Software Engineering und Projektmanagement

Qualitätssicherung und Qualitätsmanagement

„It‘s more about good enough than it is about right or wrong“ – James Bach

2015W

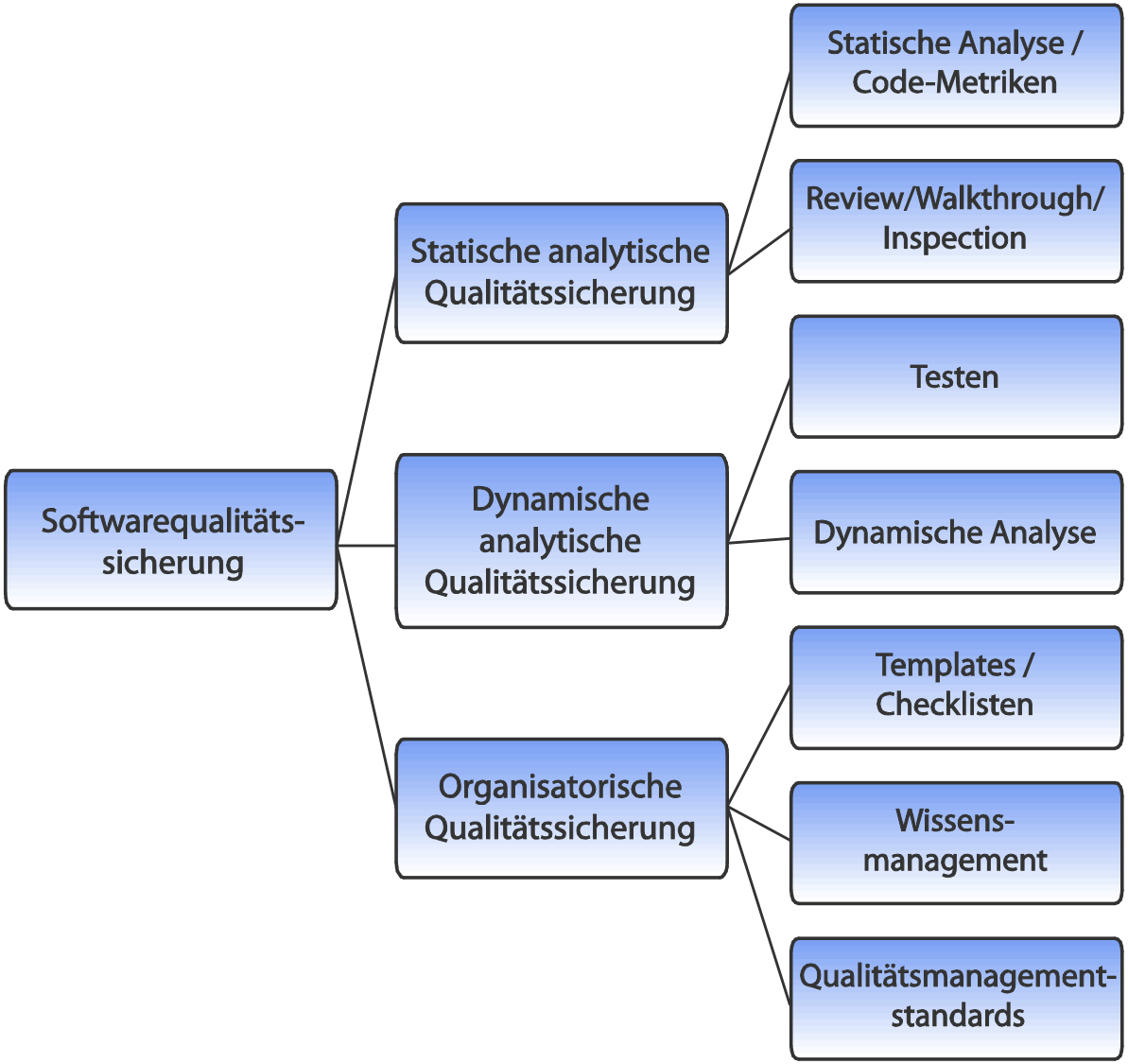
Univ. Lektor Dipl.-Ing. Mag. Roland Breiteneder

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

Lernziele

- **Motivation und Ziele** der Softwarequalitätssicherung: Was sind die Ziele der Softwarequalitätssicherung und welche Motivation steckt dahinter?
- **Qualitätskostenmodelle** und **Kostenoptimierung**: Was ist ein Qualitätskostenmodell und welche Arten von Kosten können grundsätzlich unterschieden werden? Wie kann eine Optimierung der Qualitätssicherungskosten erzielt werden?
- **Organisatorische Qualitätssicherung**: Welche organisatorischen Maßnahmen werden im Rahmen eines umfangreichen Qualitätsmanagements getroffen?
- **Statische und dynamische Qualitätssicherung**: Was sind statische und dynamische Qualitätssicherungsmaßnahmen und wie werden sie eingesetzt?
- **Qualitätsmanagementstandards**: Welche anerkannten Standards gibt es und welche Prinzipien und Konzepte stecken dahinter?

Software-Qualitätssicherungsmethoden



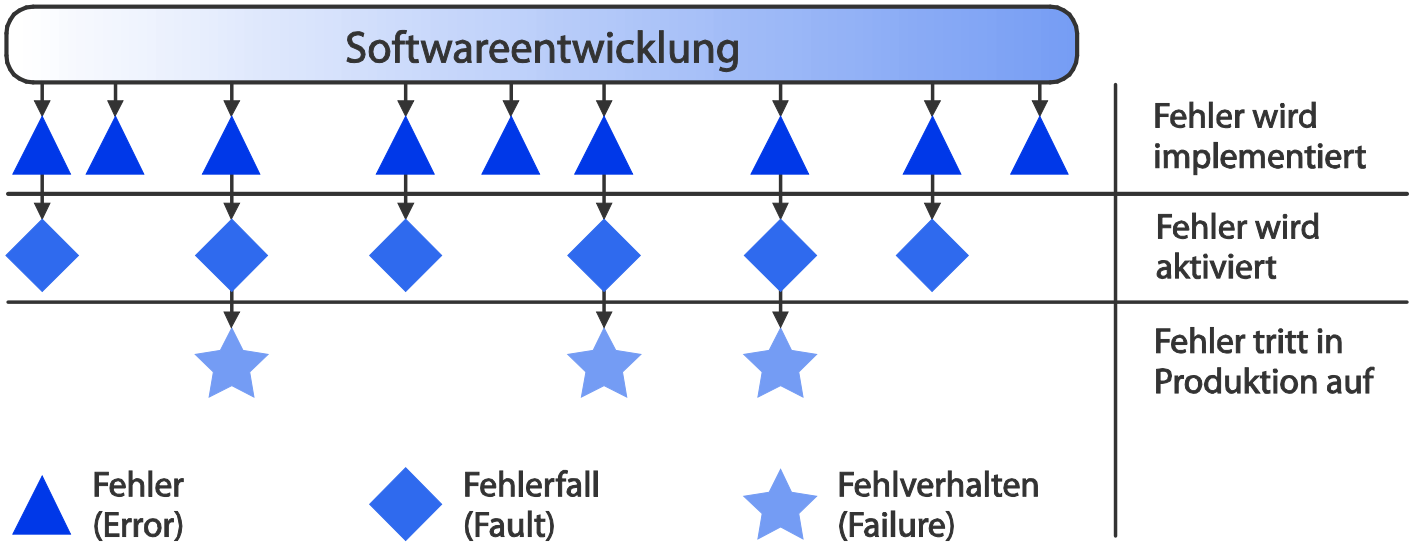
Softwarequalitätsfaktoren

- **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

Softwarefehler, -fehlerfall und -fehlverhalten

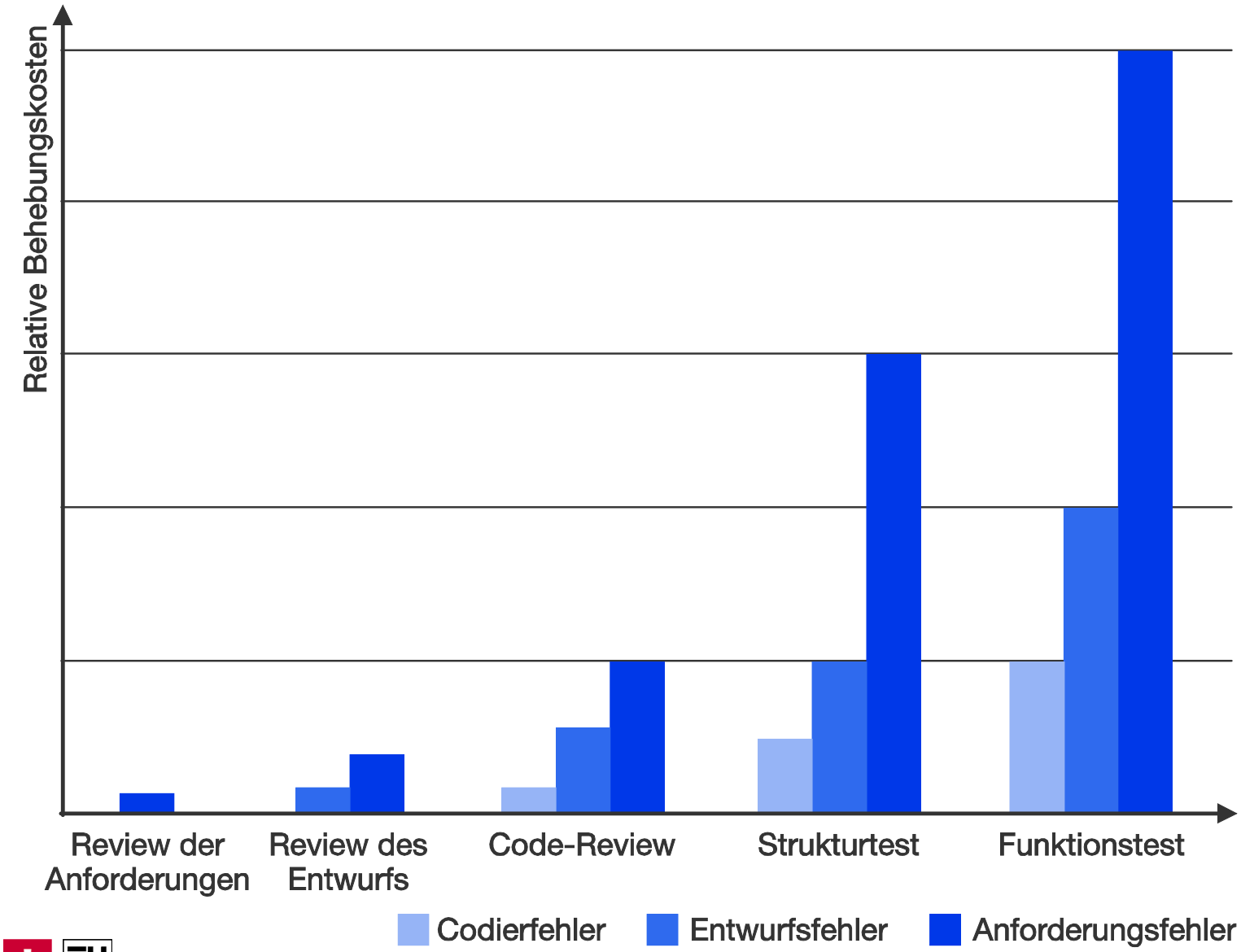
- Ursprung von Softwarefehlverhalten liegt in Softwarefehlern, verursacht von Softwareentwicklern während der Programmierung
- Ein Fehlerfall muss ausgelöst werden, um zu einem Fehlverhalten zu führen
- Häufig werden Fehlerfälle gar nicht oder nur sehr selten ausgelöst



Ursachen von Softwarefehlern

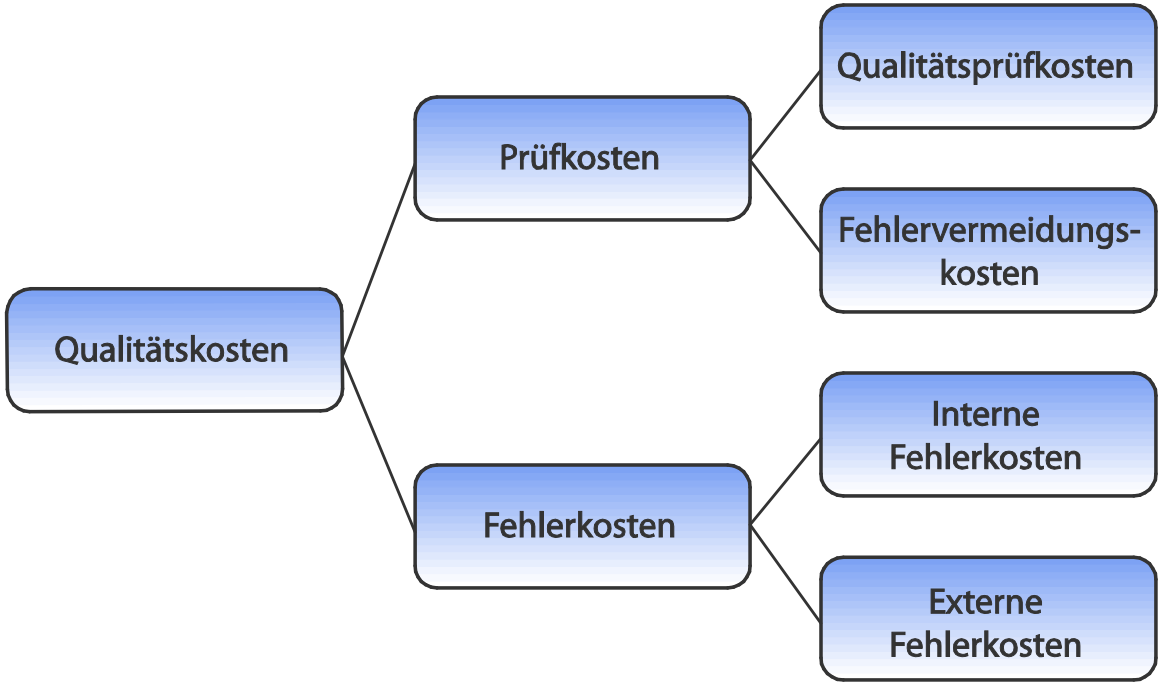
- Fehlerhafte Definitionen in den Anforderungsdokumenten
- Falsche Interpretation von Kundenanforderungen
- Logische Designfehler aufgrund fehlerhafter Anforderungen von Designexperten, Softwareentwickler und Analysten
- Fehlinterpretationen des Designdokuments
- Sprachliche Fehler in der Programmiersprache
- Fehler bei der Test-Datenauswahl
- Nichteinhaltung von standardisierten Codierungs- und Dokumentationsrichtlinien

Fehlerentstehung und Fehlerentdeckung

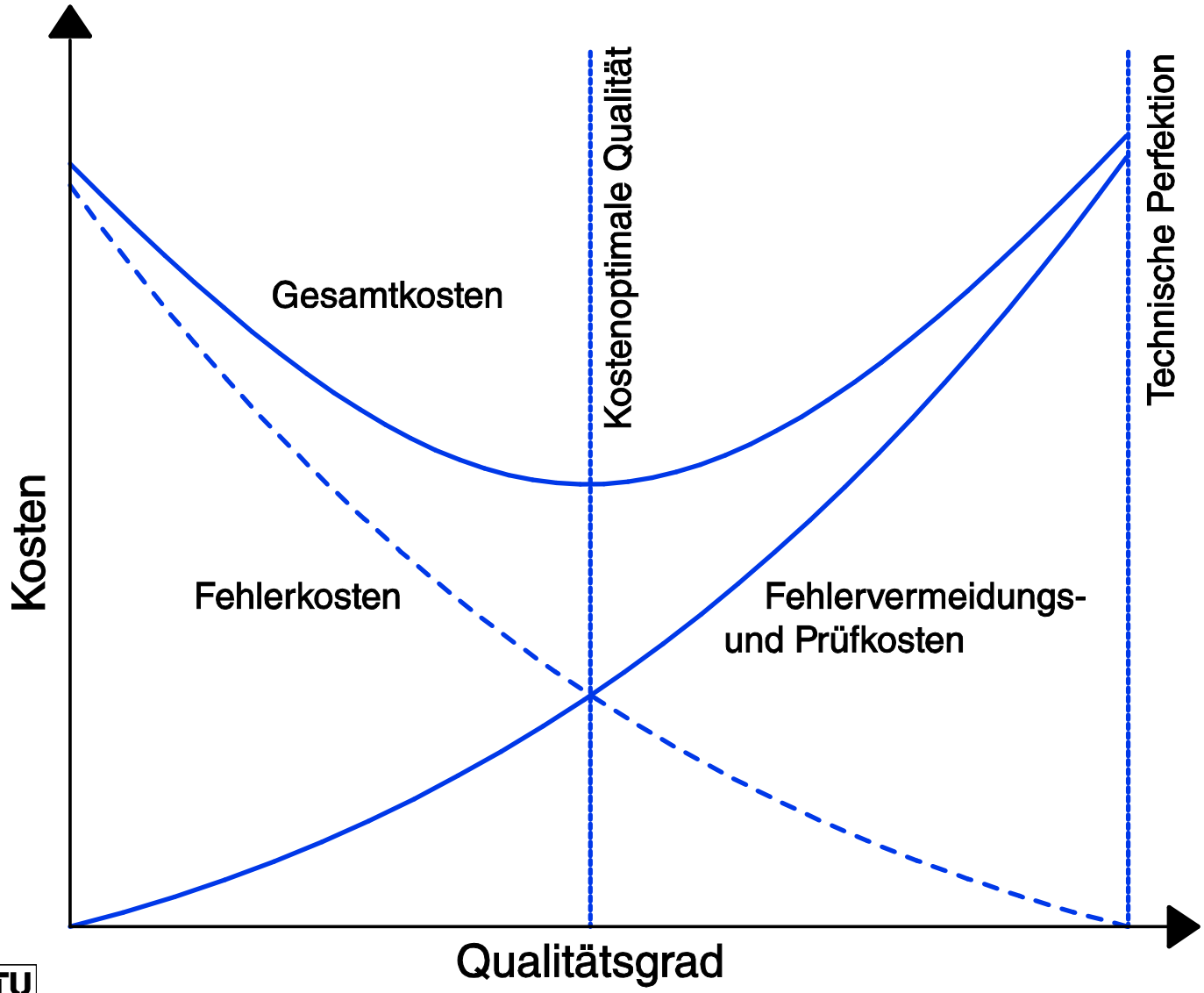


Kosten der Qualitätssicherung

- 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



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**
- **Produktmetriken messen die Software oder Teile davon, ohne dabei zu berücksichtigen, wie das Produkt entstanden ist oder warum es sich gerade in einem bestimmten Zustand befindet**

  - Umfangsmetrik
    - LOC Lines of Code

  - McCabe-Metrik
    - *zyklomatische Zahl V(g)*

Softwaremetriken

Mit dem Ansatz der objektorientierten Programmierung wurden neue Metriken notwendig, welche auch die Beziehungen von Objekten untereinander und deren Struktur berücksichtigten

Objektorientierte Metriken:

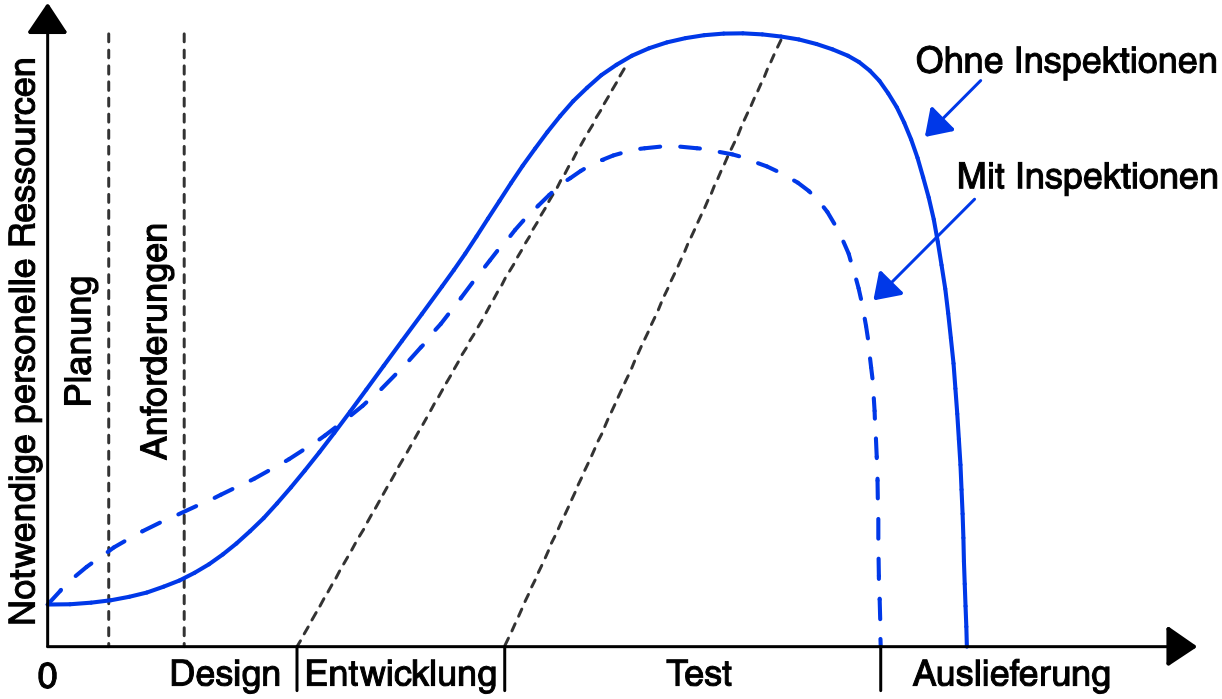
- CBO (coupling between objects)
- DIT (depth of inheritance tree)
- NOC (number of children)
- RFC (response for a class)
- WMC (weighted methods per class)
- LCOM (lack of cohesion in methods)

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**

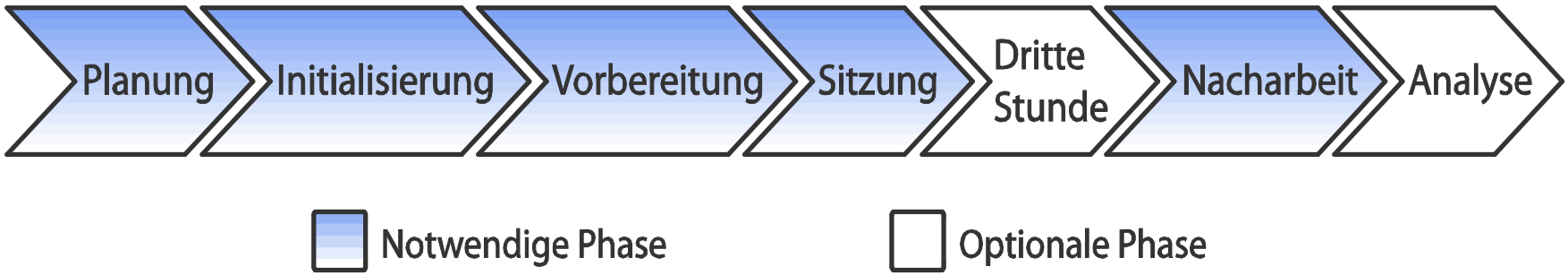
- Sind jene Review-Mitglieder, welche die Aufgabe haben, das zu prüfende Material in der Vorbereitung zu begutachten und in der Sitzung über ihre Erfahrungen zu berichten

Leser

- Ist für die Aufbereitung der zu prüfenden Softwareelemente und für die zeitlich angemessene Durchführung der Sitzung verantwortlich

Review-Prozess

- Der Review-Prozess beginnt mit der *Planungsphase*.
- Das Review selbst wird erst durch einen *Initialisierungsvorgang* gestartet.
- Den Abschluss eines Review bildet die Freigabe.
- Ein Review besteht aus den Schritten *Vorbereitung, Sitzung* und *Nacharbeit*.
- Im Anschluss an das durchgeführte Review wird versucht, anhand von Metriken und Analysen Aussagen über die Effektivität des Review zu treffen



Review-Messung

Anhand der gesammelten Daten soll es möglich sein festzustellen, ob die Ergebnisse im erwarteten Rahmen liegen. Barnard et al. (1994) beschreibt folgende neun *Schlüsselmetriken*:

- Total noncoment lines for source code inspected, in thousand (Gesamtanzahl an inspizierten Codezeilen ohne Kommentare in Tausend (KLOC))
- Average lines of code inspected (Durchschnittlich inspizierte Codezeilen)
- Average preparation rate (Durchschnittliche Vorbereitungsrate)
- Average inspection rate (Durchschnittliche Inspektionsrate)
- Average effort per KLOC (Durchschnittlicher Aufwand pro KLOC)
- Average effort per fault detected (Durchschnittlicher Aufwand für einen gefundenen Fehler)
- Average faults detected per KLOC (Durchschnittlich gefundene Fehler pro KLOC)
- Percentage of reinspection (Prozent der Re-Inspektionen)
- Defect-removal efficieny (Fehlerentfernungseffizienz)

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

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 die Aufgabe, in der Vorbereitung nach positiven Argumenten des zu prüfenden Softwareelementes zu suchen

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

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

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

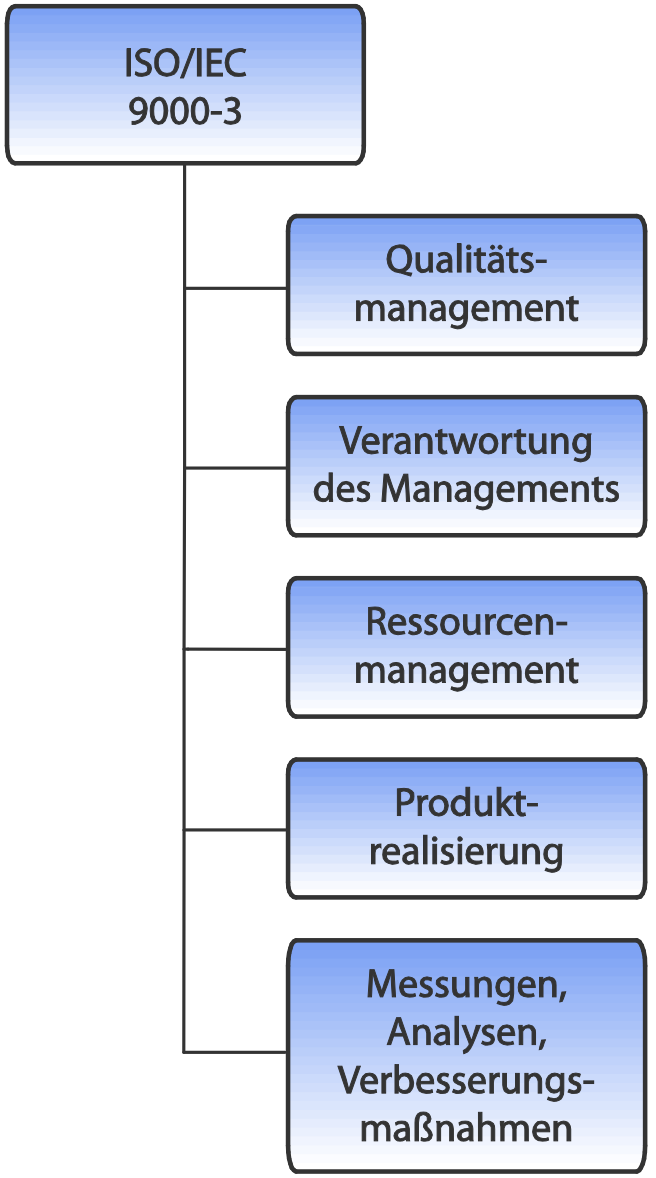
- Qualitätsmanagement beinhaltet aufeinander abgestimmte **Tätigkeiten** zum **Leiten** und **Lenken** einer **Organisation** bezüglich Qualität
- Sollen die Entwicklung eines Unternehmens hin zu einem **hohen Qualitätslevel** und zu **Kontinuität** unterstützen
- Standards verbessern zudem das **wechselseitige 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

ISO 9001 und ISO 9000-3

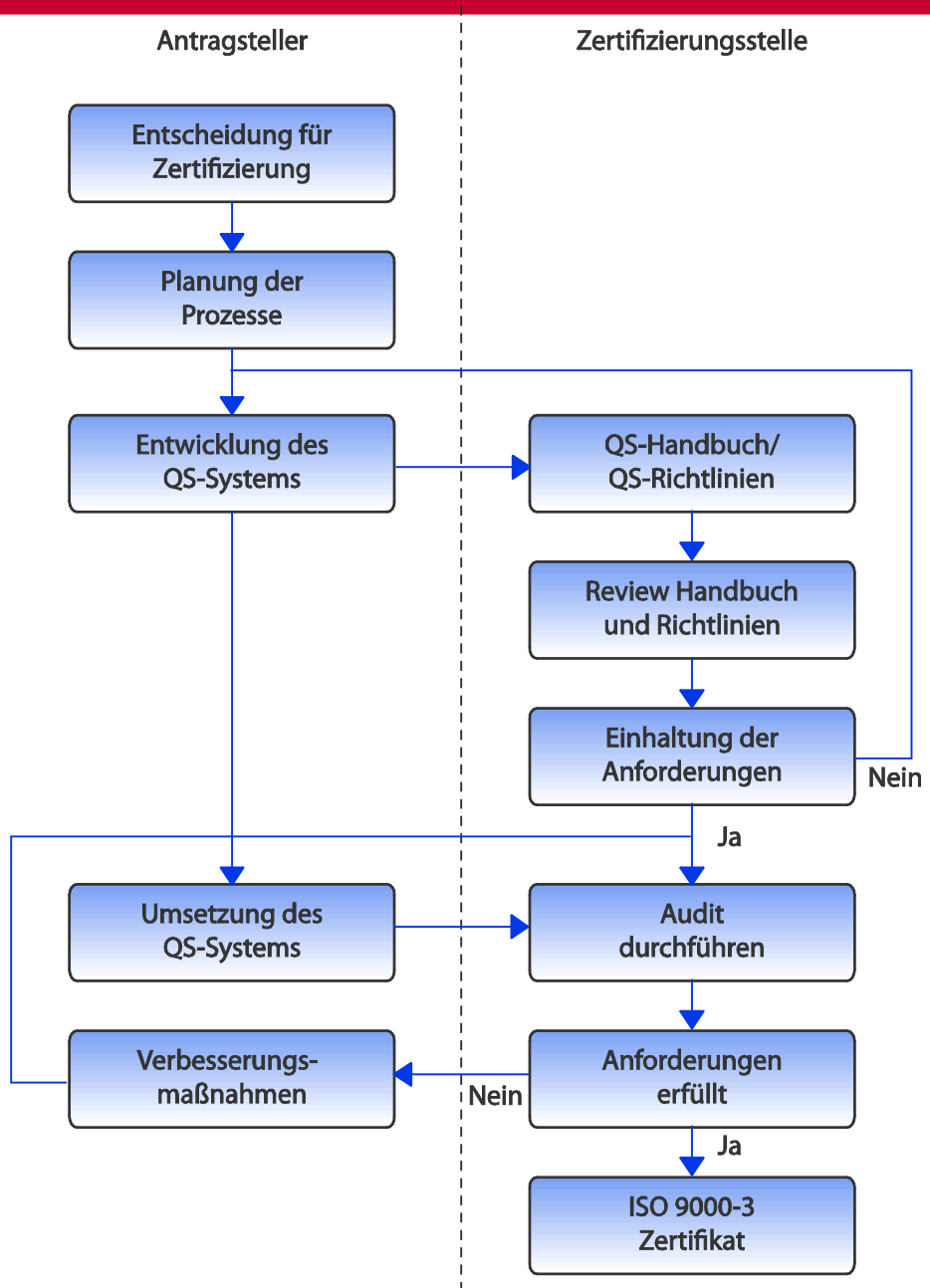
- **Primär steht hier der Prozessgedanke im Vordergrund**
- **Jeder Prozess orientiert sich an der Erfüllung der Bedürfnisse des Kunden**
- **Im Vergleich zu anderen Qualitätsmanagementsystemen steht ein Zertifikat im Vordergrund, das den Nachweis eines Unternehmens liefert, normenkonform zu sein**
- **Der ISO 9000-3-Standard formuliert acht Prinzipien**
  - *Kundenorientierung*
  - *Prozessorientierung*
  - *Mitarbeiterführung*
  - *Zuständigkeit der Mitarbeiter*
  - *Kunden-Lieferanten-Verhältnis*
  - *Kontinuierliche Verbesserung*
  - *Betrachtung von Prozessen als System*
  - *Sachliche Herangehensweise bei der Entscheidungsfindung*

Der ISO 9000-3-Standard beschreibt 22 zu erfüllende Anforderungen, die in folgende Gruppen aufgeteilt sind

- Qualitätsmanagementsystem
- Verantwortung des Managements
- Ressourcenmanagement
- Produktrealisierung
- Messungen, Analysen und Verbesserungsmaßnahmen



ISO 9001 und ISO 9000-3 Zertifizierungsprozess



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

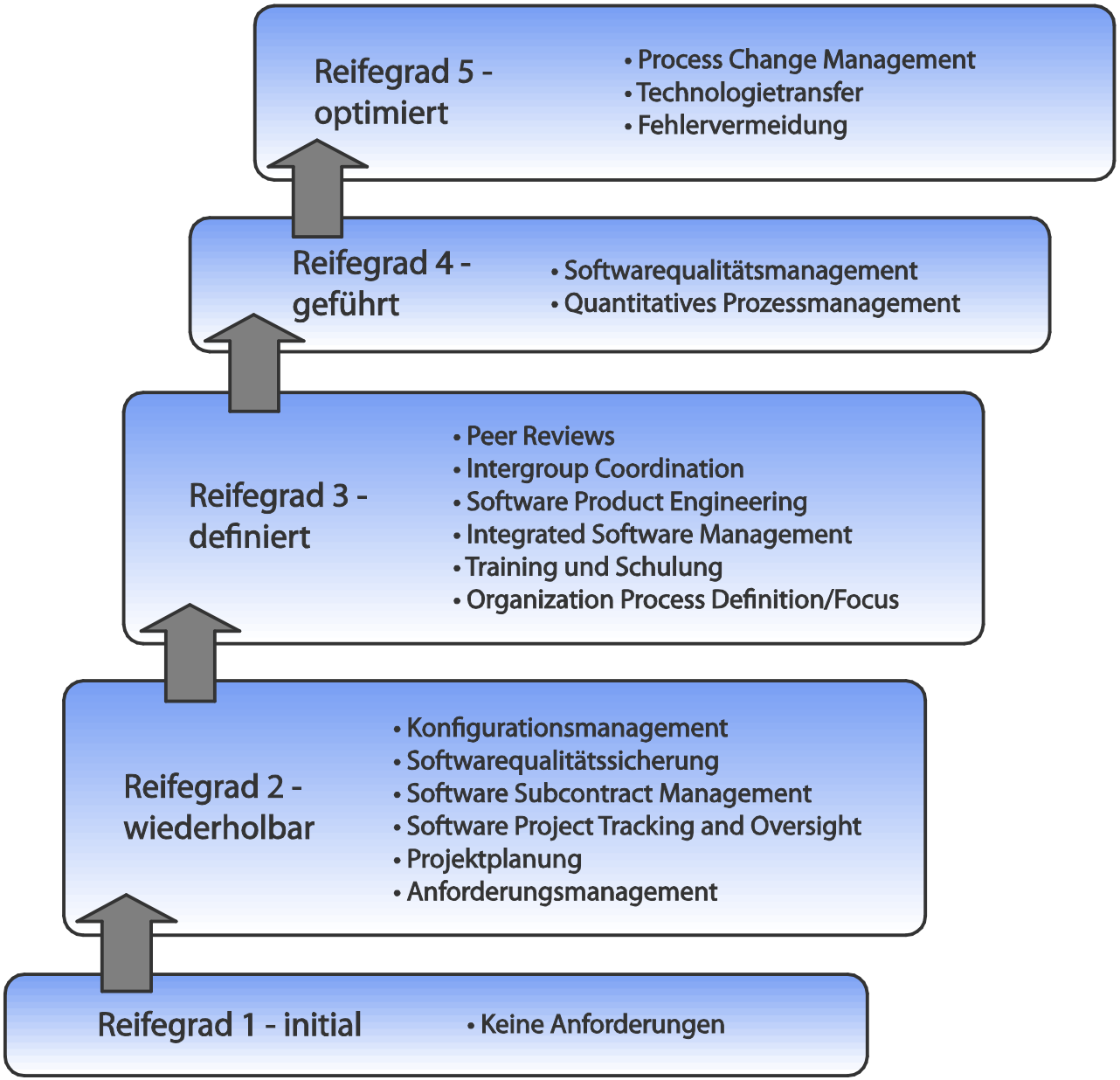
CMMI

Prinzipien und Konzepte

- **Quantitative Managementmethoden** erhöhen die Fähigkeit eines Unternehmens, die Qualität zu kontrollieren und die Produktivität zu steigern

- Die Anwendung des **Fünf-Ebenen**-Capability Maturity Model (Reifegrade) ermöglicht die Bewertung der Leistungen / Prozesse und bestimmt die **Aufwände**, die zur Erreichung der nächsten Stufe notwendig sind

CMMI Reifegradstufen



SPICE – ISO/IEC 15504

- **SPICE** (*Software Process Improvement for Capability Determination*) wurde Anfang der 90er Jahre als gemeinsame Initiative von ISO und IEC (International Electrotechnical Commission) entwickelt
- Grundsätzlich sind SPICE und CMMI sehr ähnlich aufgebaut. Zielsetzungen und Forderungen dieser beiden Modelle überdecken sich zu ca. 80 – 90%.
- Somit kann SPICE als **Konkurrent** zum US-dominanten CMMI gesehen werden
- SPICE besteht aus einem **zweidimensionalen Referenzmodell**.
- Zu diesem Referenzmodell gehören eine **Prozessdimension** und eine **Fähigkeitsdimension** zur Reifegradbestimmung der einzelnen Prozesse

SPICE – ISO/IEC 15504

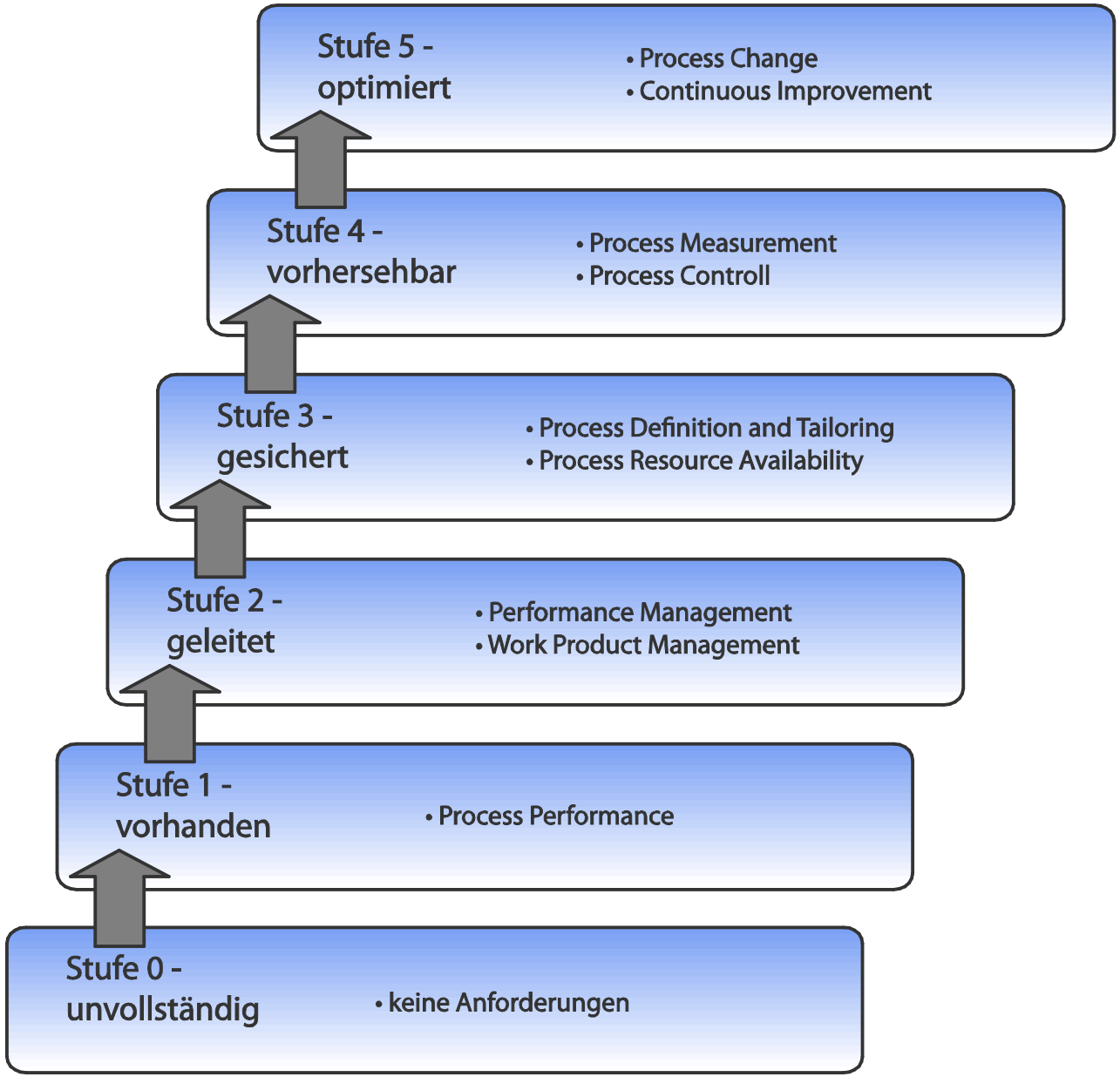
Prinzipien und Ziele

- **Zusammenführung** verschiedenster existierender **Assessment-Methoden** zu einem zusammenhängenden Systemmodell
- **Universelle Einsetzbarkeit**, um alle Kategorien von Softwareprojekten sowie deren Kunden und Zulieferer zu unterstützen
- **Hohe Professionalität**
- Erreichung hoher **internationaler Akzeptanz** als **weltweiter Standard**

Die einzelnen Elemente von SPICE verlangen eine strukturierte Vorgehensweise mit folgenden Zielsetzungen:

- Das Verstehen des **Entwicklungsstandes** der **eigenen Prozesse**, um Verbesserungen vornehmen zu können
- Die Eignung der eigenen Prozesse in Bezug auf **Anforderungen** oder Klassen von Anforderungen zu bestimmen
- Die Eignung der Prozesse anderer Organisationen und Organisationseinheiten in Bezug auf **Verträge** oder Klassen von Verträgen zu bestimmen

SPICE – ISO/IEC 15504
Reifegradstufen



SPICE – ISO/IEC 15504

Erfüllungsgrade

- nicht erreicht (not achieved) 0 – 15 %
- teilweise erreicht (partially achieved) 16 – 50 %
- weitestgehend erreicht (largely achieved) 51 – 85 %
- vollständig erreicht (fully achieved) 86 – 100 %

Prozesskategorien

- Engineering (ENG)
- Customer-Supplier (CUS)
- Management (MAN)
- Support (SUP)
- Organisation (ORG)

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.
- **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.

Zusammenfassung

- **ISO 9000-3**: Primär steht hier der Prozessgedanke im Vordergrund. Jeder Prozess orientiert sich an der Erfüllung der **Bedürfnisse des Kunden**. Im Vergleich zu anderen Qualitätsmanagementsystemen steht ein **Zertifikat** im Vordergrund, das den Nachweis eines Unternehmens liefert, normenkonform zu sein.
- **Capability Maturity Modell Integration (CMMI):** Dieses Modell 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.
- **SPICE (Software Process Improvement and Capability Determination):** Dieses Modell ist dem CMM-Standard sehr ähnlich. SPICE stellt auch ein spezifisches Framework für Assessments von Softwareprozessen für alle Phasen und Bereiche der Softwareherstellung bereit.