

# Software Engineering

## Vorgehensmodelle

**Prof. Dr. Peter Jüttner, Technische Hochschule Deggendorf**  
**Dr. Christian Flug, Continental AG**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

4.1. Wasserfall

4.2. V-Modell

4.3. Iterative / Agile Methoden

4.3.1. RUP

4.3.2. Extreme Programming

4.3.3. Scrum

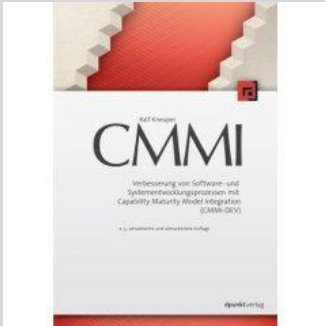
4.4. Prozessbewertung

4.4.1. CMMI

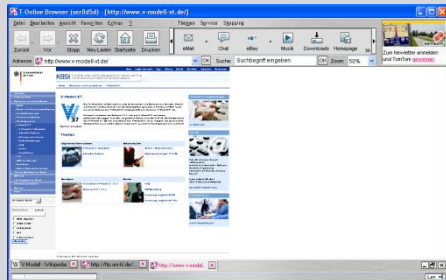
4.4.2. SPICE

## Literatur

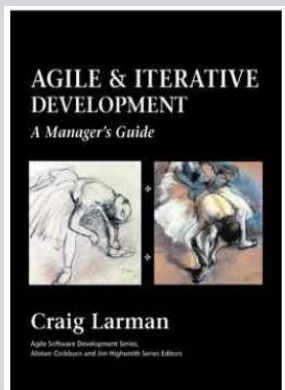
**Ralf Kneuper, CMMI, Dpunkt Verlag**



**[www.v-modell-xt.de](http://www.v-modell-xt.de)**



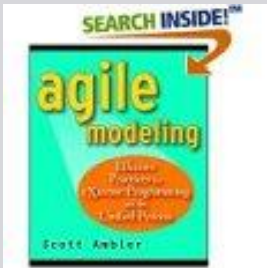
**Craig Larman, Agile & Iterative Development,  
Addison-Wesley, ISBN 0131111558**



# Literatur



**K. Hörmann, L. Dittmann, B. Hindel, M. Müller: SPICE in der Praxis. Interpretationshilfe für Anwender und Assessoren. dpunkt Verlag, Heidelberg 2006, ISBN 3-89864-341-7**



**S.W. Ambler, Agile Modelling: Effective Practices for EXtreme Programming and the Unified Process, Wiley & Sons**

# Software Prozess

Ein **(Software) Prozess** beschreibt eine Abfolge von Tätigkeiten zusammen mit deren Ergebnissen

Gemeinsamkeiten von Softwareprozessen

- Spezifikation: Definition der Funktion, Randbedingungen (Anforderungen)
- Entwicklung: Erstellen (Design, Implementierung) und Qualitätssicherung
- Validierung, Test: Sicherstellen, dass die Software tut, was der Kunde will
- Evolution: Weiterentwicklung wegen sich ändernden Wünschen oder Randbedingungen

# Software Prozess

Softwareprozesse unterscheiden sich in

- der Anzahl, Art und Koordination der Prozessschritte
- Anzahl und Art der entwickelten Produkte

Ein **Vorgehensmodell** ist eine vereinfachte/abstrahierte Beschreibung eines Softwareprozesses

# Software Prozess

Ein **Vorgehensmodell** legt fest:

- durchzuführende Aktivitäten
- Reihenfolge der Tätigkeiten (Entwicklungsstufen, Phasen)
- Definition der Zwischenergebnisse / Endergebnisse (Inhalt, Layout)
- Endekriterien
- Verantwortlichkeiten und Kompetenzen
- Notwendige Qualifikationen der Mitarbeiter
- Zu verwendende Standards, Vorlagen, Richtlinien, Methoden und Werkzeuge

Vorgehensmodelle, die in Phasen strukturiert sind, nennt man auch Phasenmodelle

# Software Prozess

## Vorgehensmodelle

- Wasserfall-Modell
- Prototyping, evolutionäre Entwicklung
- Formale Entwicklung
- Wiederverwendungsorientierte Entwicklung
  - Opportunistisch, ungeplant
  - Strategisch (Produktlinien)
- Prozesswiederholende Modelle
  - Inkrementelle Entwicklung
  - agile Entwicklung
- Modellbasierte Entwicklung



## Randbemerkung

### Achtung:

**Der Erfolg eines Software Projekts hängt nicht an einem Faktor allein, d.h.**

**weder durch**

- ***neue Methoden***
- ***neue Tools***
- ***neue Vorgehensmodelle***

**werden im SW Engineering Quantensprünge ausgelöst, sondern kontinuierliche Verbesserungen in kleinen Schritten erreicht**



**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

#### 4.2. V-Modell

#### 4.3. Iterative / Agile Methoden

##### 4.3.1. RUP

##### 4.3.2. Extreme Programming

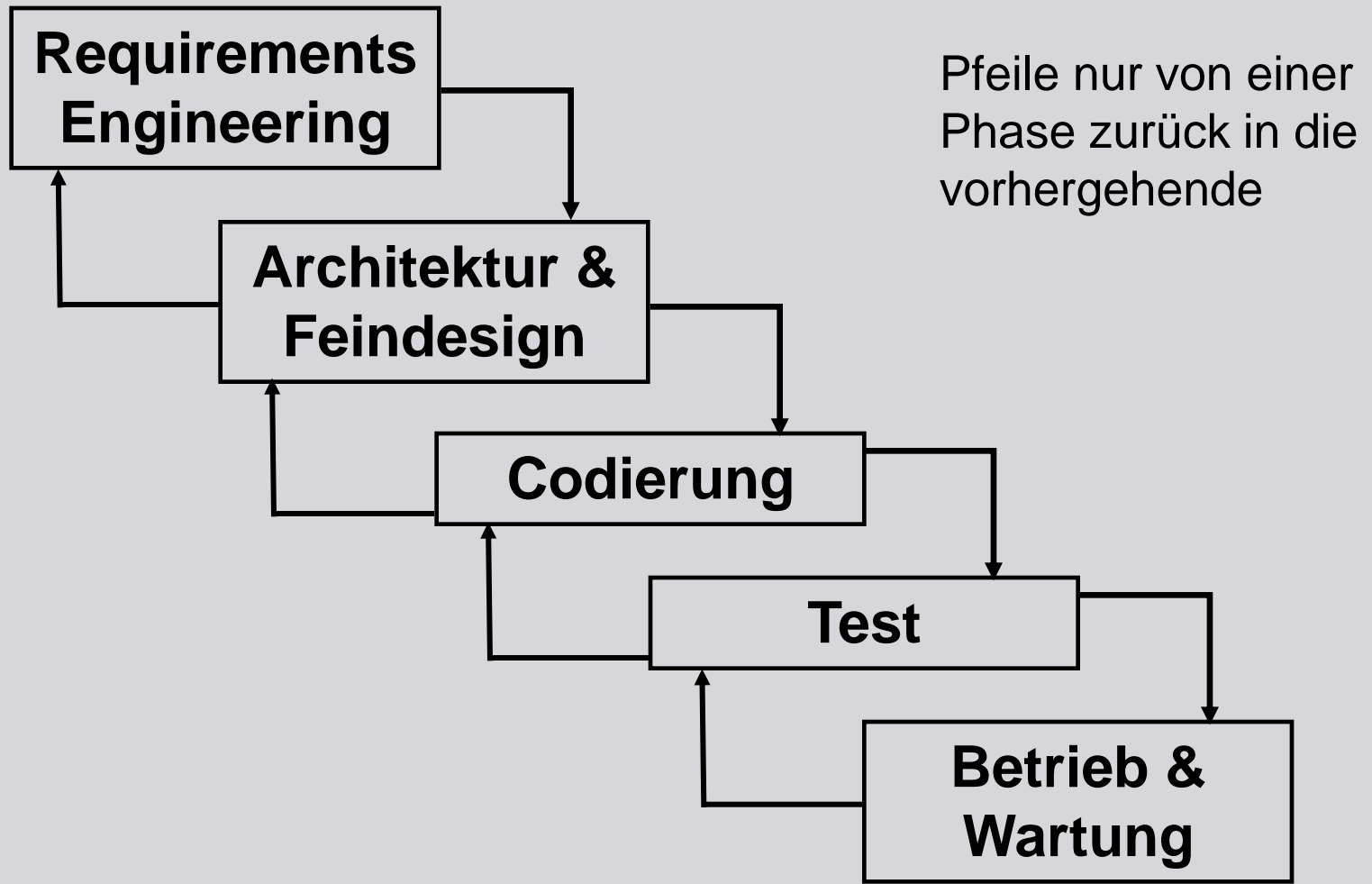
##### 4.3.3. Scrum

#### 4.4. Prozessbewertung

##### 4.4.1. CMMI

##### 4.4.2. SPICE

# Wasserfall Modell



# Wasserfall Modell

- Ältestes Vorgehensmodell (Boehm, ca. 1980)
- Strikte Reihenfolge von Projektphasen
- Korrekturzyklen werden nur auf die jeweils vorangehende Phase beschränkt
- Vollständiges Ausführen einer Phase, Dokumentation als Abschluss
- Nachfolgephase beginnt erst nach vollständigem Abschluss einer Phase
- Der Auftraggeber und Anwender ist nur in der Planungsphase und bei der Abnahme beteiligt.

# Wasserfall Modell



## Vorteile:

- Einfaches und klares Vorgehen
- Relativ einfach umzusetzen
  - Planung
  - Steuerung
- (Relativ) gut anwendbar bei klaren Anforderungen (z.B. Standards)
- Erzwingt eine gewisse Disziplin bei den Entwicklern

# Wasserfall Modell

## Nachteile:

- Entspricht nicht der Realität:
  - Anforderungen sind zu Projektstart nicht immer (bzw. nie) zu 100% klar
  - zusätzliche Anforderungen ergeben sich erst im Lauf des Projekts
  - änderungsunfreundlich
- Phasen können sich gegenseitig bedingen
- Schwache Einbindung des Auftraggebers bzw. Anwenders
- spätes Erkennen von Risiken, die sich aus der Implementierung ergeben (z.B. Hardwareprobleme, Toolprobleme)

# Wasserfall Modell

## Nachteile:

- spätes Erkennen von Fehlern aus der frühen Phasen, die erst in spätere Phasen erkennbar werden (z.B. falsche Architektur-entscheidungen)
- Dokumentation bekommt einen manchmal zu hohen Stellenwert
- Zeitverzug aus frühen Phasen schlägt voll durch auf spätere Phasen
- Im Extremfall ist das Produkt nicht brauchbar, die Entwicklung muss neu von vorne begonnen werden
- Späte Änderungen und Fehlerbehebungen im Code führen zu veralteter Dokumentation



**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Extreme Programming

#### 4.3.3. Scrum

### 4.4. Prozessbewertung

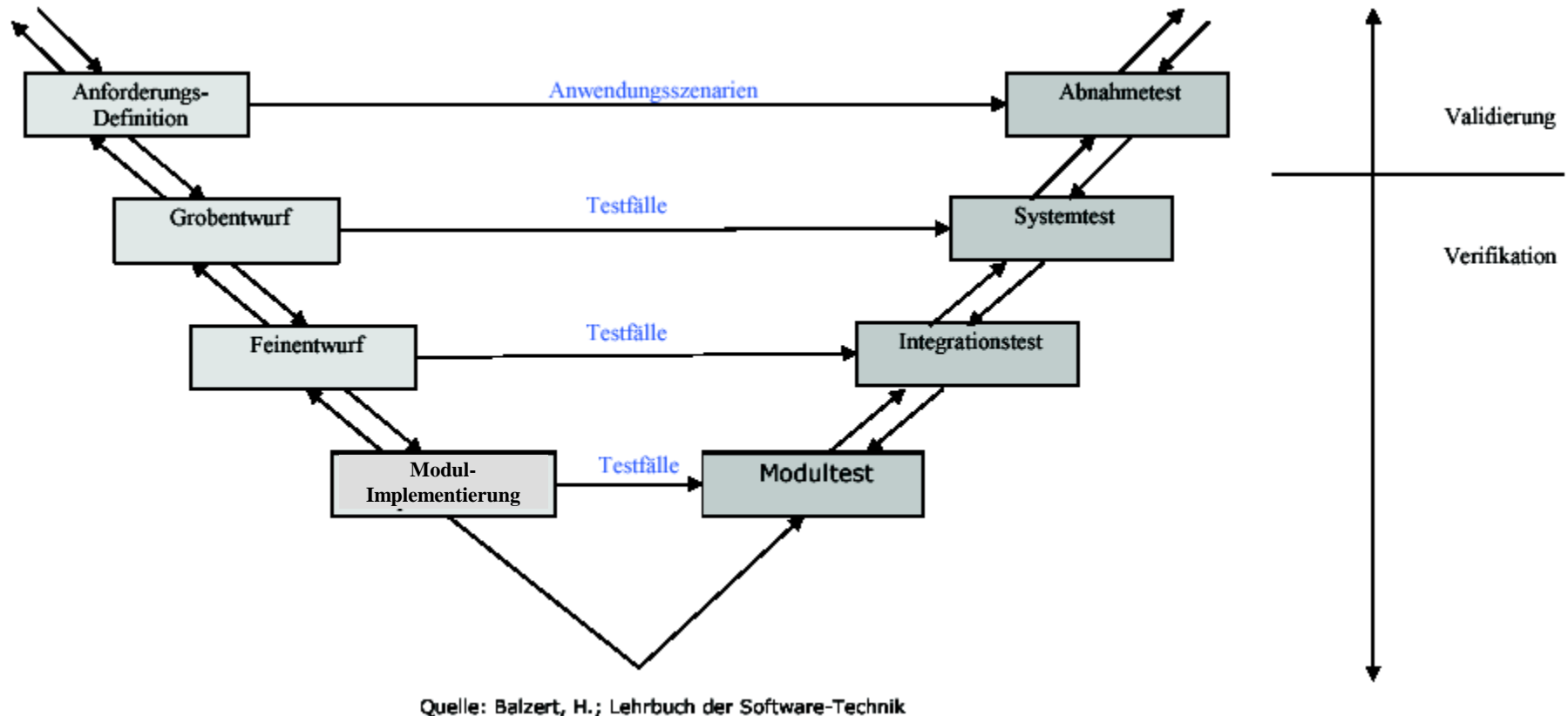
#### 4.4.1. CMMI

#### 4.4.2. SPICE

# V- Modell

- Verbessertes Wasserfallmodell
    - Ergänzt um Qualitätssicherungsmaßnahmen in allen Phasen
    - Standardisiert
  - Ziel: Entwurfsfehler frühzeitig zu erkennen
  - Aber: Nachteile des Wasserfallmodells bleiben erhalten
    - starre Reihenfolge der Phasen
    - Ergebnis in einem Durchlauf entwickelt
- ➔ Weiterentwicklung und Verbesserung zum V-Modell-XT

# V- Modell



## V-Modell-XT

- Verbesserung der Unterstützung von Anpassbarkeit, Anwendbarkeit, Skalierbarkeit und Änder- und Erweiterbarkeit des V-Modells
- Berücksichtigung des neuesten Stands der Technologie und Anpassung an aktuelle Vorschriften und Normen
- Erweiterung des Anwendungsbereiches auf die Betrachtung des gesamten Systemlebenszyklus im Rahmen von Entwicklungsprojekten
- Einführung eines organisationsspezifischen Verbesserungsprozesses für Vorgehensmodelle

**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Extreme Programming

#### 4.3.3. Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

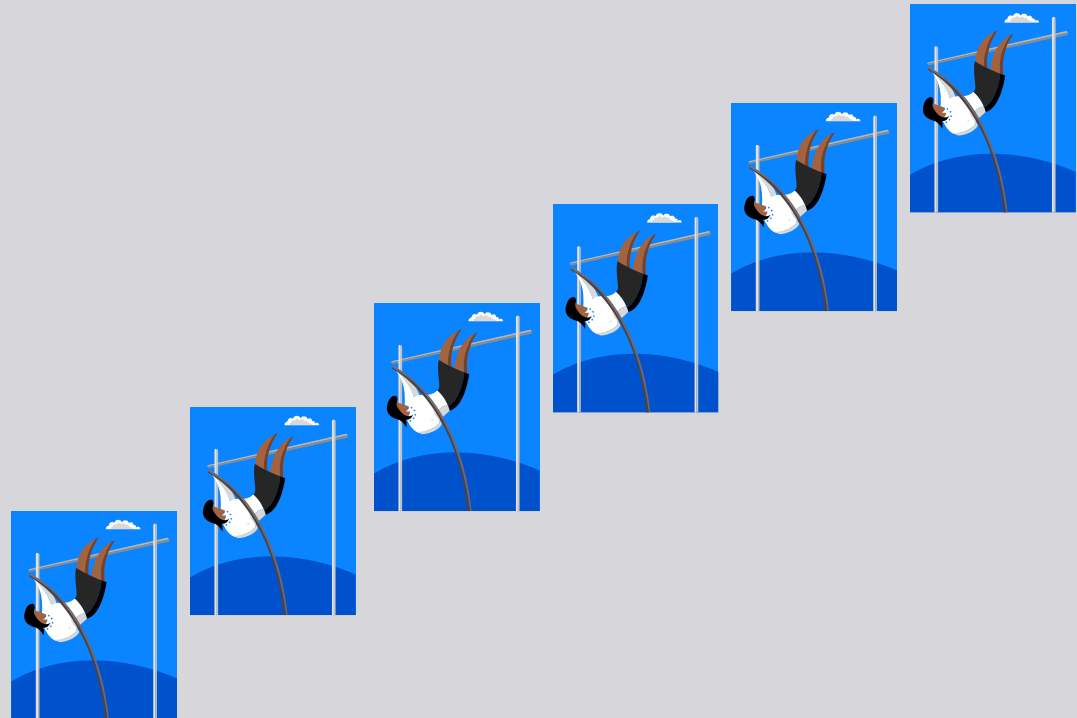
# Inkrementelles & iteratives Vorgehen

## Lösung: Inkrementelles bzw. iteratives Vorgehen

statt:



besser:





# Inkrementelles & iteratives Vorgehen

## Charkteristika / Ziele

- Schrittweise Vorgehen
- Möglichst schnell lauffähigen Code
- **Lauffähiger Code ist das im Vordergrund stehende Produkt**
- Nicht alle Anforderungen auf einmal implementieren
- Mit den Iterationen wachsender Funktionsumfang
- Direkte und permanente Einbindung des Auftraggebers, Kunden, Anwenders

# Inkrementelles & iteratives Vorgehen

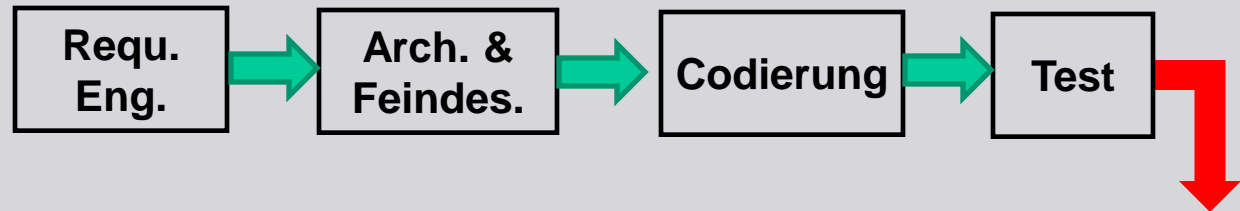
## Charakteristika / Ziel

- Frühzeitiges Erkennen von Risiken und Problemen (z.B. falsche Architektur)
- Frühzeitiges Lösen von Problemen und Minimieren von Risiken
- Wissen nicht (nur) in der Dokumentation, sondern in den Köpfen möglichst vieler Entwickler

# Inkrementelles & iteratives Vorgehen

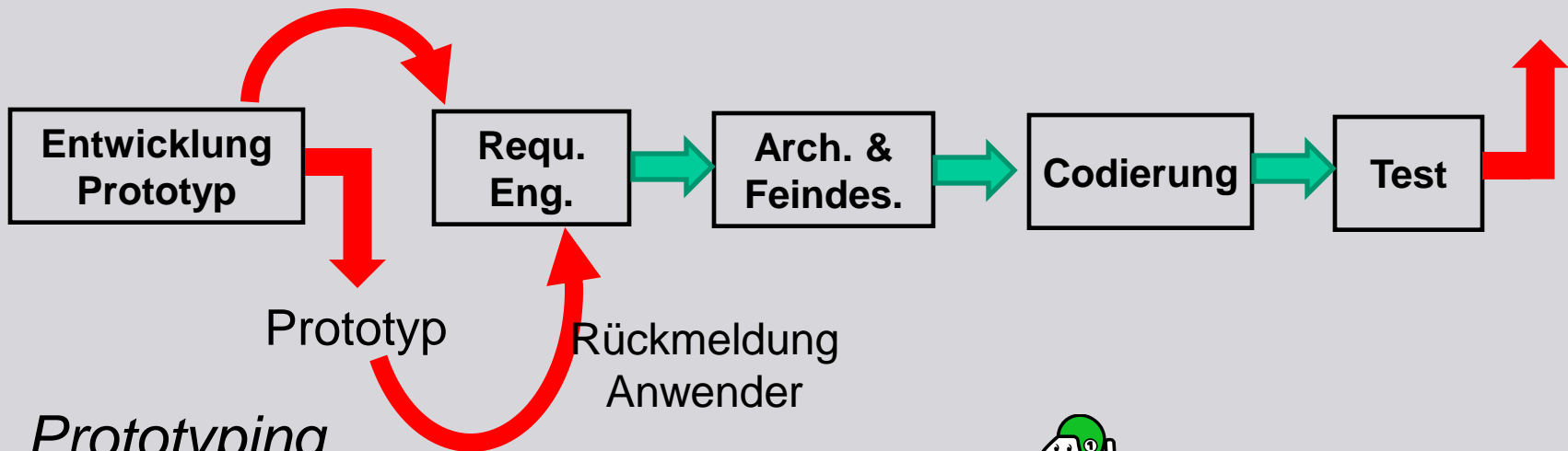
## Erster Ansatz: Prototyping

Wasserfall

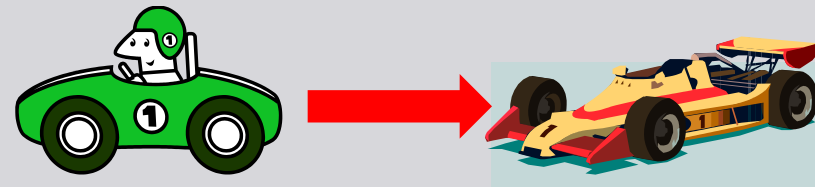


Erfahrungen, Lösungsansätze

Endprodukt



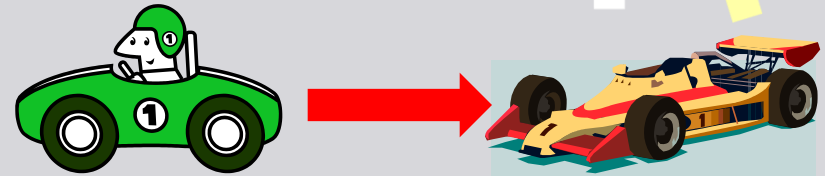
Prototyping



# Inkrementelles & iteratives Vorgehen

## Erster Ansatz: Prototyping

- Prototyp
  - Erste Implementierung einer zu entwickelnden Software
  - Mit Schwerpunkt auf die wichtigsten Aspekte des späteren Anwenders
  - Reduzierter Aufwand (Entwicklungsprozess, Qualitätssicherung)
  - Mit den Zielen
    - Darstellen der Machbarkeit
    - Erfahrungsaufbau in der Entwicklung
    - Abstimmung der Anforderungen mit dem Kunden / Anwender
  - Code ist Wegwerfprodukt, Wiederverwendung von Erfahrung, Konzepten, Anforderungen, Tests, ...



# Inkrementelles & iteratives Vorgehen

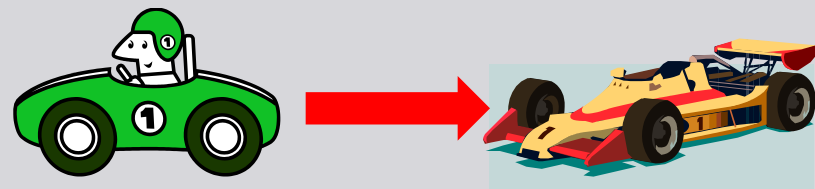
## Erster Ansatz: Prototyping

- **Vorteile**

- Geringeres Entwicklungsrisikos durch frühzeitige stärkere Rückkopplung.
- Prototypen sind durch geeignete Werkzeuge schnell darstellbar. („**Rapid Prototyping**“)

- **Nachteile**

- Evtl. höherer Entwicklungsaufwand
- Gefahr, dass ein Prototyp nicht weggeworfen wird.
- Gefahr des Übersehens wichtiger Randbedingungen



**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2 Agile Methoden: Extreme Programming

#### 4.3.3. Agile Methoden: Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

# Iterative & inkrementelle Methoden – RUP

## Allgemeines

- ***"Rational Unified Process"***, I. Jacobsen, 1999
- Entwickelt von Rational, jetzt IBM
- Produkt, unterstützt durch zahlreiche Tools
- „The Rational Unified Process - An Introduction“, P. Kruchten
- Ziele
  - Allgemeingültiges Prozeßmodell
  - Einsatz bewährter Entwicklungspraktiken



# Iterative & inkrementelle Methoden – RUP

## Struktur

- Einteilung in 4 zeitlich aufeinanderfolgende Phasen
  - Inception (Konzeptphase)
    - Analyse Anforderungen, Konzeption, Planung Vorgehen und Qualitätssicherung
  - Elaboration (1. Entwurf)
    - Fachkonzept, Grobkonzept, Feinkonzept, Entwurf System-Architektur, Entwurf Test-Konzept, Qualitätssicherung.
  - Construction (Konstruktion)
    - Entwurf Software-Architektur, Design, Codierung, Test, Abnahme beim Kunden.
  - Transition (Übergabe an den Kunden)
    - Software-Lieferung, Inbetriebnahme

# Iterative & inkrementelle Methoden – RUP

## Struktur

In den Phasen werden Iterationen durchgeführt, die sich in die folgenden Aktivitäten gliedern:

- Geschäftsmodellierung (Business Modelling)
- Anforderungsanalyse (Requirements)
- Entwurf (Analysis and Design),
- Implementierung
- Test
- Auslieferung (Deployment)
- Je nach Phase werden die Aktivitäten mit unterschiedlichem Aufwand durchgeführt.
- Jede Iteration wird durch einen Meilenstein (formales Review) abgeschlossen.

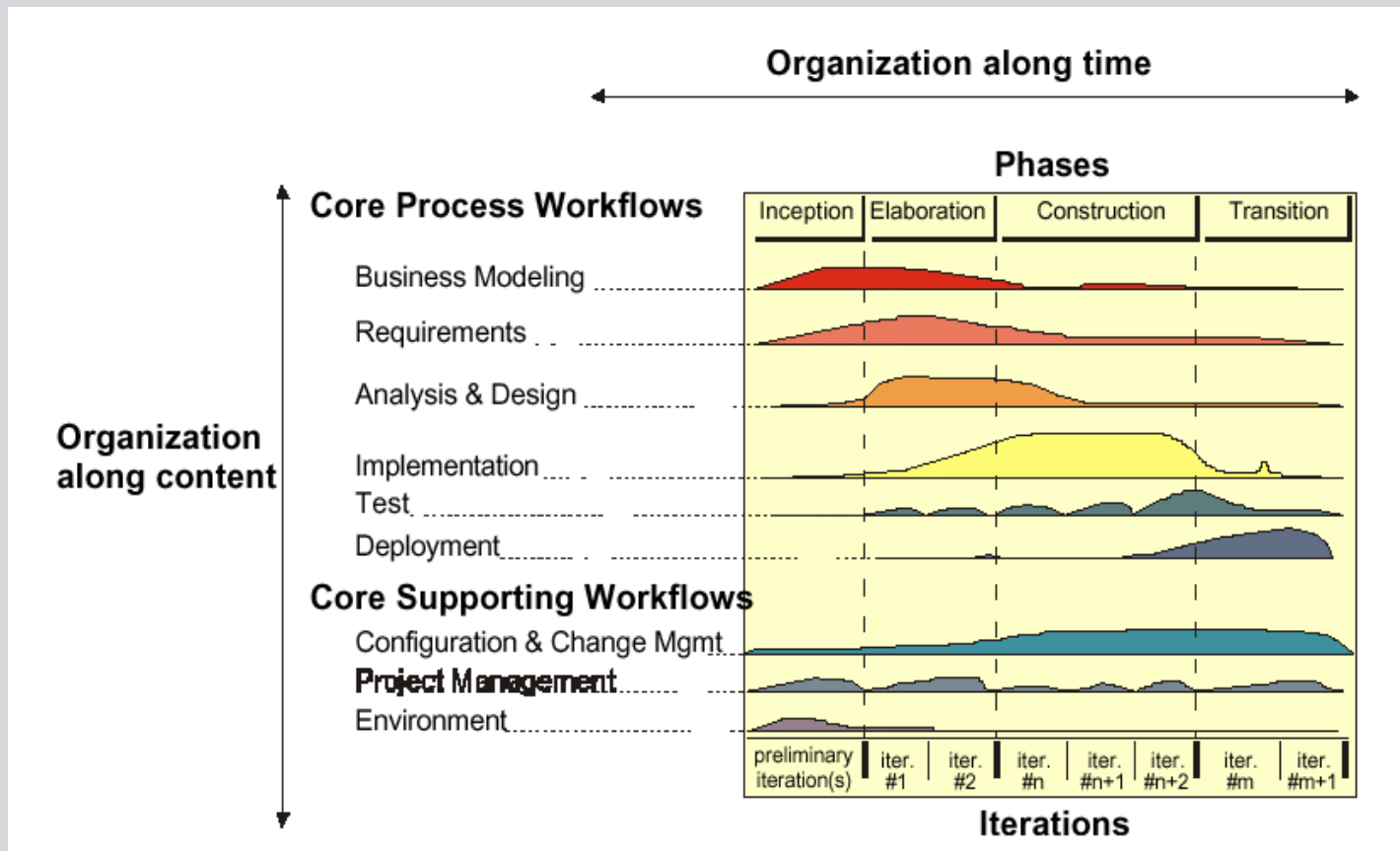
# Iterative & inkrementelle Methoden – RUP

## Struktur

Begleitende, unterstützende Aktivitäten (Core Supporting Workflows)

- Konfigurations- und Änderungsmanagement (config. & Change Mgmt)
- Projektmanagement.

# Iterative & inkrementelle Methoden – RUP



# Iterative & inkrementelle Methoden – RUP

## RUP beinhaltet die wesentlichen Elemente:

Worker (Rollen)	:	wer
Activities (Tätigkeit)	:	wie
Workflows (zeitlicher Ablauf)	:	wann
Artifacts (Ergebnis)	:	was

**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Agile Methoden: Extreme Programming

#### 4.3.3. Agile Methoden: Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

# Agile Methoden

## 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.

(Quelle: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas  
<http://www.agilemanifesto.org>, 2001)




# Agile Methoden - Prinzipien

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most effective method of conveying information to and within a development team is face-to-face conversation.

(Quelle: <http://www.agilemanifesto.org>, 2001)

# Agile Methoden - Prinzipien

- 
7. Working software is the primary measure of progress.
  8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
  9. Continuous attention to technical excellence and good design enhances agility.
  10. Simplicity—the art of maximizing the amount of work not done—is essential.
  11. The best architectures, requirements, and designs emerge from self-organizing teams.
  12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

(Quelle: <http://www.agilemanifesto.org>, 2001)

# Agile Methoden – Extreme Programming

- Mitte der 90er entwickelt von Kent Beck, publiziert 1999 in *Extreme Programming Explained*
- Methode: Bewährte Praktiken der SW Entwicklung ins „Extreme“ zu treiben, z.B.
  - Design → ständiges (Re-)Design (Refactoring)
  - Code Reviews → ständige Code Reviews (Pair Programming)
  - Testen → permanentes Testen
    - Modultest (Pair Programming)
    - Integration
    - Funktionale Tests (Kunde)
  - Risikomanagement → ständiges Managen von Risiken
  - „40-Stunden-Woche“ (keine permanenten Überstunden)
  - Respekt und Mut
  - Tägliche Projektmeetings
  - Aktive Mitarbeit des Kunden

# Agile Methoden – Extreme Programming

## Ziele:

- schnelle Ergebnisse
  - es steht immer ein lauffähiges Produkt zur Verfügung
- schnelle Änderungen
- maximaler Output
- Konzentration auf das Wesentliche
  - wesentliche Features zuerst implementieren
- Entwickler meistern schwierige Situationen gemeinsam (Pair Programming)
- zufriedene Mitarbeiter
- verteiltes Wissen (keine Wissensmonopole)

# Agile Methoden – Extreme Programming

## Pair Programming

- immer zwei Entwickler arbeiten gemeinsam an einem Stück SW, u.U. an einem Computer
  - wechselnden Rollen (Codierer, Tester, Reviewer)
- 
- ➔ permanenter Wissensaustausch
  - ➔ keine Wissensmonopole
  - ➔ kein Wissensverlust bei Personalwechsel
  - ➔ gemeinsame Verantwortung
  - ➔ gemeinsame Problemlösung



**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2 Agile Methoden: Extreme Programming

#### 4.3.3. Agile Methoden: Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

# Agile Methoden - Scrum

## Charakteristika

- Kurze Entwicklungszyklen → Sprint(~4 Wochen)
- Feature orientiert (statt layer-orientiert)
- Lauffähige Software ist **das Hauptziel**, an dem der Projektfortschritt gemessen wird
- Daily Build
- Test first
- Refactoring (Überarbeiten der Software z.B. nach Erkennen von Schwachstellen)



# Agile Methoden - Scrum



## Charakteristika

- Ähnlich Extreme Programming
- Intensive Interaktion mit dem Kunden
- Intensive Kommunikation im Entwicklungsteam (Tägliches kurzes Scrum Meeting)
- Requirements Workshop(s)
- Architektur Workshop(s)
- Requirements Backlog (Scrum spezifisch)
  - Aktuelle Übersicht, welche Anforderungen schon umgesetzt sind und welche gerade in Bearbeitung sind

# Agile Methoden - Scrum

## Charakteristika (speziell bei SCRUM)

- Project Backlog
  - Tagesaktuelle Übersicht, welche Aufwände bereits erbracht wurden (im Projekt und innerhalb einer Iteration)
  - Tagesaktuelle Übersicht, welche Aufwände in der aktuellen Iteration noch zu erbringen sind.
- Selbstorganisierende Teams (z.B. bzgl. der Rollen im Team)
- Team entscheidet, welche Anforderungen im nächsten Zyklus entwickelt werden
- Team kündigt ggf. Anforderungen für einen Zyklus ab
- Dokumentation nicht zwingend, ggf. informell

# Agile Methoden - Scrum

## Diskussion

- Selbstorganisierende Teams ?
- Feature Orientierung versus Layer orientierte Implementierung ?
- Entwicklungszyklus 4 Wochen ?
- Eigenverantwortlichkeit des Teams ?
- Dokumentation ?
- Erfordert auf jeden Fall extreme Disziplin bei den Entwicklern



# Agile Methoden - Scrum

## Folgerung

- Vor der Anwendung im eigenen Projekt detailliert kritisch hinterfragen und im Zweifel geeignet anpassen!



**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Extreme Programming

#### 4.3.3. Scrum

#### 4.4. Prozessbewertung

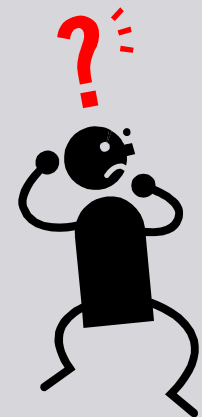
##### 4.4.1. CMMI

##### 4.4.2. SPICE

# Bewertung und Messung von Entwicklungsprozessen

## Motivation

- Wie „gut“ ist mein Entwicklungsprozess ?
  - an sich
  - im Vergleich zu anderen Prozessen
- Was heißt, ein Prozess ist „gut“ oder ein Prozess ist „besser“ als ein anderer?
- Wie kann ich meinen Prozess verbessern?

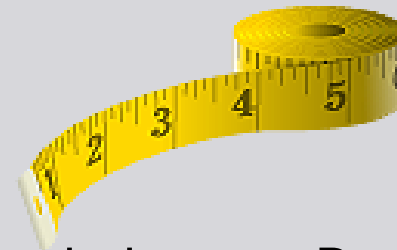


# Bewertung und Messung von Entwicklungsprozessen

## Motivation

Wie wird Prozessqualität gemessen ?

1. indirekt durch Messen der Qualität des Ergebnisses, z.B.
  - Aufwand
  - Kosten
  - Fehler
  - Kundenzufriedenheit
  - Nachteile
    - Messung erst am Ende des Projekts möglich
    - Verbesserungen erst für das folgende Projekt
    - Vergleichbarkeit schwierig
    - fehlender systematischer Ansatz für Verbesserungen





# Bewertung und Messung von Entwicklungsprozessen

## Motivation

Wie wird Prozessqualität gemessen ?

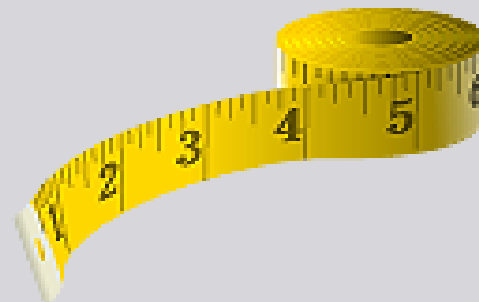
2. direkt durch Messen des Prozesses basierend auf einem entsprechenden Maßstab.

- Messen bevor ein Produkt vorliegt
- unabhängig von der Domäne
- Verbesserungen schon für das laufende Projekt

➔ Standards zur Prozessmessung

➔ CMMI

➔ SPICE



# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Extreme Programming

#### 4.3.3. Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

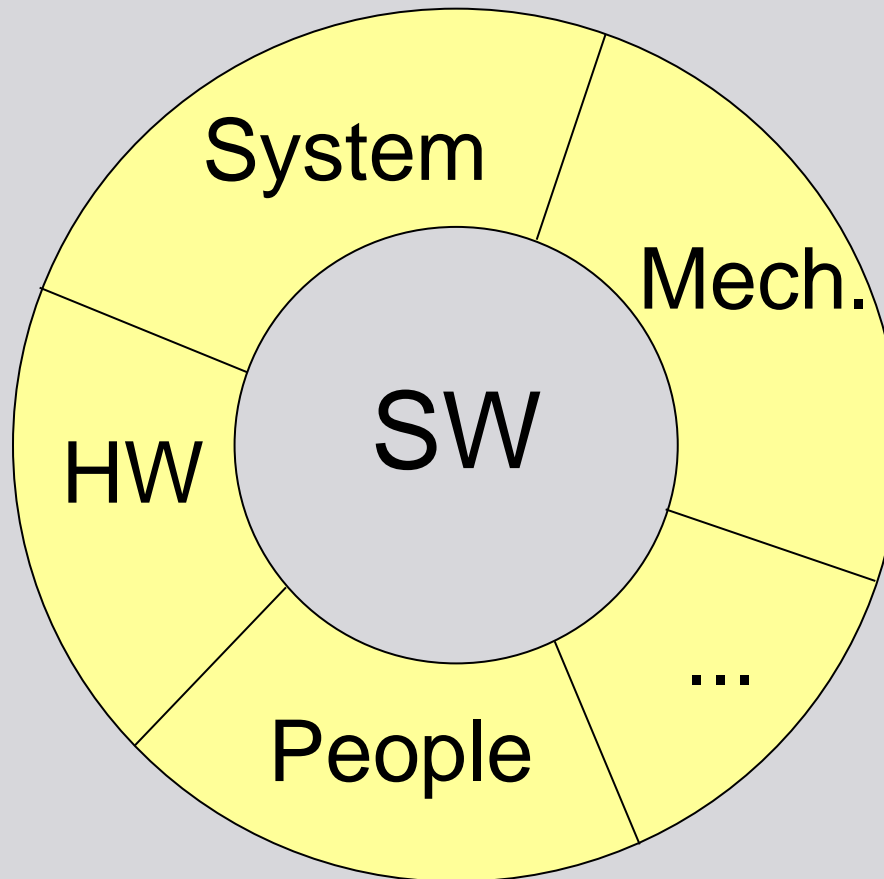
# Bewertung und Messung von Entwicklungsprozessen

## CMMI

- Capability Maturity Model Integration
- Ursprünglich CMM (nur für SW)
- 1986 das Software Engineering Institute (SEI) der Carnegie Mellon University/Pittsburgh, startet auf Initiative des US-Verteidigungsministeriums die Entwicklung eines Systems zur Bewertung der Reife von Softwareprozessen.
- 1991 Capability Maturity Model 1.0
- 1993 CMM Version 1.1
- 1997 Start CMMI Entwicklung (Integration = SW + System + HW + Mechanik + People + ...)
- 2000 CMMI 1.0
- 2002 CMMI Release
- 2006 CMMI 1.2

® Capability Maturity Model, Capability Maturity Modeling, Carnegie Mellon, CMM and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University

# Bewertung und Messung von Entwicklungsprozessen



CMMI

# Bewertung und Messung von Entwicklungsprozessen

## CMMI

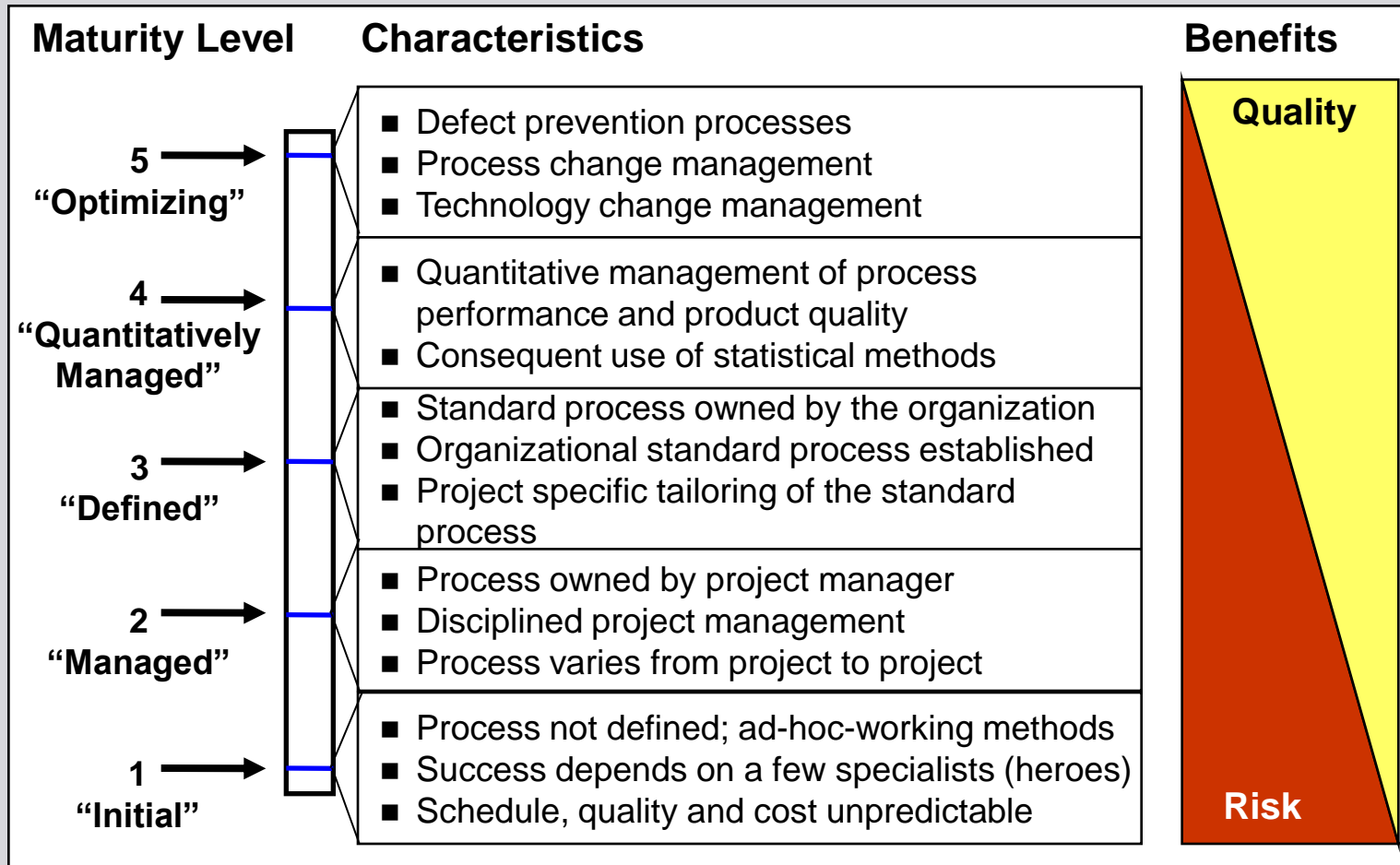
- definiert weder einen konkreten Prozess noch konkrete Prozessverbesserung.
  - ➔ Anwendbarkeit auf unterschiedliche Organisationen und Organisationsgrößen
  - ➔ Anwendbarkeit in verschiedene Domänen (z.B. Automotive, Medizin, Anlagentechnik)
  - ➔ Es gibt nicht „die eine“ richtige Umsetzung von CMMI
- Fokus auf das „**Was zu tun ist**“, nicht auf das „**Wie etwas zu tun ist**“
- kontinuierlichen Prozessverbesserung als Ziel
- Definition der Anforderungen und Kriterien an eine professionelle Entwicklungsorganisation

# Bewertung und Messung von Entwicklungsprozessen

## CMMI

- bewertet den Prozess in der Theorie und Praxis
  - Prozess auf dem Papier (Theorie)
  - konkrete(s) Projekt(e) (Umsetzung der Theorie in der Praxis)
  - alle Aspekte des Projekts , sowohl technisch als auch organisatorisch (z.B. Planung, Planverfolgung, Kommunikation)
  - Aufgaben der Firmenorganisation, z.B.
    - Bereitstellung von Ressourcen
    - Durchführung von Trainingsmaßnahmen
    - Prozessverbesserung
    - Technologiemanagement
  - durch zertifizierte Auditoren

# Bewertung und Messung von Entwicklungsprozessen



© Capability Maturity Model, Capability Maturity Modeling, Carnegie Mellon, CMM and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University

# Bewertung und Messung von Entwicklungsprozessen

## CMMI - Zertifizierung

- aufwendig, nur für größere Organisationen sinnvoll
- Fragebogen, der alle sog. Key Process Areas abfragt
- enthält Bewertung und i.a. Vorschläge zur weiteren Verbesserung in Richtung des nächsten CMMI Levels
- 2-3 Jahre werden meist benötigt um von einem Level auf das nächste zu kommen, d.h. Sprünge von 1 auf 5 sind unwahrscheinlich



# Bewertung und Messung von Entwicklungsprozessen

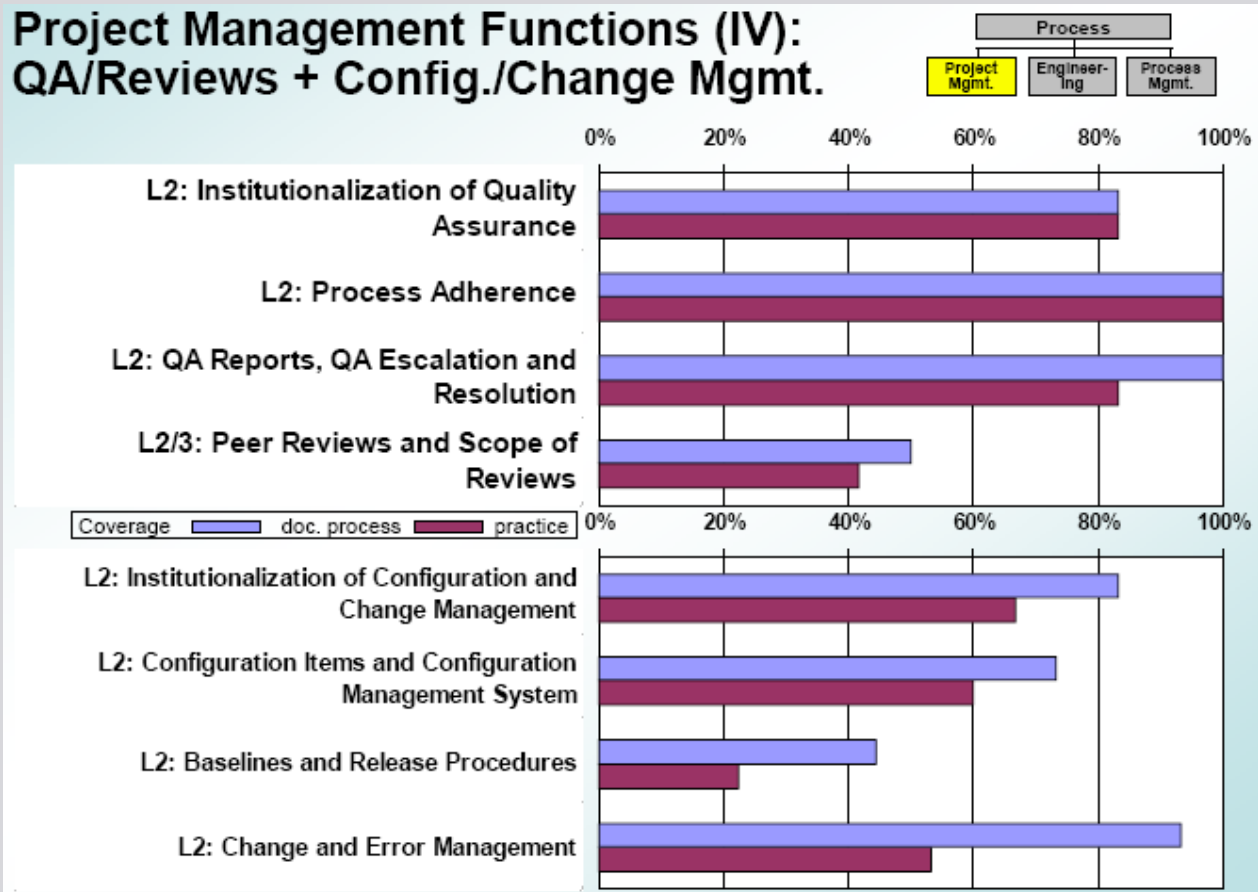
## Beispiel CMMI Level 2 – Fragen\*)

	Yes	No	Does Not Apply	Don't Know
3 Does the project follow a written organizational policy for managing the system requirements allocated to software? .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments:				
4 Are the people in the project who are charged with managing the allocated requirements trained in the procedures for managing allocated requirements? .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments:				
3 Do all affected groups and individuals agree to their commitments related to the software project? .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments:				

\*) Quelle: [www.sei.cmu.edu](http://www.sei.cmu.edu)

# Bewertung und Messung von Entwicklungsprozessen

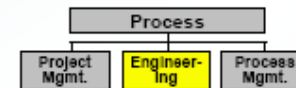
## Beispiel CMMI – Assessment Ergebnisse



# Bewertung und Messung von Entwicklungsprozessen

## Beispiel CMMI – Assessment Ergebnisse

### Engineering Functions (II): Summarized Findings



#### Definition of Features and Architecture (TS):

- ☹ There is a general idea (not yet a process) how to break down system requirements to components and how to describe that using Doors.
- 😊 HW interfaces (PIN usage) are usually documented in separate documents.
- 😊 Defined process for SW requirements management (SRS, PSRS).
- ☹ IP activities more on organizational level than in projects.
- ☹ RM tool only used on system level so far. Therefore only limited tracing by naming conventions.
- ☹ Safety criticality of increasing relevance for BC, but lack of process and awareness

#### SW Implementation & Test (SW):

- 😊 SW architects established, documentation (using UML) ongoing
- ☹ Design documents without formal description
- 😊 Coding guidelines for C enforced by tool and code reviews.
- 😊 SW test process (3 phases, new focus on module test, automation) defined, piloted and ready for roll out.

# Bewertung und Messung von Entwicklungsprozessen

## **CMMI – Erwartungen durch höheres Level**

- Höhere Termin- und Budgettreue
- Zuverlässigeres Erreichen der geplanten Ergebnisse
- Geringere Kosten, höhere Produktivität und bessere Qualität
- Bessere Vorhersagbarkeit des Projektverlaufs
- Höhere Kundenzufriedenheit
- Höhere Mitarbeiterzufriedenheit

(Identische Erwartungen knüpfen sich auch an höhere SPICE Levels)

# Inhalt

## 4. Vorgehensmodelle / Prozeßmodelle (Beschreibung, Vorteile, Nachteile, Bewertung)

### 4.1. Wasserfall

### 4.2. V-Modell

### 4.3. Iterative / Agile Methoden

#### 4.3.1. RUP

#### 4.3.2. Extreme Programming

#### 4.3.3. Scrum

### 4.4. Prozessbewertung

#### 4.4.1. CMMI

#### 4.4.2. SPICE

# Prozessbewertung / Prozessmessung SPICE

## SPICE (Software Process Improvement and Capability Determination) oder ISO/IEC 15504

- internationaler Standard zur Durchführung von Bewertungen (Assessments) von Unternehmensprozessen mit Schwerpunkt auf der Softwareentwicklung.
- Basiert u.a. auf CMMI
- 2006 internationaler Standard (IS)
- Fokus auf Verbesserung von Prozessen in der eigenen Organisation und auf die Prozessfähigkeit von Lieferanten (Capability Determination)
- Automotive SPICE als „Teilmenge“, definiert durch die (Automobil-) *Hersteller Initiative Software* (HIS, u.a. VW, Daimler, BMW)

# Prozessbewertung / Prozessmessung SPICE

## SPICE

- Bewertung des Entwicklungsprozesses durch Assessments
- Unterteilung in Prozess- und Reifegraddimension, d.h.
  - Unterteilung in (Sub-)Prozesse
  - Jeder (Sub-) Prozess erhält seinen eigenen Reifegrad
- Unabhängige Bewertung der Prozessdimensionen
- Identifikation von Verbesserungsmöglichkeiten

# Prozessbewertung / Prozessmessung SPICE

## Prozesskategorien

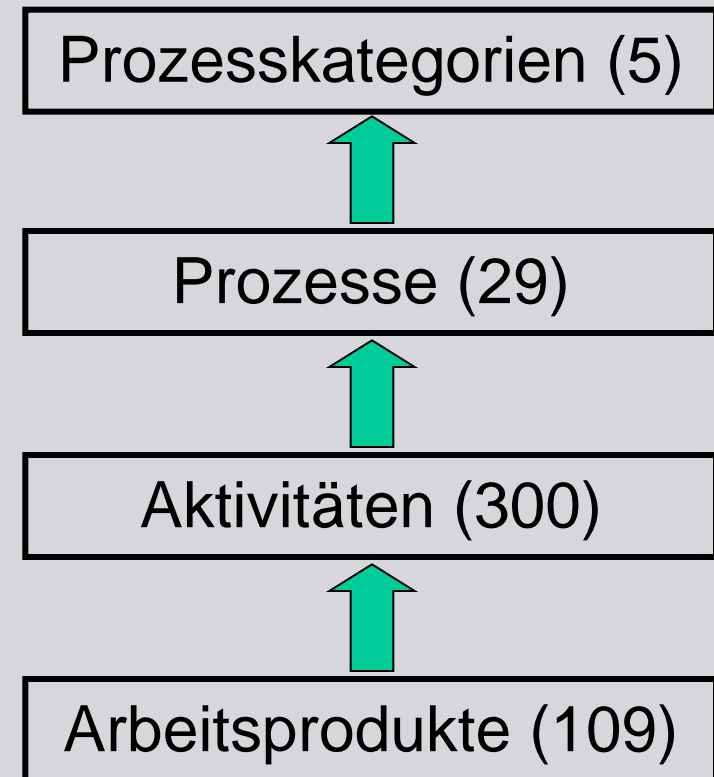
- Kunden-Lieferanten (Akquisition, Betreuung, Lieferung, Einsatz)
- Entwicklungsprozess (Engineering)
- Unterstützende Prozesse (Konfigurationsmanagement)
- Managementprozess (Projektmanagement)
- Organisationsprozess (Erreichen von Organisationszielen, Personalmanagement)



# Prozessbewertung / Prozessmessung SPICE

## SPICE-Referenzmodell:

- 29 Prozesse, zugeordnet jeweils einer Prozesskategorie;
- Jeder Prozess selbst wird durch grundlegende Aktivitäten beschrieben;
- Aktivitäten definieren Aufgaben, um das Prozessziel zu erreichen;
- Arbeitsprodukte sind Ein- und Ausgaben für Prozesse.



# Prozessbewertung / Prozessmessung SPICE

## SPICE-Reifegrade

- Die Einstufung der Leistungsfähigkeit von Prozessen erfolgt durch Prozessattribute, die in Reifegradstufen eingeteilt sind.
- Der nächste Reifegrad enthält Hinweise für Prozessverbesserung
- Messbare Prozess-Attribute („vollständig erfüllt“, „weitgehend erfüllt“, „teilweise erfüllt“, „nicht erfüllt“) beurteilen Prozesse

Rating		Scale	Comment
N	Not achieved	0% - 15%	There is little or no evidence of achievement of the defined attribute in the assessed process.
P	Partially achieved	16% - 50%	There was evidence of a sound systematic approach to and achievement of the defined attribute in the assessed process. Some aspects of achievement may be unpredictable.
L	Largely achieved	51% - 85%	There was evidence of a sound systematic approach to and significant achievement of the defined attribute in the assessed process. Performance of the process may vary in some areas or work units
F	Fully achieved	86% - 100%	There was evidence of a complete and systematic approach to and full achievement of the defined attribute in the assessed process. No significant weaknesses existed across the defined organizational unit.

# Prozessbewertung / Prozessmessung SPICE

## SPICE- Reifegrade

5. Stufe: Optimierender Prozess (Der Prozess wird kontinuierlich verfeinert und verbessert)

4. Stufe: Vorhersagbarer Prozess (Der Prozess ist quantitativ verstanden und kontrolliert)

3. Stufe: Etablierter Prozess (Die Ausführung des Prozesses ist standardisiert)

2. Stufe: Gesteuerter Prozess (Die Ausführung des Prozesses wird geplant und gesteuert)

1. Stufe: Durchgeführter Prozess (Der Zweck des Prozesses wird erfüllt)

0. Stufe: Unvollständiger Prozess



# Prozessbewertung / Prozessmessung SPICE

## SPICE-Fragenkatalog – Beispiel Anforderungserhebung

1. Ist eine dauerhafte Kommunikation mit dem Kunden eingerichtet?
2. Sind die vereinbarten Kundenanforderungen definiert und als
3. Baseline festgelegt?
4. Ist ein Änderungsmechanismus geschaffen, mit dessen Hilfe die Änderungen der Kundenanforderungen bewertet und in die Basis Anforderungen integriert sind? (Die Änderungen der Anforderungen beruhen dabei auf den sich ändernden Kundenbedürfnissen.)
5. Ist ein Mechanismus für die ständige Überwachung der Kundenbedürfnisse eingeführt?
6. Ist ein Mechanismus eingeführt, mit dem gewährleistet wird, dass die Kunden den Status und die Verwendung ihrer Anfragen bequem ermitteln können?

# Prozessbewertung / Prozessmessung SPICE

## SPICE-Ergebnis (Beispiel)

- |    |   |  |  |
|----|---|--|--|
| 6  | L | BP 6 Develop detailed design   | <ul style="list-style-type: none"><li>- Zu großen Teilen Reverse-Engineering, u.a. auch wegen Übernahme</li><li>- Das Detailed-Design befindet sich thw. nur auf Code-Niveau (kein Mehrwert zu Code!)</li></ul>  |
| 7  | P | BP 7 Develop Test Criteria   | <ul style="list-style-type: none"><li>- Test Kriterien fehlen. Es wird auf Ebene SDD (gegen SDD) nicht getestet.</li></ul>   |
| 8  | L | BP 8 Verify Software Design  | <ul style="list-style-type: none"><li>- für Architektur Delta-Review (Zuordnungstabelle war aber z.B. nicht angepasst)</li><li>- 2 Delta-Reviews für die Klemmensteuerung mit nur 3 info-Statements</li></ul>  |
| 9  | P | BP 9 Ensure consistency and bilateral traceability to software requirements          | <ul style="list-style-type: none"><li>- Abbildung Requirements -&gt; Funktion via Tabelle in SDD dargestellt.<br/>Stichproben ergaben Fehler, was nicht verwundert, da diese Form der Darstellung nicht praktikabel ist.</li><li>- SW-Architektur und Detailed-Design werden nicht zur Ableitung von SW-Integrationstestfällen verwendet</li></ul> |
| 10 | L | BP 10 Ensure consistency and bilateral traceability to software architectural design | <ul style="list-style-type: none"><li>- Keine Verifikationskriterien und daher auch keine Berücksichtigung eben dieser</li><li>- Ansonsten prinzipiell i.O. da für jeden Block in Architektur eine SDD existiert</li></ul>   |

# Vergleich CMMI - SPICE

SPICE	CMMI
Reifegrade für einzelne Prozesse ➔Man kann nicht “SPICE Level3 erreichen”	Reifegrad der gesamten Organisation ➔Man kann CMMI Level 3 erreichen
Unabhängig von der Domäne, anpassbar, z.B. Automotive SPICE	Unabhängig von der Domäne starr
Ein Level pro Prozess (0-5)	Ein Level für die Organisation (1-5)
Assessment erfordert detaillierte Sachkenntnis des Auditors	Assessment erfordert detaillierte Sachkenntnis des Auditors
4 Antwortstufen	2 Antwortstufen
Eigenbewertung	Fremdbewertung

# Prozessbewertung / Prozessmessung – zum Schluss

## Bei Risiken und Nebenwirkungen ...

- Ohne Unterstützung des Managements (aktives Sponsoring) nicht sinnvoll
- Prozessverbesserung braucht Aufwand und Zeit
- Prozesse zu definieren ist (relativ) einfach, angemessene Prozesse einzuführen ist schwierig
- Prozessverbesserung darf nicht zum Selbstzweck werden
- Fokus bei CMMI und SPICE heute eher auf „konventionelle“ Prozesse (Wasserfall)
- Wie funktioniert Prozessverbesserung bei agilen Prozessen?
- Prozess ist nicht Alles!



# Übung

## Erfolgsfaktoren

Welche Faktoren außer Prozessqualität können den Erfolg / Misserfolg eines Projekts noch beeinflussen ?

Begründen Sie Ihr Ergebnis!

10 min, arbeiten Sie ggf. zusammen mit einem Partner





**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**