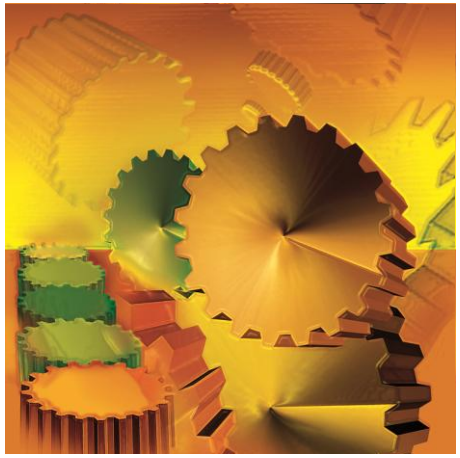
Software Engineering und Projektmanagement 2.0 VO



EINHEIT 8

Vorgehensmodelle, Qualitätssicherung & -management

“I'm not a great programmer; I'm just a good programmer with great habits.” Kent Beck (creator of XP)

WS2016/17 – 15. Dezember 2016

Dr. Raoul Vallon
www.inso.tuwien.ac.at

Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Grundlagen und Bedeutung von Vorgehensmodellen

- **Softwareprodukte werden immer umfangreicher und komplexer**
- **Effektive und effiziente Softwareentwicklung 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**

- **Es existiert kein optimaler Entwicklungsprozess, eine Art Universalprozess, für jedes Softwareprojekt:**
  - Je nach Projektart ist ein geeignetes Vorgehen auszuwählen
  - Bisher keine einheitliche Systematik der Softwareentwicklung
  - Mittlerweile haben sich bereits einige sogenannte State-of-the-Art-Software-Entwicklungsprozessmodelle etabliert

Die drei Steuergrößen eines Projektes

- **Kosten**
- **Umfang und Inhalt (Qualität)**
- **Zeit**

Wählen Sie zwei:

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

Aktuelle Herausforderungen: Verteilte Softwareentwicklung

- Kommunikation
- Kontrolle
- Koordination

Fundamentale Prozessmodelle

- **Grundlegenden Idee des Phasenmodells**

- **Phasen dieses Modells sind in unterschiedlicher Form in jedem Prozessmodell enthalten**

Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

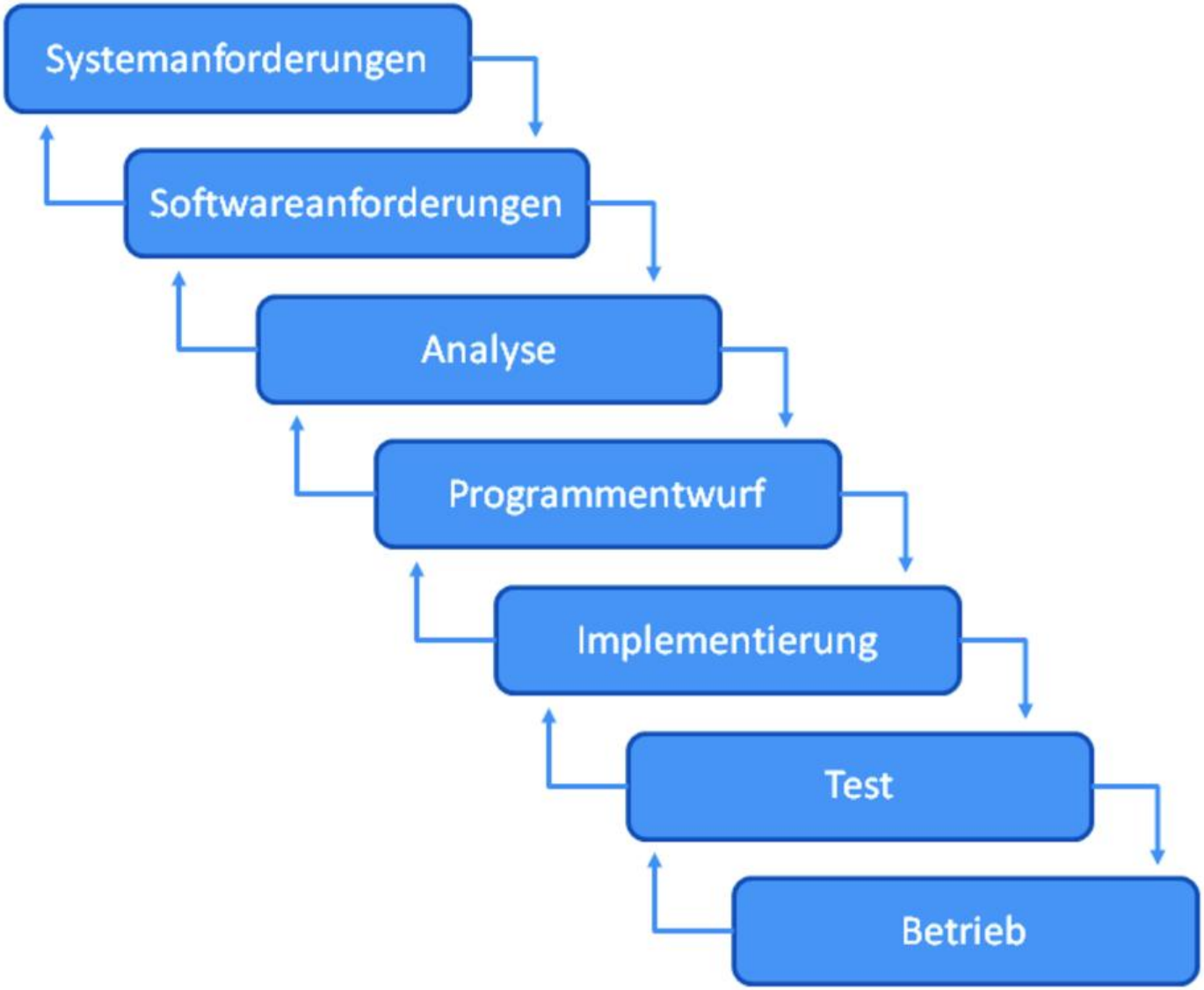
II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

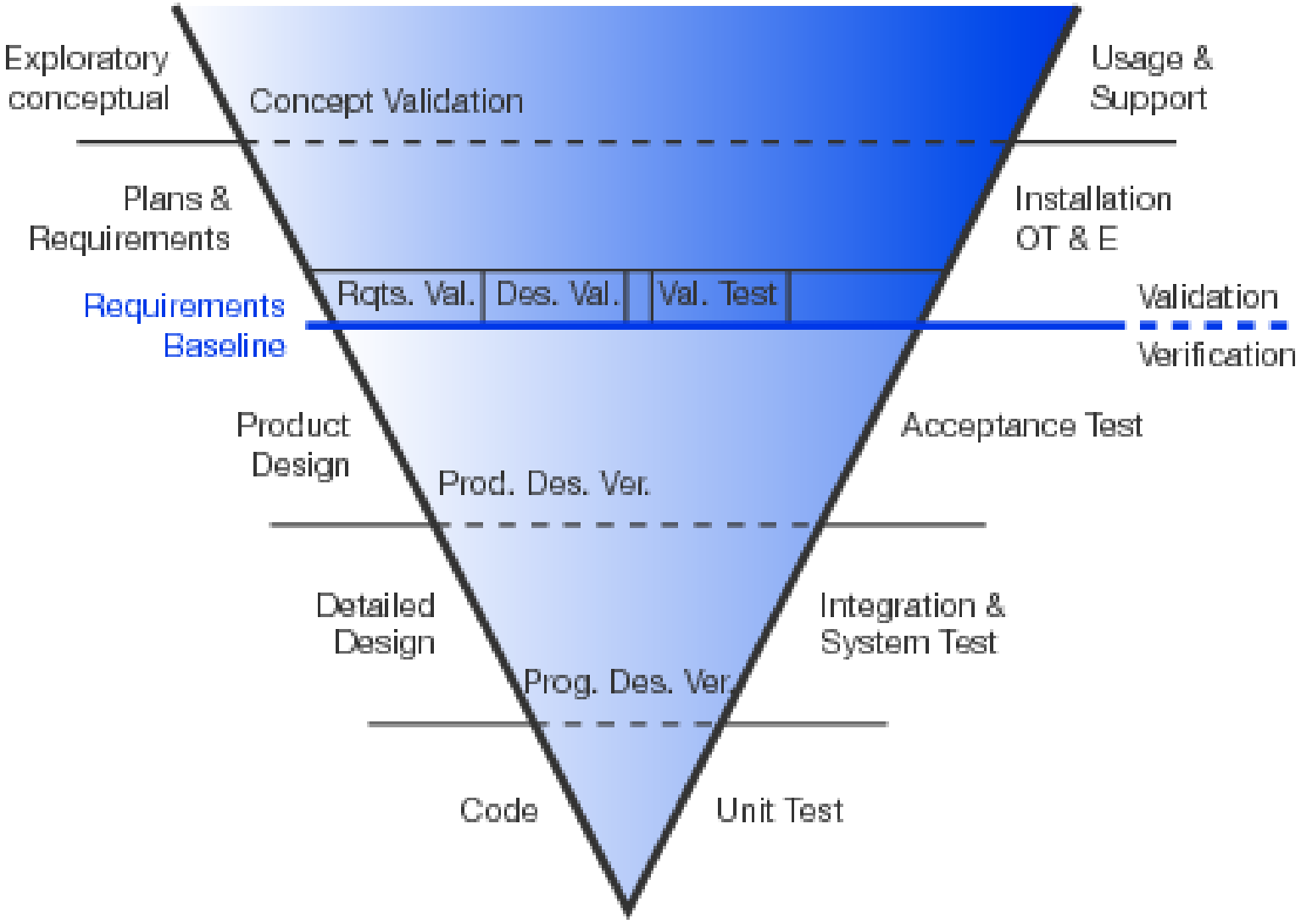
Wasserfallmodell nach Royce 1970



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 risikoreich betrachtet und hat zur Entwicklung der iterativen Modelle geführt**

Fundamentales V-Modell nach Böhm



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 97 & XT - Entwicklungsphilosophie

V-Modell 97:

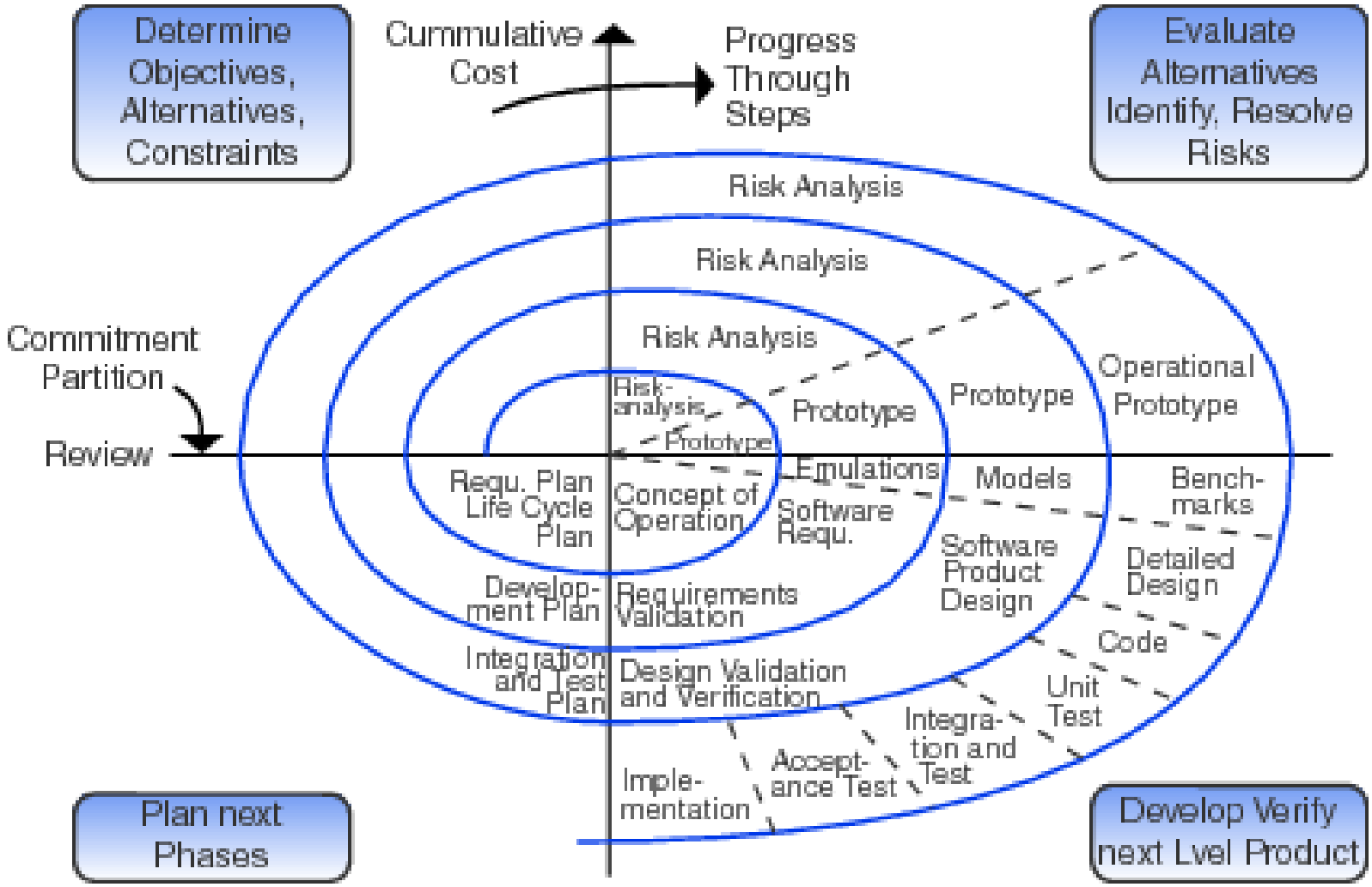
- Die Entwicklung im V-Modell 97 erfolgt inkrementell
- Zu Beginn wird die Gesamtfunktionalität abgegrenzt und die einzelnen Funktionen des Softwaresystems werden priorisiert
- Die einzelnen Funktionen werden dann gemäß ihrer Priorität zu Ausbaustufen zugeordnet und im Rahmen dieser umgesetzt
- Das Ergebnis einer Stufe wird den Endanwendern zur Verfügung gestellt
- Der Umfang des Systems wächst somit ausgehend von einer Basisfunktionalität stetig an

- **V-Modell XT („eXtreme Tailoring“) – 21.7.2015 v2.0**
  - Skalierbarkeit hinsichtlich verschiedener Projektgrößen (reduzierte Menge an Aktivitäten und Produkte für kleinere Projekte)
  - Mehr Orientierung Richtung agilem Vorgehen: keinerlei Vorschriften über die zeitliche Abfolge von Vorgehensbausteinen

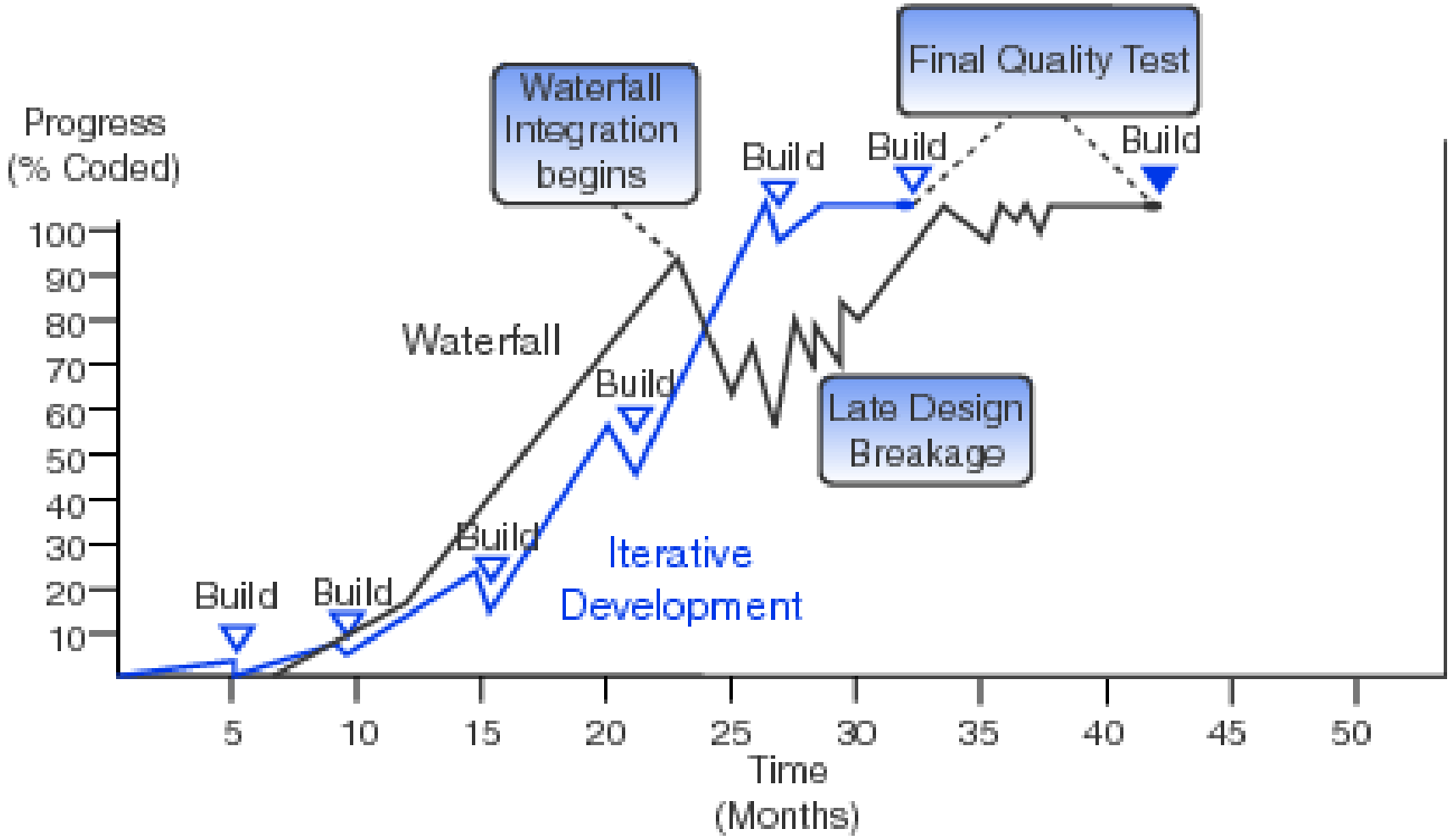
Spiralmodell nach Böhm 1988

- **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

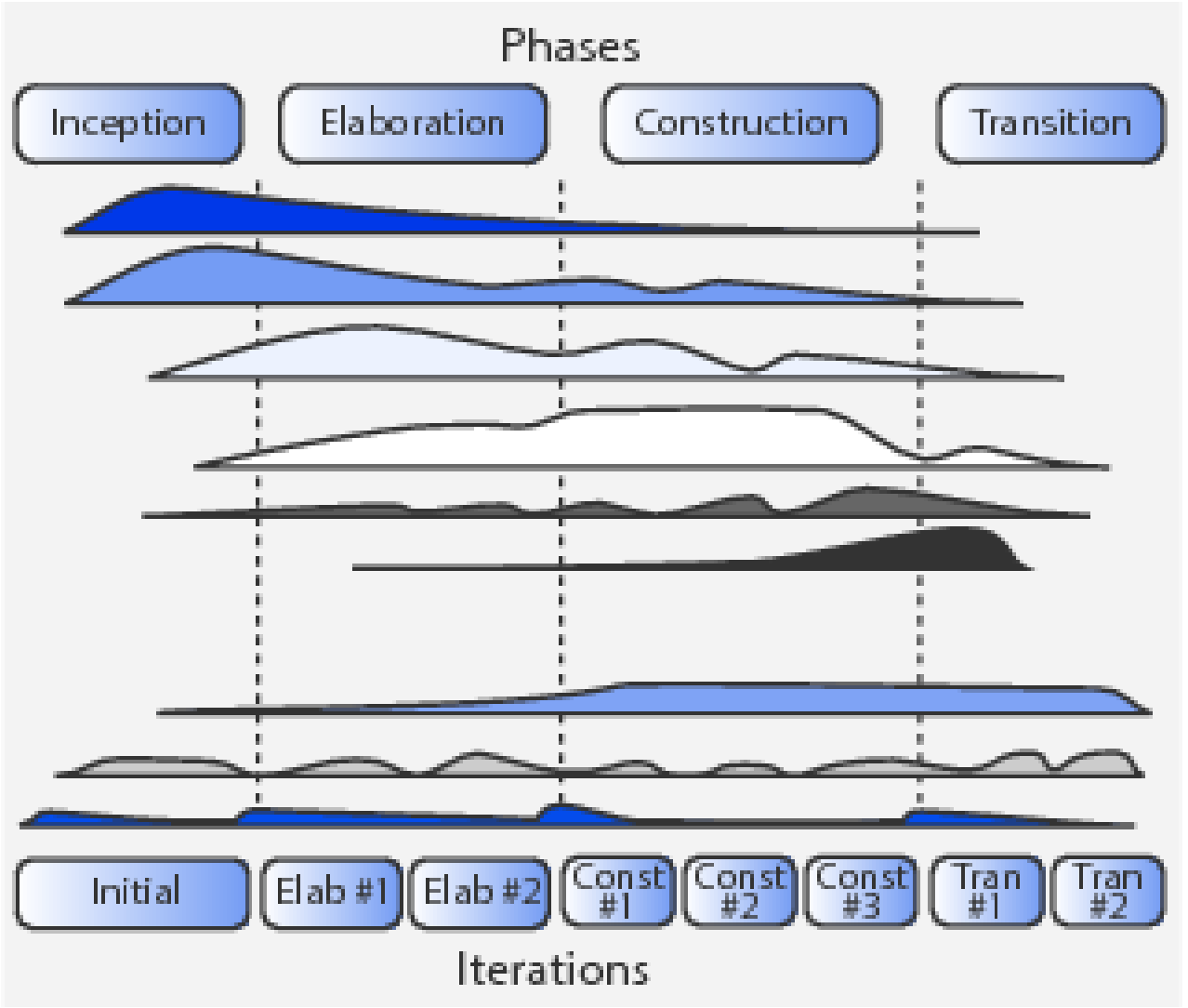


Beispiel für eine Iterative Entwicklung

Industriell geprägte Modelle: Rational Unified Process 1998

- **Prozess-Framework, das von IBM Rational gepflegt, ständig weiterentwickelt und momentan in der Version 9.0 (2006) ist**
- **Eines der am meisten verbreiteten und detailliert beschriebenen Prozessmodelle**
- **Der Rational Unified Process (RUP) beruht auf folgenden Vorgehensweisen:**
  - Software iterativ entwickeln
  - Anforderungen managen
  - Komponentenbasierte Architekturen verwenden
  - Software visuell modellieren (UML)
  - 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 mit u.a. 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**

Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Agile Software Development (Agile Manifesto) 2001

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

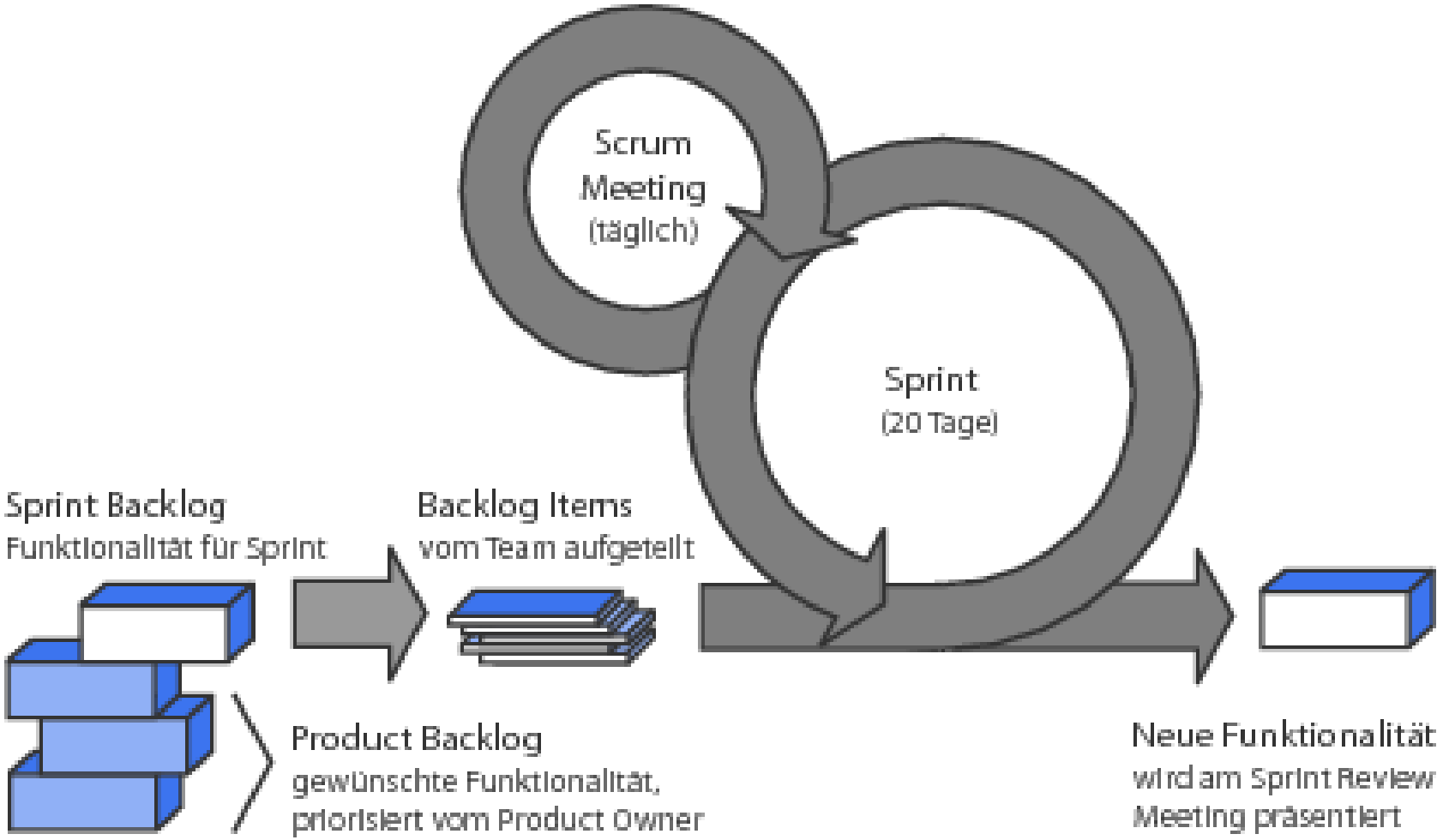
- ***Individuals and interactions*** **over processes and tools**
- ***Working software*** **over comprehensive documentation**
- ***Customer collaboration*** **over contract negotiation**
- ***Responding to change*** **over following a plan**

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

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



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)

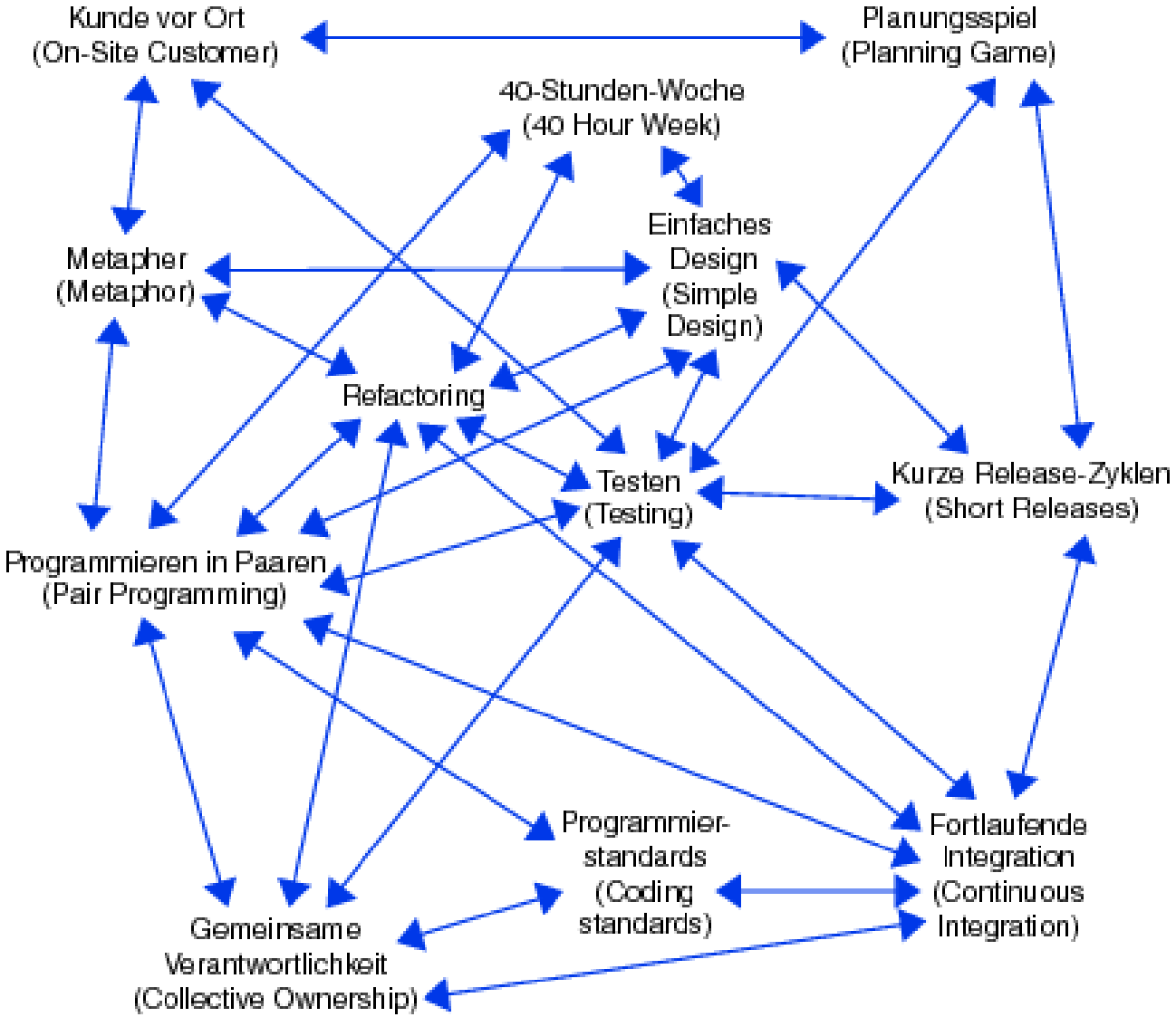
Mut

- zur Offenheit und Transparenz
- notwendige Änderungen durchzuführen

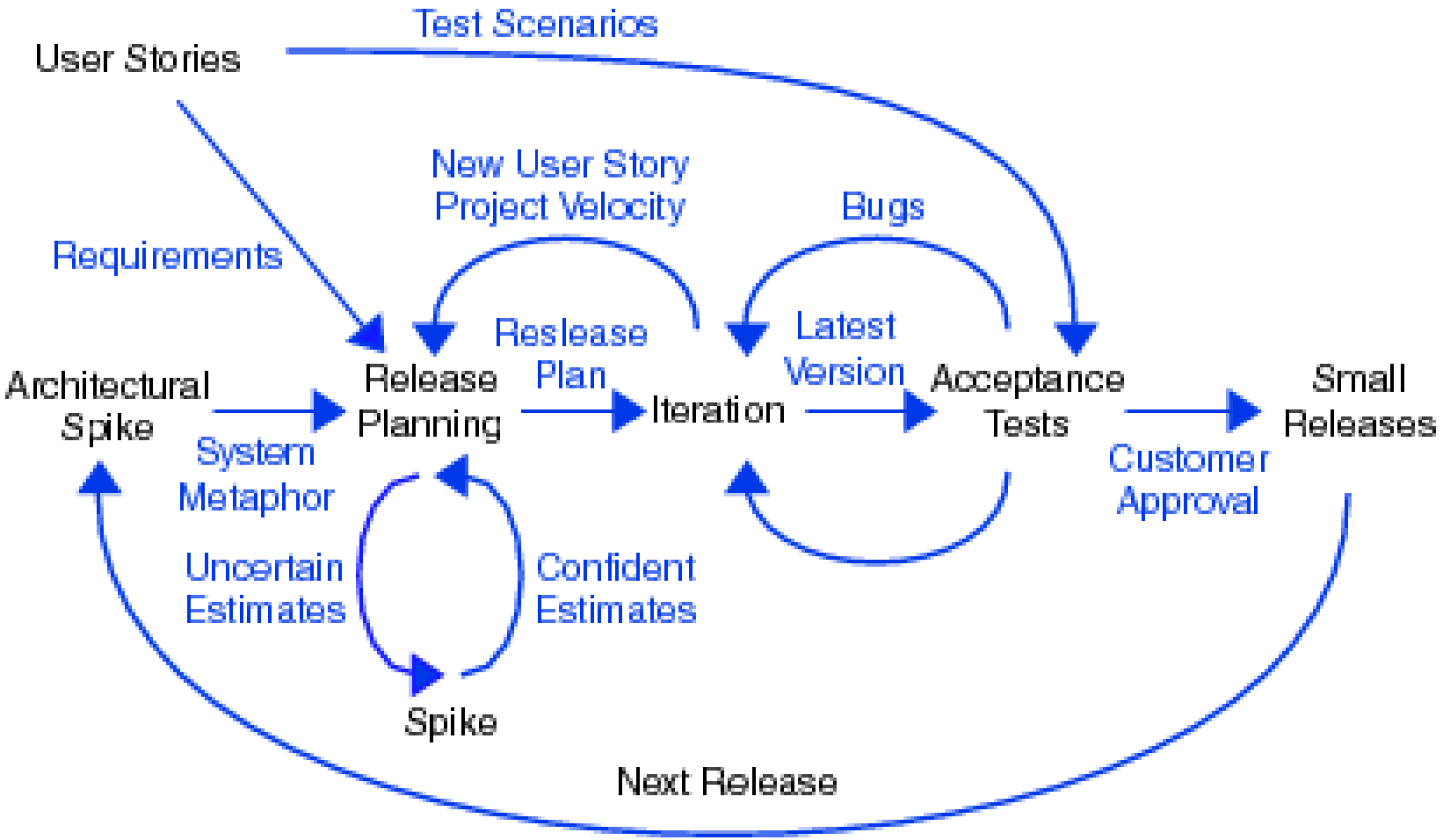
Respekt

- vor sich selbst, dem Team und dem Produkt

eXtreme Programming- XP-Techniken



eXtreme Programming - Entwicklungsphilosophie



XP@Scrum

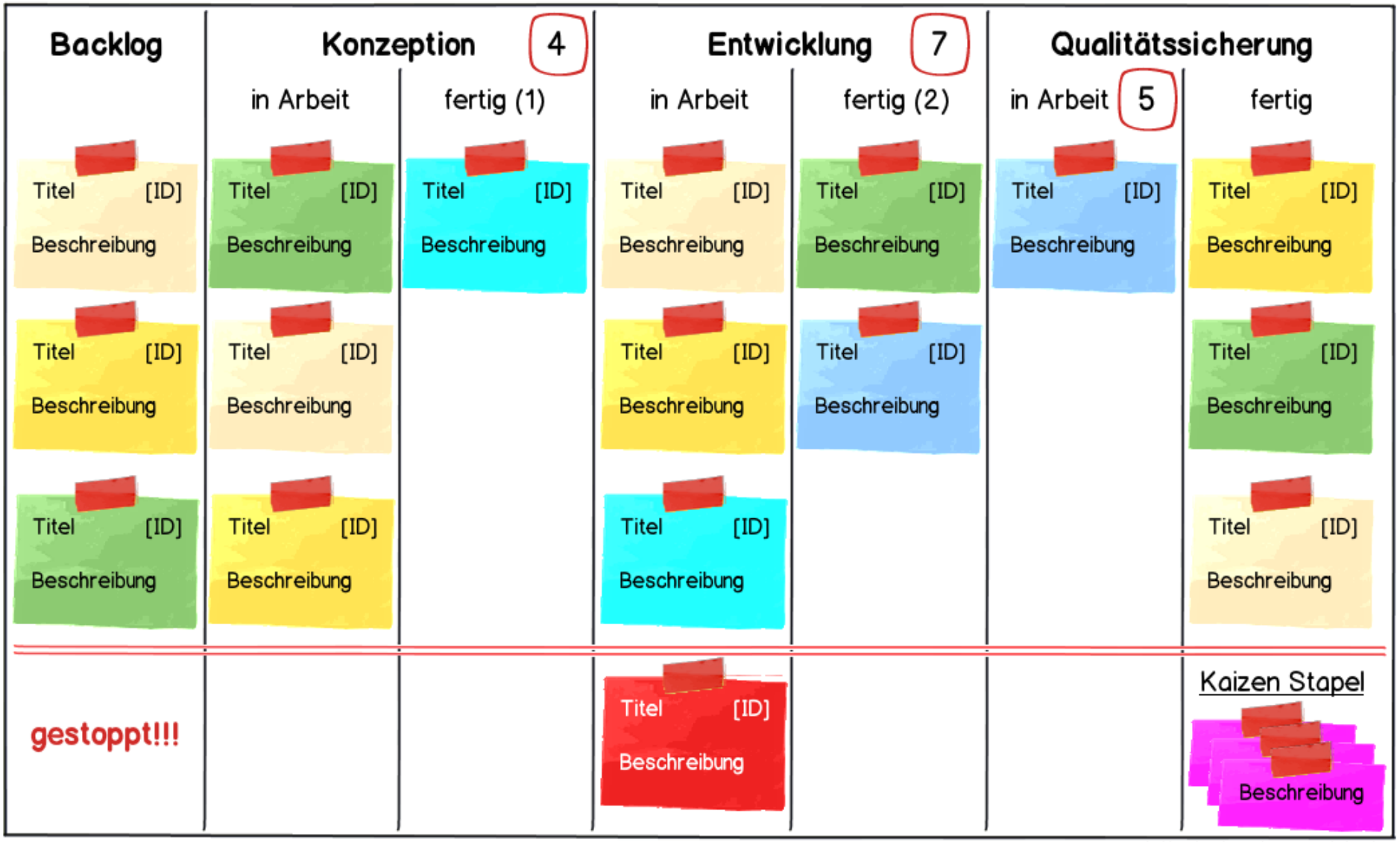
In der Praxis sehr beliebt:

- **Scrum Prozess inkl. Meetings, Rollen und Vorgehensmodell**
- **Zusätzliche Feedbackschleifen über XP-Techniken:**
  - Pair Programming
  - Code Reviews
  - Continuous Integration
  - etc.
- **Scrum als Vorgehensmodell, erweitert mit XP Entwicklungstechniken**

Kanban

- **Lean Management/Thinking: Ursprünglich aus der Automobilindustrie (1978 Toyota Production System)**
- **Von Poppendieck 2003 für die Softwareentwicklung adaptiert:**
  - Eliminate waste
  - Amplify learning
  - Decide as late as possible
  - Deliver as fast as possible
  - Empower the team
  - Build integrity in
  - See the whole
- **Keine vorgeschriebenen Rollen, Meetings, Artefakte, außer:**
  - Kontinuierlicher “Pull” Workflow
  - Kanbanboard
  - Work In Progress Limit

Kanban Board Beispiel



Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

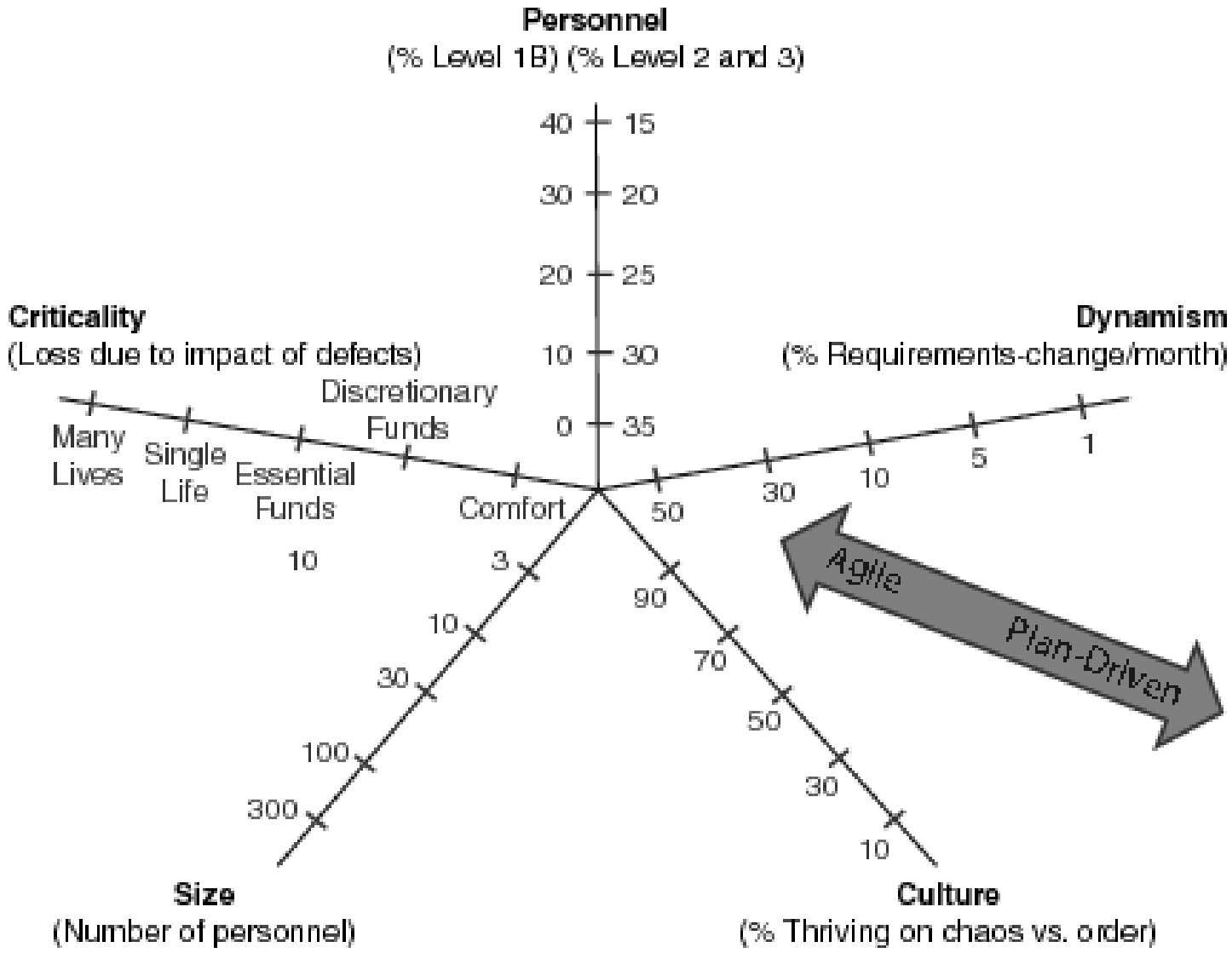
Agile versus traditionelle Vorgehensmodelle

- **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 i.d.R. 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 Thurner 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



Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

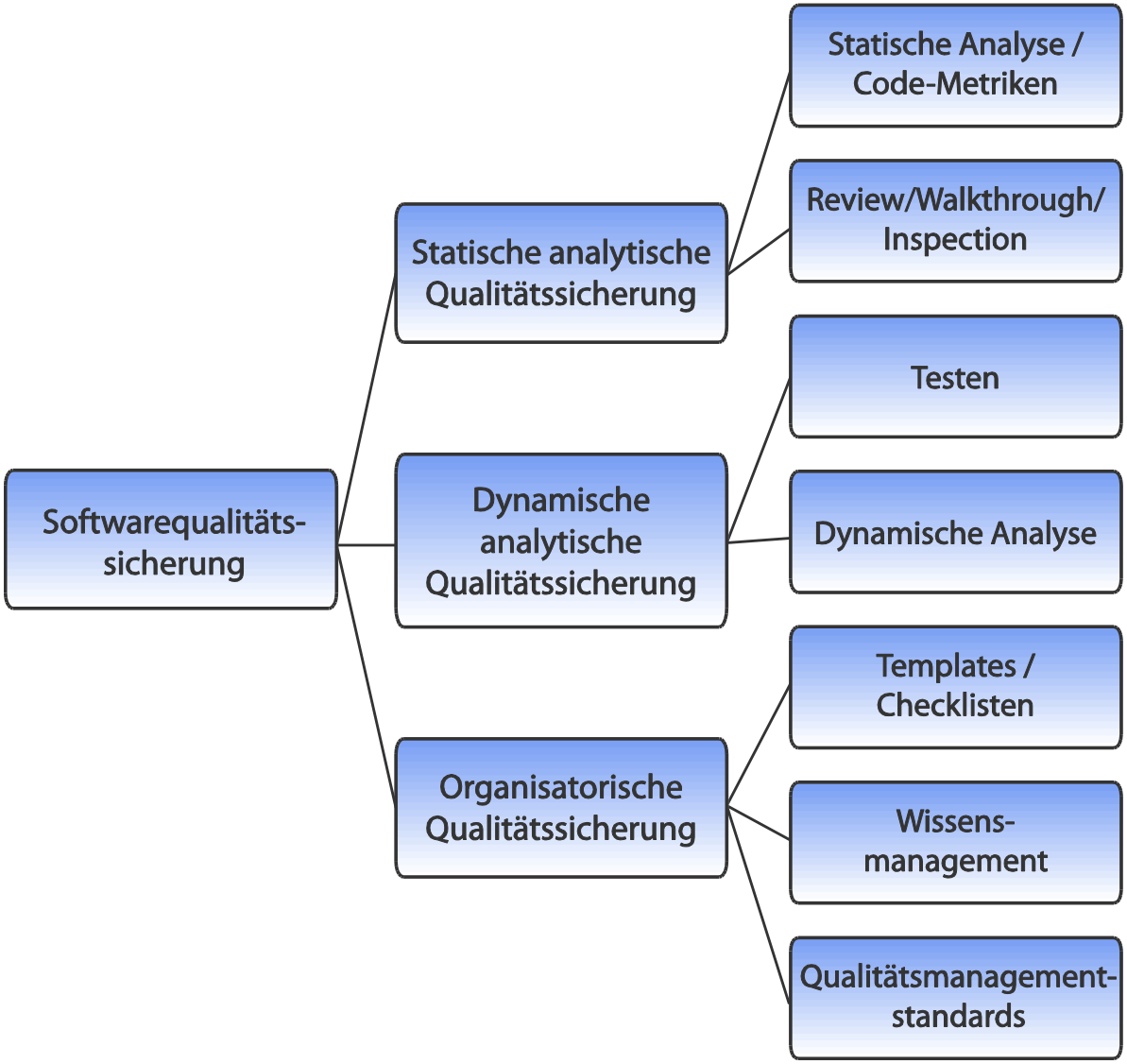
II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Software-Qualitätssicherungsmethoden



Softwarequalitätsfaktoren

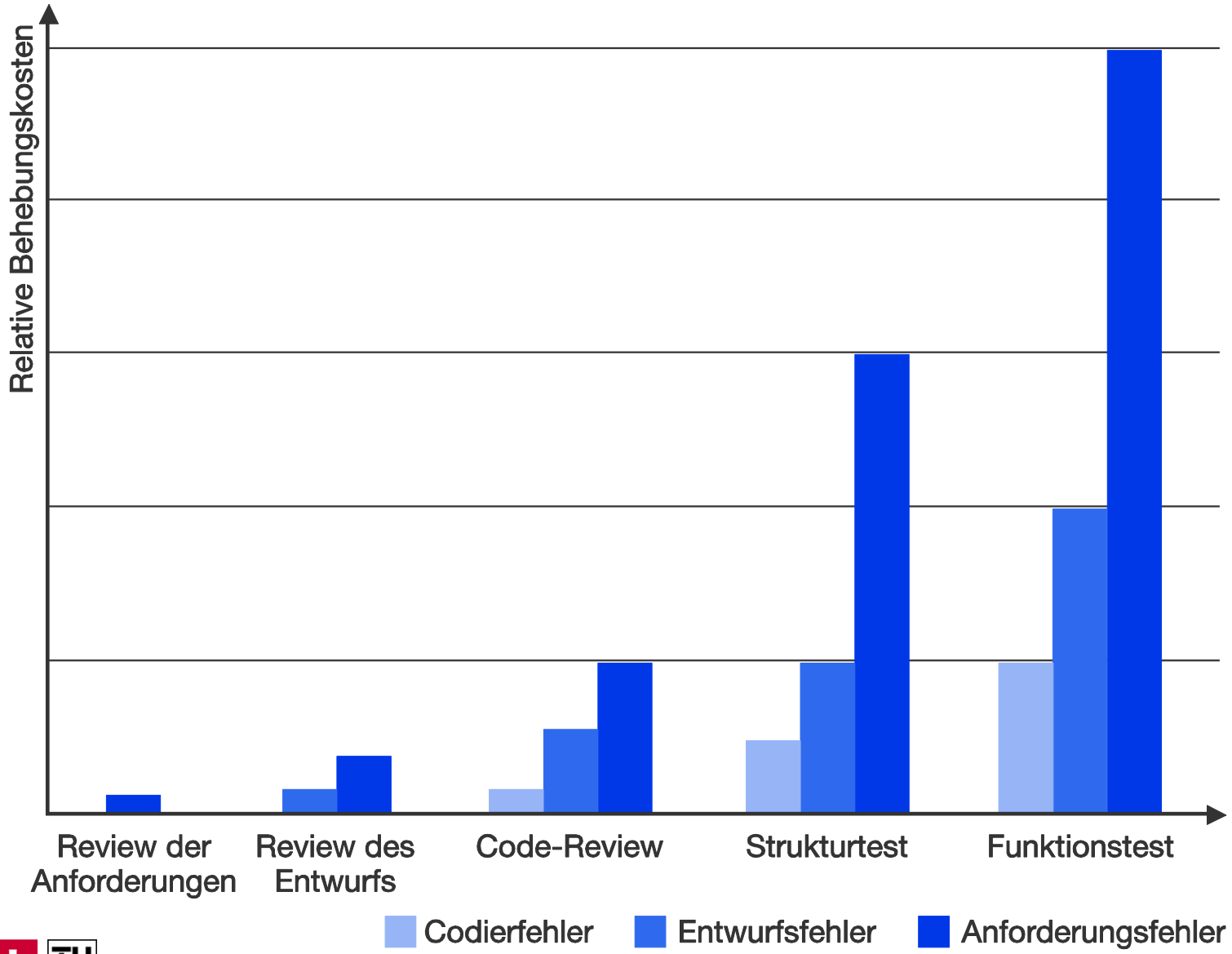
- **Qualitätsfaktoren ermöglichen Aussagen über die Qualität von Software zu treffen**

- **Einen Ansatz Qualitätsfaktoren zu definieren und in Gruppen zusammenzufassen beschreibt die ISO/IEC 9126**
  - Funktionalität
  - Zuverlässigkeit
  - Benutzbarkeit
  - Effizienz
  - Änderbarkeit
  - Übertragbarkeit

Ursachen von Softwarefehlern

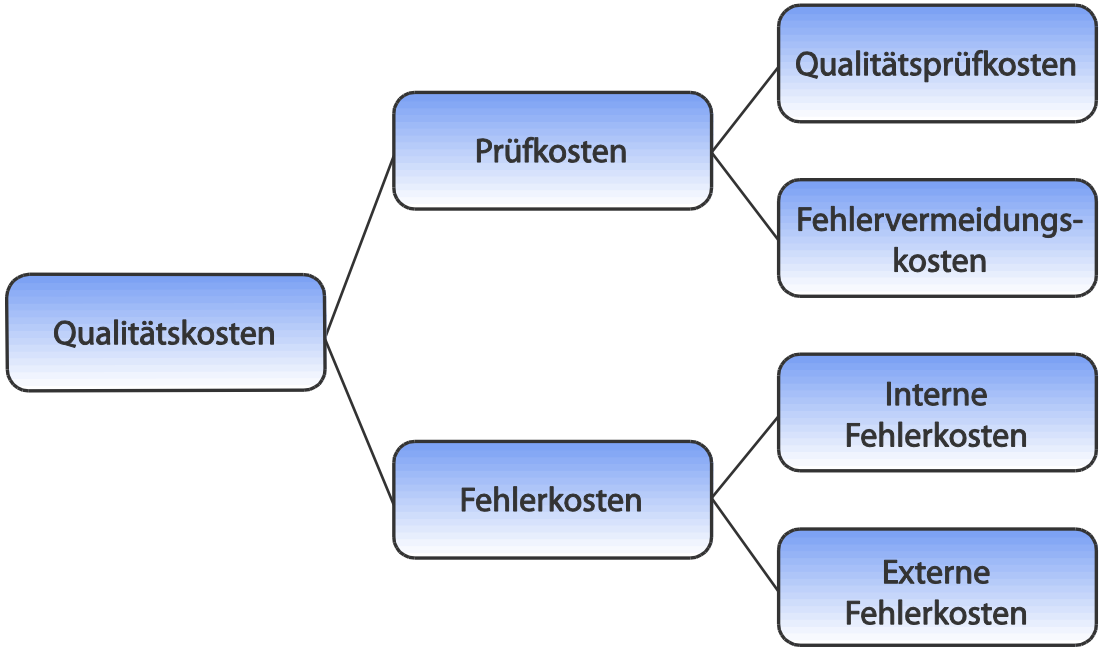
- **Anforderungen:**
  - Fehlerhafte Definitionen in den Anforderungsdokumenten
  - Falsche Interpretation von Kundenanforderungen
- **Designdokumente:**
  - Logische Designfehler aufgrund fehlerhafter Anforderungen von Designexperten, Softwareentwickler und Analysten
  - Fehlinterpretationen des Designdokuments
- **Fehler bei der Test-Datenauswahl**
- **Nichteinhaltung von Coding Standards**

Fehlerentstehung und Fehlerentdeckung

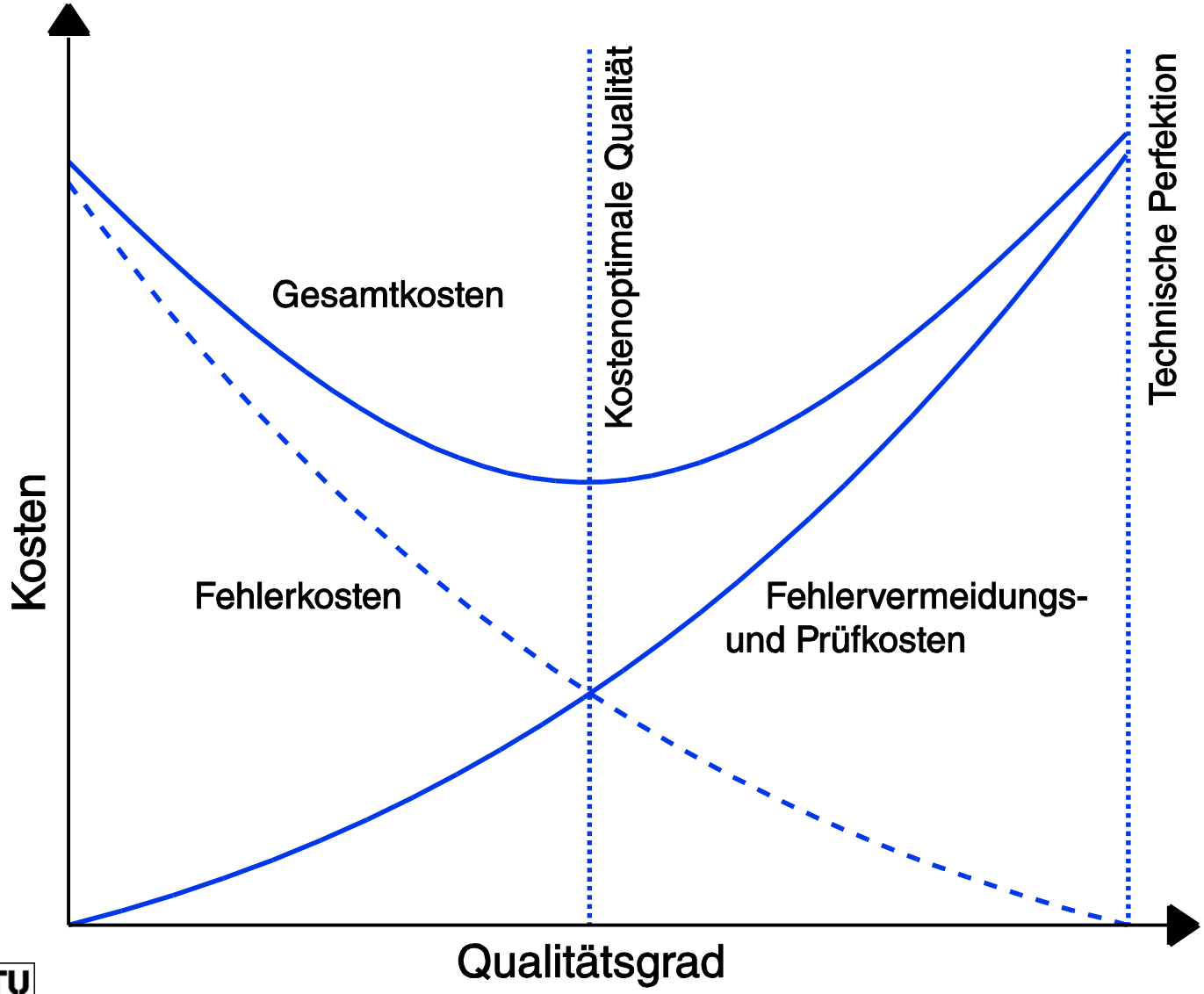


Kosten der Qualitätssicherung

- Kosten setzen sich aus den Herstellungskosten und den Qualitätskosten zusammen
- Wichtiges Ziel in der Softwareproduktion ist die Vermeidung von Fehlern und deren Folgekosten
- Qualitätskostenmodell nach Feigenbaum:



Kostenoptimierung



Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Statische Qualitätssicherung

- Analytische Aktivitäten zur Prüfung eines Testobjektes, ohne dass dieses dynamisch ausgeführt wird
- Prüfung von Testobjekten bereits sehr früh im Projektlebenszyklus möglich, noch bevor ausführbarer Code zur Verfügung steht
- Statische Codeanalysen ermöglichen die Prüfung einzelner Programmteile, bereits lange vor der Integration zu einem lauffähigen Gesamtsystem
- Mit Hilfe der Strukturanalyse kann die innere Qualität ermittelt und über den gesamten Entwicklungsprozess auf einem hohen Niveau gehalten werden. Basis einer Strukturanalyse sind Abhängigkeitsgraphen und Qualitätsmetriken
- Überprüfung der Einhaltung von Entwicklungsrichtlinien durch syntaktische Prüfung
- Fehlermusteranalyse prüft den Quellcode gegen typische Fehlermuster

Softwaremetriken

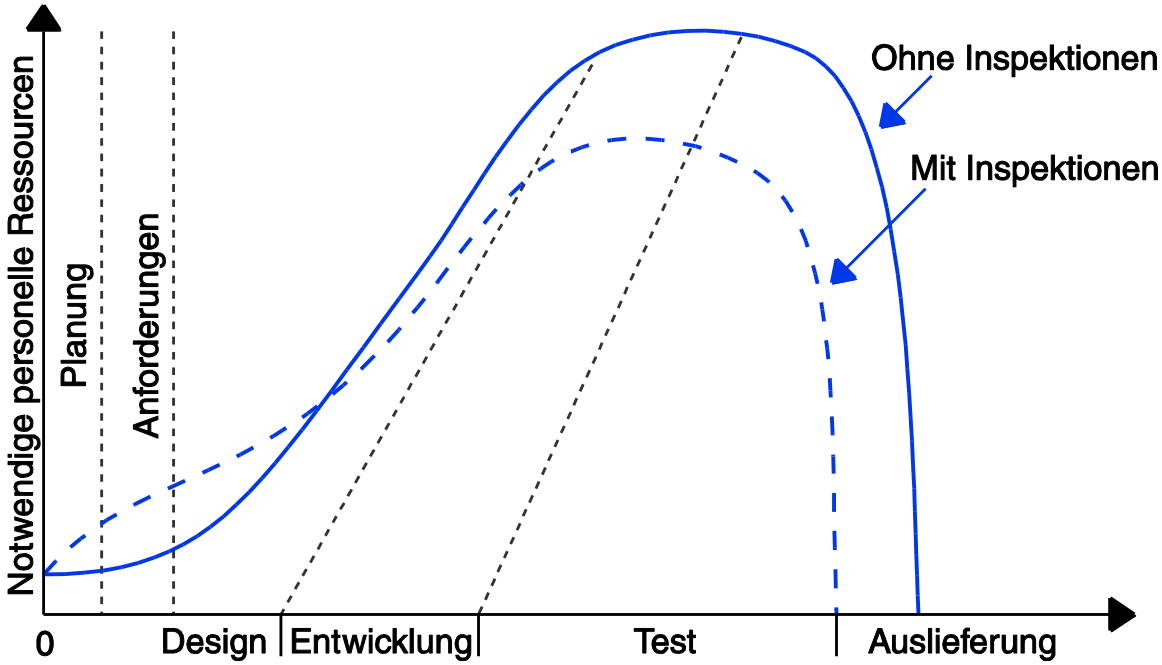
- **Ermöglichen, die Qualität von Software zu interpretieren**
- **Bei Prozessmetriken handelt es sich um quantitative Daten des Softwareentwicklungsprozesses, wie z.B.:**
  - Fehler: wie schnell werden Fehler im Prozess entdeckt?
  - Welche Aufwände verursachen Sie? Mitarbeiteraufwände, Zeitaufwände, Kostenaufwände, …
- **Produktmetriken messen die Software oder Teile davon selbst, wie z.B.:**
  - Umfangsmetrik: Lines of Code (LOC)
  - Entwurfsqualität: Modularität, Kopplung, …
  - Produktqualität: Testergebnisse, Testabdeckung, …

Reviews - Definition

- **Unter dem Begriff Review versteht man allgemein eine formell organisierte Zusammenkunft von Personen zur inhaltlichen oder formellen Überprüfung eines Produktteils (Dokument, Programmcode etc.) nach vorgegebenen Prüfkriterien und -listen (IEEE-Standard 1028, 2008)**
- **Grundsätzlich können alle Entwicklungsergebnisse mit Reviews geprüft werden**
- **Es gibt viele verschiedene Ausprägungen von Reviews, wobei für alle definierte Regeln und Zuständigkeiten existieren**

Vorteile von Reviews

- Reviews lassen sich somit schon sehr früh im Entwicklungsprozess durchführen, noch bevor ausführbare Programme zum Testen vorliegen
- Dadurch kann die Latenzzeit eines Fehlers in der Entwicklung wesentlich verkürzt werden
- Die Kosten für die Behebung des Fehlers sind somit geringer und kompensieren den Review-Aufwand



Rollen im Review-Prozess

Manager

- Ist jener Projektleiter, der den Auftrag zur Erstellung des Testobjektes gegeben hat und sich für die Freigabe verantwortlich zeichnet.

***Moderator* oder Review-Leiter**

- Er plant, organisiert und leitet das Review. Er sucht die Teilnehmer aus, stellt die benötigten Unterlagen des Softwareelements bereit und organisiert die Review-Sitzung.

Schreiber

- Er notiert alle während der Sitzung anfallenden Erkenntnisse des Review-Teams in einem Protokoll.

Rollen im Review-Prozess

Autor

- Ist der Urheber des Testobjektes oder Repräsentant des Teams, das das Testobjekt erstellt hat

***Gutachter* oder Reviewer**

- Sind jene Review-Mitglieder, welche die Aufgabe haben, das zu prüfende Material in der Vorbereitung zu begutachten und in der Sitzung über ihre Erfahrungen zu berichten

Leser

- Ist für die Aufbereitung der zu prüfenden Softwareelemente und für die zeitlich angemessene Durchführung der Sitzung verantwortlich. Er geht das zu überprüfende Material durch, soll aber explizit nicht der Autor sein

Typen von Review-Verfahren

Stellungnahme

- Der Autor stellt einem oder mehreren Kollegen Kopien zum Lesen zur Verfügung.
- Diese begutachten die Unterlagen und retournieren die Papiere mehr oder weniger ausführlich kommentiert an den Autor.
- Anschließend analysiert er die Stellungnahmen und korrigiert, falls er es für notwendig erachtet, seine Arbeit

Walkthrough

- Weniger formelle Form des Review
- Regen Diskussionen bzw. zu einer hohen Interaktion zwischen dem Vortragenden und den Teilnehmern an

Technisches Review

- Beurteilung eines Softwareprodukts durch ein qualifiziertes Team hinsichtlich der beabsichtigten Verwendung und die Identifikation von Abweichungen von der Spezifikation oder von verwendeten Standards
- Starke Anlehnung an die Inspektion

Typen von Review-Verfahren

Inspektion

- Formellste und wirkungsvollste Review-Verfahren
- Zweck einer Inspektion ist das Finden und die Identifikation von Anomalien im Source-Code.
- Es soll ein Softwareelement auf die Erfüllung seiner Spezifikation und die Einhaltung von Richtlinien und Standards untersucht werden

Round-Robin-Review

- Versuch das Prinzip eines herkömmlichen Review auf den Kopf zu stellen
- Gutachter erhalten abwechselnd die Möglichkeit, die Führungsrolle zu übernehmen (verschiedene Testobjekte)

Peer-Review

- Der wesentlichste Unterschied zur Inspektion ist der, dass der Prozess nicht definiert ist und keine speziellen Techniken zur Fehlersuche eingesetzt werden
- Das Team, ad-hoc zusammengestellt oder als permanente Einrichtung, bestimmt selbst die Aufgabenaufteilung und Vorgehensweise

Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Dynamische Qualitätssicherung

- **Prüfen und Überwachen des Systemverhaltens gegenüber dem spezifizierten Verhalten während der Laufzeit**
- **Dazu wird das System mit Testdaten ausgeführt und beobachtet**
  - Testen
  - Dynamische Analyse:
    - Untersuchung der Interaktion von verschiedenen Komponenten
    - Das zu analysierende Programm mit Hilfe von speziellen Werkzeugen (Debugger) ausgeführt
    - Vorgang des Nachverfolgens von Information (*Programm-Trace)*
    - Informationen darüber, an welcher Stelle im Programm Unregelmäßigkeiten aufgetreten sind

Inhalt

I Vorgehensmodelle

1 Traditionelle Modelle

2 Agile Modelle

3 Traditionell oder agil: Entscheidungsfaktoren

II Qualitätssicherung und Qualitätsmanagement

1 Statische Qualitätssicherung

2 Dynamische Qualitätssicherung

3 Organisatorische Qualitätssicherung

Organisatorische Qualitätssicherung

Ziel: Bereitstellung von Infrastrukturkomponenten zur Vermeidung von Softwarefehlern oder zumindest zur Verringerung der Fehlerrate

Wissensmanagement

- Wissensmanagement kann als Prozess der Weiterentwicklung von Ideen des organisationalen Lernens verstanden werden
- Der Fokus liegt auf der Verbesserung einer Organisation auf allen Ebenen durch den gezielten Umgang mit der Ressource Wissen, insbesondere im Bereich des Qualitätsmanagements
- Dieses Wissen umfasst sämtliche Bestandteile, über die eine Organisation zur Lösung ihrer Aufgaben verfügt, wie z.B:
  - Fähigkeiten
  - Erfahrungen
  - Fertigkeiten
  - Normen
  - Routinen

Organisatorische Qualitätssicherung

Konfigurationsmanagement

- Erstreckt sich über alle Softwareentwicklungsaktivitäten im Laufe des Softwarelebenszyklus
- Verwaltet sämtliche Objekte wie z.B. Spezifikationen, Dokumentationen, Änderungen von Anforderungen oder Sourcecode
- Ein Konfigurationsmanagement-System muss auch in der Lage sein, Versionslinien, Änderungsstände und Freigaben der „Configuration Items“ zu steuern
- Ablauf eines typischen Konfigurationsmanagement-Prozesses:
  - Objekt wird eingereicht.
  - Objekt wird identifiziert, versioniert und archiviert.
  - Objekt wird zur Prüfung freigegeben (Test, Review, …).
  - Objekt wird freigegeben.
  - Änderung wird eingereicht.
  - Änderung wird veranlasst

Organisatorische Qualitätssicherung

Templates

- Templates helfen, das Format, den Aufbau und den Inhalt von Dokumenten in Bezug auf Vollständigkeit und Qualität zu sichern
- Durch Vorgabe der Struktur und des Aufbaus von Dokumenten kann wertvolle Zeit für die Vorbereitung gespart werden.
- Außerdem können Reviews effizienter und effektiver durchgeführt werden, wenn die Struktur der Dokumente bekannt ist
- Beispiele: Testpläne, Testfallbeschreibungen, Testberichte, Review-Berichte

Checklisten

- Enthalten in Frageform gekleidete Richtlinien und Hypothesen von vermuteten Schwachstellen im zu prüfenden Produkt
- Diese Fragen sind so formuliert, dass sie mit „ja“ oder „nein“ beantwortet werden müssen

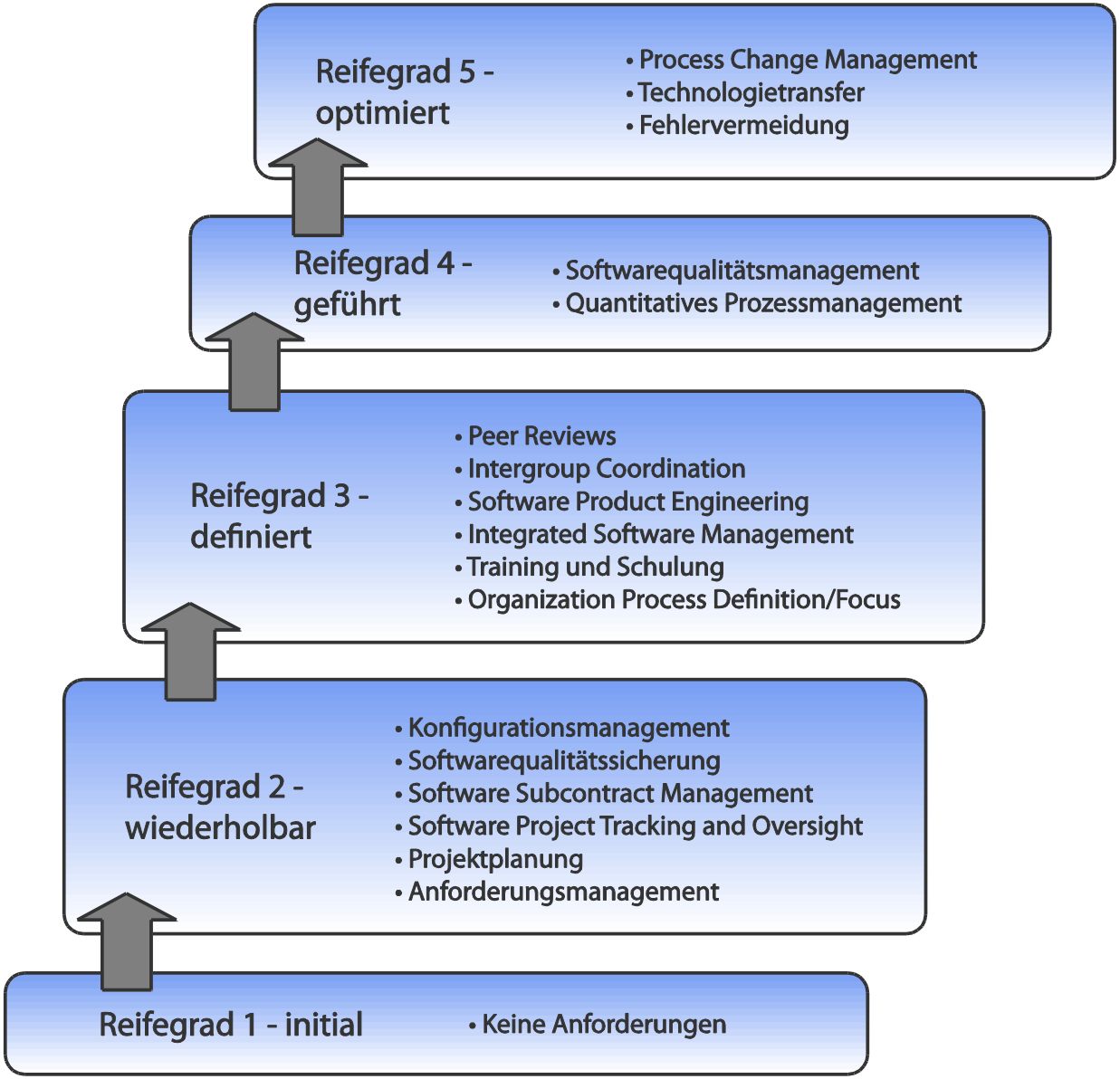
Qualitätsmanagementstandards

- Qualitätsmanagement beinhaltet aufeinander abgestimmte **Tätigkeiten** zum **Leiten** und **Lenken** einer **Organisation** bezüglich Qualität
- Sollen die Entwicklung eines Unternehmens hin zu einem **hohen Qualitätslevel** und zu **Kontinuität** unterstützen
- Standards verbessern zudem das **wechselseitige Verständnis** und die **Koordination** unterschiedlicher Entwicklungsteams, speziell aber auch zwischen Entwicklung und Wartung
- Durch eine höhere **Transparenz** und **Berechenbarkeit** eines Qualitätssicherungssystems kann der Kunde von einem gewissen Qualitätsniveau ausgehen und bleibt so eher von unerwünschten Überraschungen verschont als bei intuitiver Vorgehensweise
- Grundsätzlich können Qualitätsmanagementstandards in **Zertifizierungs-** und **Assessmentstandards** (*Bewertungsstandards*) unterschieden werden

CMM and CMMI

- CMM-Modell wurde in den 80er Jahren, auf Initiative des **US-Verteidigungsministerium**, vom SEI (*Software Engineering Institute*) entwickelt
- Ziel, die **Vergleichbarkeit** und **Qualität** von **Softwareprozessen** im Rahmen der Vergabe an externe Lieferanten zu verbessern
- Mit der Zeit jedoch verbreitete sich das Modell. 1993 wurden für diverse Entwicklungsdisziplinen weitere Modelle entwickelt, wie z.B. SE-CMM
- 2003 ein vereinheitlichter und modularer Nachfolger, das **CMMI** (*Capability Maturity Model Integration*), entwickelt
- CMMI Liefert eine **Analyse** des **Ist- Zustandes eines Unternehmens** und ist kein international anerkanntes Zertifikat wie ISO 9000-3
- Ein CMM-Assessment beurteilt die **Fähigkeit** einer Organisation, Software unter bestimmten Rahmenbedingungen erstellen zu können

CMMI Reifegradstufen



Zusammenfassung

- **Organisatorische Maßnahmen** schaffen die notwendigen **Rahmenbedingungen** und sorgen dafür, dass **analytische Maßnahmen** stattfinden können

- **Statische analytische** Maßnahmen (z.B. Reviews) und **dynamische analytische Maßnahmen** (z.B. Tests) hingegen versuchen, Fehler und Mängel zu erkennen und zu lokalisieren

- Da es nicht möglich ist, fehlerfreie Software herzustellen, muss das **Verhältnis** zwischen **Fehlerkosten**, **Fehlervermeidungskosten** und **Qualität** optimiert werden.

Zusammenfassung

- **Kostenoptimale Qualität** wird erreicht, wenn die Kosten für die Vermeidung und Prüfung von Fehler genauso hoch sind wie die Kosten für deren Behebung. Das ist auch jener Punkt, an dem die **Gesamtkosten** am geringsten sind.

- Weitere Vermeidungs- und Prüfmaßnahmen sind **nicht mehr wirtschaftlich**, da mit steigendem Aufwand die Fehlerfindungsrate deutlich sinkt. Es wird also immer schwieriger und aufwändiger, Fehler zu finden.

- **Capability Maturity Modell Integration (CMMI):** Dieses Modell liefert eine Analyse des Ist- Zustandes eines Unternehmens. Ein CMM-Assessment beurteilt die Fähigkeit einer Organisation, Software unter bestimmten Rahmenbedingungen erstellen zu können.