V03 - Vorgehensmodelle

Softwaretechnik 1

Wintersemester 2021/2022

Prof. Dr.-Ing. Jasminka Matevska

(jasminka.matevska@hs-bremen.de)

Hochschule Bremen, Fakultät Elektrotechnik und Informatik, TI, ISTI, DSI & ATMEI

Rechtliche Hinweise

- Die Rechte an geschützten Marken liegen bei den jeweiligen Markeninhabern
- Die Rechte an referenzierte Literatur, Folien, Notizen und sonstigen Materialien liegen bei den jeweiligen Autoren
- Diese Veranstaltung benutzt Materialien von Prof. Dr. Spillner (HSB in Ruhestand) und Prof. Dr. Hasselbring (Universität Kiel)
- Alle Rechte an den Materialien zu dieser Veranstaltung liegen bei ihrem Autor, Prof. Dr.-Ing. Jasminka Matevska
- Jede Form der teilweisen oder vollständigen Weitergabe, Speicherung auf Servern oder Nutzung in Lehrveranstaltungen, die nicht von dem Autor selbst durchgeführt werden, erfordert seine schriftliche Zustimmung. Eine schriftliche Zustimmung ist darüber hinaus für jede kommerzielle Nutzung erforderlich
- Für inhaltliche Fehler kann keine Haftung übernommen werden

Der **rote** Faden

- Einführung
- Entwicklungsprozess
- **Vorgehensmodelle**
- Anforderungsanalyse
- Entwurf
- Objektorientierter Entwurf mit UML
- Test
- Qualitätssicherung
- Konfigurationsmanagement
- Implementierung
- Projektmanagement

Gliederung

- Vorgehensmodell - Definition
- Wasserfallmodell/Phasen und Ergebnisse
- Wasserfall-Verbesserung
  - Iterative/inkrementelle/evolutionäre Modelle
  - Das Spiral-Modell
  - Prototyping zur Unterstützung der Iterationen
  - Rational Unified Process (RUP)
- Agile Vorgehensmodelle
- Das V-Modell

Vorgehensmodell

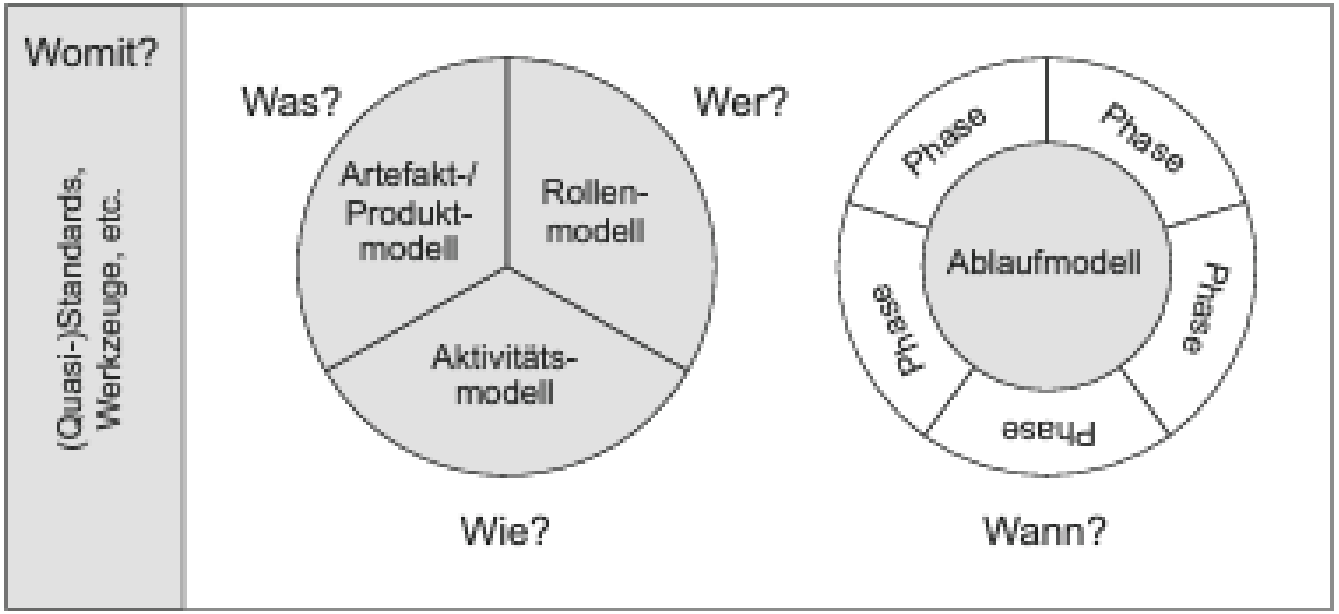
- Was ist ein Vorgehensmodell?
- Wozu können Vorgehensmodelle nutzen?
- Kennen Sie ein Vorgehensmodell?

Was ist ein Vorgehensmodell?

- Ein Vorgehensmodell ist das Bindeglied zwischen den Aufgaben des Projektmanagements und technischen Aufgaben der Softwaretechnik
- Vorgehensmodelle bilden die Grundlage für Projektplanung, -Überwachung und –Steuerung
- Ein Vorgehensmodell
  - Beschreibt systematische, ingenieurmäßige und quantifizierbare Vorgehensweisen, um Aufgaben einer bestimmten Klasse wiederholbar zu lösen
  - Beschreibt die Schnittstellen des Projekts zur Organisation, in die das Projekt eingebettet ist
  - Ist ein Instrument zur Integration von Methoden, die für die Lösung eng umgrenzender Teilaufgaben verwendet werden

[Broy, Kuhrmann, 2013]

Bestandteile eines Vorgehensmodells



Frage	Prozess-element	Beispiel
Was?	Artefakt	Projektplan, Modelle, Code, Berichte, Anwenderdokumente
Wer?	Rolle	Projektleiter, Entwickler, Tester
Wie?	Aktivität	„Erstelle UML-Modell“, „Schreibe Unit Tests“

[Broy, Kuhrmann, 2013]

Vorgehensmodell

- Ein Vorgehensmodell beantwortet die Fragen:
  - **Wer** erarbeitet **Was** und **wie** geht er dabei vor?
  - **Welche** Phase (logisch zusammenhängende Aktivitäten) wird **wann** durchgeführt?
  - **Wann** und unter **welchen Kriterien** kann ein Übergang zwischen den Phasen stattfinden?
- Grundlegende Elemente/Bestandteile:
  - Artefakte
  - Rollen
  - Aktivitäten/Phasen
  - Werkzeuge/Standards

Das 0. Vorgehensmodell der Software-Entwicklung

- Code & Fix
  1. Schreibe ein Programm!
  2. Finde und behebe die Fehler in dem Programm!
- Nachteile:
  - Bei der Behebung von Fehlern wird das Programm häufig so umstrukturiert, dass es schlechter verständlich wird und weitere Fehlerbehebungen immer teurer werden
- → Erkenntnis: eine **Entwurfsphase** vor der Programmierung wird benötigt

Weiterhin Probleme trotz Entwurfsphase

- Selbst gut entworfene Software wird vom Kunden/Endbenutzer oft nicht akzeptiert

  → Dies führt zu der Erkenntnis, dass eine **Anforderungsdefinition** vor dem Entwurf benötigt wird

- Fehler sind schwierig zu finden, da Tests schlecht vorbereitet und Änderungen unzureichend durchgeführt werden

  → Dies führt zu einer separaten **(Integrations-) Testphase**

- Auch nach der Fertigstellung muss die Software weiterentwickelt werden

  → Dies führt uns zum **Wasserfallmodell**

Übung

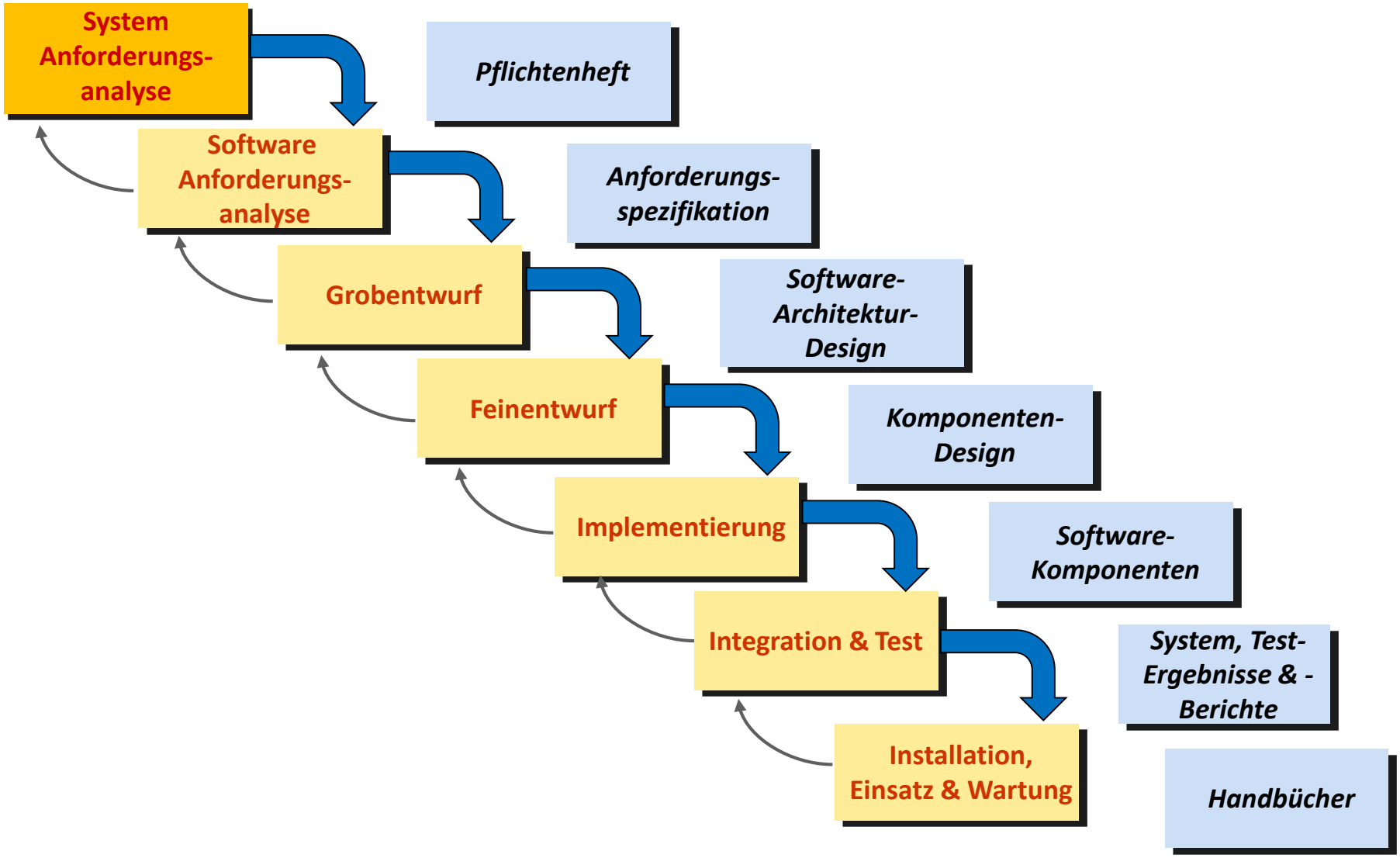
- Was ist ein Vorgehensmodell?
- Welche Fragen beantwortet ein Vorgehensmodell?
- Welche Bestandteile hat ein Vorgehensmodell?
- Was sind die Nachteile/Probleme des 0. Vorgehensmodell der Software-Entwicklung bzw. der ersten Erweiterung?

Wasserfallmodell

http://pixabay.com/en/waterfall-water-river-green-283145/

- Produkte einer Phase fallen wie bei einem Wasserfall in die nächste Phase
- Erstes umfassendes Modell für Software Engineering
- Basiert auf dem **stagewise model** von Benington [1956]
- Weiterentwicklung u.a. durch Royce [1970 & 1987] und Boehm [1981]

Wasserfallmodell



Eigenschaften des Wasserfallmodells

- Software wird in sukzessiven Stufen entwickelt
  - Jede Aktivität ist in der richtigen Reihenfolge und in der vollen Breite vollständig durchzuführen
  - Der Entwicklungsablauf ist sequentiell. Jede Aktivität muss beendet sein, bevor die nächste anfängt
  - Am Ende jeder Phase stehen fertige Ergebnisse und Dokumente
  - Rückkopplungsschleifen zwischen den Stufen sind nur begrenzt möglich
- Nutzerbeteiligung ist nur in den ersten Phasen vorgesehen
- Das Modell ist einfach, verständlich und benötigt relativ wenig Koordinierungsaufwand
- **Wichtig als Grundlage für alle anderen Modelle → Phasen und Ergebnisse**

Anforderungsanalyse: Was?

- Ergebnis: **Pflichtenheft**
- Basiert evtl. auf einer Machbarkeitsstudie
- Produziert funktionale und nicht-funktionale Systemanforderungen, Zeitverhalten der einzelnen Funktionen, Schnittstellen, Ziel-Hardwareumgebung, etc.
- Eine der schwierigsten Aufgaben in der Software-Entwicklung besteht darin, die tatsächlichen Anforderungen der Kunden/Anwender zu ermitteln
- Interaktion zwischen Kunden/Anwender und Entwickler ist nötig

Grob- und Feinentwurf: Wie?

- Ergebnis: **Systemarchitekturdesign, Komponentendesign**
- Der Grob- bzw. Architekturentwurf produziert Spezifikation der System-Architektur, Schnittstellen zwischen den Komponenten mit Definition deren Import-/Export-Beziehungen
- Der Feinentwurf spezifiziert dann die Eigenschaften der einzelnen Komponenten

Implementierung, Integration und Test

- Implementierung → **programming-in-the-small**
  - Ergebnis: **Software-Komponenten**
  - Produziert Modulrümpfe in einer ausführbaren Sprache (Programmiersprache) nach Vorgabe der Systemarchitektur
  - Testet einzelne Module (Unit-Tests)
  - Erstellt Programmdokumentation
- Integration und Test → **programming-in-the-large**
  - Ergebnis: **Integriertes Softwaresystem, Testpläne, Testergebnisse und Berichte**
  - Fügt die einzelnen Module zum System zusammen
  - Produziert Testpläne, Testtreiber und Testergebnisse
  - Führt Tests (Funktionale und Belastungstests) von Modulen (Teilsysteme, Komponenten) und vom Gesamtsystem durch

Installation, Inbetriebnahme und Abnahme

- Ergebnis: **Technische Handbücher, Benutzungshandbücher**
- Produziert Installation der lauffähigen Software auf den Zielmaschinen, Installationsanleitung, Überführung vorhandener Daten, Schulung, Nutzungsbeschreibung
- Mit der Übernahme verbunden ist im Allgemeinen ein Abnahmetest
- Ab diesem Zeitpunkt unterliegt das Produkt dann der Wartung und Pflege

Wartung

- Ergebnis: **aktualisiertes Softwaresystem und Dokumentation**
- Ziel
  - Sicherung der korrekten und konsistenten Funktionsfähigkeit während des Einsatzes → **zufriedener Kunde**
  - Verbesserung des Rufs und der Marktposition des Anbieters
- Aufgaben
  - Korrektur von Fehlern (Programmfehler, Spezifikationsfehler, Analysefehler, Dokumentationsfehler, ...)
  - Anpassungen
  - Erweiterungen
- Wichtig für den Betrieb u.a.
  - Überwachungsmöglichkeiten (Monitoring)
  - Unterstützung zur Fehlerlokalisation und –Diagnose
  - Adaption bzw. Selbstadaption
  - Online-Tests

Probleme mit dem Wasserfallmodell 1/2

- Vollständige Durchführung aller Phasen ist nicht immer sinnvoll
- Sequentielle Phasendurführung problematisch
  - Späte Änderungen fließen nicht in die vorrangegangenen Phasen
  - System-Probleme werden oft sehr spät erkannt
- Sequentielle Phasendurchführung ist in der Realität nicht machbar. Häufig sind Rückschritte nötig weil:
  - Falsche Anforderungsdefinition: falsch verstanden oder gesagt, nicht realisierbar
  - Tatsächlich Kundenanforderungen können oft nur durch den praktischen Einsatz eines Systems überprüft bzw. ermittelt werden
  - Anforderungen nicht vollständig bekannt
  - Anforderungen verändern sich im Laufe der Entwicklung

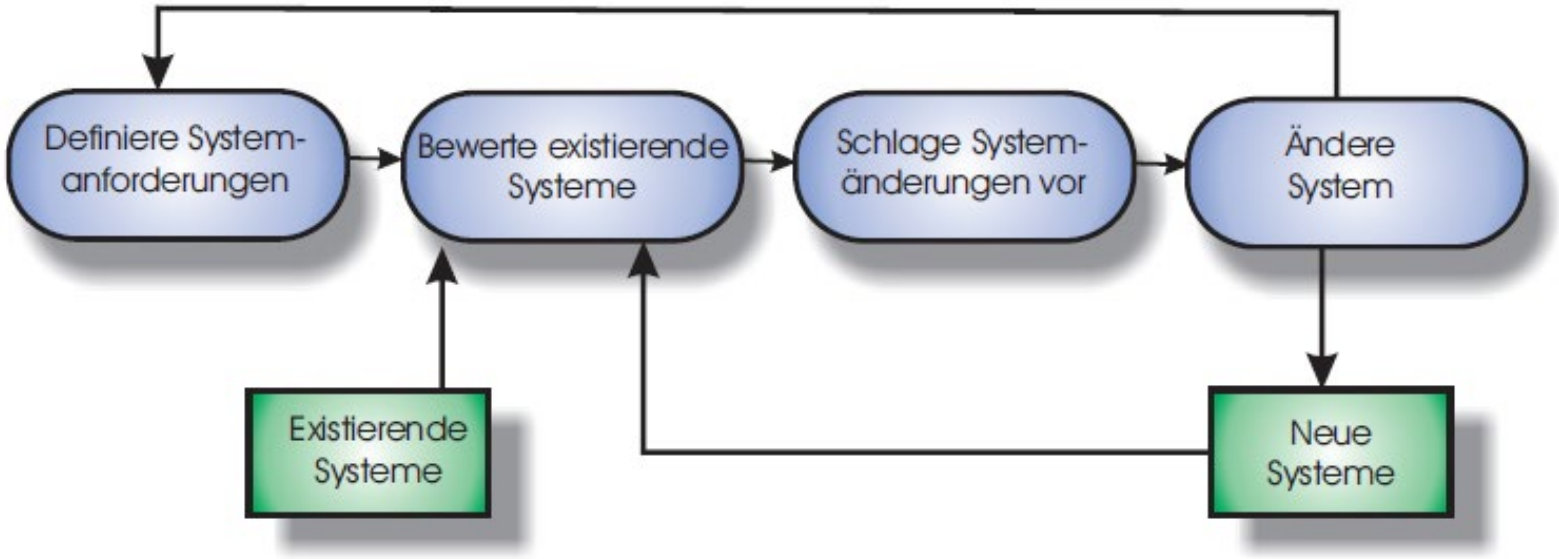
Probleme mit dem Wasserfallmodell 2/2

- Gefahr, dass die Dokumentation wichtiger wird als das eigentliche System
- Risikofaktoren werden u.U. zu wenig berücksichtigt, da der einmal festgelegte Entwicklungsablauf durchgeführt wird
- Nur die (optionale) Machbarkeitsstudie dient der einmaligen Risikoabschätzung
- Unrealistische Kosten- und Aufwandschätzung insbesondere für spätere Phasen
- In der Praxis werden die einzelnen Phasen daher nicht streng sequentiell, sondern nebenläufig durchlaufen
  - Höhere Effizienz, aber auch höherer Koordinationsaufwand

Übung

- Was sind die Eigenschaften des Wasserfallmodells?
- Welche Phasen sieht das Wasserfallmodell vor?
- Welche Ergebnisse bzw. Produkte werden in den Phasen erzeugt?
- Erklären Sie den Unterschied zwischen programming-in-the-small und programming-in-the-large
- Welche Wartungsaufgaben gibt es?
- Welche Probleme gibt es mit dem Wasserfallmodell?

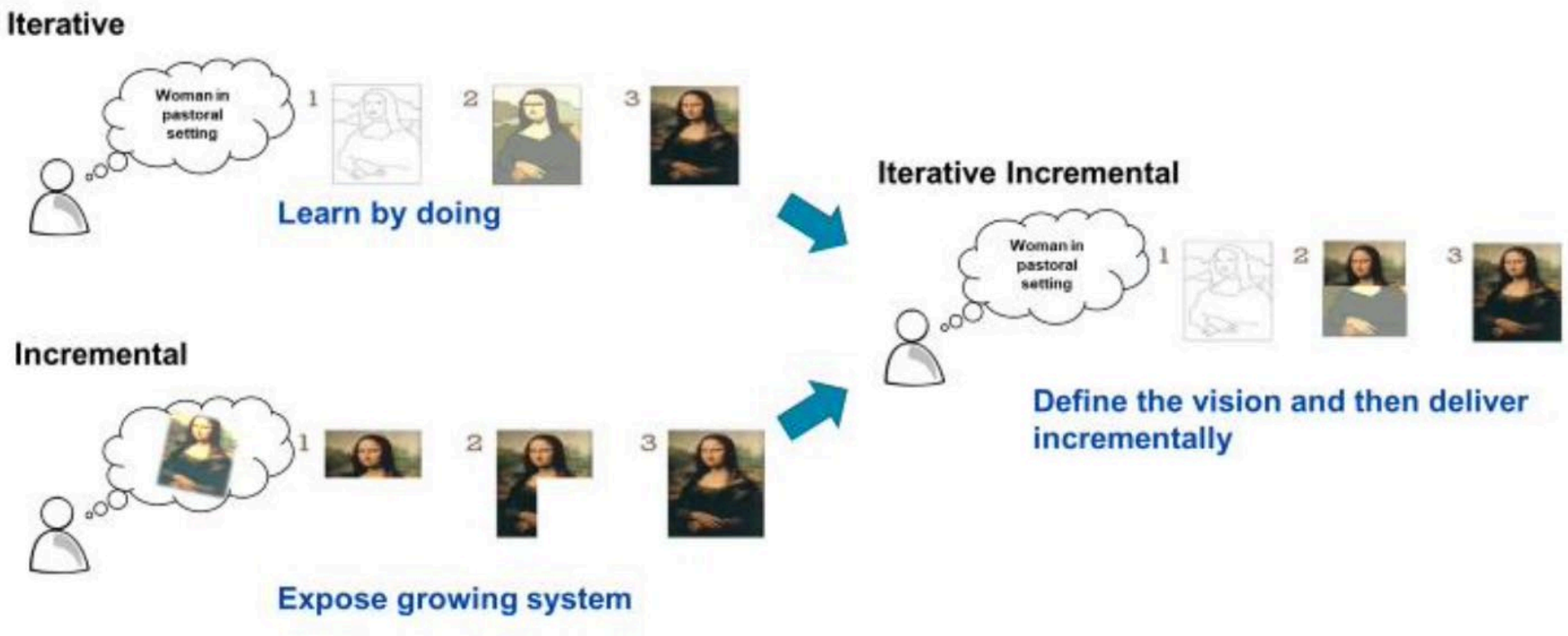
Wasserfall-Verbesserung:
Iterative, inkrementelle und evolutionäre Vorgehensmodelle



[Sommerville, 2012]

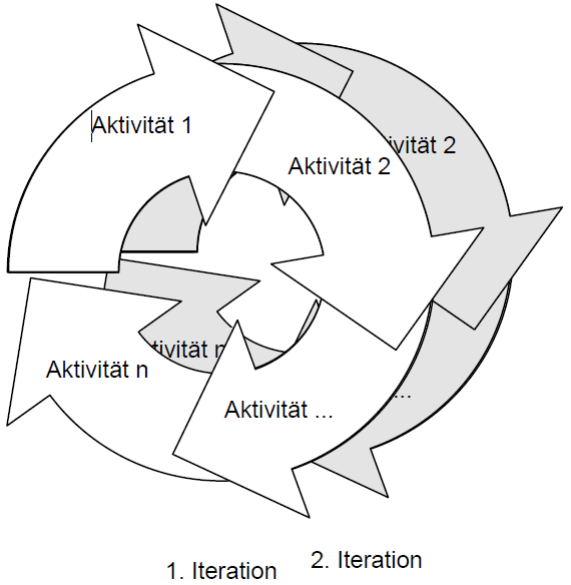
- Iteration (von lat. *iterare*) → Wiederholen
  - Prozess des mehrfachen Wiederholens gleicher oder ähnlicher Handlungen zur Annäherung an eine Lösung oder ein bestimmtes Ziel. [Wikipedia]
- Inkrement (von lat. *incrementare*) → Vergrößern
  - Pro Inkrement wächst der Umfang und/oder die Anzahl der Funktionen.

Iterativ/inkrementelle Vorgehensweise



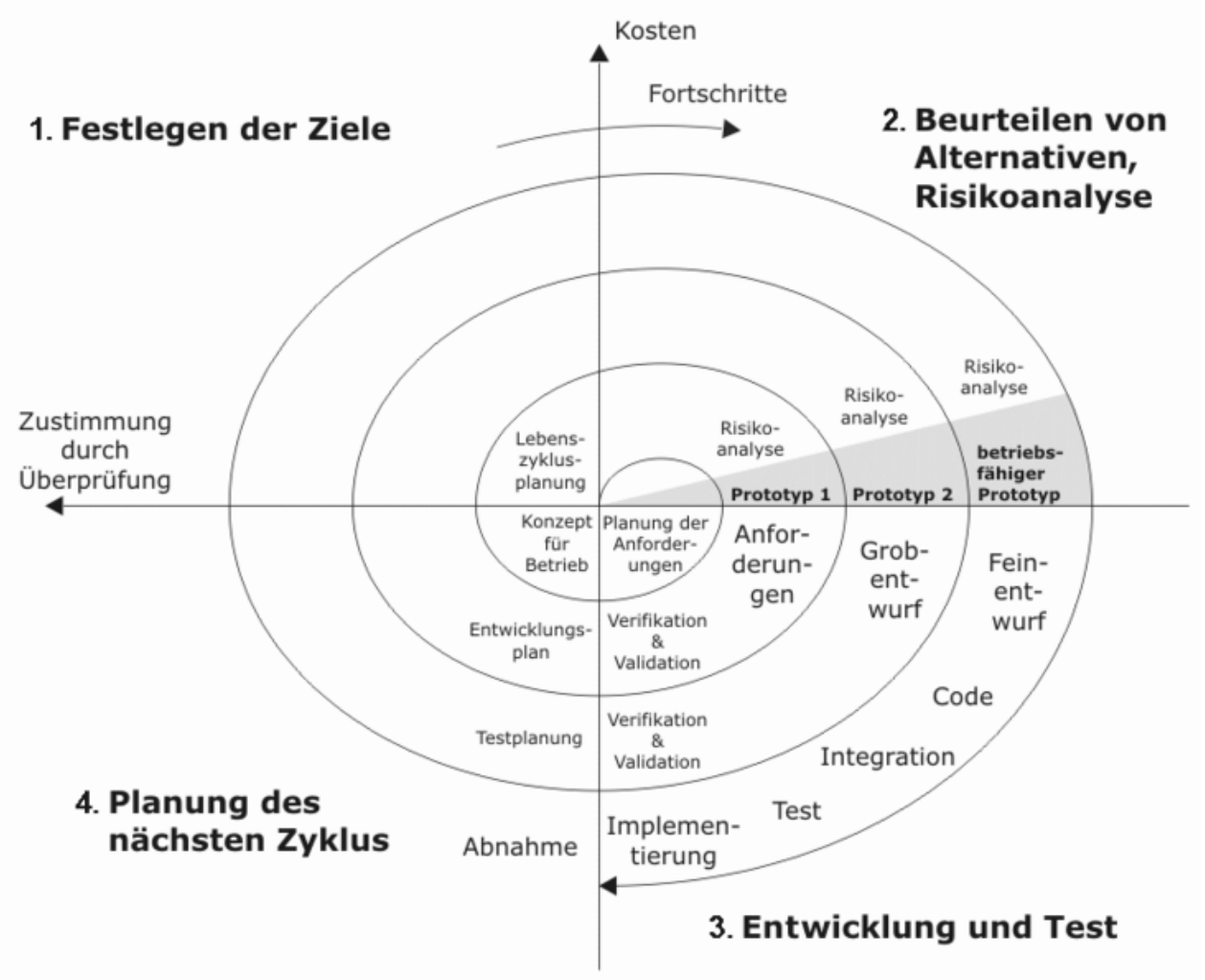
Iterativ/inkrementelle Vorgehensmodelle

- Das System wird schrittweise in sog. Inkrements entwickelt
  - Zuerst wird eine eingeschränkte Funktionalität entwickelt
  - Jeder neue Inkrement erweitert die Funktionalität des Systems
- Das ermöglicht die Risiken und Kosten besser zu kontrollieren
- Jeder Inkrement erfordert implizit min. eine Iteration
- Agile Methoden (z.B. Scrum) stellen eine besondere Variante der inkrementell/iterativen Vorgehensmodellen



Iteratives/evolutionäres Vorgehen: Das Spiralmodell

- Boehm, 1988
- Ziel: Minimierung der Projektrisiken
- Prototypgetrieben
- Wiederholtes Durchlaufen der klassischen Entwicklungsphasen in Iterationen von jeweils vier Schritten
  1. Analyse
  2. Evaluierung
  3. Realisierung
  4. Planung

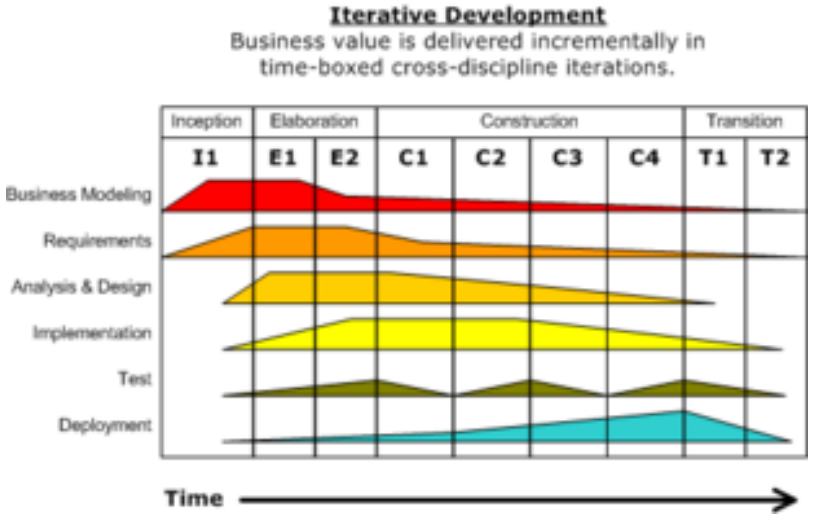


Rational Unified Process (RUP)

- Der Rational Unified Process (RUP) ist ein objektorientiertes, aktivitätsgetriebenes Vorgehensmodell, das seit 1999 von Rational (später IBM Rational) gepflegt und als kommerzielles Produkt vertrieben wird
- Initial von Kruchten 1998 vorgestellt
- RUP ist ein spezifisches Vorgehensmodell zur Softwareentwicklung auf Basis der Unified Modelling Language (UML)
- Im Kern folgt der RUP den folgenden drei Grundprinzipien:
  - RUP ist Anwendungsfall-getrieben
  - Die Architektur steht im Zentrum der Planung
  - Das Vorgehen zur Entwicklungszeit ist iterativ/inkrementell

Grundkonzepte

- RUP unterscheidet zwischen einzelnen Lebenszyklusabschnitten (Phasen) inkl. der Iterationen in den einzelnen Abschnitten
  1. Inception (Konzeptionsphase)
  2. Elaboration (Entwurfsphase)
  3. Construction (Konstruktionsphase)
  4. Transition (Übergabephase)
- In jeder der vier Phasen werden Aktivitäten (Arbeitsschritte) einzelner Disziplinen gebündelt
  - Business Modeling
  - Requirements
  - Analysis & Design
  - Implementation (iterativ)
  - Test
  - Deployment (Auslieferung)



RUP-Gebirge [Kruchten 2003]

Prototyping zur Unterstützung der Iterationen

- Der Einsatz von Prototypen unterstützt die Arbeit in Iterationen
- Prototyping ist keine ad-hoc Vorgehensweise sondern systematisiert die Erstellung ablauffähiger Demonstratoren
- Typen von Prototypen (Zweck)
  - Evolutionär
  - Explorativ
  - Experimentell
- Typen von Prototypen (Abdeckung)
  - Horizontal
  - Vertikal

[nach Floyd 1984]

Evolutionäre Prototypen

- Ziel ist, mit den Anwendern/Kunden von einer initialen Version per Evolution zu einem nutzbaren System zu kommen
- Voraussetzung sind einigermaßen gut verstandene (Teil-) Anforderungen
- Dieser Prototyp wird in mehreren inkrementellen Schritten zu einem vollständigen Produkt weiterentwickelt
- Wesentlich ist bei dieser Art der Prototypenerstellung die konsequente Beachtung von Entwicklungs- und Qualitätsstandards
- Typische Verwendung im Spiral-Modell

[nach Floyd 1984]

Explorative Prototypen (Throw-away Prototype)

- Ausschließliches Ziel: die Anforderungen besser zu verstehen
- Dieser Prototyp wird entwickelt, um neue Erkenntnisse zu gewinnen und Ideen zu überprüfen
- Die Beziehung zwischen Prototyp und fertiger Software ist nicht festgelegt
- Wird in agilen Methoden regelmäßig eingesetzt (Scrum, Design Thinking)

[nach Floyd 1984]

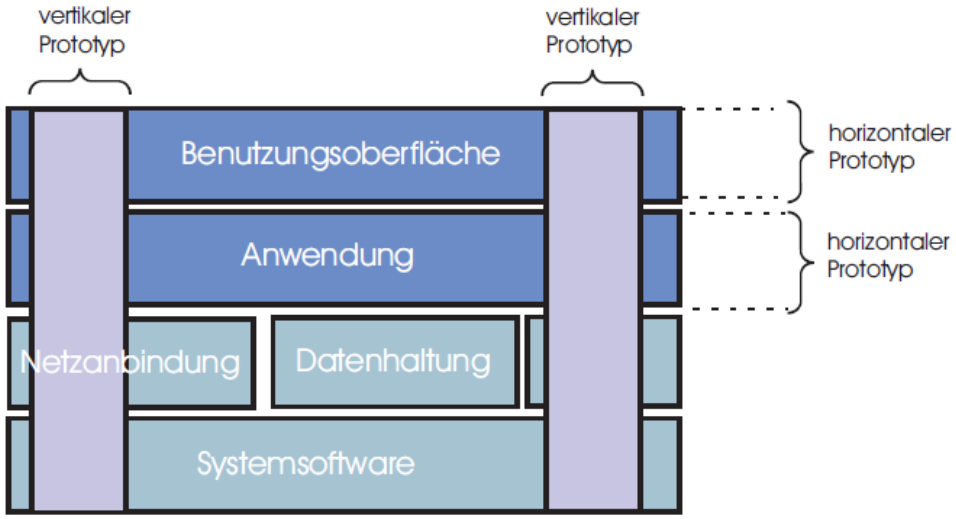
Experimentelle Prototypen

- Dienen dem Erkunden der technischen Machbarkeit
- Werden nur zur Testzwecken mit so geringem Aufwand wie möglich unter Vernachlässigung von Qualitätsanforderungen erstellt
- Dieser Prototyp wird verwendet, um für bereits bekannte Zielsetzungen und feststehende Anforderungen Lösungsmöglichkeiten zu untersuchen, zu bewerten und zu vergleichen
- Der Prototyp ist dabei nicht Teil der zu entwickelnden Software

[nach Floyd 1984]

Horizontale- vs. Vertikale Prototypen

- Horizontaler Prototyp
  - Realisiert nur spezifische Ebenen des Systems möglichst vollständig
- Vertikaler Prototyp
  - Implementiert ausgewählte Teile des Zielsystems vollständig durch alle Ebenen hindurch
  - Geeignet, wenn die Funktionalitäts- und Implementierungs-optionen noch offen sind



Probleme der iterativ/inkrementell/evolutionären Vorgehensmodelle

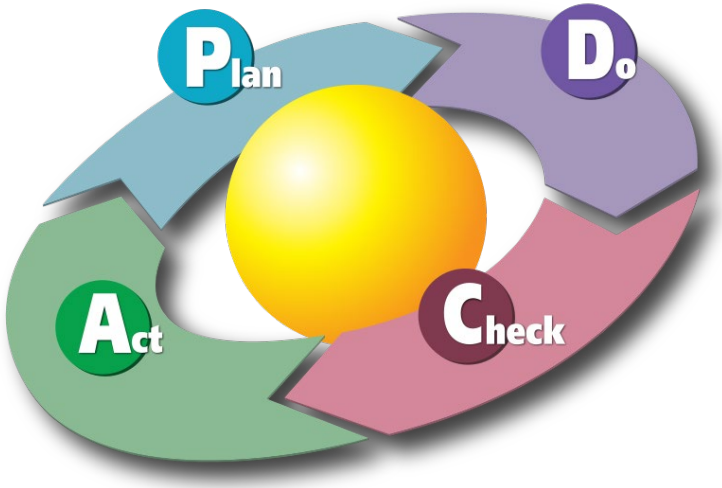
- Der Prozess ist nicht wirklich sichtbar
- Systeme sind oft schlecht strukturiert
- Vertragsgestaltung
  - Festpreise bei Vertragsabschluss
  - Dienstleistungsverträge besser, allerdings erfordern eine gute Vertrauensbasis zwischen Auftraggeber und -nehmer.
- Anwendbarkeit
  - Skalierbarkeit? Kleine bis mittelgroße (interaktive) Systeme
  - Für Teile von größeren Systemen (z.B. die Benutzungsschnittstelle)
  - Für Systeme mit kurzer Lebensdauer (falls bekannt)
  - Reine Softwaresysteme (z.B. Informationssysteme für Banken, Versicherungen, Einkaufsportale)

Übung

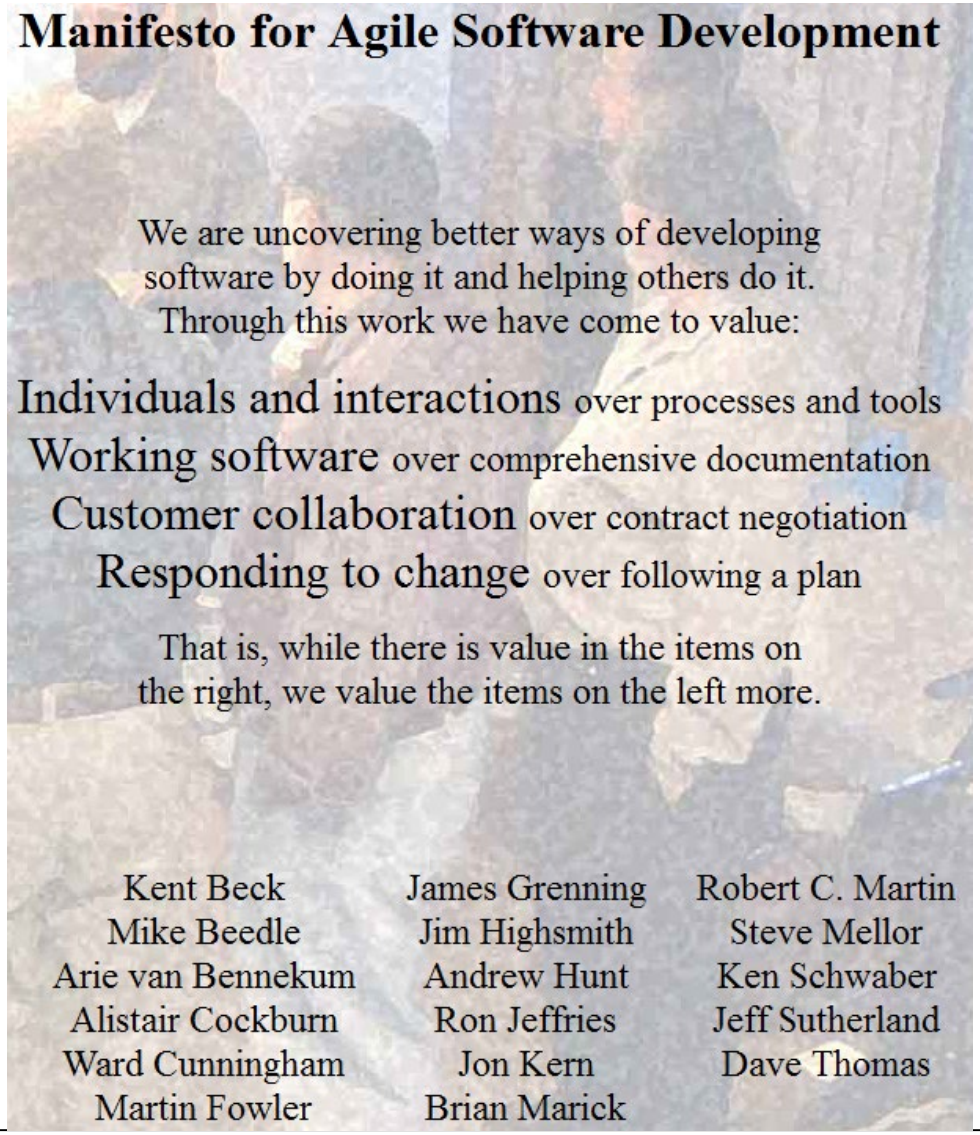
- Was ist die wichtigste Eigenschaft eines inkrementell/iterativen bzw. evolutionären Vorgehensmodells?
- Was sind die Unterschiede zwischen einer Iteration und eines Inkrements?
- Was sind die Eigenschaften des Spiralmodells?
- Welche Eigenschaften und Grundprinzipien hat der Rational Unified Process (RUP)?
- Welche Phasen und Aktivitäten sieht RUP vor?
- Welche Disziplinen sind in RUP definiert?
- Welche Arten von Prototypen gibt es? Welchen Zweck haben diese?
- Welche Probleme gibt es mit den inkrementell/iterativen und evolutionären Vorgehensmodellen?

Agile Vorgehensmodelle

- Inkrementeller Ansatz zusammengesetzt aus sehr vielen kleinen Iterationen
- Selbstorganisiertes autonom agierendes Team (cross-functional)
- Frühe und kontinuierliche Auslieferung an den Kunden
- Am Ende jeder Iteration folgt ein Reviews und ggf. Re-Evaluation von Projektprioritäten
- → ein einfaches „Inspect & Adapt“ Framework



Manifesto for Agile Software Development
[http://agilemanifesto.org/]

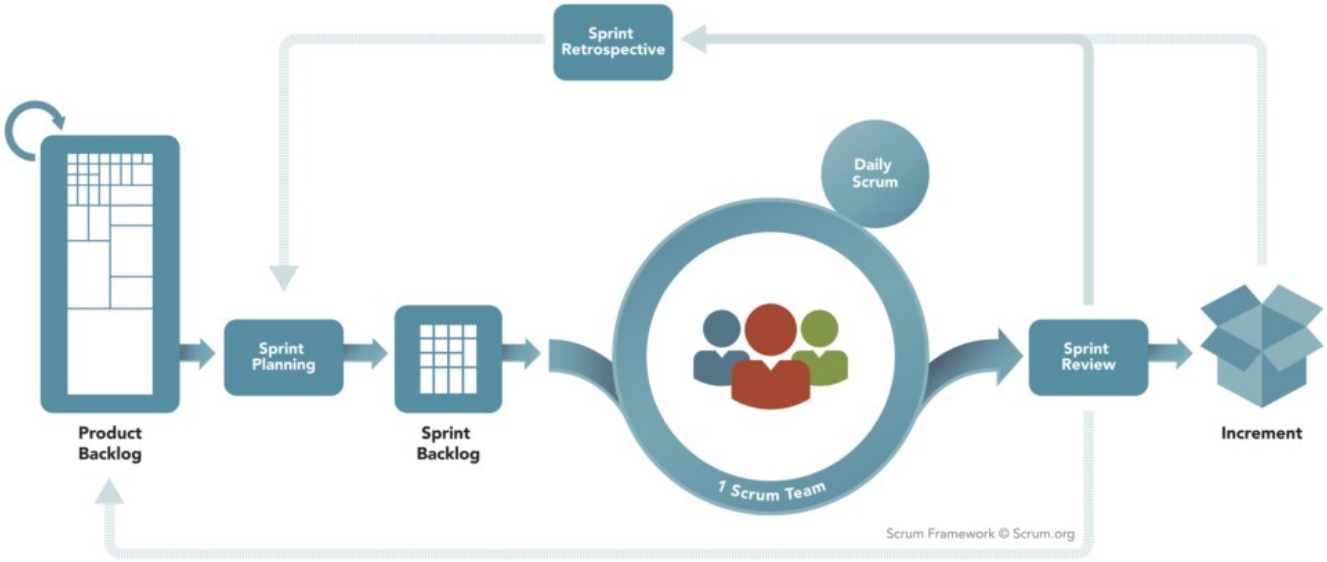


Agile Software-Entwicklung mit SCRUM 1/3

- Sprint → Iteration & sog. Timebox
  - Arbeitsabschnitt, in dem ein Inkrement einer Produktfunktionalität implementiert wird
  - Dauer: ca. 2 Wochen bis zu 1 Monat
- Produkt Backlog → „agiles“ Pflichtenheft
  - Wird vom Product Owner gepflegt
  - Enthält alle bekannten Anwendungsfälle (User Stories), funktionalen und nicht-funktionalen Anforderungen (Tasks)
  - Beschreibt das Installieren der Entwicklungs-, Integrations- und Testumgebung
- Sprint Backlog
  - Auszug aus dem Product Backlog, der im aktuellen Sprint realisiert wird
- Während des Sprints
  - Keine Änderungen zum Sprint-Ziel
  - Teamzusammensetzung bleibt fest

Agile Software-Entwicklung mit SCRUM 2/3

- Treffen (bei einem Sprint von ca. 1 Monat)
  - Daily Scrum (ca. 15 Min)
  - Sprint Planning Meeting (ca. 2 Std. / Woche)
  - Sprint Review Meeting (ca. 4 Std.)
  - Sprint Retrospektive (ca. 3 Std.)
- Ziel: am Ende jedes Sprints lauffähige Software liefern



Agile Software-Entwicklung mit SCRUM 3/3

- SCRUM Rollen
  - Scrum Master
  - Team
  - Product Owner
  - Stakeholders (user, customer, vendors)
  - Managers
- Cross-Functional Team
  - Ein Team ist immer mit allen Fähigkeiten ausgestattet, um das gewünschte Produkt zu erstellen
  - Rollen wechseln von einer Iteration zu der anderen. Ist das realsitisch?
  - Ist möglichst klein und besteht aus Vollzeitmitgliedern an einem Standort
  - Agiert autonom und selbstorganisiert
    - Entscheidet selbst, wie viele Anforderungen in einem Inkrement umgesetzt werden
    - Legt Arbeitsschritte und Arbeitsorganisation selbst fest

Feature Driven Development

- FDD stellt den Feature-Begriff in den Mittelpunkt
  - Jedes Feature stellt Mehrwert für den Kunden dar
  - Entwicklung anhand eines Feature-Plans
- Chefarchitekt - wichtige Rolle - hat Überblick
- Prozess- und Rollenmodell ist sehr kompakt
- FDD-Projekte durchlaufen fünf Prozesse.
  - 1. Entwickle ein Gesamtmodell (Rollen: alle Projektbeteiligte)
  - 2. Erstelle eine Feature-Liste (Rollen: in der Regel nur die Chefprogrammierer)
  - 3. Plane je Feature (Rollen: Projektleiter, Entwicklungsleiter, Chefprogrammierer)
  - 4. Entwurf je Feature (Rollen: Chefprogrammierer, Entwickler)
  - 5. Konstruiere je Feature (Rollen: Entwickler)

[http://de.wikipedia.org/wiki/Feature_Driven_Development]

Kanban - Ursprung

- Kanban (japanische bedeutet Signalkarte‘ (kan ‚Signal‘, ban ‚Karte‘)
- Kanban ist eine Methode der Produktionsprozesssteuerung
- Das ursprüngliche Kanban-System wurde 1947 von Taiichi Ohno in der japanischen Toyota Motor Corporation entwickelt, um die Produktivität zu steigern.
- Idee: „Es müsste doch möglich sein, den Materialfluss in der Produktion nach dem Supermarkt-Prinzip zu organisieren, das heißt, ein Verbraucher entnimmt aus dem Regal eine Ware bestimmter Spezifikation und Menge; die Lücke wird bemerkt und wieder aufgefüllt“.

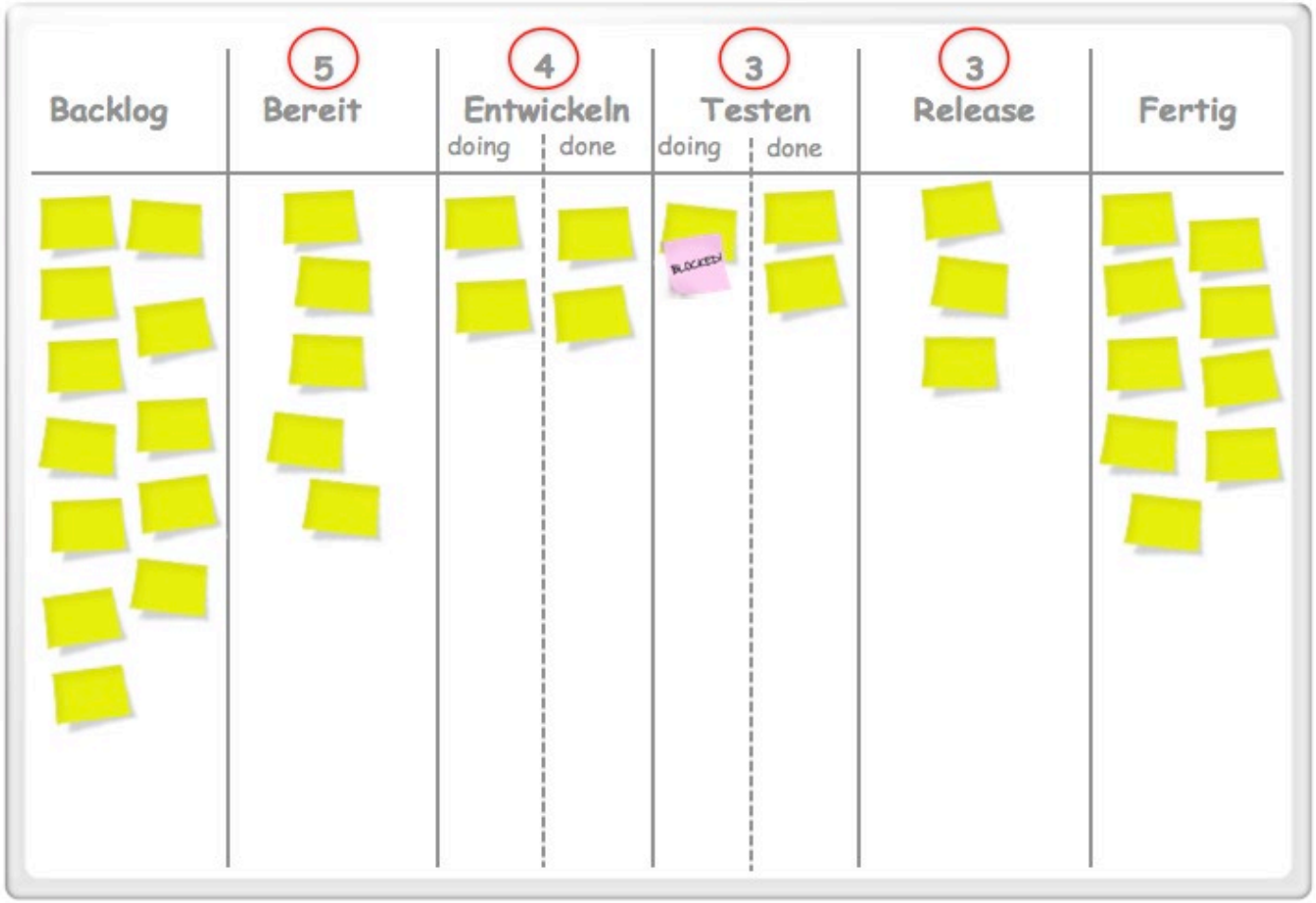
[https://de.wikipedia.org/wiki/Kanban]

Kanban zur Softwareentwicklung

- Kanban ist eine agile Methode für evolutionäres Change Management
- Der bestehende Prozess wird in kleinen Schritte (evolutionär) verbessert
  - Viele kleine Änderungen werden durchgeführt (anstatt einer großen)
  - Das Risiko für jede einzelne Maßnahme wird dadurch reduziert
- Iterationen sind optional
- Es gibt keine vorgegebenen Rollen
- Schätzungen und Priorisierung sind optional
- Neue Anforderungen können zu jedem Zeitpunkt an das Team gegeben werden, falls Kapazitäten frei sind
- Die Durchlaufzeit (Cycle Time) wird als Basis-Metrik für Planung und Prozessverbesserung verwendet

[https://de.wikipedia.org/wiki/Kanban_(Softwareentwicklung)]

Kanban Board zu Visualisierung



[it-agile.de]

eXtreme Programming

- Test first (vor Codierung)
- Permanenter Test
- Kunde »vor Ort«
- Anforderungen als Benutzungsbeispiele (user story)
- Inkrementelles Vorgehen
- Fortlaufende Integration
- Einfache Architektur (refactoring)
- Paar-Programmierung
- Gemeinsame Verantwortlichkeit

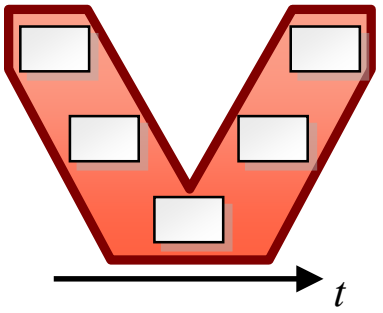
Bewertung der Agilen Methoden

- Vorteile
  - Kundenorientiert → Anforderungen werden iterativ in Zusammenarbeit mit dem Kunden entwickelt
  - Selbstorganisiertes autonom agierendes Team
  - Schnellere Implementierung der gewünschten Änderungen
  - Lauffähige Software am Ende jeder Iteration
  - Ergebnisse der Reviews und Retrospektiven „fließen“ unmittelbar in die nächste Iteration ein → laufende Verbesserung
- Nachteile
  - Qualität im industriellen Sinne (z.B. für sicherheitskritische Anwendungen) nicht immer gewährleistet
  - Skalierbarkeit schwierig
  - Häufige Iterationen und Selbstorganisation erfordern ein sehr “enges” Monitoring
  - Gefahr der ständigen Änderung der Anforderungen seitens des Kunden, da keine Baseline definiert
  - Hohes Risiko bei der Kostenabschätzung und Zeitplanung

Übung

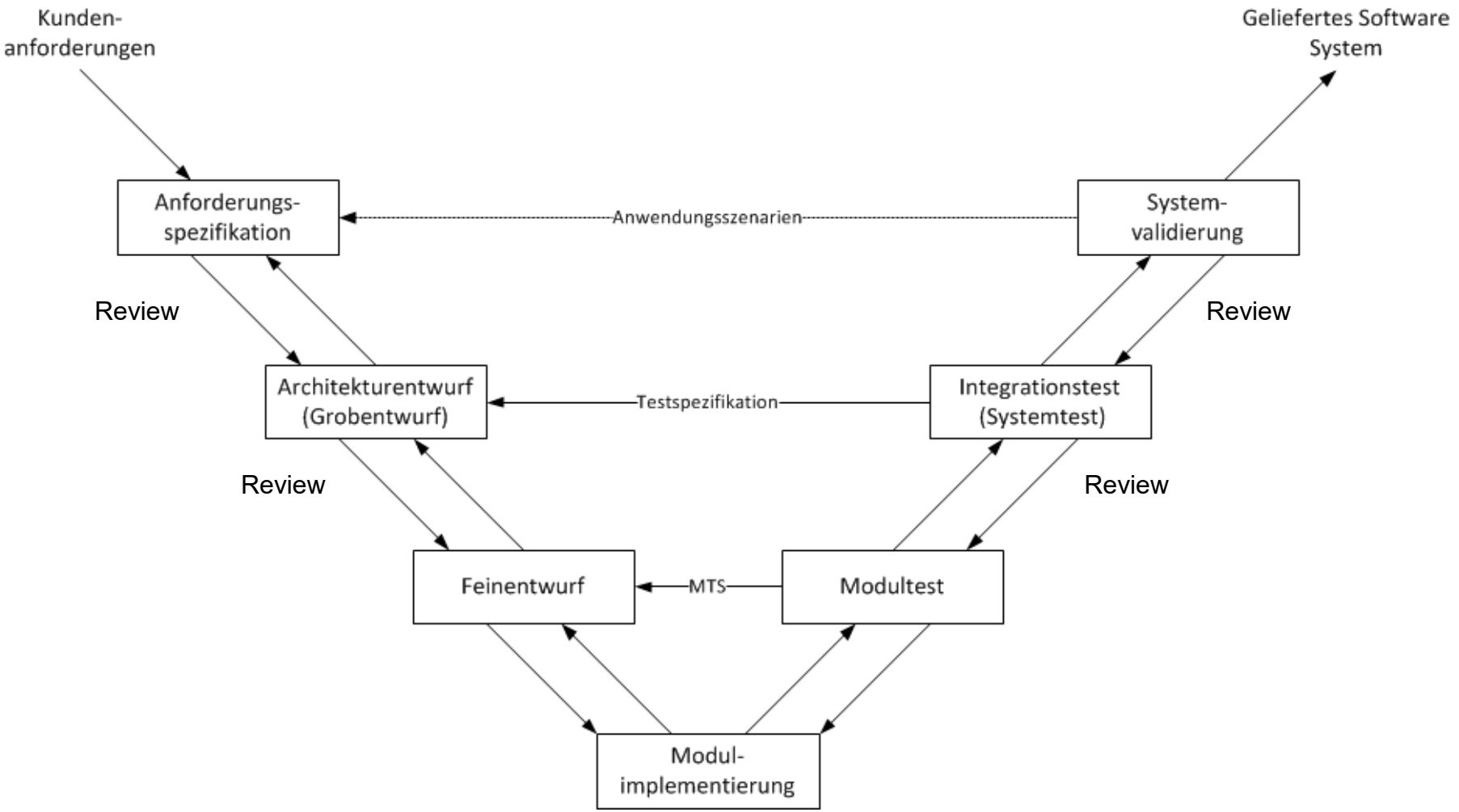
- Was sind die Eigenschaften eines Agilen Vorgehensmodells?
- Was besagt das Agile Manifesto?
- Welche Rollen gibt es im Scrum?
- Was zeichnet ein Scrum Team aus?
- Wie läuft ein Sprint ab?
- Was ist ein Produkt Backlog?
- Welche Vor- und Nachteile haben die Agilen Methoden?

V-Modell



- Erweiterung des Wasserfallmodells
- Verbreitet das V-Modell 97, Grundlage Boehm, 1984
- Integriert die Qualitätssicherung in das Wasserfallmodell
- Verifikation und Validierung der Teilprodukte sind wesentliche Bestandteile des V-Modells
- Kommunikation mit den Kunden an Phasenenden (meistens als Review)
- Einsatzbereich: sicherheitskritische software-intensive technische Systeme
  - Behörden
  - Militär (Standards DOD MIL-STD-2167, MOD 00-55)
  - Automobilindustrie
  - Luft- und Raumfahrt (ISO/IEC 15288:2008, ECSS)
  - Energieversorgung

V-Modell für die Software-Entwicklung

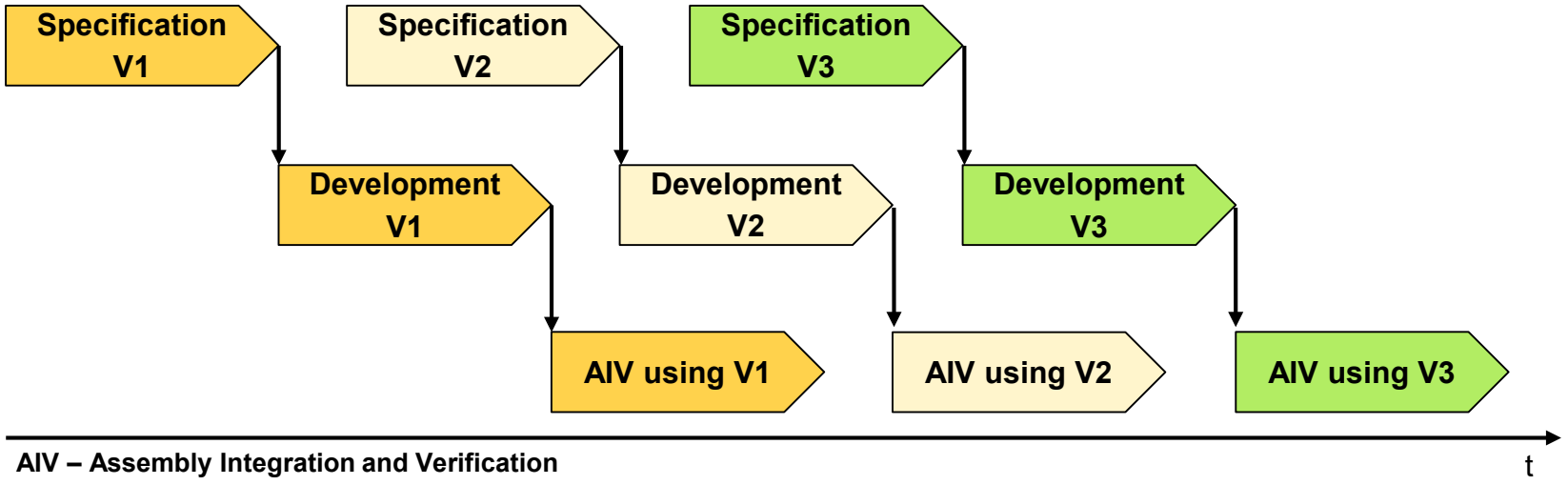


V-Modell

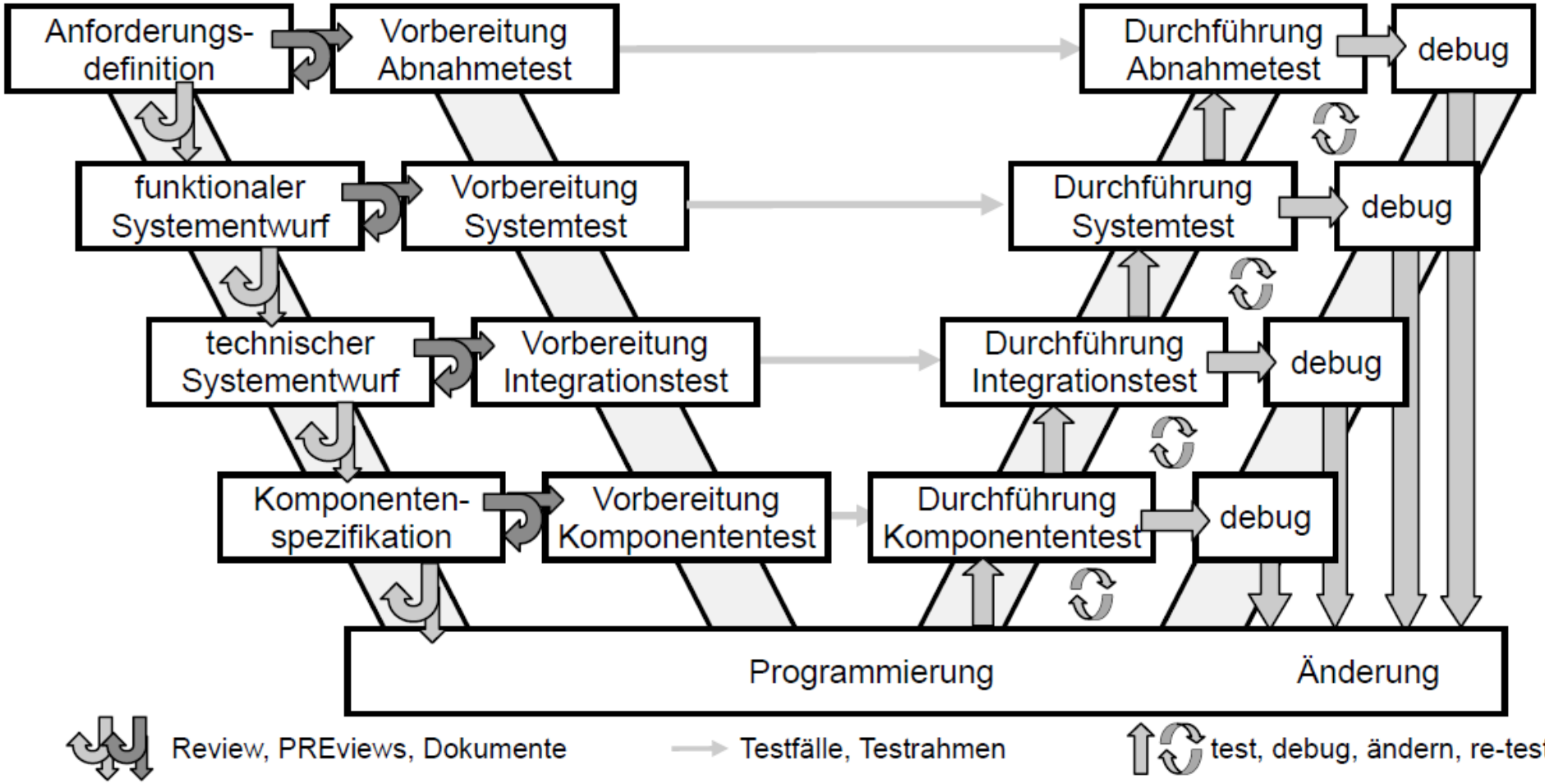
- Vorteile
  - Integriert die Qualitätssicherung in das Wasserfallmodell
    - Regelmäßige Reviews am Ende jeder Phase
    - Einbeziehung der Kunden zu fest definierten „Review-Zeitpunkten“
  - Ermöglicht formale Qualifikation (Verifikation und Validation) sicherheitskritischer Systeme
  - Standardisiert durch Militärstandards (z.B. ECSS, DOD MIL-STD-2167, MOD 00-55, ISO/IEC 15288:2008)
  - Ermöglicht gute Kostenabschätzung und Zeitplanung
- Nachteile
  - Klare Vorstellung über Anforderungen seitens des Kunden erwartet
  - (Fast) vollständige Spezifikation der Anforderungen vor der Entwicklung vorausgesetzt
  - Lieferung der Software erfolgt (spät) am Ende des Entwicklungs- und Qualifizierungsprozesses

V-Modell – inkrementell/iterativ

- Vorteile gegenüber dem klassischen V-Modell
  - Einbindung des Auftraggebers bei definierten Meilensteinen (Ende von Inkrements bzw. Iterationen)
  - Integration und Lieferung von definierten Versionen nach jedem Inkrement
  - Parallelisierung der Entwicklung verschiedener Subsysteme/Komponenten
  - Früher Test auf System Level möglich



W-Modell (erweitertes V-Modell)



[Spillner, A., et al.: Praxiswissen Softwaretest - Testmanagement, Kap. 3.4, dpunkt, 2. Auflage, 2008“]

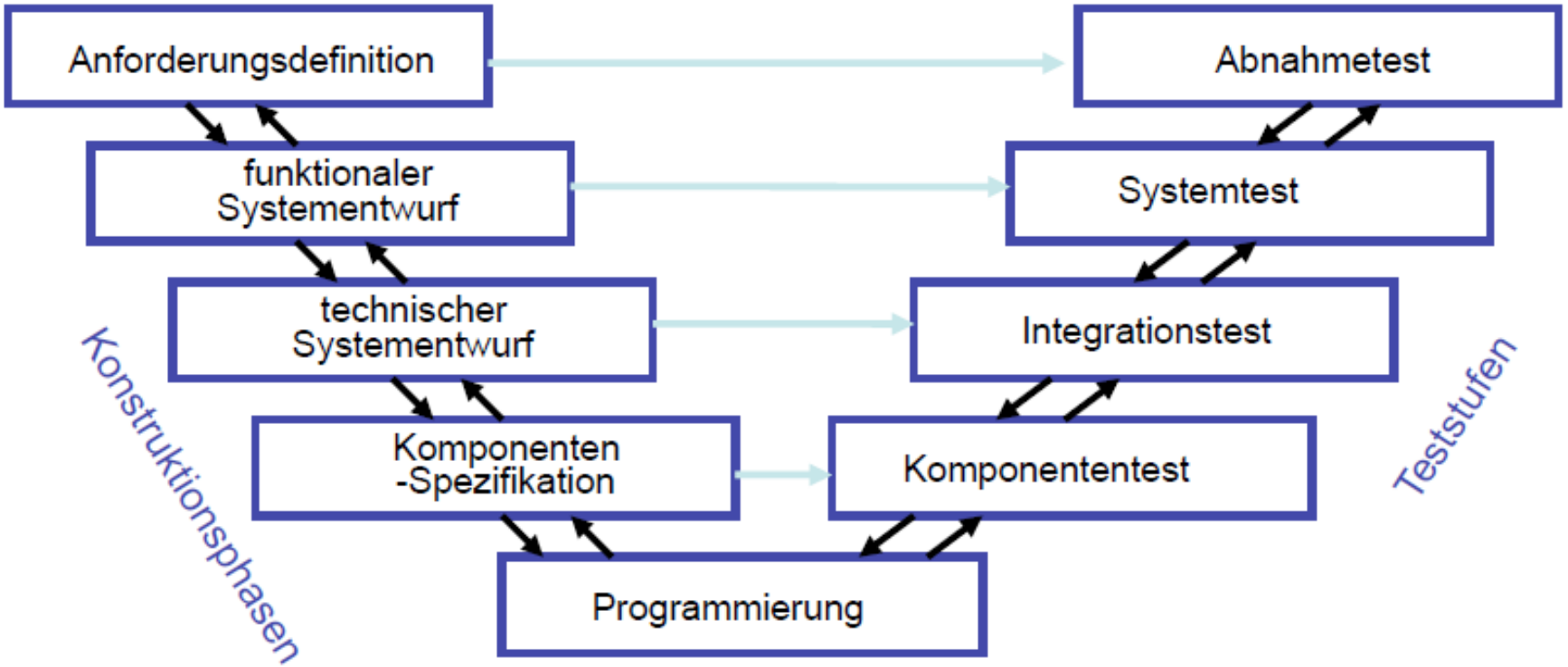
Was ist und warum W-Modell?

- Charakteristische Merkmale:
  - W-Modell beruht auf dem weit verbreiteten V-Modell
  - Frühzeitliche Testaktivitäten durch Vorbereitung der Tests auf allen Ebenen (d.h.: Definition der Testfälle aus den Anforderungen und Entwicklung der Testprozeduren)
  - Testaktivitäten (Test-V) parallel zu Entwicklungsaktivitäten (Entwicklungs-V)
  - Kürzere Zyklen: testen-debuggen-fixen
  - Frühe Einbindung von PA (Product Assurance) und von Testteams
- Beweggründe:
  - Zeitersparnis durch kürzere Iterationszyklen
  - Qualitätssteigerung durch frühe Einbindung von Testteams und PA
  - Kostenersparnis

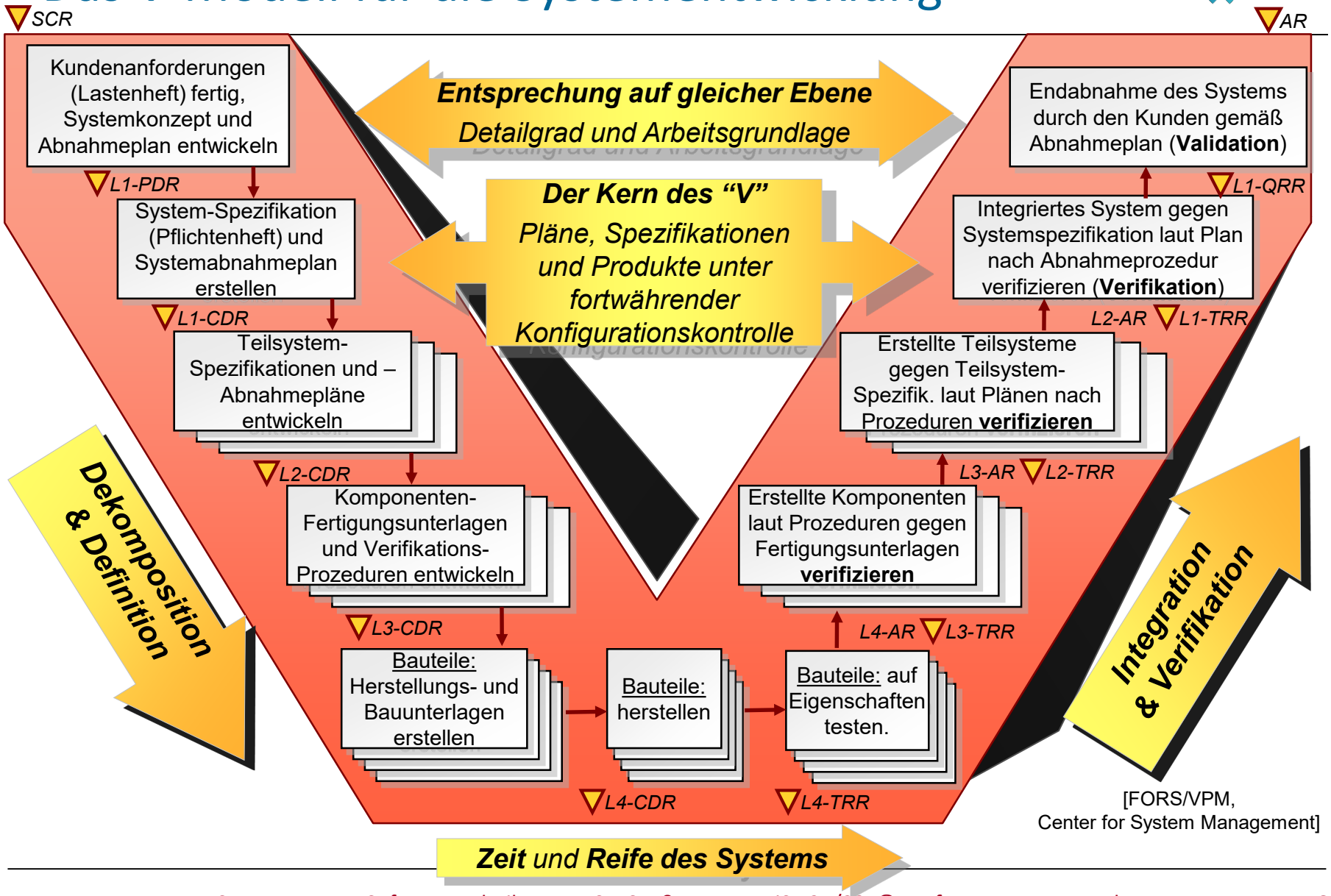
Das V-Modell XT (Extreme Tailoring)

- Entstanden innerhalb eines Forschungsprojektes der TU München als Weiterentwicklung des V-Modell 97
- Das V-Model XT Referenzmodell wird durch das Bundesministerium für Innern verantwortet
- Eigenschaften
  - Generisches Standardvorgehensmodell, das durch die Unternehmen bzw. Projekte angepasst (tailored/zugeschnitten) werden sollte
  - Artefaktorientiert (produktorientiert)
- Inhalte
  - Vorgehensbausteine
  - Ablaufbausteine
  - Entscheidungspunkte
  - Auftraggeber-/Auftragnehmer-Schnittstelle
  - Projektrollen

Das V-Modell für die Systementwicklung



Das V-Modell für die Systementwicklung



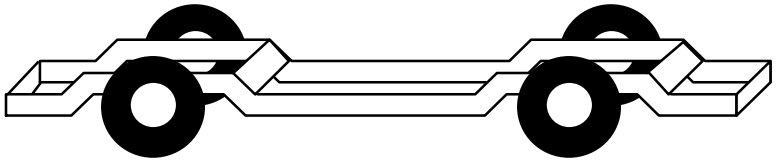
[FORS/VPM, Center for System Management]

Varianten des V-Modells für die Systementwicklung

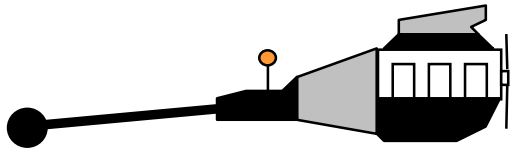
- Inkrementelle Entwicklung mit abschließender Lieferung
- Inkrementelle Entwicklung mit mehreren Lieferungen
- Evolutionäre Entwicklung (kontinuierliche Verbesserung)

Inkrementelle Entwicklung + eine Lieferung

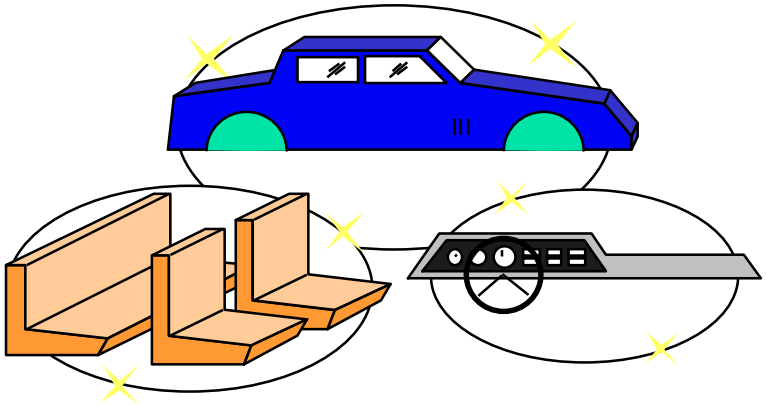
Inkrement 1 – Chassis und Federung



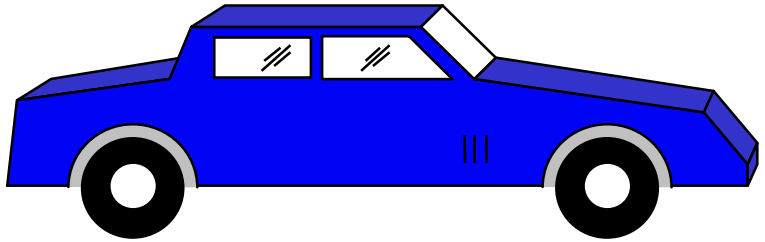
Inkrement 2 – Triebstrang



Inkrement 3 – Gehäuse, Sitze, Armaturen

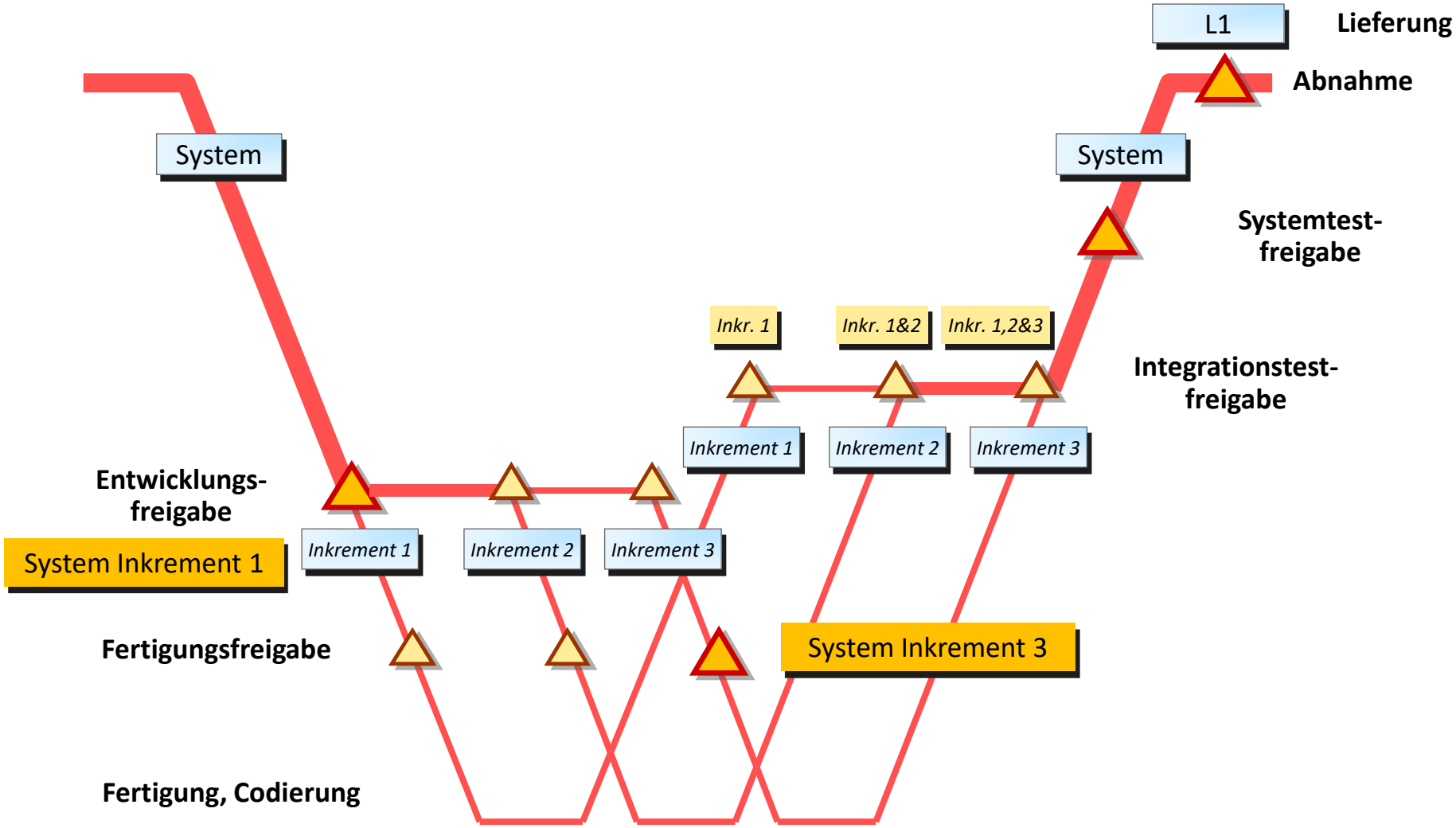


Zusammenbau



[FORS/VPM, Center for System Management]

Inkrementelle Entwicklung + eine Lieferung

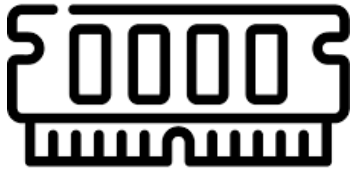


[FORS/VPM, Center for System Management]

Inkrementelle Entwicklung + mehrere Lieferungen

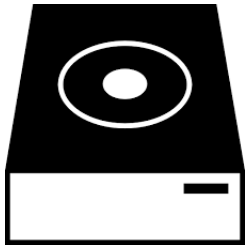
Phase 1 – Funktionsfähiger Computer

Phase 2 – RAM hinzugefügt



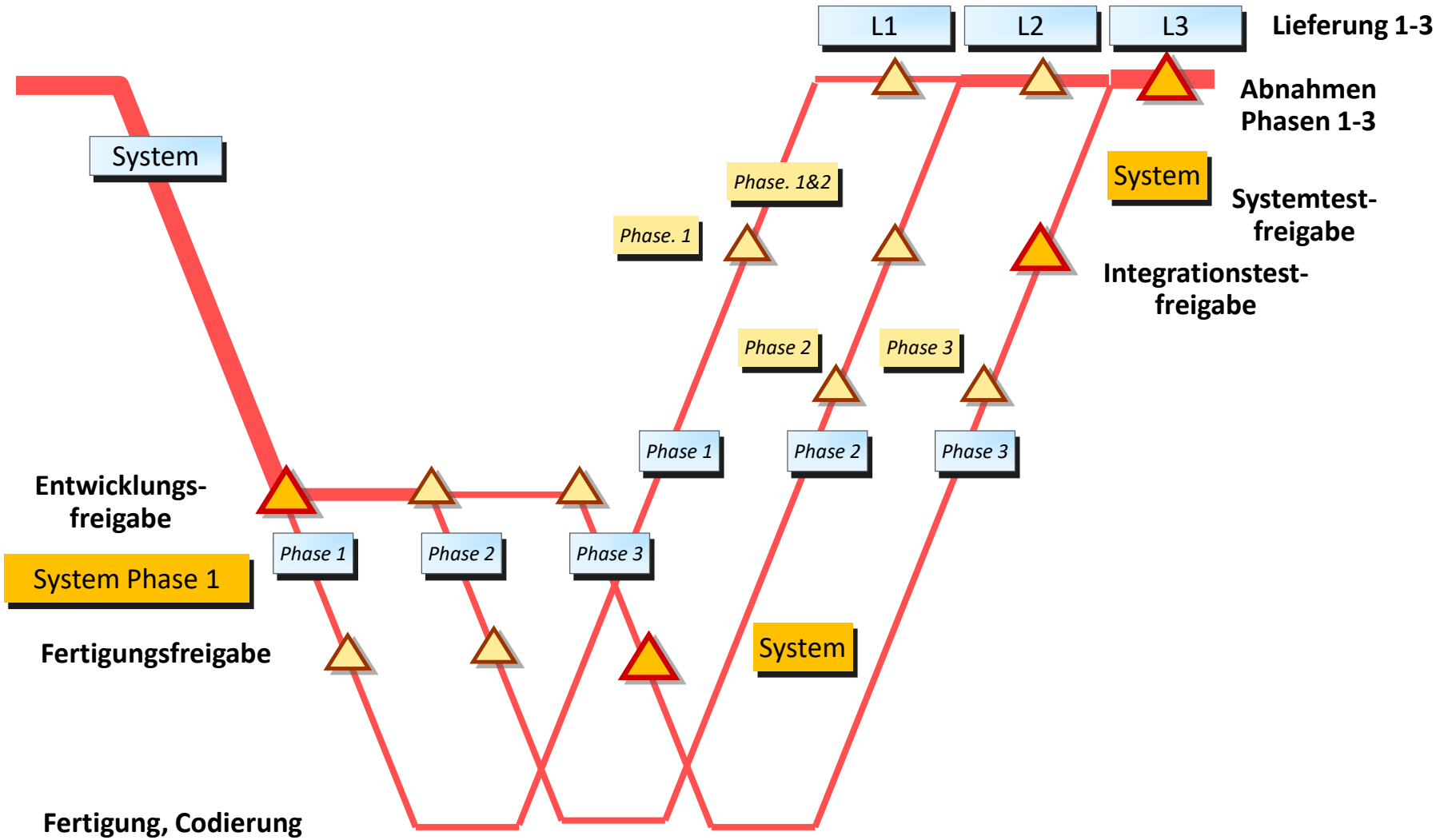
Phase 3 – bessere Grafikkarte hinzugefügt

Phase 4 – DVD-Laufwerk entfernt, SSD eingebaut



[Idee: FORS/VPM, Center for System Management, Bilder: flaticon.com]

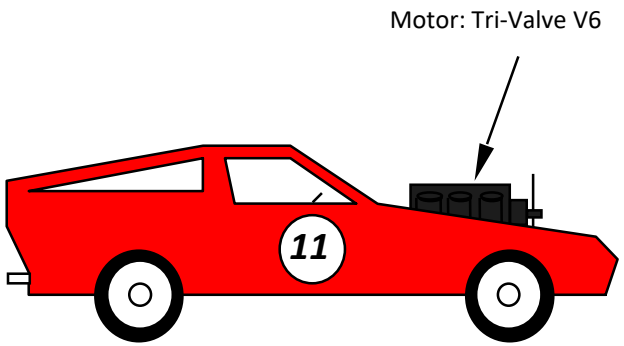
Inkrementelle Entwicklung + mehrere Lieferungen



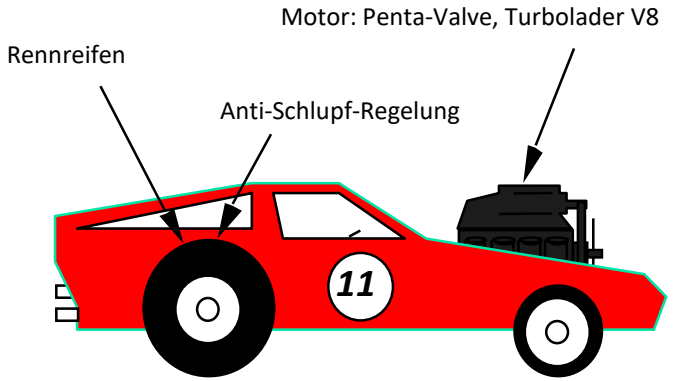
[FORS/VPM, Center for System Management]

Evolutionäre Entwicklung: Hochleistungsmotorsport

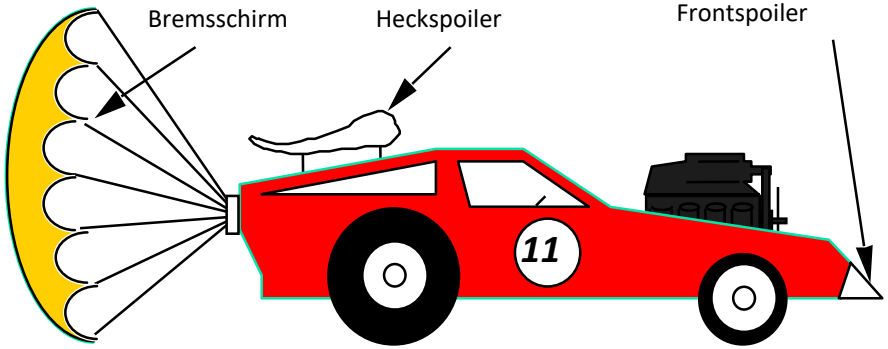
Initiale Version – Dragster



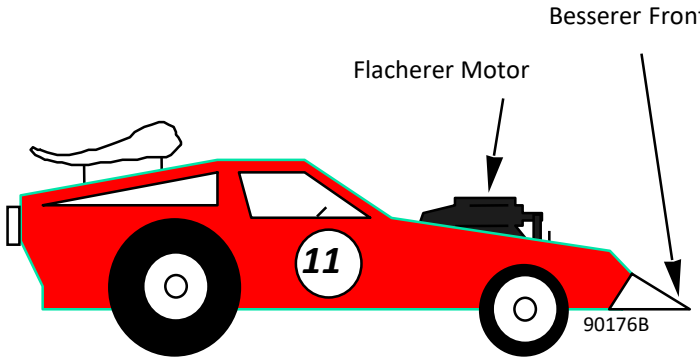
Version 2 – Mehr Leistung und verbesserte Bremsen



Version 3 – Verbesserung der Fahreigenschaften

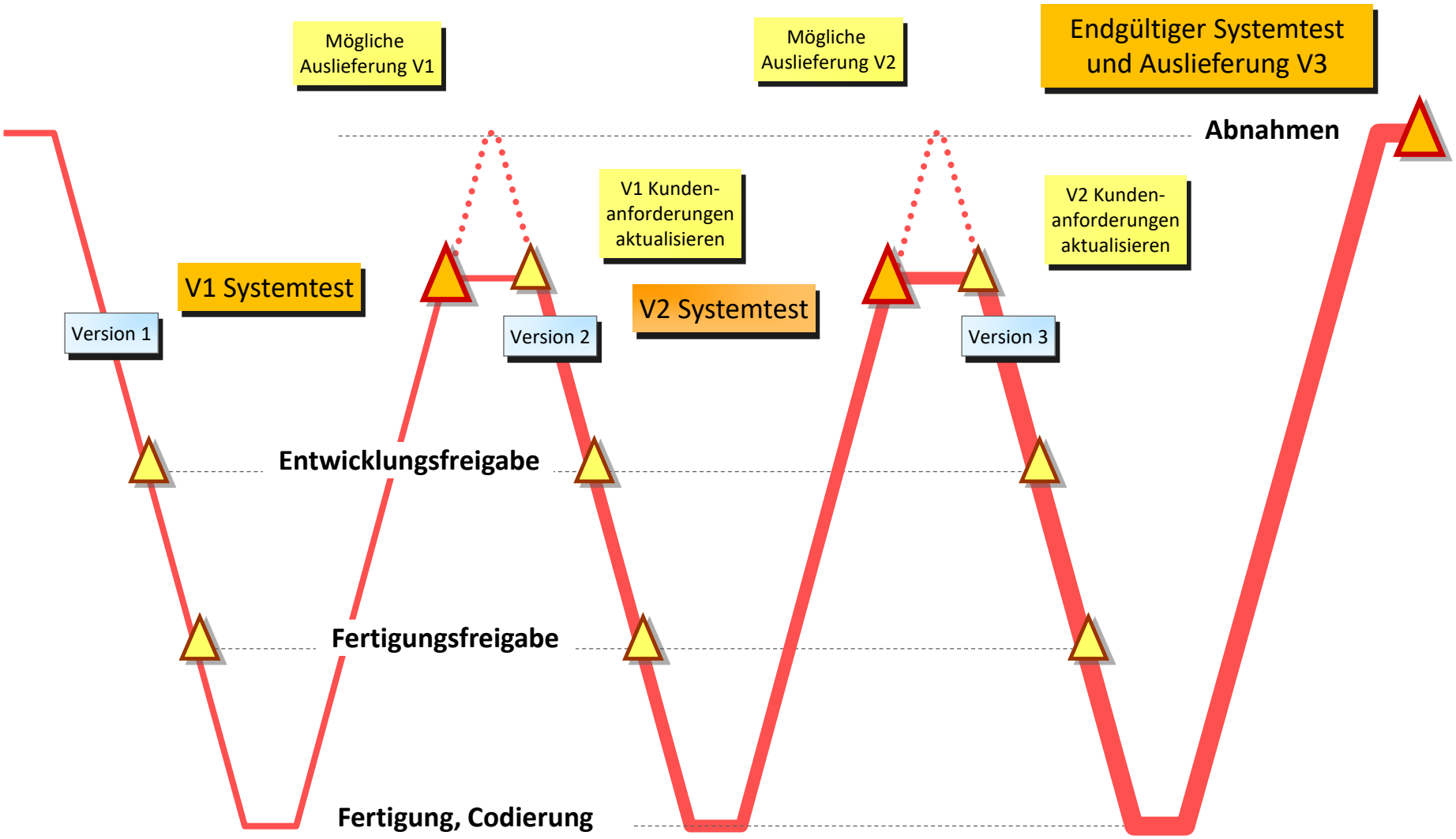


Version 4 – Verfeinerung



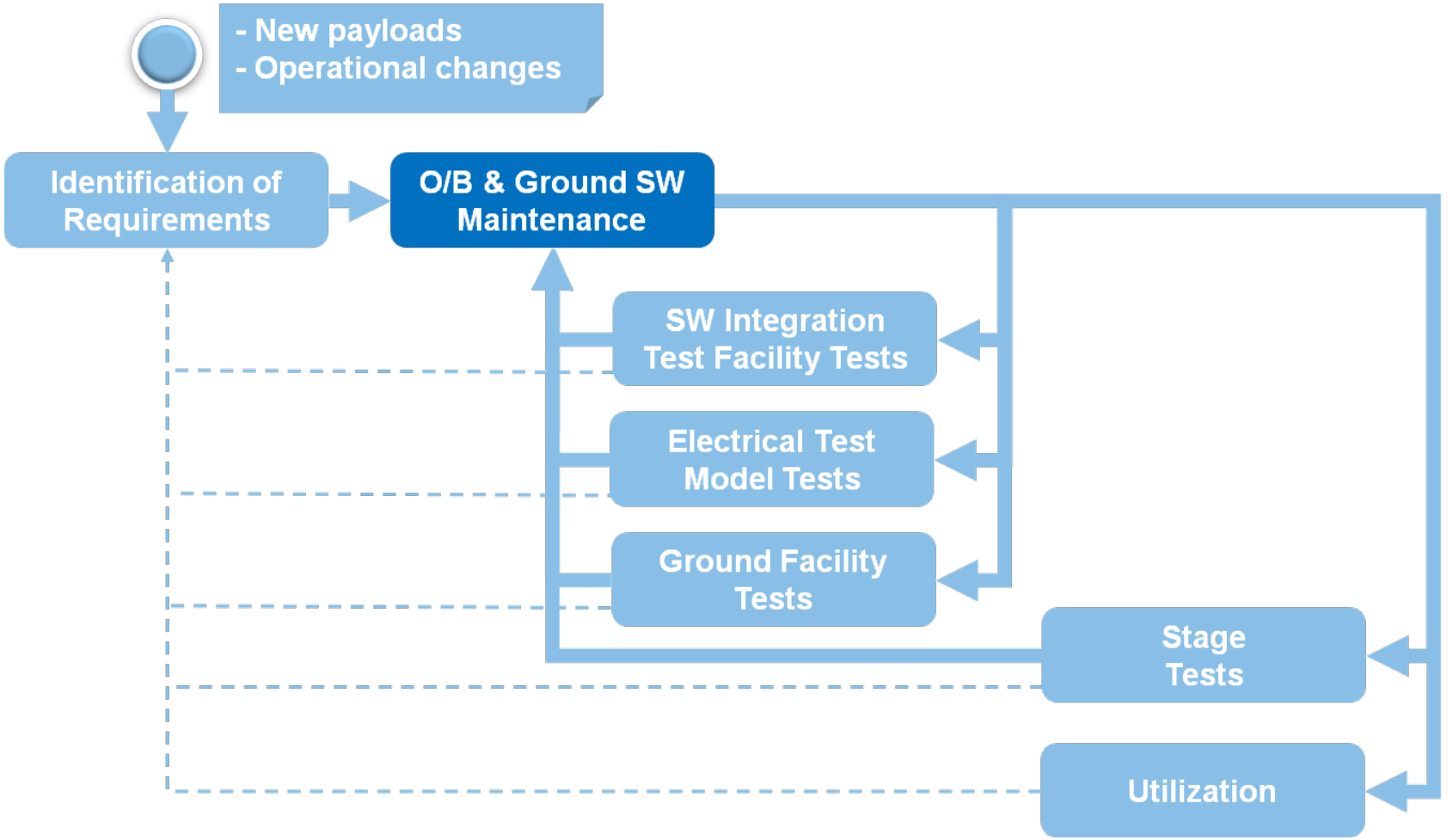
[FORS/VPM, Center for System Management]

Evolutionäre Entwicklung



[FORS/VPM, Center for System Management]

Beispiel: Columbus iterativ/inkrementelles V-Model



Übung

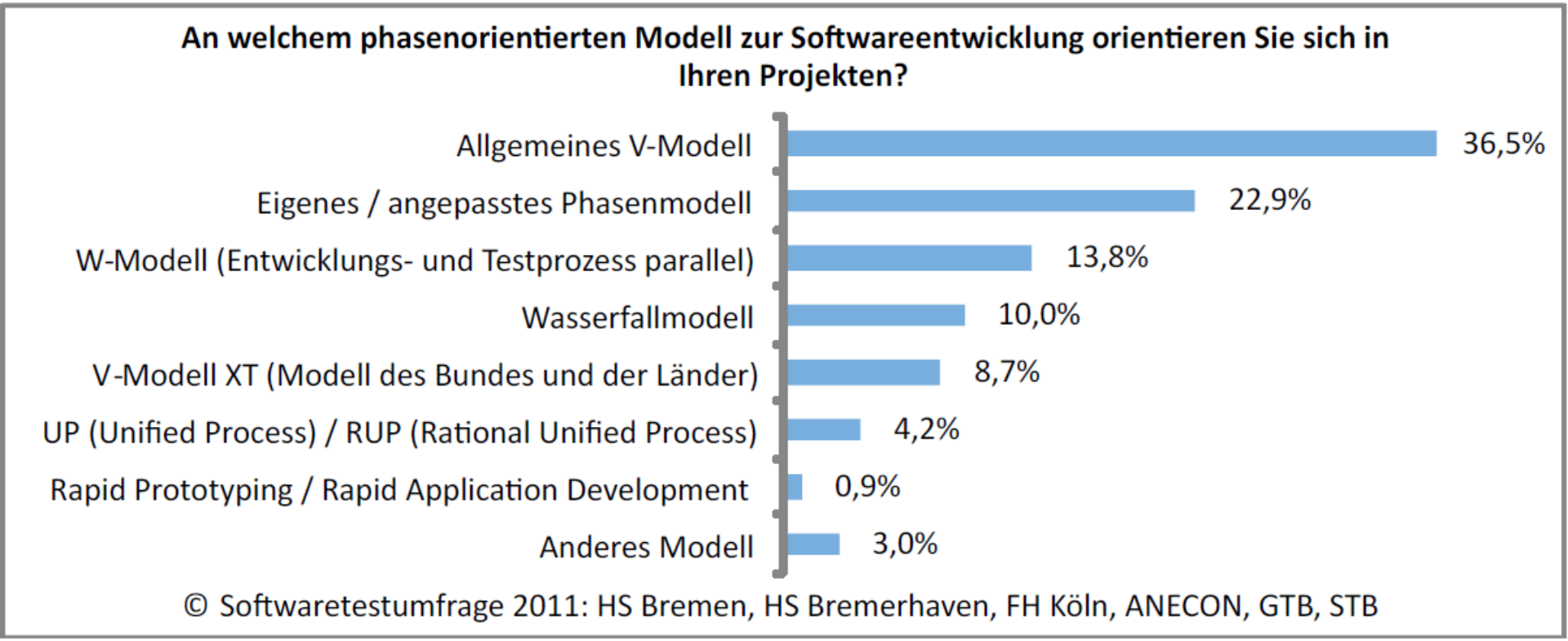
- Beschreiben Sie das V-Modell
- Wo wird das V-Modell eingesetzt?
- Welche Vor- und Nachteile hat das klassische V-Modell?
- Welche Varianten des V-Modells für die Software-Entwicklung kennen Sie?
- Welche Varianten des V-Modells für die System-Entwicklung kennen Sie?

Umfrageergebnisse - Vorgehensmodelle

- Umfrage von verschiedenen Unternehmen im D, CH und A, 2011 (http://www.softwaretest-umfrage.de)
  - 54,2 % der Unternehmen setzen ein Phasenorientiertes Vorgehensmodell ein
  - 28,6 % der Unternehmen setzen ein Agiles Vorgehensmodell ein
  - 17,2 % der Unternehmen setzen gar kein Vorgehensmodell ein

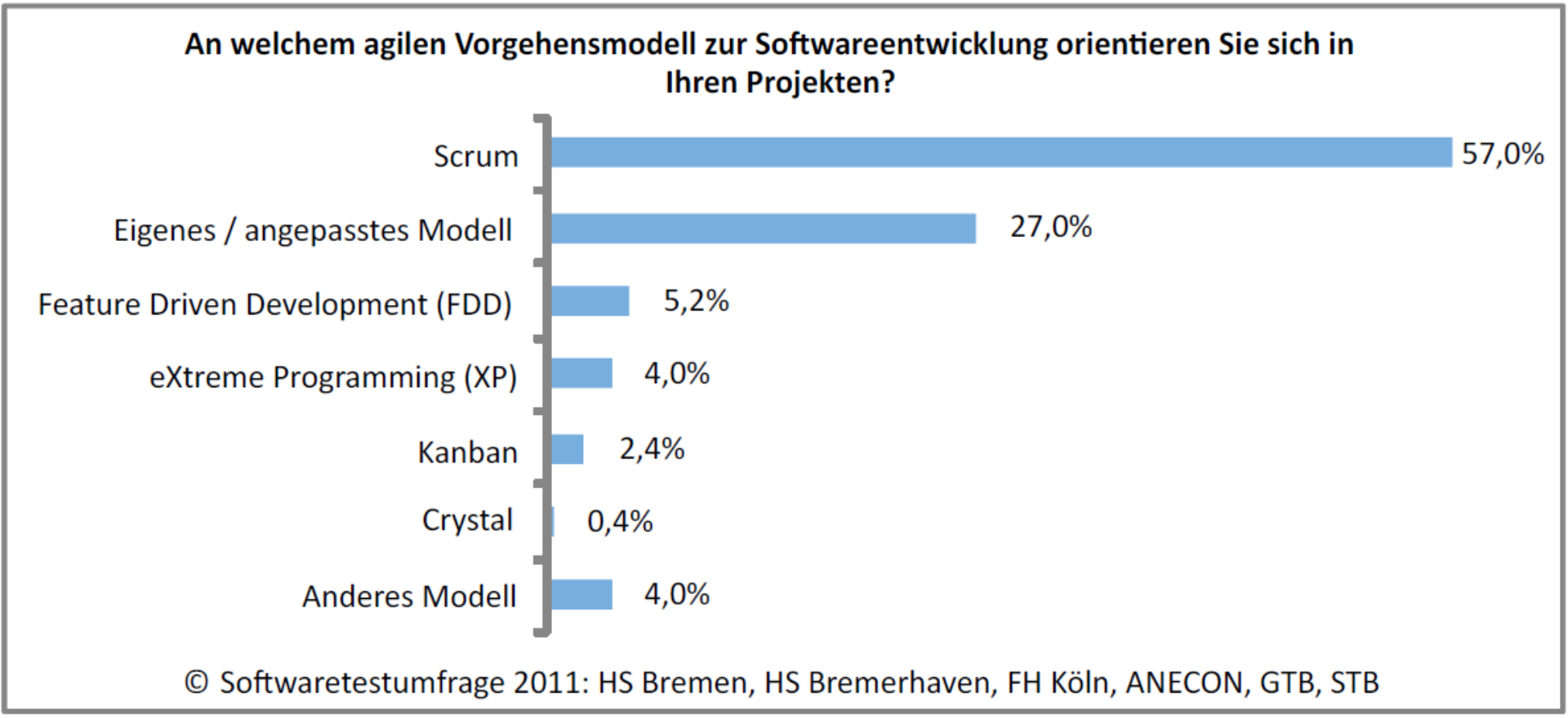
Umfrageergebnisse – Phasenvorgehensmodelle

- Umfrage von verschiedenen Unternehmen im D, CH und A, 2011 (http://www.softwaretest-umfrage.de)



Umfrageergebnisse - agile Vorgehensmodelle

- Umfrage von verschiedenen Unternehmen im D, CH und A, 2011 (http://www.softwaretest-umfrage.de)



Zusammenfassung - Was Sie beantworten können sollten

- Ein Vorgehensmodell ist … und beantwortet die folgenden Fragen …
- Bestandteile eines Vorgehensmodells sind …
- Was sind die Nachteile/Probleme des 0. Vorgehensmodell der Software-Entwicklung bzw. der ersten Erweiterung?
- Eigenschaften/Phasen/Ergebnisse des Wasserfallmodells sind …
- Ein inkrementell/iterativen bzw. evolutionären Vorgehensmodell ist …
- Es gibt x,y,z… Prototypen
- Eigenschaften des klassischen V-Modells sind …
- Eigenschaften eines Agilen Vorgehensmodells sind …
- Beispiel für Agile Vorgehensmodelle sind …
- **+ alle Fragen in den Übungsfolien!**

Literaturhinweise

- **Primär: Materialien im AULIS!**
- I. Sommerville - Software Engineering, 2012
- C. Rupp, S. Queins, die SOPHISTen - UML 2 glasklar: Praxiswissen für die UML-Modellierung, 2012
- E. Gamma, R. Helm, R. E. Johnson - Design Patterns. Elements of Reusable Object-Oriented Software, 1996
- P. Clements - Documenting Software Architectures: Views and Beyond, 2003
- L. Bass, P. Clements, R. Kazman - Software Architecture in Practice, 2012
- ISO/IEC/IEEE 15288:2015 - Systems and software engineering -- System life cycle processes
- ISO/IEC/IEEE 16326:2009 - Systems and software engineering -- Life cycle processes -- Project management
- ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models
- A. Spillner, T. Linz - Basiswissen Softwaretest, 2019
- M. Broy, M. Kuhrmann - Projektorganisation und Management im Software Engineering, 2013
- S. Röpstorff, R. Wiechmann - Scrum in der Praxis, 2016