Software Engineering und Projektmanagement 2.0 VO

Vorlesung

2015W
Vorgehensmodelle

www.inso.tuwien.ac.at

INSO - Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Lernziele

- Die Bedeutung und Ziele von Vorgehensmodellen im Software Engineering
- Die Entstehung unterschiedlicher Vorgehensmodelle
- Der Unterschied zwischen konservativen und modernen Vorgehensmodellen
- Auswahl von Vorgehensmodellen: Wann setze ich welches Vorgehensmodell ein?
- Adaption und Einführung von Vorgehensmodellen

Grundlagen und Bedeutung von Vorgehensmodellen

- Softwareprodukte werden immer umfangreicher und komplexer
- Effektive und effiziente Software-Entwicklungsprozessmodelle und ihre praktische Umsetzung gewinnen demnach immer mehr an Bedeutung und sind wesentlich am Geschäftserfolg eines Softwareunternehmens beteiligt
- Ziel ist es, die Softwareentwicklung auf eine etablierte, ingenieurmäßige Basis zu stellen, um Verbesserungen in der Produktivität, Qualität und Vorhersagbarkeit zu erreichen
- Bisher keine einheitliche Systematik der Softwareentwicklung
- Es existiert kein optimaler Entwicklungsprozess, eine Art Universalprozess, für jedes Softwareprojekt
- Je nach Projektart ist ein geeignetes Vorgehen auszuwählen
- Mittlerweile haben sich bereits einige sogenannte State-of-the-Art-Software-Entwicklungsprozessmodelle etabliert

Die drei Steuergrößen eines Projektes

Kosten

Umfang und Inhalt (Qualität)

Zeit

Pick Two


We offer three kinds of service:
GOOD - CHEAP - FAST
You can pick any two
GOOD service CHEAP won't be FAST
GOOD service FAST won't be CHEAP
FAST service CHEAP won't be GOOD

Entstehungsgeschichte der Prozessmodelle nach Böhm

1950 – Software wie Hardware

1960 – Softwareentwicklung als Kunstfertigkeit

1970 – Formalismus

1980 – Produktivität, Wiederverwendung, Objekte, Peopleware

1990 – plangetrieben vs. agil

2000 – Hybride Modelle

Fundamentale Prozessmodelle

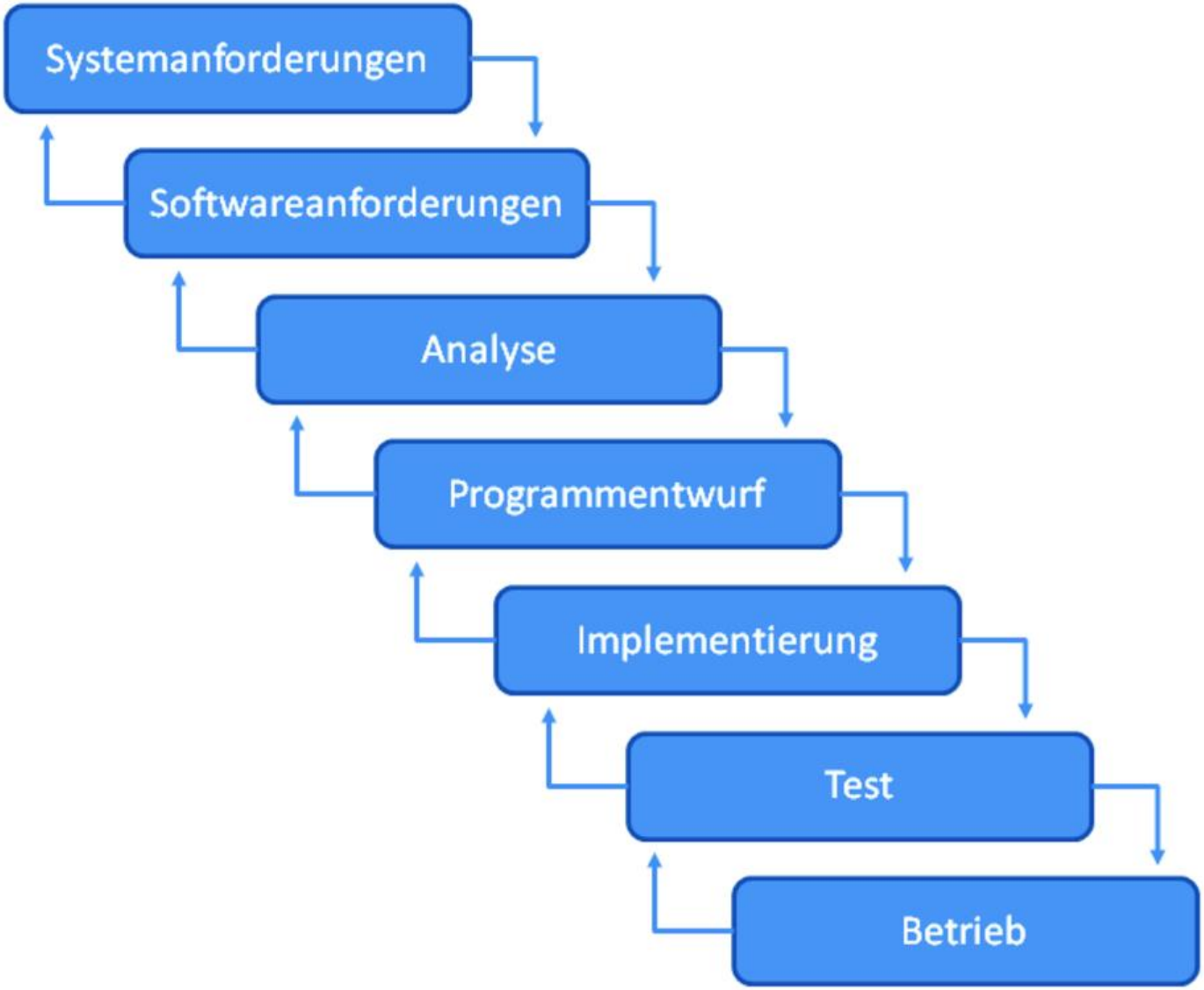
Grundlegenden Idee des Phasenmodells

Phasen dieses Modells sind in unterschiedlicher Form in jedem Prozessmodell enthalten

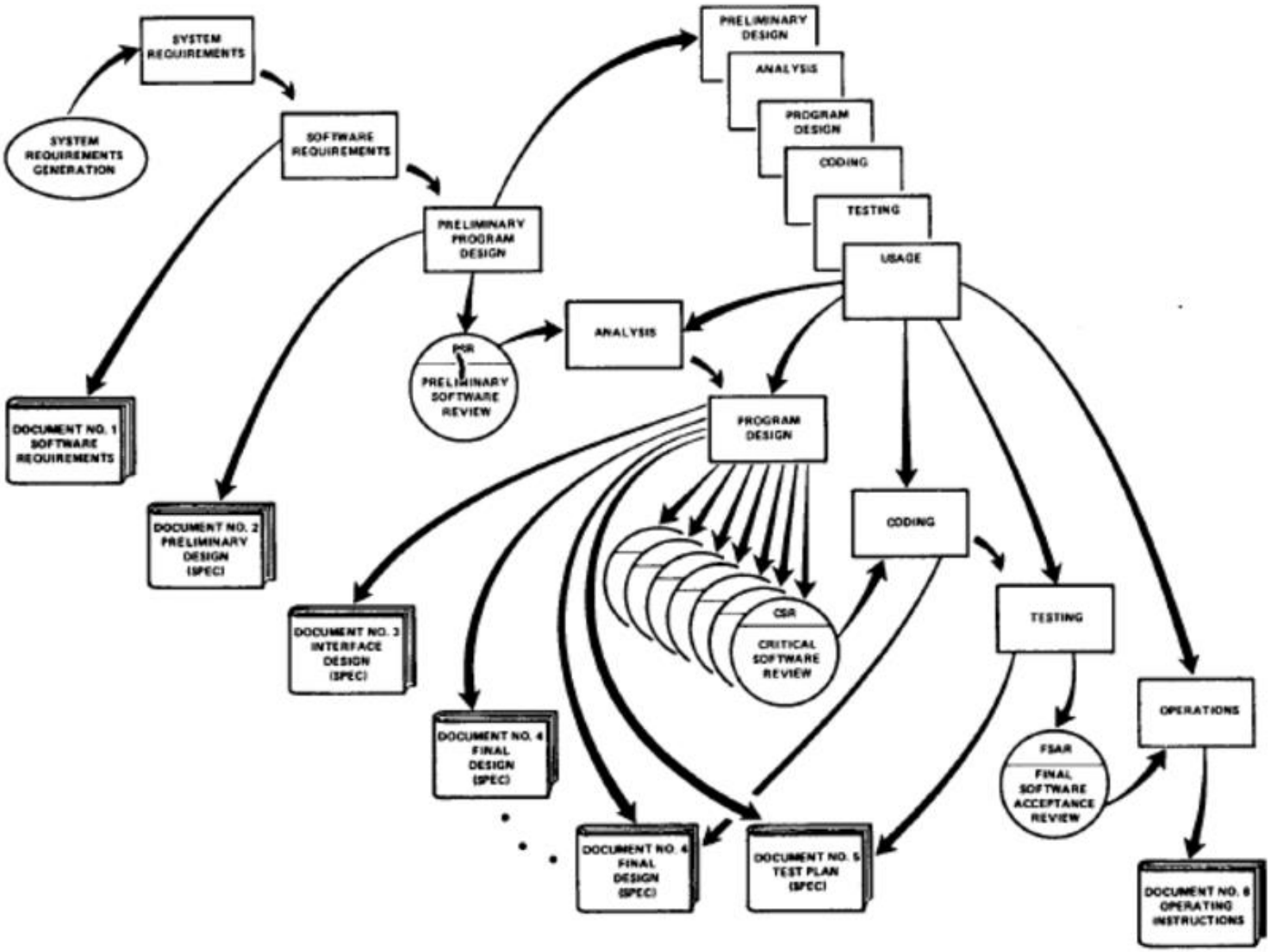
Wasserfallmodell nach Royce 1970

- Das am weitesten verbreitete Vorgehensmodell, von dem es viele unterschiedliche Varianten und Ausprägungen gibt
- Modell in seiner Grundform vollständig sequentiell
- Lässt in einer Erweiterungsstufe einen Rückwärtsschritt von einer Phase auf den direkten Vorgänger zu
- Ein Arbeitsschritt kann erst dann abgeschlossen werden, wenn alle vorgesehenen Produkte fertig gestellt wurden
- Berücksichtigt wesentliche Dokumente und Review-Schritte im Modell
- Wird in der modernen Softwareentwicklung als inhärentes Risiko betrachtet und hat zur Entwicklung der iterativen Modelle geführt

Fundamentales Wasserfallmodell nach Royce



Vollständiges Wasserfallmodell nach Royce



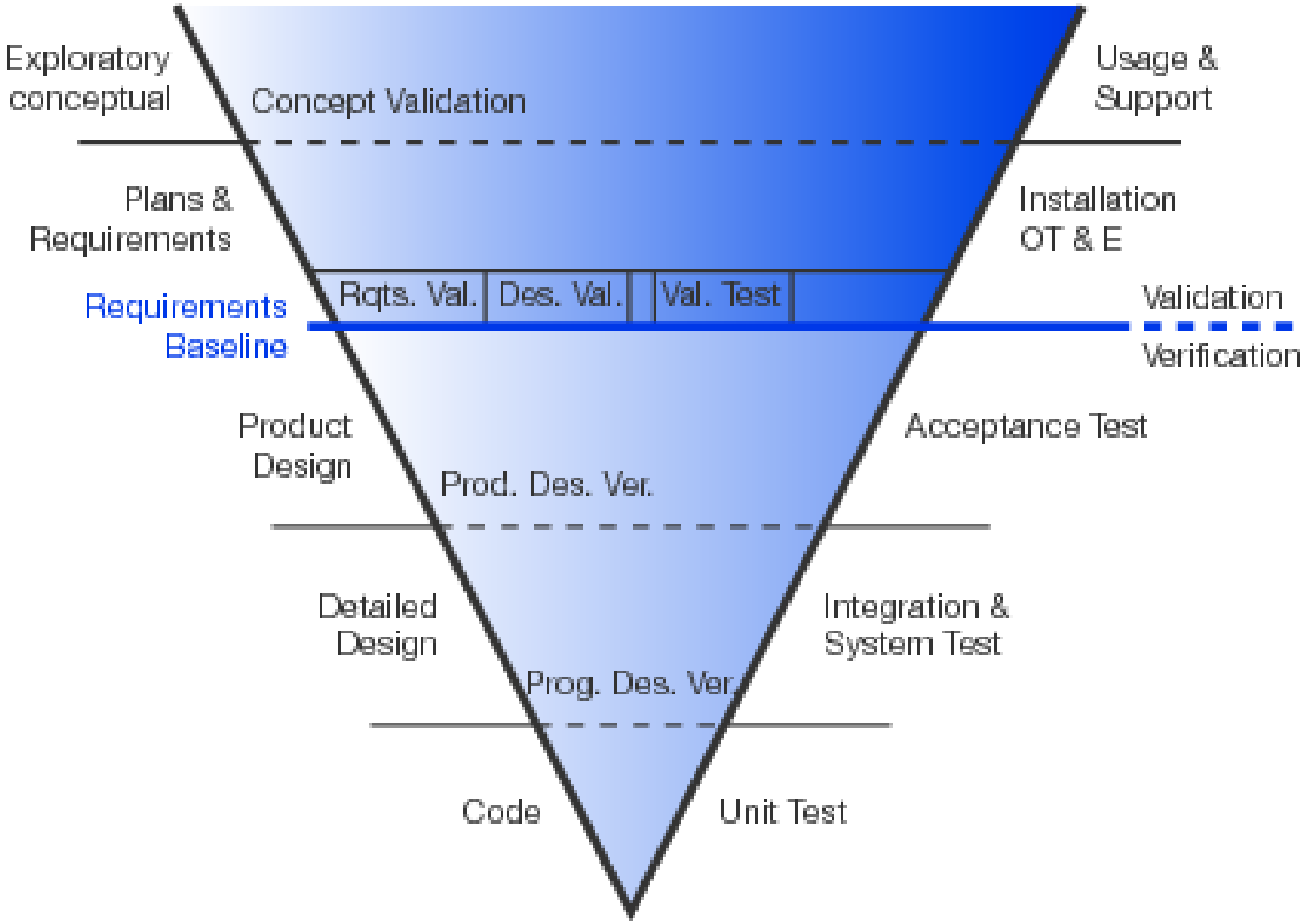
Fundamentales V-Modell nach Böhm

Basis für viele aktuelle Entwicklungsmodelle, die insbesondere im öffentlichen Sektor zum Einsatz kommen (V-Model 97, V-Model XT, BVM)

Das V-Modell ist auf folgenden Eigenschaften aufgebaut:

- Korrelierende Phasen
- Validierung und Verifikation

V-Modell


Exploratory conceptual
Concept Validation
Usage & Support
Plans & Requirements
Installation OT & E
Rqts. Val.
Des. Val.
Val. Test
Requirements Baseline
Validation
Verification
Product Design
Acceptance Test
Prod. Des. Ver.
Detailed Design
Integration & System Test
Prog. Des. Ver.
Code
Unit Test

Spiralmodell nach Böhm

Berücksichtigt die möglichen Projektrisiken der sequentiellen Entwicklung und ist daher iterativ aufgebaut

Der gesamte Prozess ist in vier Phasen gegliedert

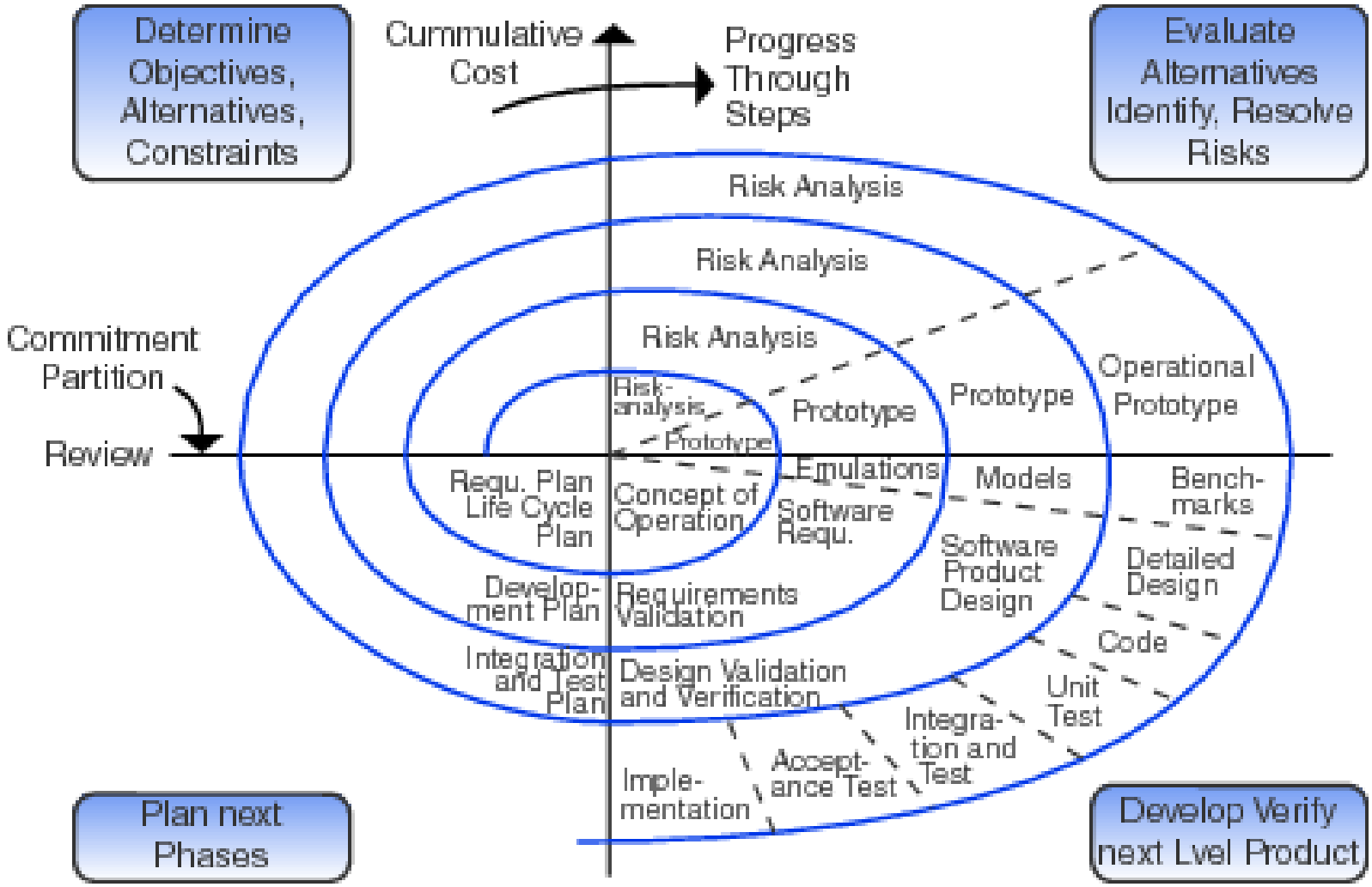
Alle Phasen werden im Rahmen einer evolutionären Softwareentwicklung mehrmals durchlaufen

Phasen:

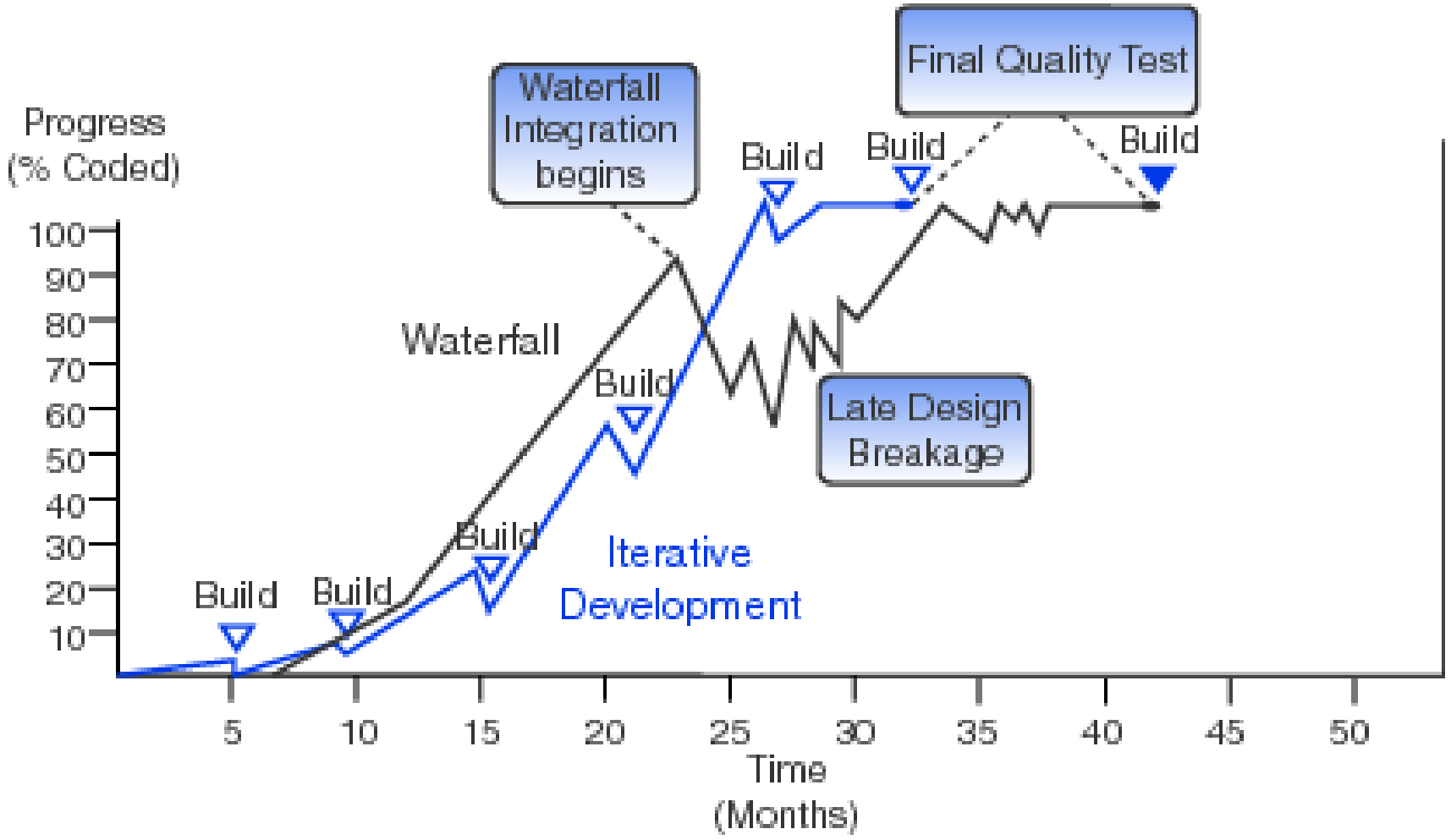
- 1. Definition von Zielen, 2. Risikoanalyse, 3. Durchführen der Arbeitsschritte, 4. Planen der nächsten Phase

Die Anzahl der notwendigen Durchläufe ergibt sich erst im Projektverlauf und ist von den der auftretenden Risiken abhängig

Spiralmodell



Vergleich Wasserfallmodell und iterative Entwicklung



Beispiel für eine Iterative Entwicklung

Industriell geprägte Modelle – Rational Unified Process

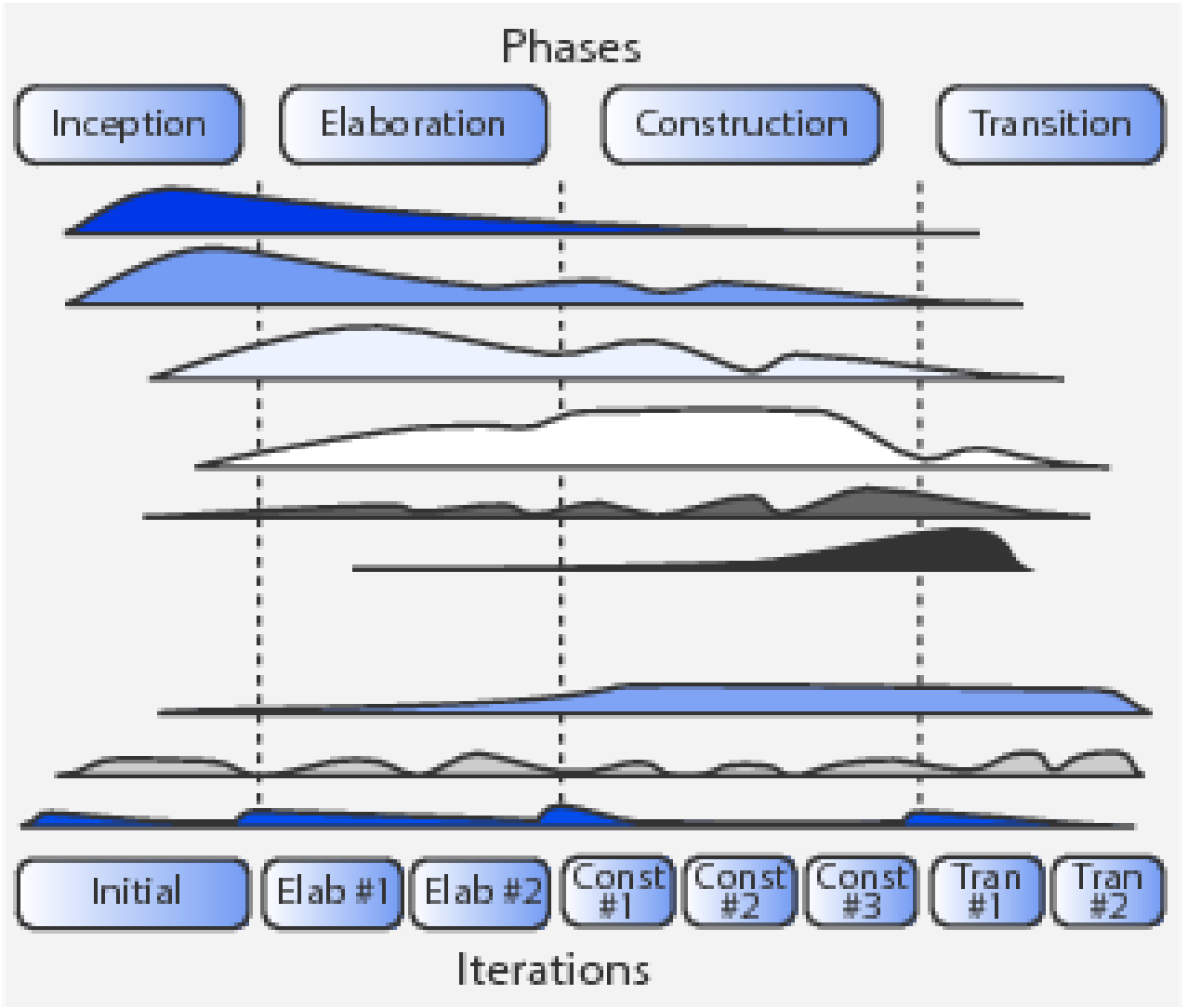
Prozess-Framework, das von IBM Rational gepflegt, ständig weiterentwickelt und momentan in der Version 7.0 ist

Eines des am meisten verbreitete und detailliert beschriebene Prozessmodell

Der Rational Unified Process beruht auf folgenden Vorgehensweisen:

- Software iterativ entwickeln
- Anforderungen managen
- Komponentenbasierte Architekturen verwenden
- Software visuell modellieren
- Softwarequalität kontinuierlich prüfen
- Änderungen kontrolliert in die Software einbringen

RUP – Entwicklungsphilosophie



Rational Unified Process (RUP) – Tailoring

- Für jedes Projekt bzw. für jede Organisation ist eine selektive Adaption vorzunehmen, um den erwarteten Nutzen auch tatsächlich zu erzielen
- Für die Anpassung (Tailoring) des RUP bietet sich das kostenpflichtige Tool IBM Rational Method Composer (RMC) an
- RMC erlaubt das Verwenden von vordefinierten RUP-Prozessmodellen
- Tool unterstützt das detaillierte Anpassen an spezifische Gegebenheiten und das Erzeugen eines komplett eigenen Modells bzw. das Mischen mit anderen Prozessmodellen
- Die Rolle, den RUP an die jeweiligen speziellen Rahmenbedingungen anzupassen, kommt dabei dem Process Engineer zu

Microsoft Solutions Framework (MSF)

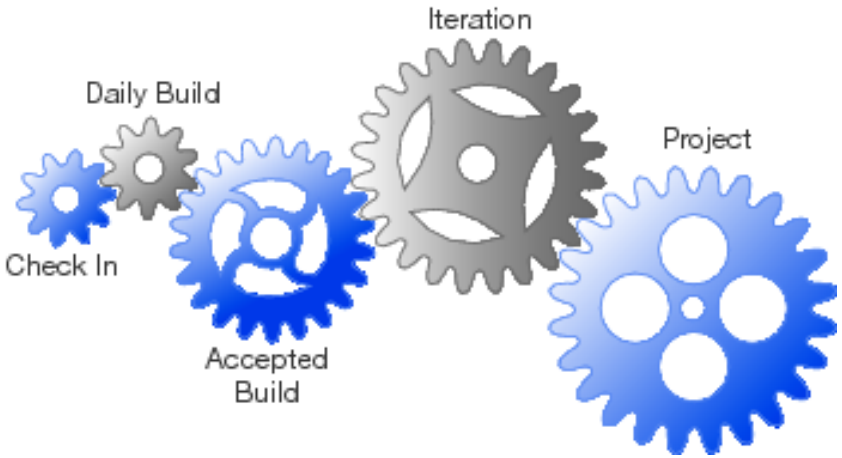
Alternativmodell zum industriell erfolgreichen RUP

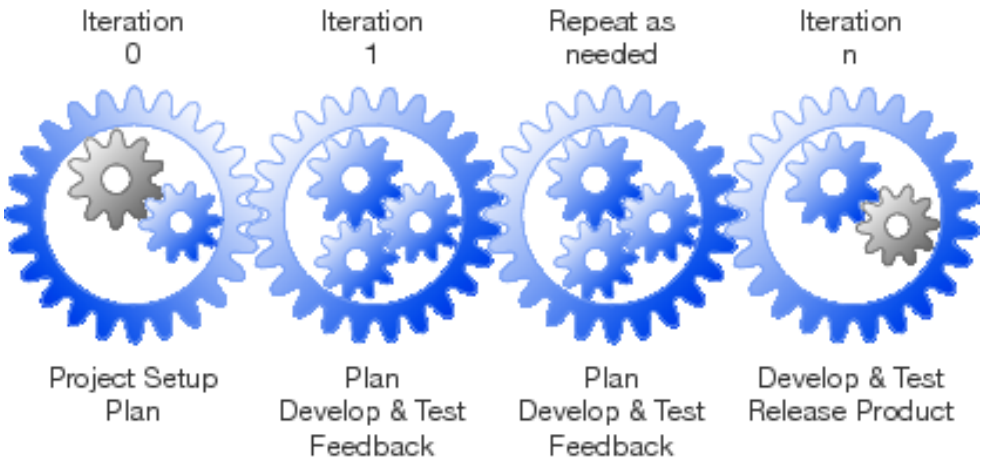
eingebunden im Microsoft-Produkt Visual Studio Team System angeboten

Visual Studio 2005 Team System wird mit zwei Ausprägungen des MSF mitgeliefert

- MSF for Agile Software Development: Diese Variante ist ausgerichtet auf eher kleinere und formlose Softwareprojekte und orientiert sich an typischen agilen Software-Entwicklungsprozessmodellen
- MSF for CMMI Process Improvement: Seitens des Herstellers wird betont, dass auch unter dieser Variante ein agiler und leichtgewichtiger Ansatz verstanden wird. Der Fokus liegt aber, durch die Ausrichtung auf eine formellere Umgebung, auf der Abwicklung größerer und komplexerer Vorhaben

MSF - Aufgaben- und Rollenverteilung





MSF - Entwicklungsphilosophie

- Ein Projekt setzt sich aus Iterationen und diese wiederum aus kleineren Einheiten, sogenannten Builds und Check Ins zusammen
- Innerhalb einer Iteration spielen Themen wie der Iteration Backlog (Katalog von zu implementierenden Anforderungen), Stakeholder Involvement (Miteinbeziehung des Kunden) und Iteration Launch bzw. Retrospective (Start-Up und Beurteilung der Iteration) eine wichtige Rolle
- Prinzip der iterativen und inkrementellen Entwicklung

Agile Software Development (Agile Manifesto)

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.

Motivation für Agile Prozessmodelle

Wurzeln in der empirischen Prozesssteuerung, die im Gegensatz zur definierten Prozesssteuerung über ständige Rückkoppelungen geführt wird

Software Development als Wissensarbeit

Software Projekt als komplexes System

- Grund für oftmaliges Scheitern klassischer Modelle
- Individuen und deren Interaktionen rücken in den Mittelpunkt

Von Personen geprägte Prozessmodelle - Scrum

Begriff bezeichnet eine hyper-produktive Produktentwicklung nach Sutherland, Schwaber & Beedle

Zentrale Werte dieses Prozessmodells :

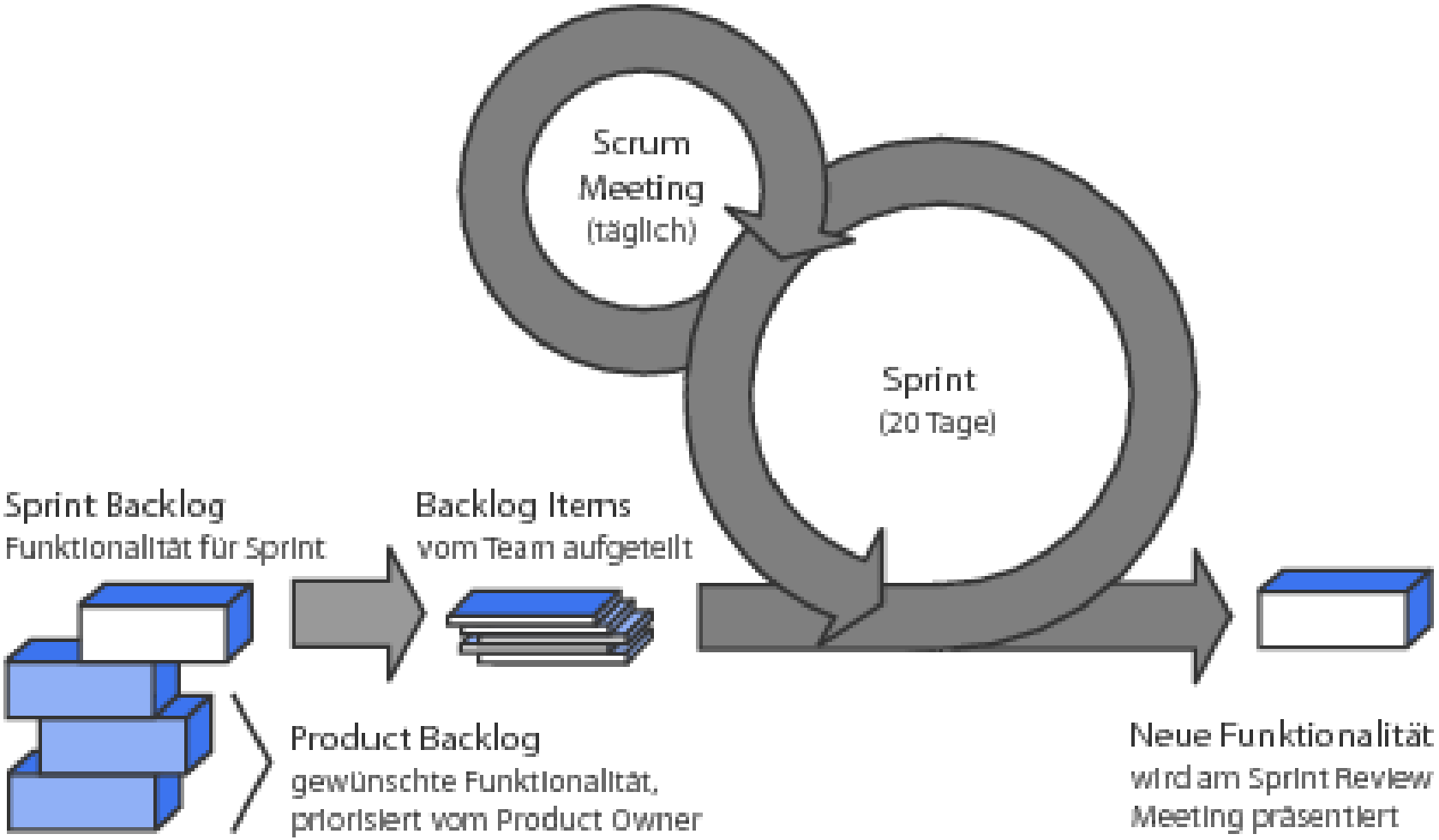
- hohe Produktivität und Anpassungsfähigkeit
- wenig Risiko und Ungewissheit
- größerer Komfort für die Projektmitglieder

Wurzeln in der empirischen Prozesssteuerung, die im Gegensatz zur definierten Prozesssteuerung über ständige Rückkoppelungen geführt wird

Scrum - Entwicklungsphilosophie

- Alle funktionalen und nichtfunktionalen Anforderungen werden in einem Product Backlog, gereiht nach Wichtigkeit, aufgelistet
- Verantwortlich für dieses Verzeichnis ist der Product Owner, der die Inhalte für den Product Backlog von den Stakeholdern erhält und die Priorisierung vornimmt
- Die eigentliche Entwicklung wird von kleinen Scrum Teams durchgeführt
- Das zentrale und wichtigste Element von Scrum ist der Sprint, der eine definierte Zeitdauer hat
- Jeder Sprint muss mit der Fertigstellung einer neuen Funktionalität enden
- Täglich findet überdies ein Daily Scrum Meeting statt, das der Scrum Master leitet
- Am Ende eines Sprints erfolgt ein Sprint Review Meeting

Scrum - Entwicklungsphilosophie



SCRUM - Elemente

Sprint: Innerhalb eines Sprints wird versucht, so schnell wie möglich zum zuvor definierten Sprint Goal zu kommen.

Daily Scrum: Dieses Meeting findet immer zur selben Zeit am selben Ort statt.

Product Backlog: Diese nach Dringlichkeit geordnete Liste aller Anforderungen kann von allen Projektbeteiligten jederzeit eingesehen werden

Increment: Bei Scrum entsteht ein Softwareprodukt in Inkrementen, wobei schon der erste, aber spätestens der zweite Sprint ein lauffähiges Softwaresystem (mit Kernfunktionalität) liefert.

Burndown Chart: In dieses Diagramm werden vom Scrum Master tagesaktuelle Aufwandsschätzungen für den aktuellen Sprint eingetragen

SCRUM – Rollen

Product Owner: Neben der Hauptaufgabe, der Erstellung und Priorisierung des Product & Release Backlogs, ist diese Person wesentlich dafür verantwortlich, dass das Team frei von Störungen und Unterbrechungen bleibt.

Scrum Team: Die Mission des Teams ist die Umsetzung der über den Sprint Backlog geforderten Funktionalität bzw. die Erreichung des über das Sprint Goal bestimmten Ziels.

Scrum Master: Die wesentliche Aufgabe dieser Rolle besteht darin, die Dynamik von Scrum zur ungestörten Entfaltung zu bringen, indem der Scrum Master einerseits auf die Umsetzung der Grundsätze und Regeln von Scrum achtet und andererseits das Team vor äußeren Einflüssen abschottet

eXtreme Programming (XP)

Eine kompakte Methode zur Entwicklung von Software in kleinen bis mittelgroßen Teams, deren Arbeit vagen oder sich rasch ändernden Anforderungen unterliegt

Hauptziel ist das termingerechte Abliefern von auftragsgemäßer Software

Charakteristische Merkmale von XP sind das eigene Wertesystem, viele Prinzipien und ein System von Techniken, die einander gegenseitig in höchstem Maße unterstützen

eXtreme Programming - Prinzipien

Kommunikation

- ständige Kommunikation zwischen allen Beteiligten (Entwicklern, Kunden, Sponsoren, Usern, ...)

Einfachheit

- Das einfachste Design welches die Aufgabe löst

Feedback

- Wissen wo man steht und was man verbessern kann (empirisches Prozessmodell)

Mut

- zur Offenheit und Transparenz
- notwendige Änderungen durchzuführen

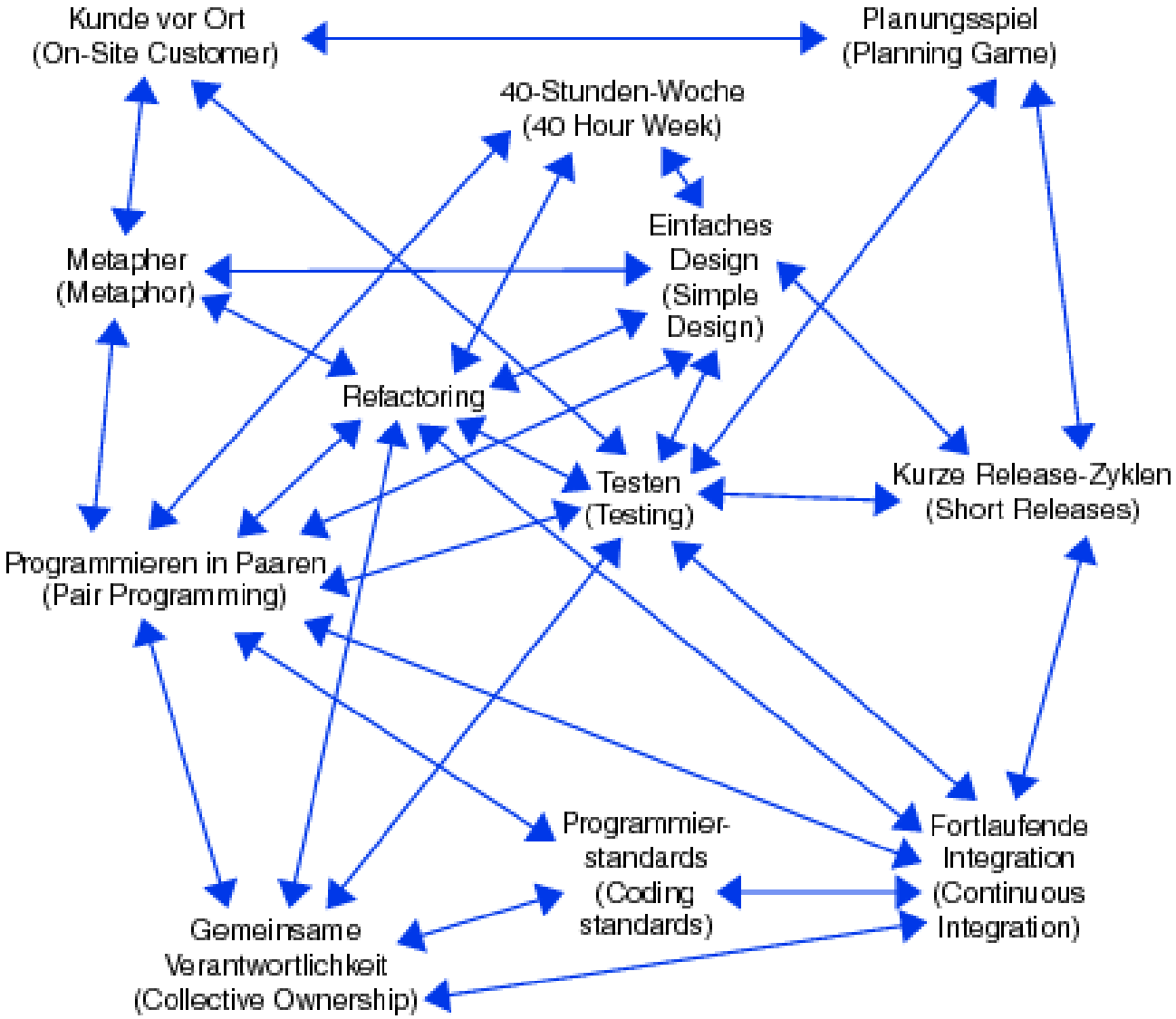
Respekt

- vor sich selbst, dem Team und dem Produkt

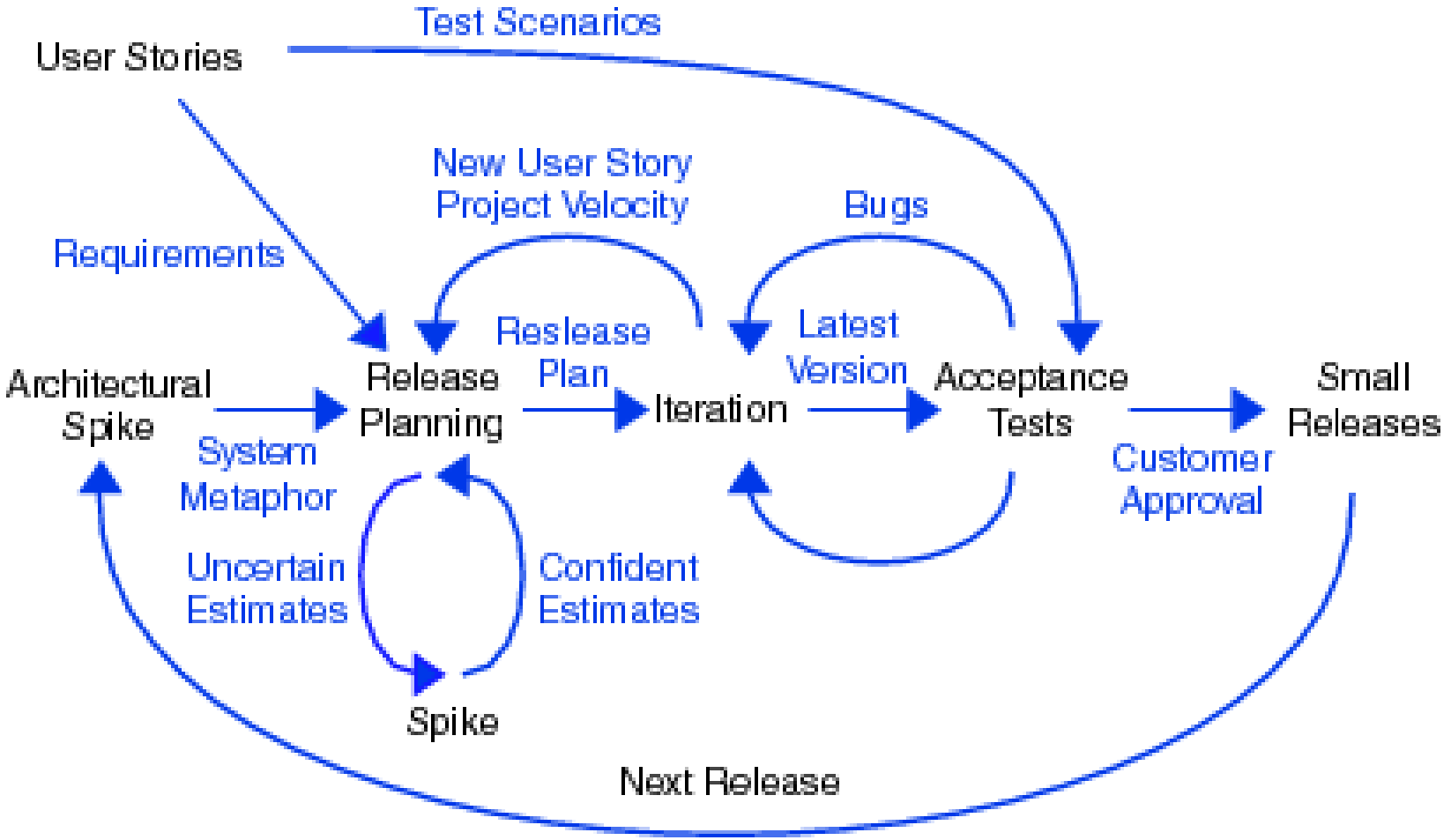
eXtreme Programming - Techniken

- Kunde vor Ort
- Planungsspiel
- Metapher
- Kurze Release-Zyklen
- Testen
- Einfaches Design
- Refactoring
- Programmieren in Paaren
- Gemeinsame Verantwortlichkeit
- Fortlaufende Integration
- Programmierstandards
- 40-Stunden-Woche.

eXtreme Programming- XP-Techniken



eXtreme Programming - Entwicklungsphilosophie



eXtreme Programming - Rollen

- Entwickler bzw. Programmierer
- Kunde
- XP-Trainer
- Tracker
- Tester
- Berater
- Big Boss

Kanban

Keine Rollen

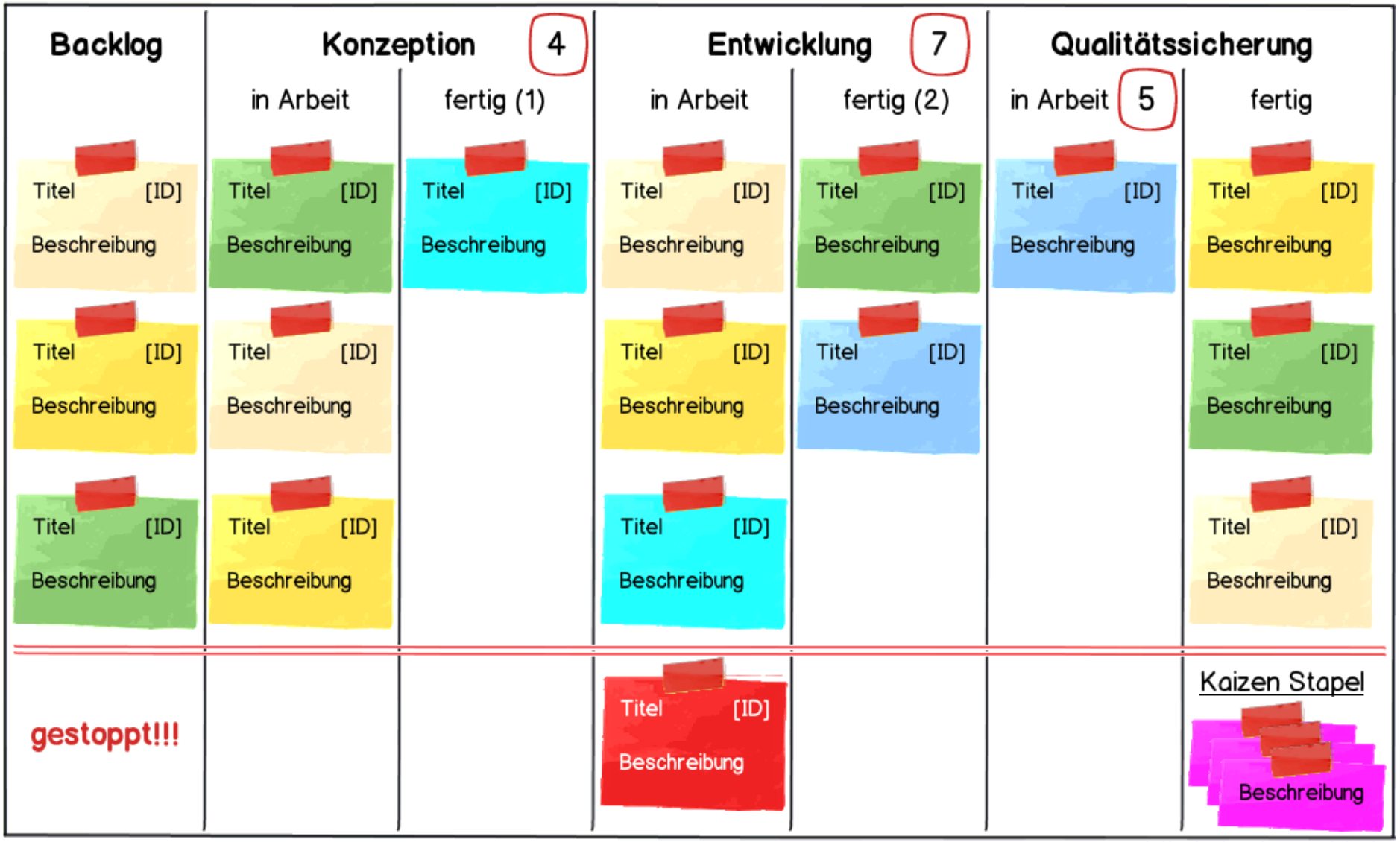
Keine Meetings

Kanbanboard

Work In Progress Limit

The Rest is up to You

Kanban Board Beispiel



V-Modell 97

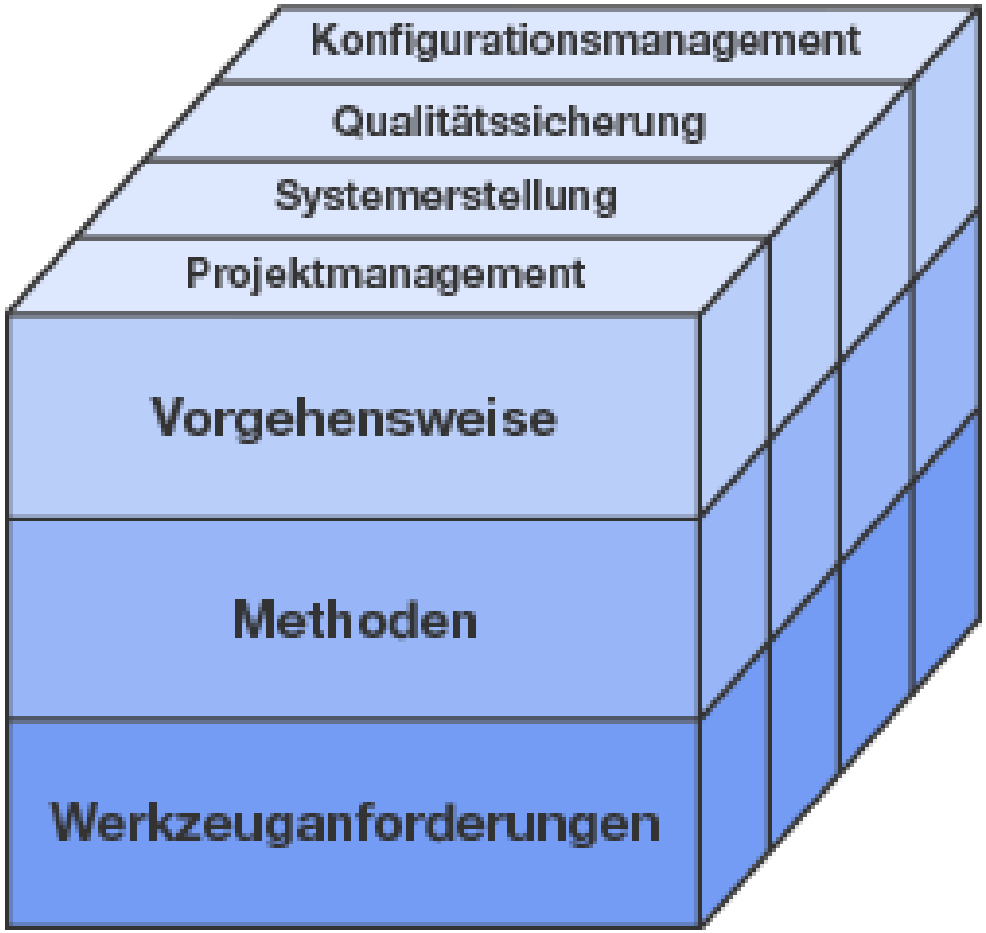
Wurde Anfang der 1990er Jahre mit dem Vorhaben entwickelt, alle bisherigen Ansätze hinsichtlich einer ingenieurmäßigen Softwareentwicklung unter einen Hut zu bringen

Dieses Prozessmodell ist ausgerichtet auf die Durchführungsphase und deckt insgesamt mehrere Aspekte ab

Es dient

- als Vertragsgrundlage bei der Auftragsvergabe
- als Arbeitsanleitung für die Softwareentwicklung
- als Kommunikationsbasis zwischen den beteiligten Parteien

V-Modell 97 - Aufbau



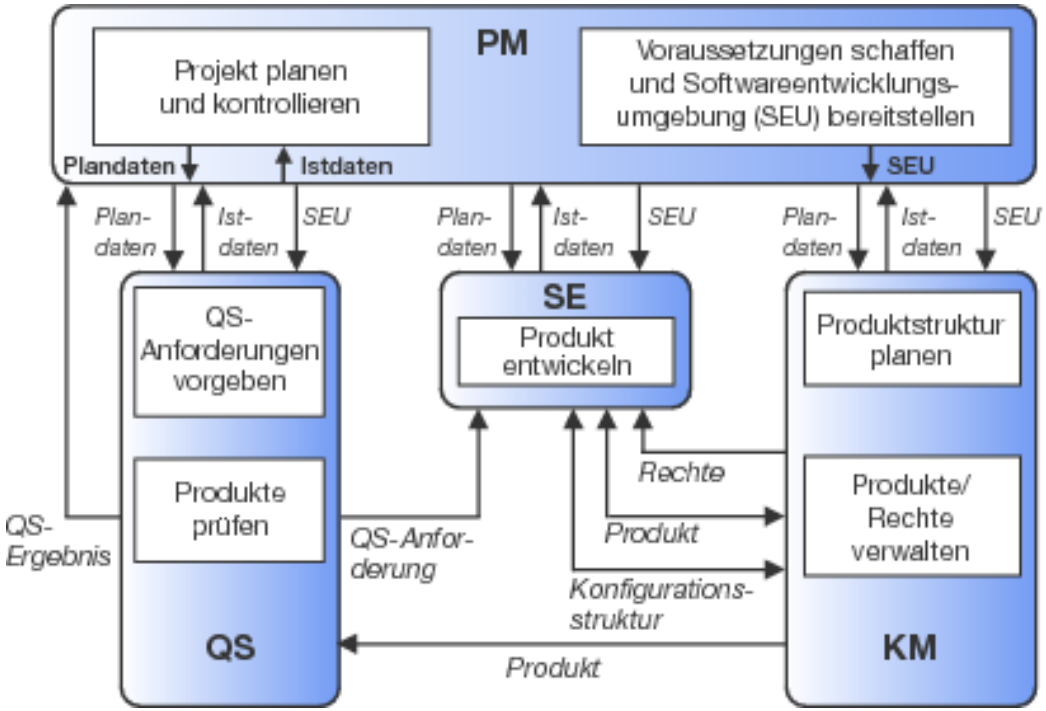
V-Modell 97 - Submodelle

Projektmanagement (PM)

Systemerstellung (SE)

Qualitätssicherung (QS)

Konfigurationsmanagement(KM)



V-Modell 97 - Entwicklungsphilosophie

Die Entwicklung im V-Modell 97 erfolgt inkrementell

Zu Beginn wird die Gesamtfunktionalität abgegrenzt und die einzelnen Funktionen des Softwaresystems werden priorisiert

Die einzelnen Funktionen werden dann gemäß ihrer Priorität zu Ausbaustufen zugeordnet und im Rahmen dieser umgesetzt

Das Ergebnis einer Stufe wird den Endanwendern zur Verfügung gestellt

Der Umfang des Systems wächst somit ausgehend von einer Basisfunktionalität stetig an

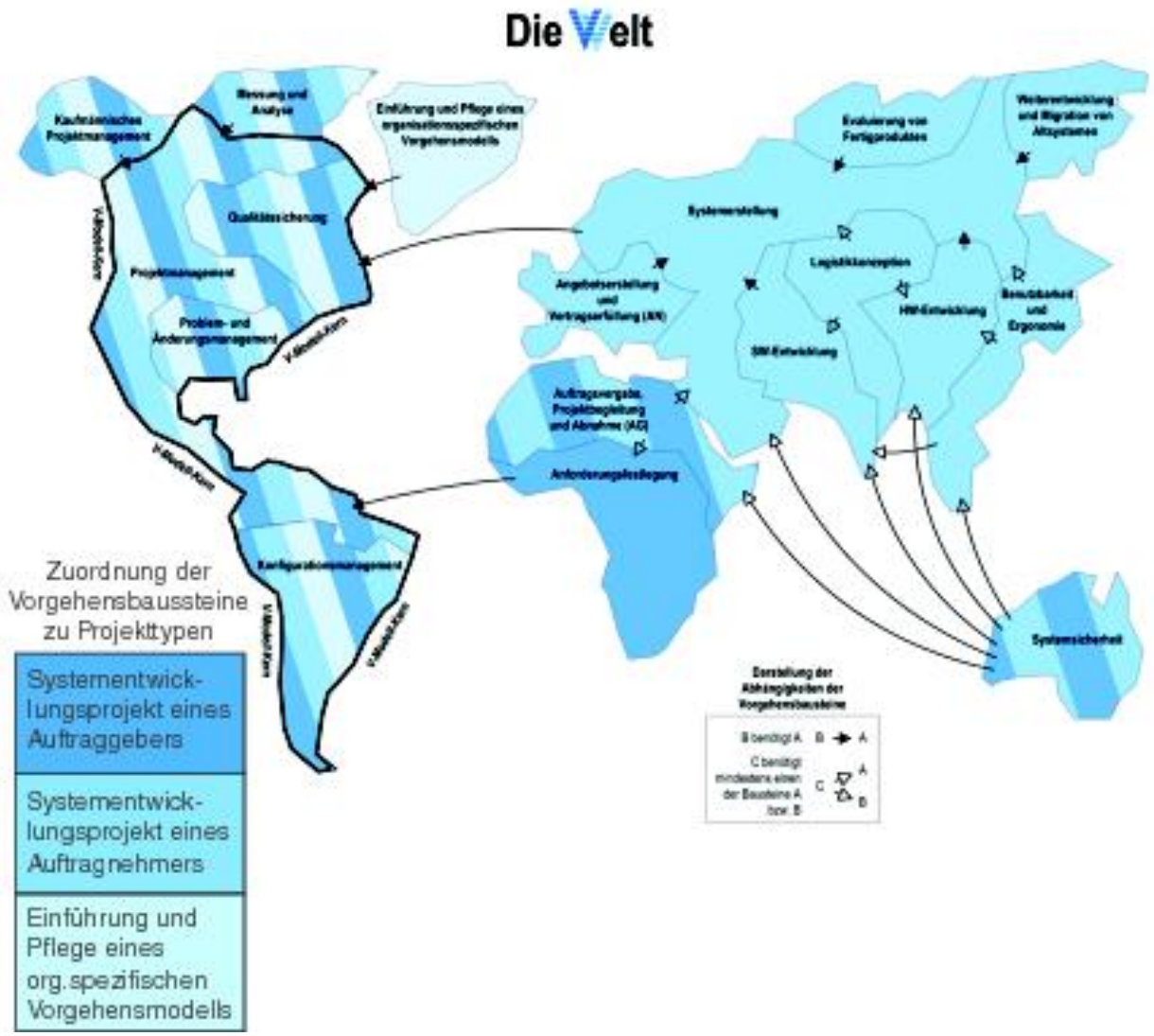
V-Modell XT

„XT“ steht bei diesem Modell für „extreme tailoring“ bzw. für „extendable“

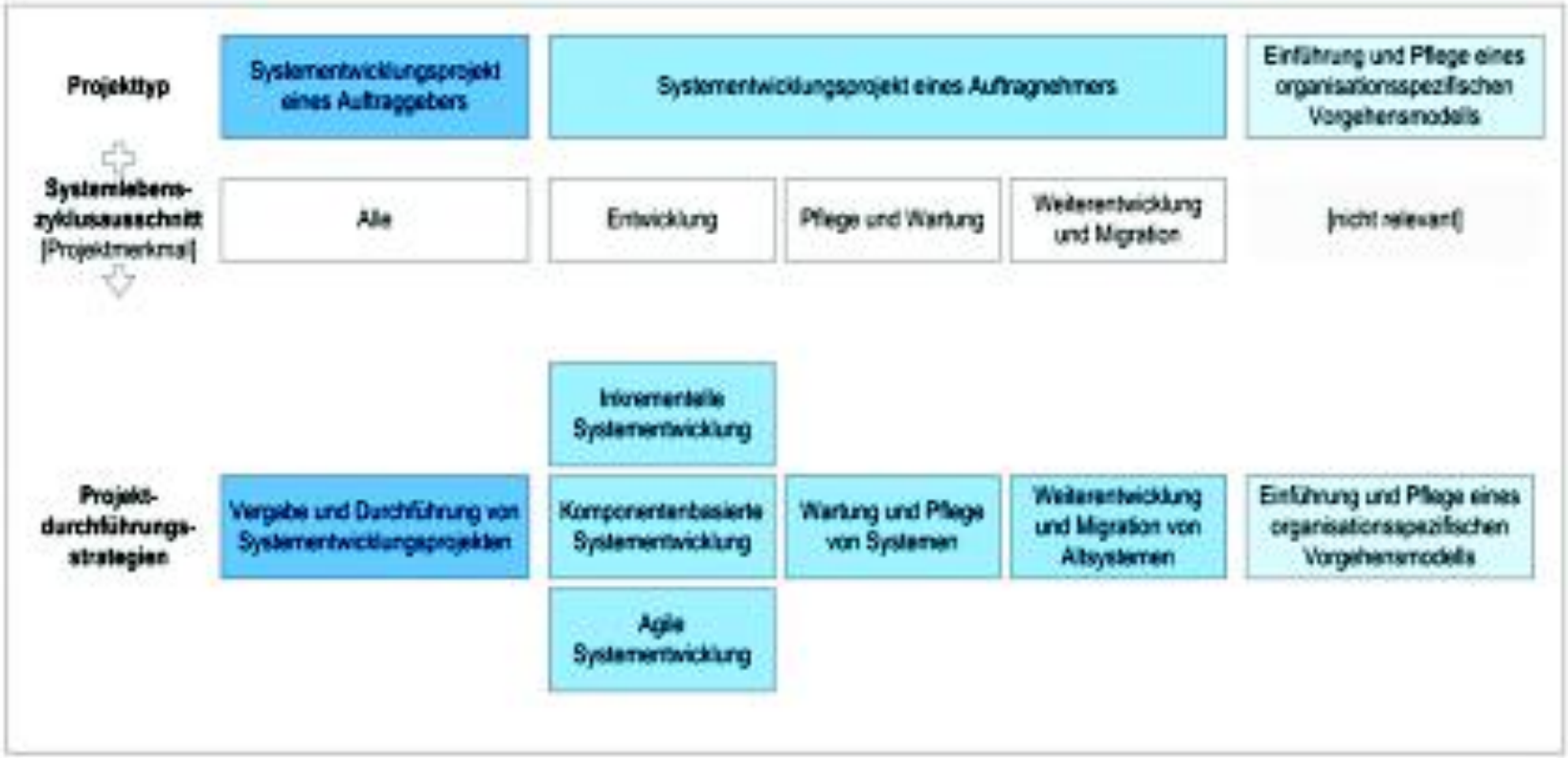
Ziele

- Bessere Anpassungs- und Anwendungsmöglichkeit bei unterschiedlichen Projekten und Organisationen
- Skalierbarkeit hinsichtlich verschiedener Projektgrößen
- Bessere Änderungs- und Erweiterungsmöglichkeiten des Modells selbst
- Anpassung an die aktuellen Normen und Vorschriften
- Betrachtung des gesamten Systemlebenszyklus
- Unterstützung der Einführung sowie Verbesserung des Modells selbst und Anpassung an organisationsspezifische Gegebenheiten

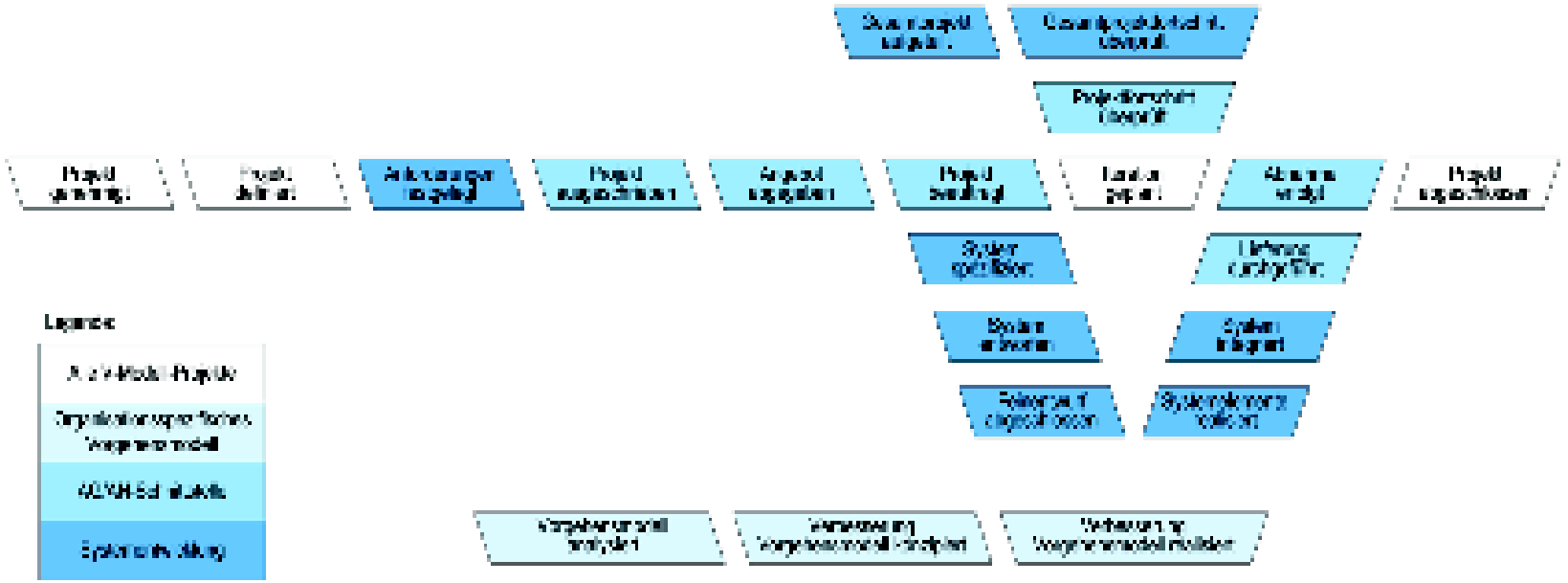
V-Modell XT - Vorgehensbausteine



V-Modell XT - Projektdurchführungsstrategien



V-Modell XT - Entscheidungspunkte



V-Modell XT - Eigenschaften

- Ziel- und ergebnisorientierte Vorgehensweise
- Produkte stehen im Mittelpunkt des V-Modells und stellen die zentralen Projektergebnisse dar
- Die Reihenfolge der Produktfertigstellung und damit die grundlegende Struktur des Projektverlaufes werden durch Projektdurchführungsstrategien und Entscheidungspunkte vorgegeben
- Auf Basis der Bearbeitung und Fertigstellung von Produkten wird die detaillierte Projektplanung und -steuerung vorgenommen

ISO/IEC 12207

ISO/IEC 12207 - Prozesse

Drei verschiedenen Arten von Prozessen

- **Primary Processes**: Primäre Prozesse zeigen Funktionen auf, die von den am Softwareprojekt beteiligten Parteien ausgeführt werden können.

- **Supporting Processes**: Diese Prozesse können von primären oder von anderen Prozessen dieser Kategorie unterstützend verwendet werden.

- **Organizational Processes**: Diese Art von Prozessen können eingesetzt werden, um Aufgaben, die organisatorische bzw. projektübergreifende Aspekte betreffen, umzusetzen und um andere Prozesse zu etablieren, zu kontrollieren und zu verbessern.

ISO/IEC 12207 - Prozesse

Agile versus traditionelle Vorgehensmodelle

- Weder agile noch plangetriebene Methoden stellen eine „Silver Bullet“ dar
- Sowohl agile als auch plangetriebene Methoden haben für gewisse Einsatzgebiete deutliche Vorteile gegenüber dem anderen Modell. Zum Beispiel sind agile Methoden für Softwareentwicklungsprojekte mit einer raschen time-to-market vorzuziehen
- In Zukunft werden sowohl agile als auch plangetriebene Aspekte eine wesentliche Rolle in der Applikationsentwicklung spielen
- Methoden werden idR ausbalanciert sein
- Es ist besser, eine Methode für seine eigenen Bedürfnisse zu ergänzen als eine Methode zu reduzieren
- Methoden sind wichtig, aber potenzielle „Silver Bullets“ liegen im Bereich „Personen“, „Werte“, „Kommunikation“ und „Management der Erwartungen“

Entscheidungsfaktoren nach Böhm und Thurner

Teamstruktur (Personnel): Boehm und Thuner nehmen in ihrem Modell an, dass die Qualifikation des Teams bzw. der beteiligten Personen beim Einsatz agiler Methoden höher sein muss als beim Einsatz plangetriebener Methoden.

Dynamik der Anforderungen (Dynamism): Je häufiger Anforderungen geändert werden, desto einfacher muss dies möglich sein.

Entwicklungskultur (Culture): Je nach organisatorischem Umfeld bzw. Entwicklungskultur können agile Methoden eingesetzt werden.

Teamgröße (Size): Es wird oftmals argumentiert, dass agile Methoden in kleinen bis mittelgroßen Teams effizient einsetzbar sind.

Kritikalität (Criticality): Wenn Menschenleben von einem Softwaresystem abhängig sind, werden natürlich viel höhere Maßstäbe an die rigorose Prüfung und Nachvollziehbarkeit auf allen Ebenen gelegt.

Entscheidungsfaktoren nach Böhm und Thurner