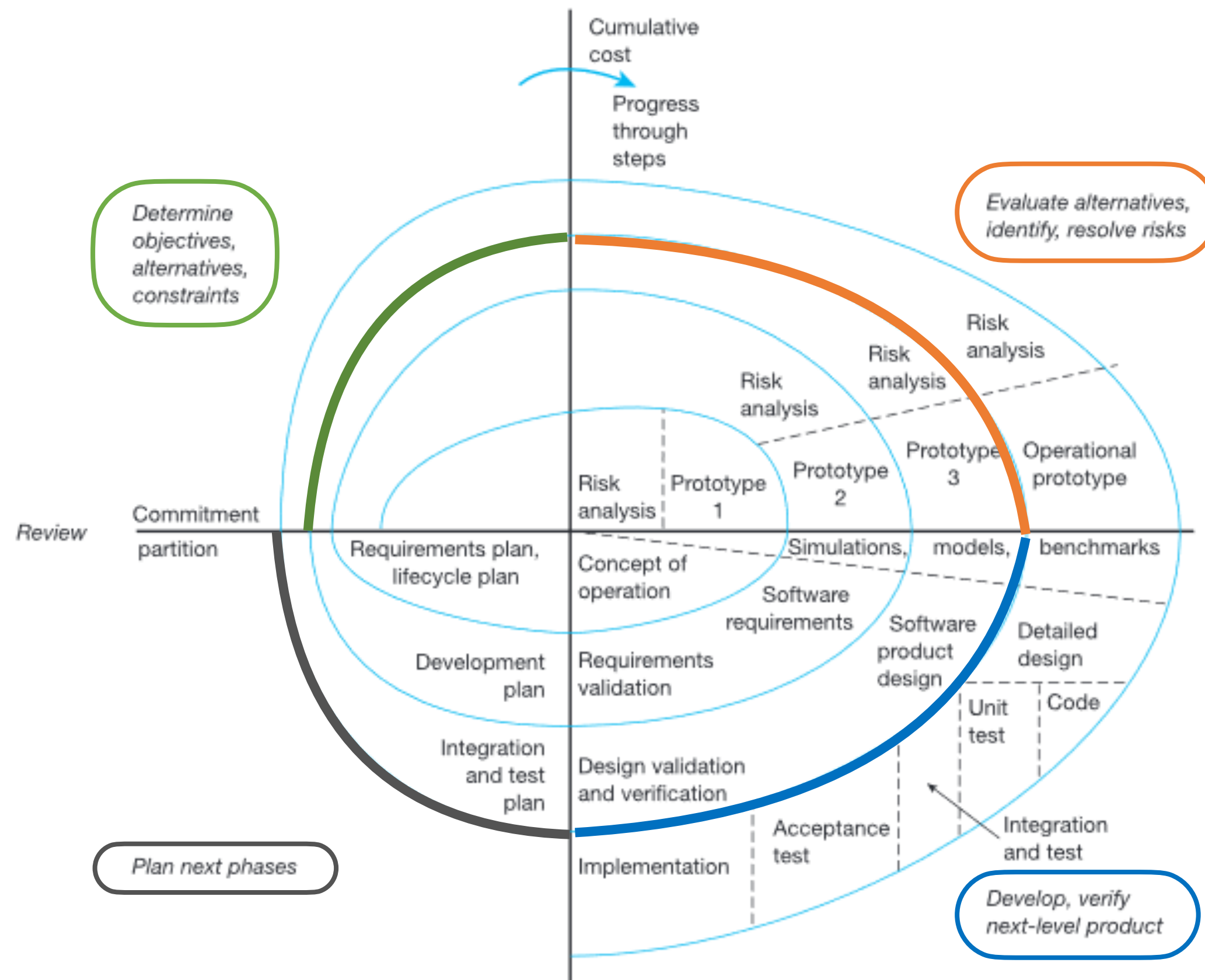
Nach Boehm 1988

## Charakteristika

- ▶ Rundenbasiert
  - **Entwicklungsziele** und Einschränkungen
  - **Risiko-Analyse und Alternativen**, Prototypen
  - **Entwicklungsphase** (Design, Programmierung, Verifikation, Validierung)
  - **Planungsphase** (für den nächsten Iterationsschritt)
- ▶ Gut für große Systeme, die das Zusammenarbeiten von mehreren Disziplinen erfordert



# DAS SPIRAL MODELL (2/2)



Nach Boehm 1988

## Vorteile

- + Risiken können früher erkannt werden
- + Volatile Anforderungen können besser berücksichtigt werden
- + Inkrementelle Auslieferung wird erleichtert
- + Anpassungsfähig!

## Nachteile

- Mehrarbeit
- Komplexeres Projektmanagement
- Theoretisches Modell

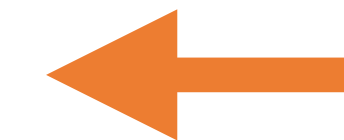
# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>



# INHALTE

<b>0</b>	<b>Einführung</b>
0.1	Definition Software Engineering
0.2	FAQs
<b>1</b>	<b>Software Entwicklungsprozesse</b>
1.1	Herausforderungen und Aktivitäten
1.2	Sequentielle Prozesse
1.3	Iterative Prozesse
1.4	Agile Prozesse
1.4.1	<i>Scrum</i>
1.4.2	<i>Kanban</i>





# AGILE PROZESS MODELLE

„**Agil**“ wird hier im Sinne von „**bereit sich zu bewegen**“, „**aktiv**“ oder „**geschickt**“ verstanden

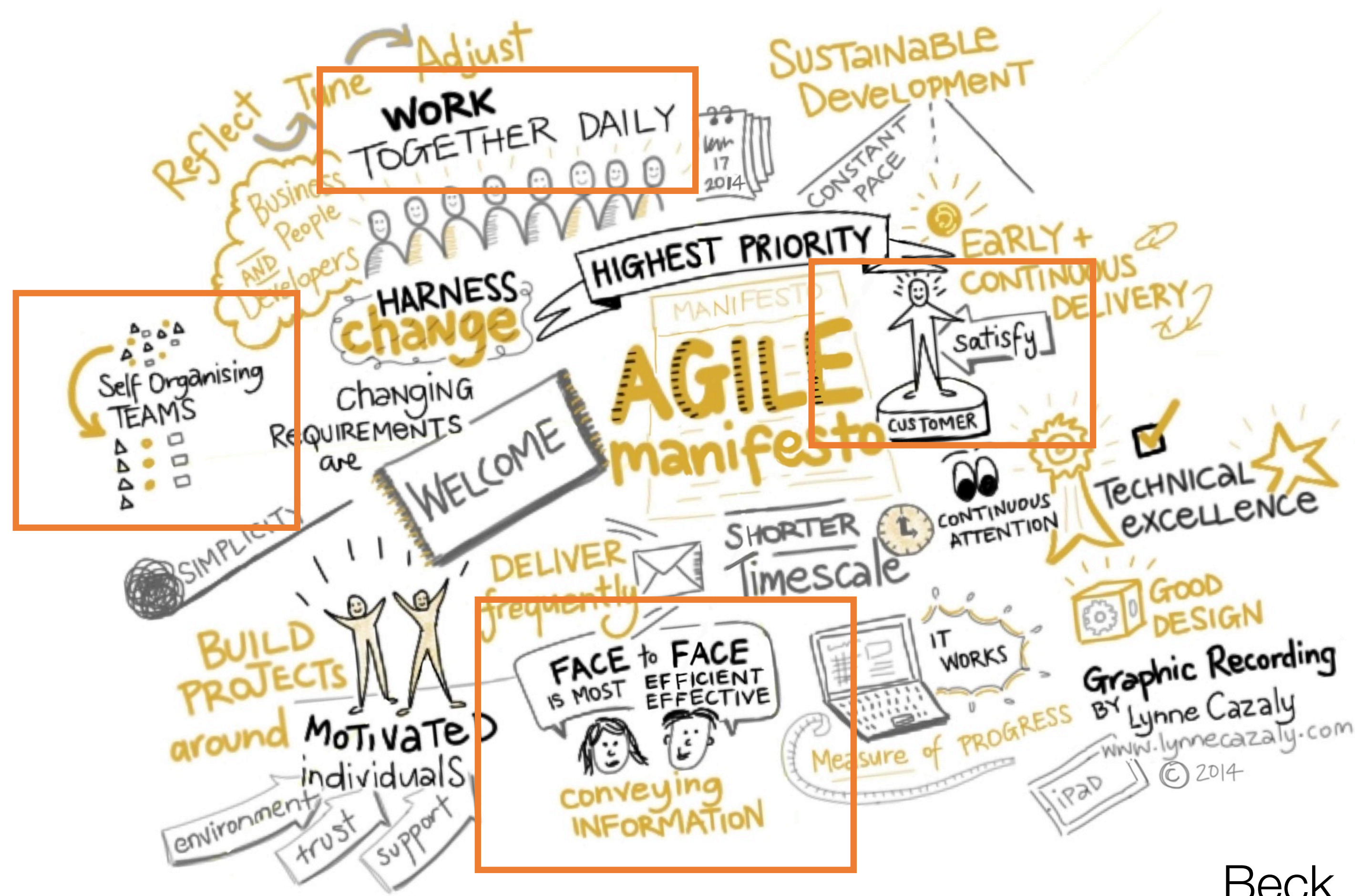
### Gegenbewegung „mit Anti-Prozessen“

- XP, SCRUM, Adaptive Software Development, FDD

### Agiles Manifest: Werte und Prinzipien

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

# DAS AGILE MANIFEST



Beck, Beedle, Cockburn 2001



# AGILE PROZESS MODELLE

## Charakteristika

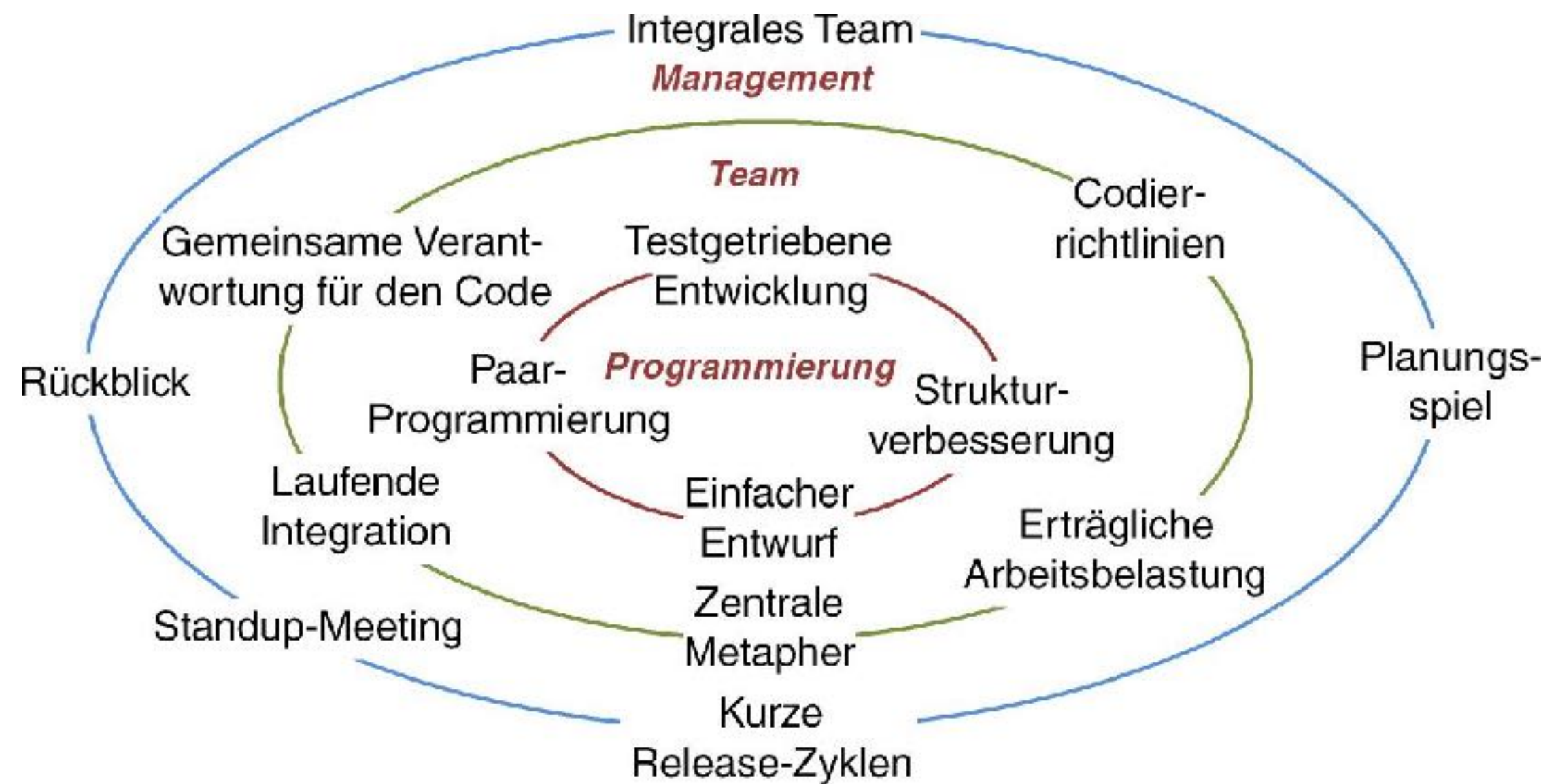
- ▶ Sie sind **iterativ** angelegt. Die Zyklendauer liegt zwischen 2 Wochen und einigen Monaten
- ▶ Die Arbeiten werden in **kleinen Gruppen** (6-8 Personen) durchgeführt
- ▶ Sie lehnen die Idee einer umfangreichen Dokumentation ab (zumindest mit Einschränkungen)
- ▶ Die **Kunden sind sehr wichtig** und deren Präsenz wird verlangt/erwünscht
- ▶ Dogmatische Regelungen werden abgelehnt

# AGILE VS. „KLASSISCHE“ MODELLE

	Bisheriger Ansatz	Agiler Ansatz
<b>Ständige Mitwirkung des Kunden</b>	Unwahrscheinlich	Kritischer Erfolgsfaktor
<b>Etwas Nützliches wird geliefert</b>	Erst nach einiger Zeit	Mindestens alle sechs Wochen
<b>Das Richtige entwickeln durch</b>	Langes Spezifizieren, Vorausdenken	Kern entwickeln, zeigen, verbessern
<b>Nötige Disziplin</b>	Formal, wenig	Informell, viel
<b>Änderungen</b>	Erzeugen Widerstand	Werden erwartet und toleriert
<b>Kommunikation</b>	Über Dokumente	Zwischen Menschen
<b>Vorsorge für Änderungen</b>	Durch Versuch der Vorausplanung	Durch „flexibel bleiben“

# EXTREME PROGRAMMING (1)

## Charakteristika

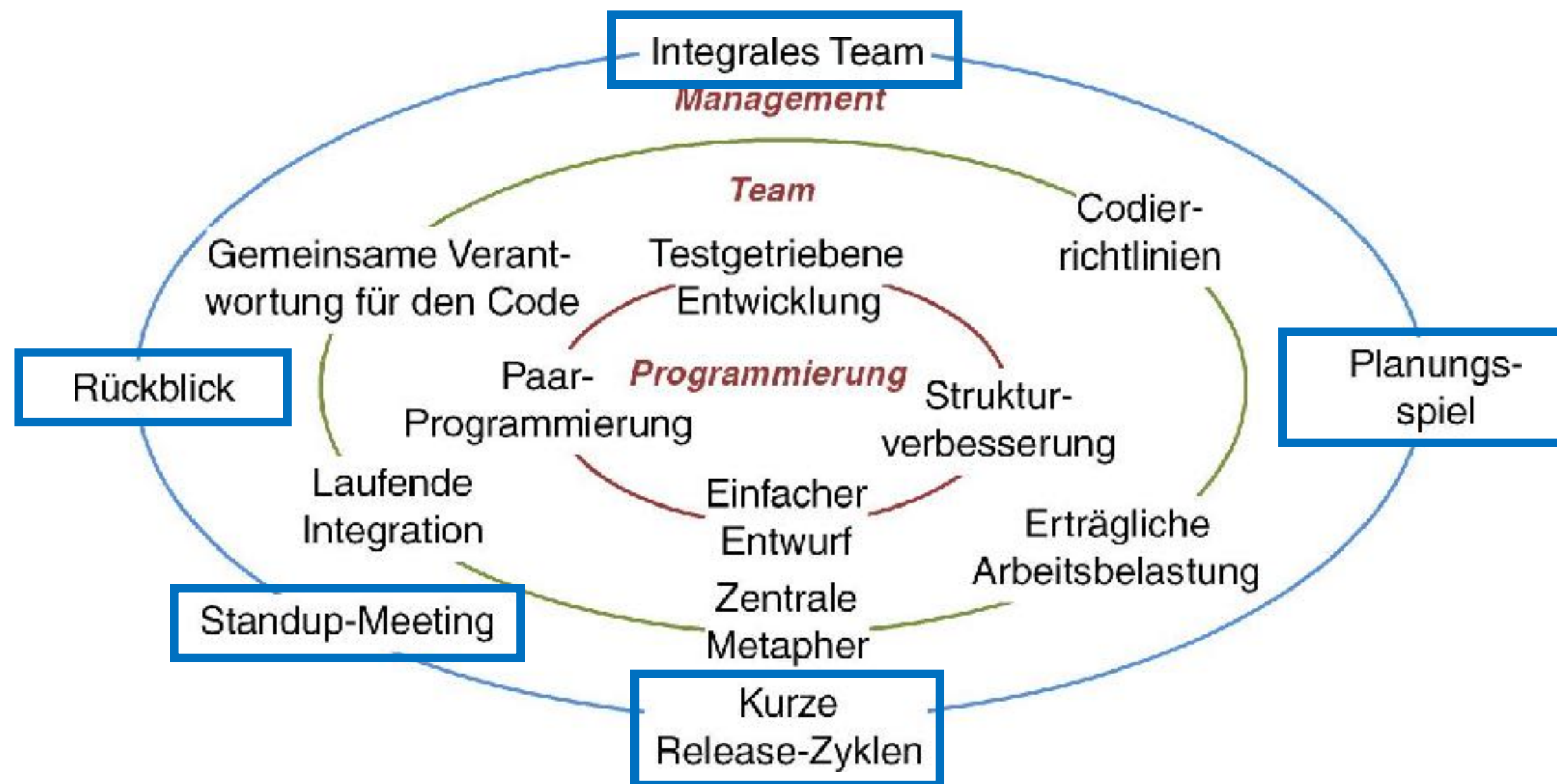


- ▶ **Werte und Prinzipien:** Kommunikation, Feedback, Einfachheit, Mut zur Entscheidung
- ▶ **Konzepte:** **Managementkonzepte**, **Teamkonzepte**, **Programmierkonzepte**
- ▶ **Rollen:** Programmierer, Kunde, XP-Trainer, Verfolger, Tester, Berater, Big Boss



# EXTREME PROGRAMMING (2)

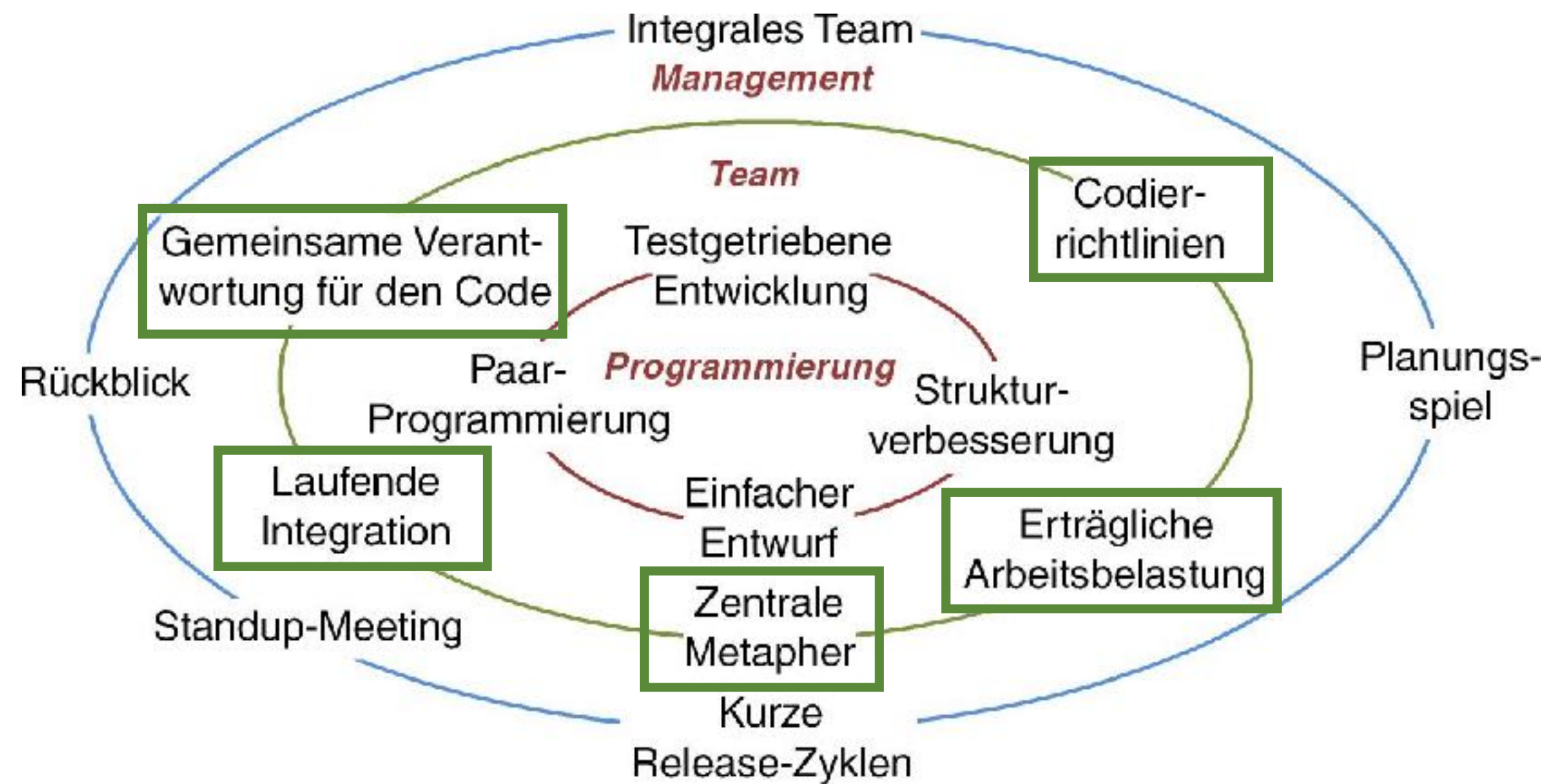
## Charakteristika



- ▶ **Werte und Prinzipien:** Kommunikation, Feedback, Einfachheit, Mut zur Entscheidung
- ▶ **Konzepte:** **Managementkonzepte**, **Teamkonzepte**, **Programmierkonzepte**
- ▶ **Rollen:** Programmierer, Kunde, XP-Trainer, Verfolger, Tester, Berater, Big Boss

# EXTREME PROGRAMMING (3)

# Charakteristika

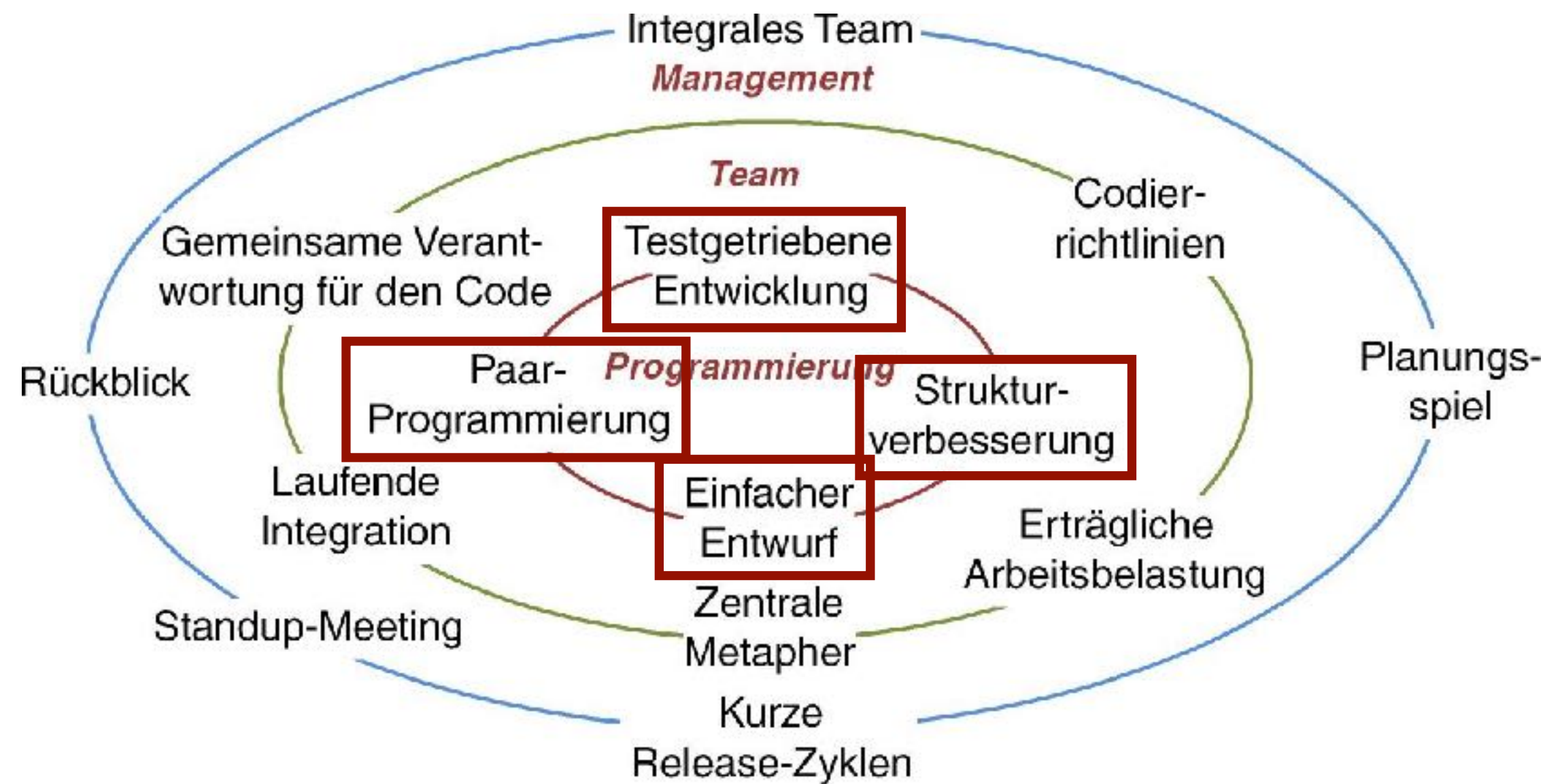


- ▶ **Werte und Prinzipien:** Kommunikation, Feedback, Einfachheit, Mut zur Entscheidung
- ▶ **Konzepte:** **Managementkonzepte**, **Teamkonzepte**, **Programmierkonzepte**
- ▶ **Rollen:** Programmierer, Kunde, XP-Trainer, Verfolger, Tester, Berater, Big Boss



# EXTREME PROGRAMMING (4)

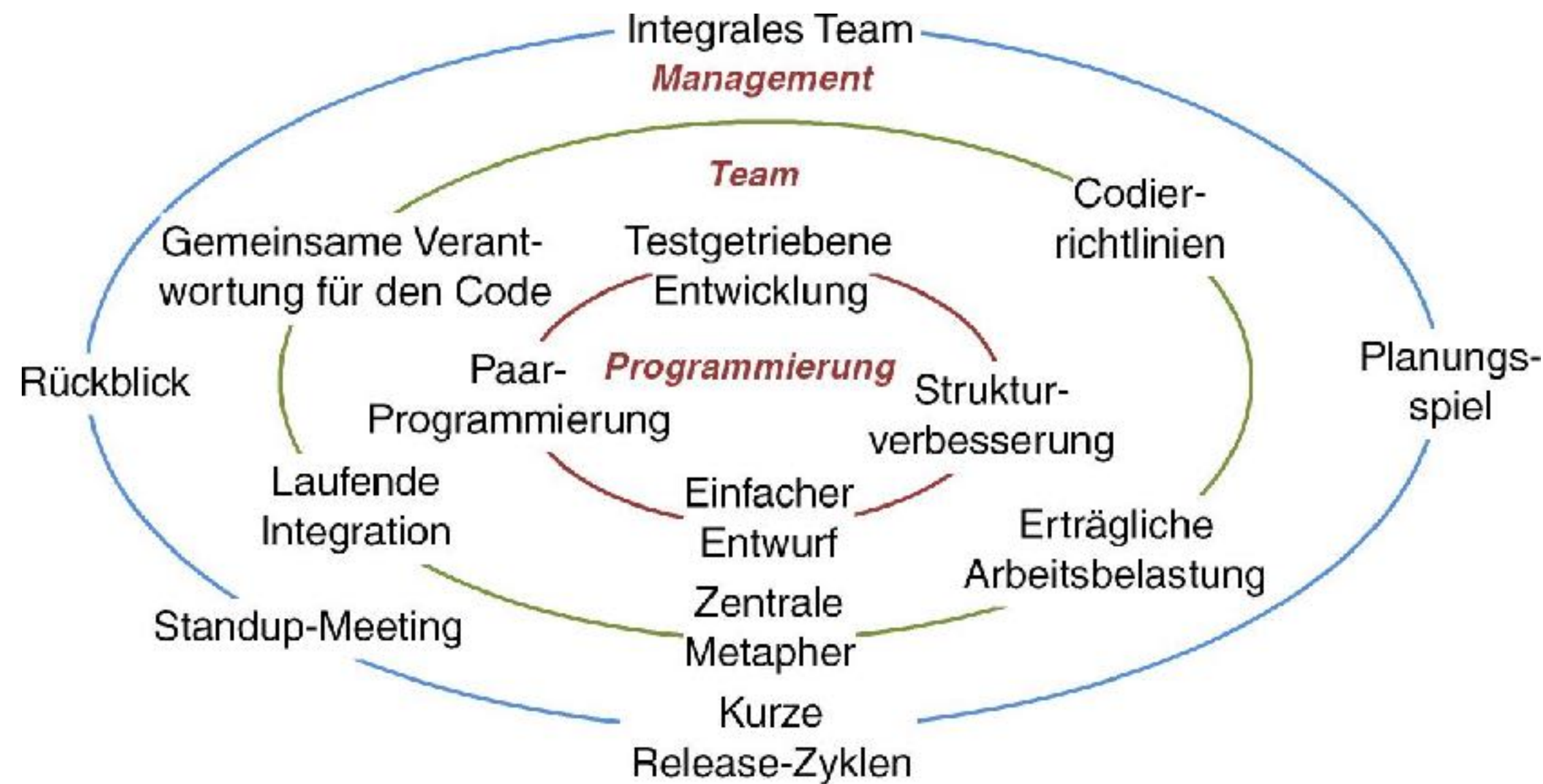
## Charakteristika



- ▶ **Werte und Prinzipien:** Kommunikation, Feedback, Einfachheit, Mut zur Entscheidung
- ▶ **Konzepte:** **Managementkonzepte**, **Teamkonzepte**, **Programmierkonzepte**
- ▶ **Rollen:** Programmierer, Kunde, XP-Trainer, Verfolger, Tester, Berater, Big Boss

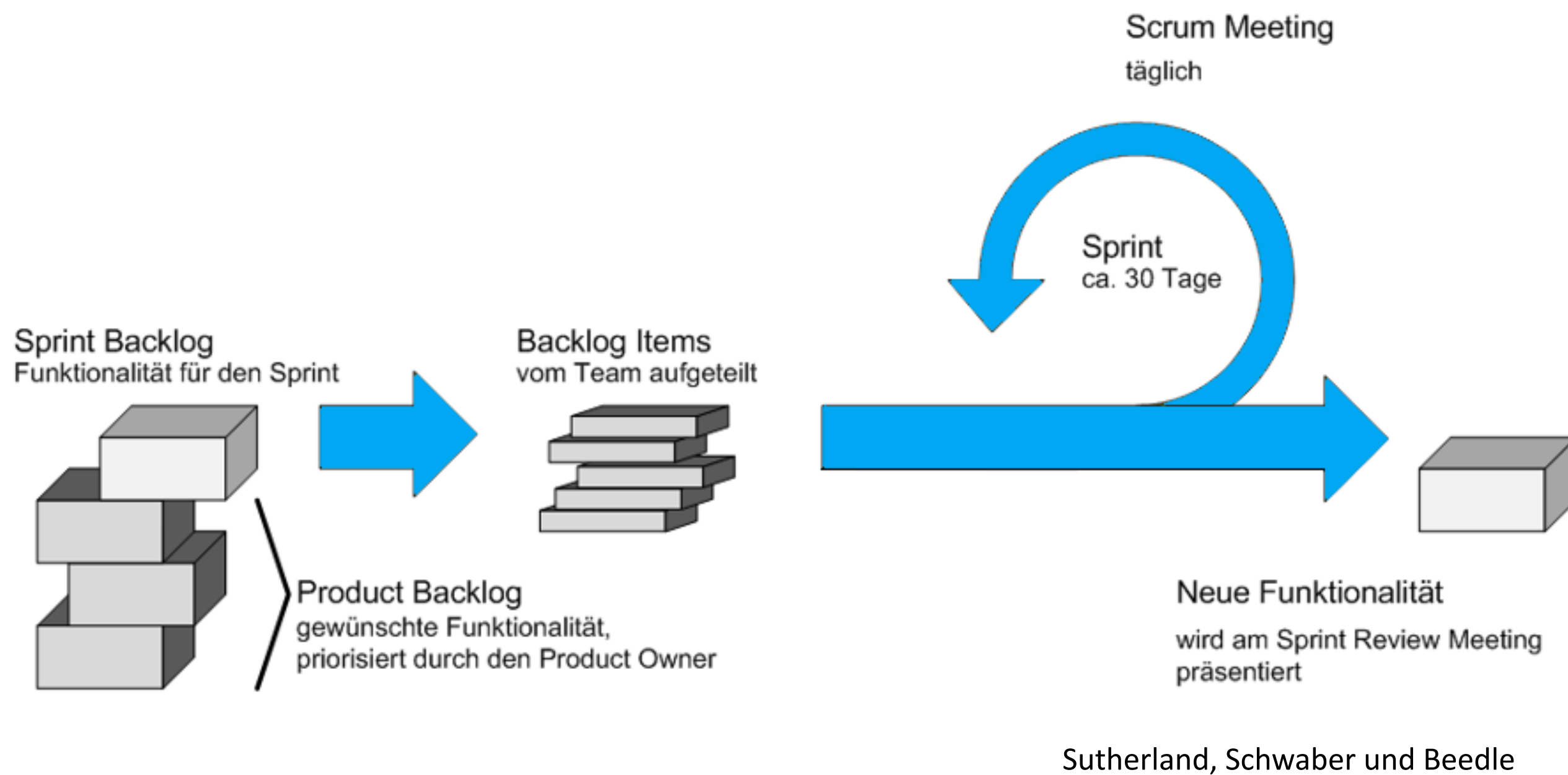
# EXTREME PROGRAMMING (5)

## Charakteristika



- ▶ **Werte und Prinzipien:** Kommunikation, Feedback, Einfachheit, Mut zur Entscheidung
- ▶ **Konzepte:** **Managementkonzepte**, **Teamkonzepte**, **Programmierkonzepte**
- ▶ **Rollen:** Programmierer, Kunde, XP-Trainer, Verfolger, Tester, Berater, Big Boss

# SCRUM



## Charakteristika

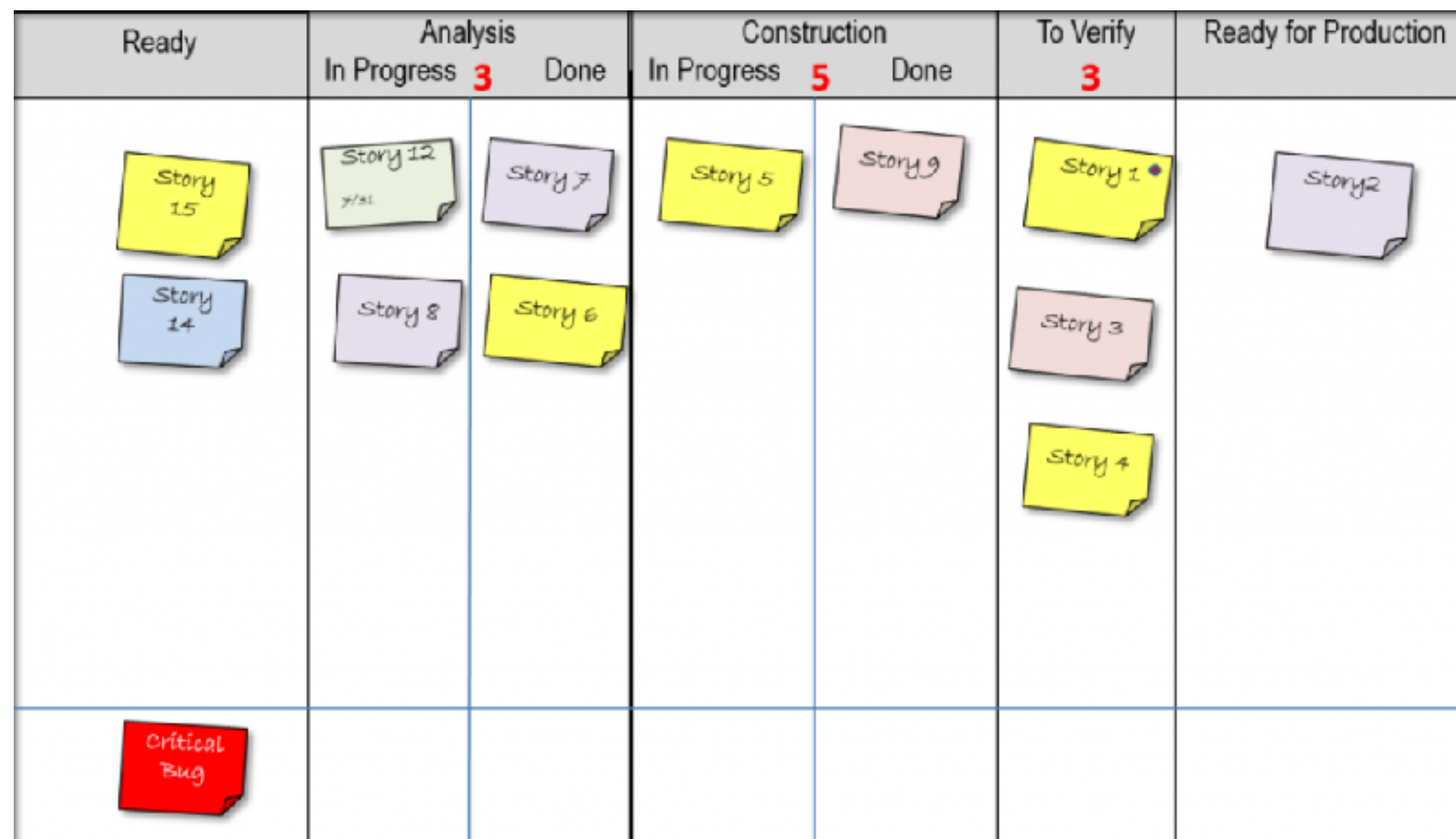
- ▶ Keine strenge Arbeitsteilung, sondern **kooperative Zusammenarbeit**
- ▶ Selbst **organisierendes Team**
- ▶ Erfolg und Misserfolg wird kollektiv verantwortet
- ▶ Terminologie aus dem Rugby
- ▶ „**scrum**“ Eroberung des Balles
- ▶ „**sprint**“ mit dem Ball um Raum zu gewinnen
- ▶ Scrum ist Rollen-basiert, iterativ und inkrementell



# KANBAN

## Charakteristika

- ▶ Orientiert sich an der selben Strategie, die Supermärkte nutzen, um ihre Regale zu füllen
- ▶ **Kanban-Board**
  - ▶ Tool zur Visualisierung
- ▶ **Kanban-Karten**
  - ▶ Jedes Aufgabenelement ist separate Karte
  - ▶ Ermöglichen **klare visuelle Darstellung** des Workflows
  - ▶ Stellen kritische Informationen zum jeweiligen Aufgabenfeld in den Vordergrund



Toyota Motor Corporation, 1947

# BEWERTUNG AGILER PROZESSE

## Anwendung

- ▶ Gut einsetzbar bei unklaren Zielen und sich ändernden Anforderungen
- ▶ innovative, zeitkritische Projekte

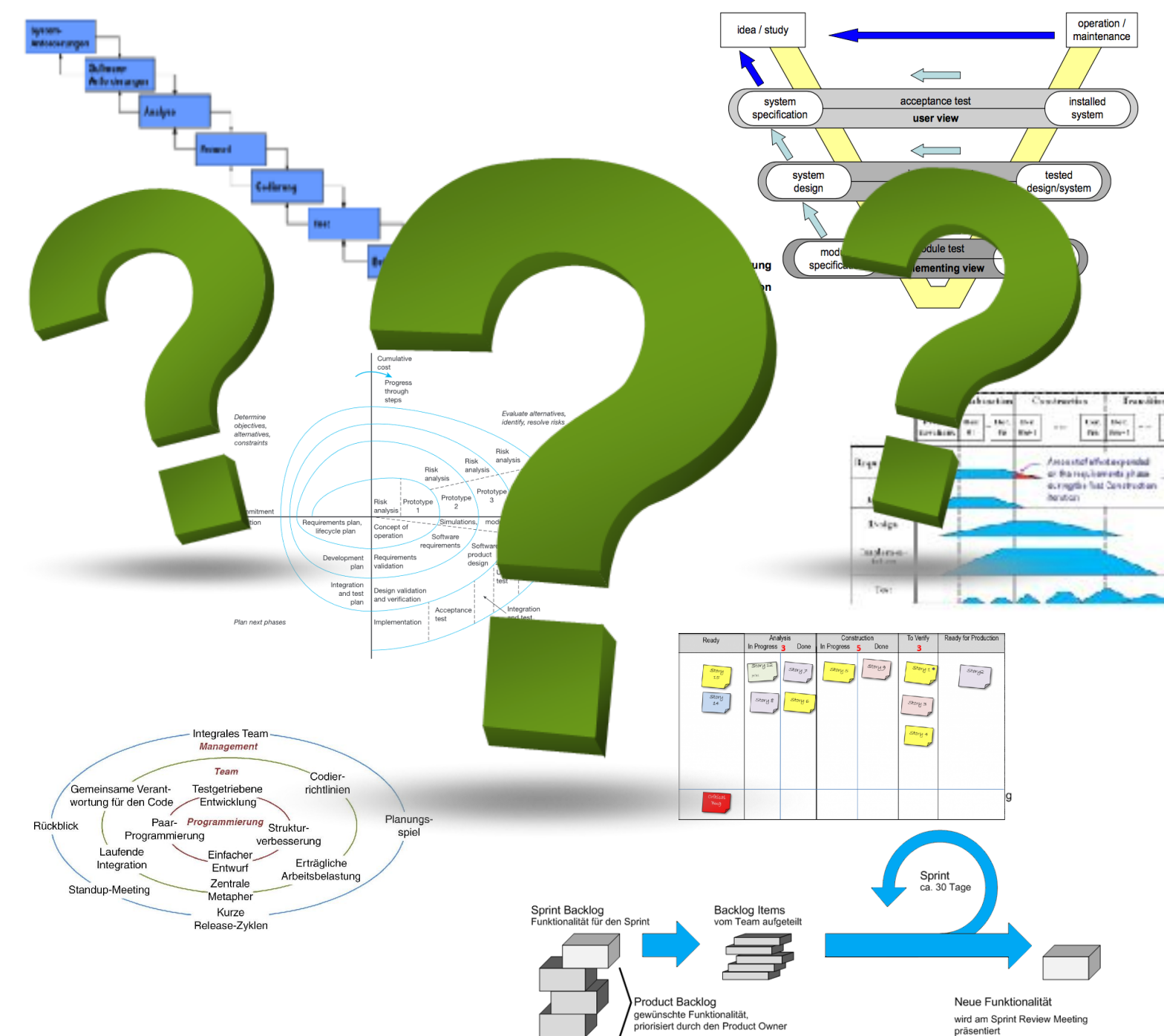
## Vorteile

- + Verspricht verbessertes Kosten/Nutzen Verhältnis
- + Verbesserte durchschnittliche Code-Qualität

## Nachteile

- Das Ergebnis ist nicht vorhersagbar
- Qualitätseigenschaften können nicht garantiert werden
- Weniger Planung
- Kunde (mangelndes Fachwissen)
- Zeitfaktor Refactoring

# WELCHES SW-PROZESS MODELL IST DAS BESTE???



Das „beste“ Modell hängt von der Organisation, dem Produkt, bzw. den Fähigkeiten der Entwickler ab (es gibt kein ideales Modell!)

## ZUSAMMENFASSUNG

### **Was sind SW-Prozess Modelle?**

- ▶ Menge an Aktivitäten und Leistungen zur Erzeugung eines SW-Produkts
- ▶ Bestehend aus: Spezifikation, Design, Implementierung, Validierung/Verifizierung, Evolution

### **Wozu brauchen wir SW-Prozess Modelle?**

- ▶ Um Projekte planbar zu machen
- ▶ Durch erhobene Anforderungen, Basis für Tests schaffen
- ▶ Kosten für Wartung senken
- ▶ Erworbenes Wissen wiederverwendbar machen

### **Welche SW-Prozess Modelle gibt es?**

- ▶ Sequentielle und Iterative Methoden: Wasserfall, V-Model, Spiralmodell, (RUP)
- ▶ Agile Methoden: XP, Scrum, Kanban, Scaled Agile

**Prozessmodelle sind nicht gut oder schlecht, sondern zum Projekt passend oder unpassend!**