Softwaretechnik C - Softwaremanagement

LE 09: Qualitätsmanagement

Fachhochschule Dortmund
University of Applied Sciences

Organisatorisches

- Vorlesungsfrei 25.12.2023 – 05.01.2024
- Vorlesung LE 12 am 09.01.2024 in Präsenz

Organisatorisches

- Abschlussveranstaltung Praktikum
  - Für die Klausur erzielbare **Bonuspunkte**: 15%
  - Geplanter **Zeitraum**: 17.01.2024
  - Inhalt / Aufbau
    - Kurze Darstellung der Produktvision
    - Darstellung des priorisierten Backlogs mit 1 User Story pro Gruppenmitglied
      - + 1 Bsp. für eine User Story zur Risikovermeidung / -minderung
      - + 1 Bsp. Für eine User Story zur konstruktiven Qualitätssicherung
    - Darstellung von 1 Risiko pro Gruppenmitglied (+ Begründung anhand Risikomatrix)
    - Darstellung des Projektplans mit 1 Meilenstein oder Aktivität pro Gruppenmitglied und min. 1 Vorgang für Akzeptanztest

Agenda

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Definition

Qualität [DIN 55350 Teil 11:1995]

- Qualität ist die Gesamtheit von **Eigenschaften** und **Merkmalen** eines Produkts oder einer Tätigkeit, die sich auf deren Eignung zur **Erfüllung** gegebener **Erfordernisse** bezieht.

Qualitätsmanagement [DIN EN ISO 9000:2005]

- Qualitätsmanagement sind aufeinander abgestimmte **Tätigkeiten** zum Leiten und Lenken einer Organisation bezüglich der Qualität, die üblicherweise das Festlegen der **Qualitätspolitik** und der **Qualitätsziele**, die **Qualitätsplanung**, die **Qualitätslenkung**, die **Qualitätssicherung** und die **Qualitätsverbesserung** umfassen.

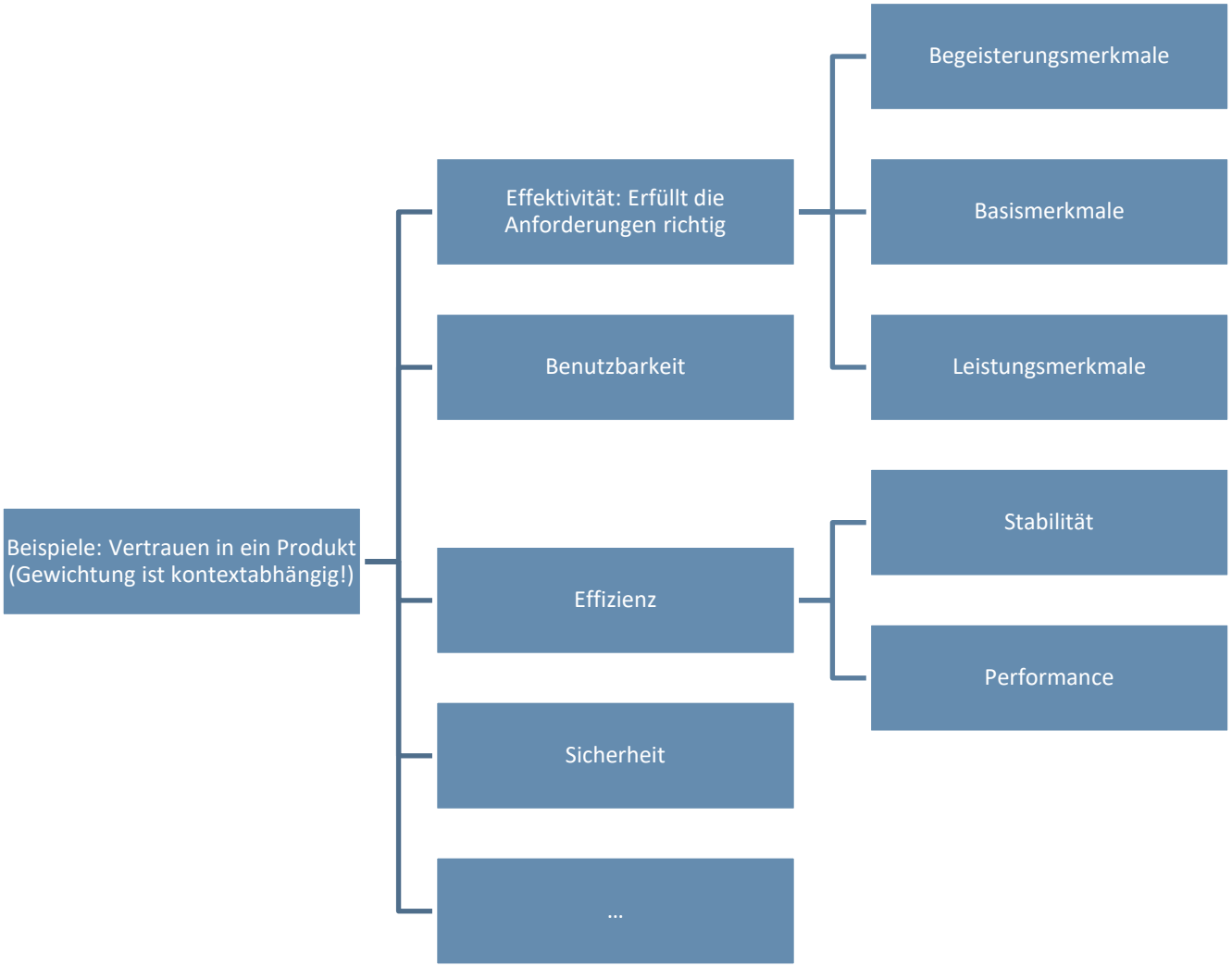
Qualitätsmanagement und Qualitätssicherung

Was bedeutet Vertrauen in ein Produkt? Was ist Grundlage für Vertrauen in die Qualität?

- Zuverlässigkeit
- Produkt wird weiterempfohlen -> mehr Gewinne / größere Nutzendenbasis
- Langlebigkeit durch Vertrauen / Qualität
- Vertrauen durch Demonstration
- Belegbarkeit durch unabhängige Prüfung
- Sicherheit und Bedienbarkeit
- Glaubwürdigkeit
- Performance

Qualitätsmanagement und Qualitätssicherung

Fachhochschule Dortmund
University of Applied Sciences



Tätigkeiten zur Schaffung von Vertrauen in die Qualität eines Produkts

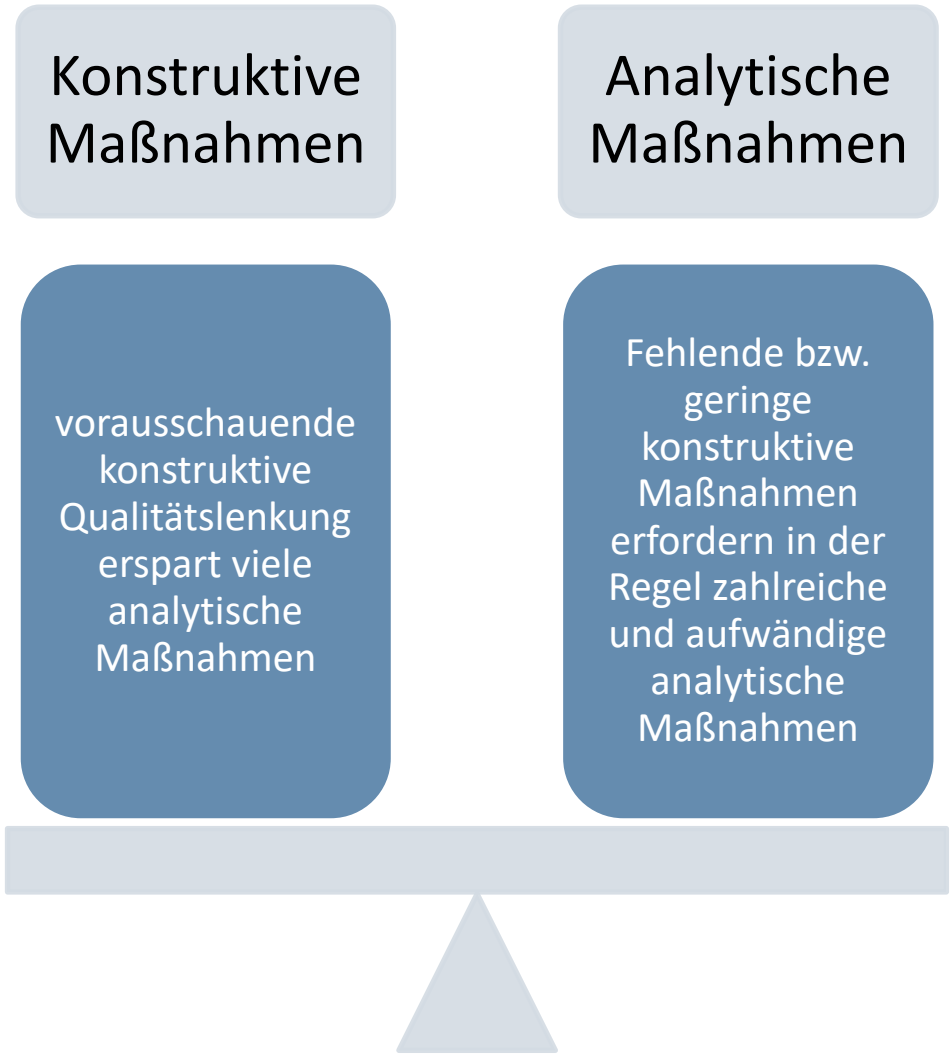
Qualitätsmanagement

Die Unternehmensführung

ist für **Zielsetzungen** und **Maßnahmen** im Hinblick auf die Erreichung und Verbesserung der Qualität verantwortlich.

hat aktiv für konsequente **Umsetzung** auf allen Hierarchieebenen zu sorgen.

Konstruktive und analytische Maßnahmen

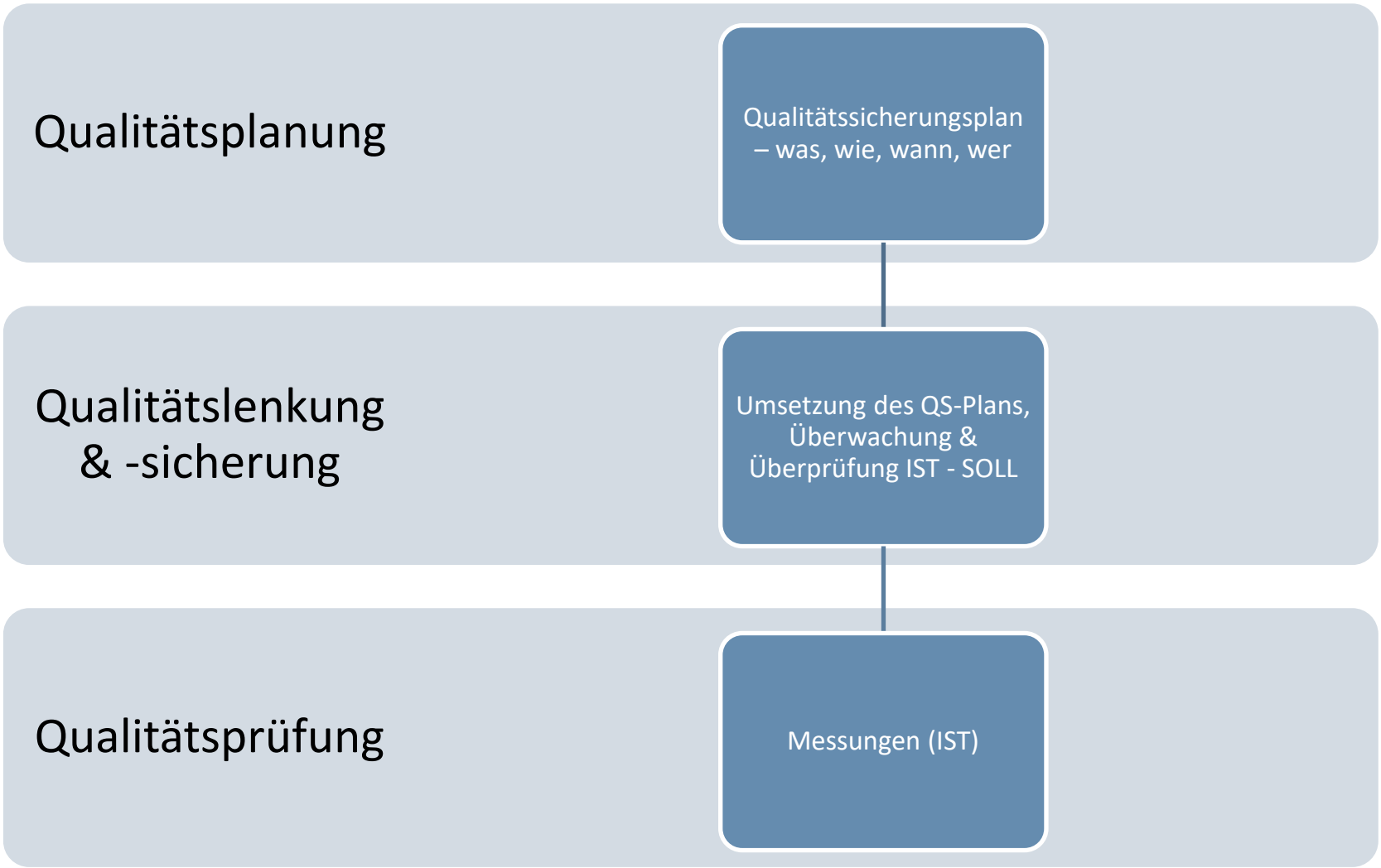


- Sind jeweils voneinander abhängig
- Generelles Ziel: Minimierung des analytischen Aufwands durch effiziente und effektive konstruktive Maßnahmen

Konstruktive und analytische Maßnahmen

- Beispiel: **konstruktive** Maßnahmen
  - Teilprodukte der SW-Entwicklung werden vor dem Eintrag in ein Repository klassifiziert und modularisiert
  - Dadurch wird die Wiederverwendung der Teilprodukte deutlich vereinfacht
- Beispiel: **analytische** Maßnahmen auf Basis konstruktiver Maßnahmen
  - Software-Modularisierung (konstruktive Maßnahme) ist essenzielle Voraussetzung für einen umfassenden Modultest (analytische Maßnahme)

Aktivitäten des Qualitätsmanagements (Einordnung)



1. Qualitätsplanung

Ziel: **Festlegung** von **Qualitätsanforderungen** an das Produkt und den Prozess in **überprüfbarer** Form

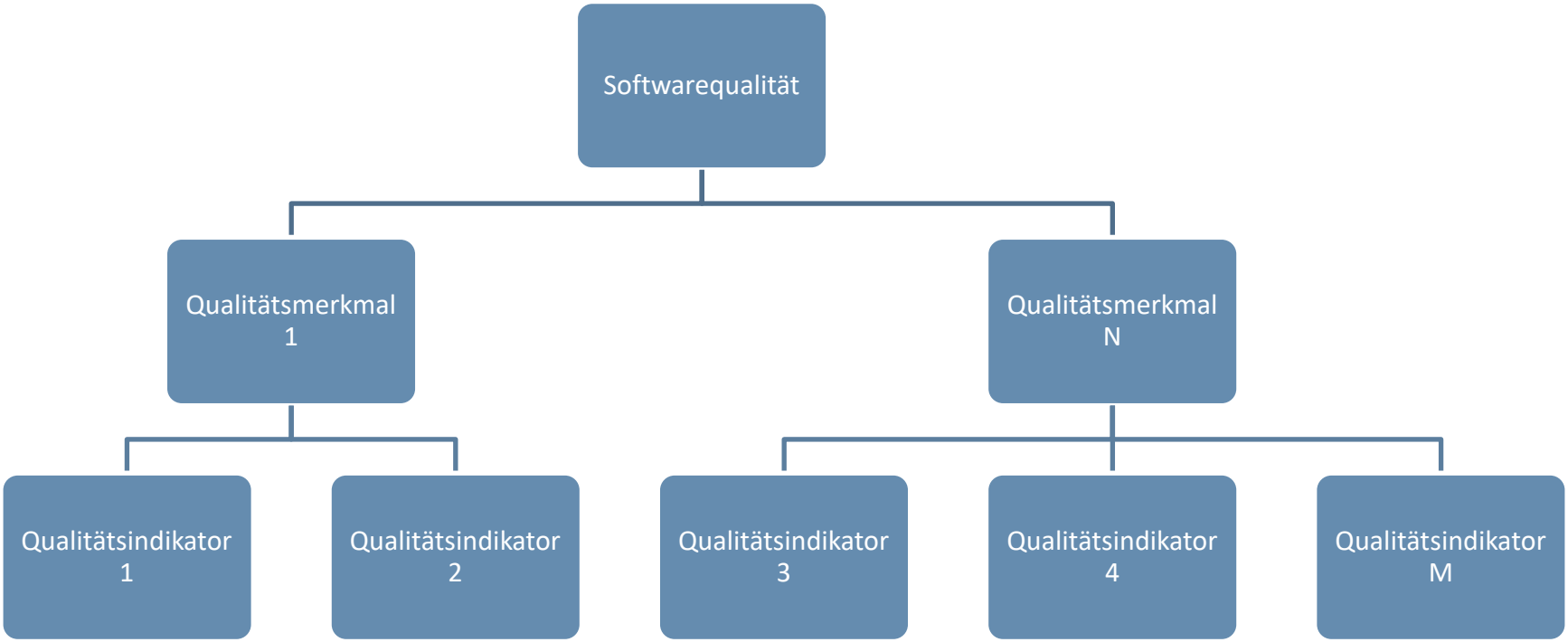
Qualitätsplanung

- Auswahl / Aufstellung eines Qualitätsmodells
- Festlegung von Soll-Qualitätsstufen für Qualitätsindikatoren

Qualitätslenkung / -sicherung

- Festlegung allgemeiner und produktspezifischer Prozesse & Methoden, bspw.
  - Messpunkte im Entwicklungsprozess
  - Methoden und Werkzeuge zur Erfassung der Messwerte

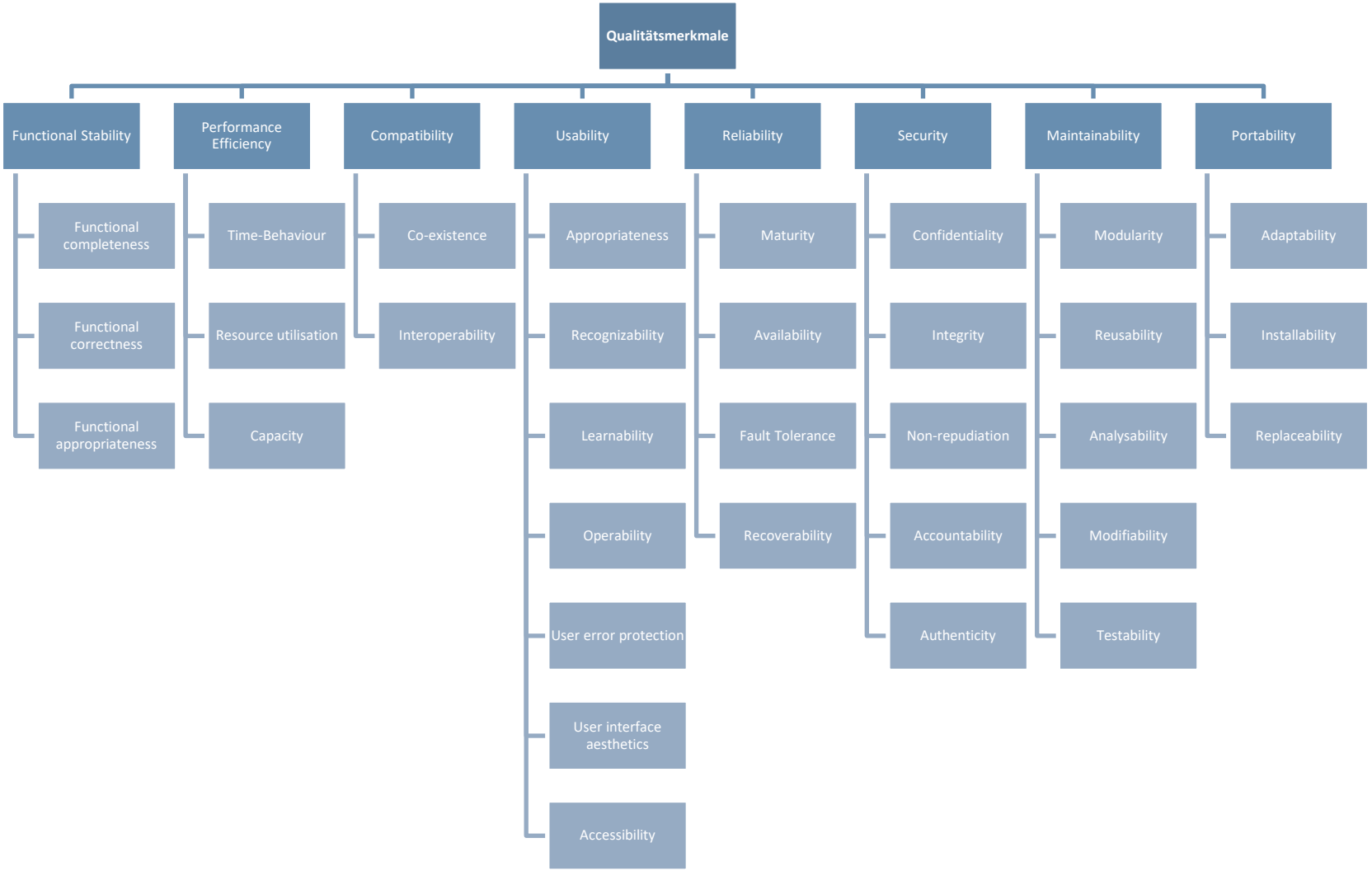
1. Qualitätsplanung



Grundlage für Qualitätsmodell: Festlegung von Qualitätsmerkmalen & insb. Qualitätsindikatoren

- Bsp. Qualitätsmerkmal: Performance
- Bsp. Qualitätsindikator: Zeitverhalten

Qualitätsmerkmale von Software nach ISO 25010



1. Qualitätsplanung

- Das Qualitätsmerkmal „**Zuverlässigkeit**“ kann bspw. über den Indikator „Anzahl der Fehler pro Monat“ sowie die Sollstufe 0,01 messbar **definiert** werden
- Für das Qualitätsmerkmal „**Zuverlässigkeit**“ muss
  - die „Anzahl der Fehler während der Nutzung“ und
  - die „Zeit der Nutzung“

  gemessen werden, um den „**Ist-Zuverlässigkeitswert**“ zu bestimmen
- Außerdem kann es sinnvoll sein,
  - die „Anzahl der Fehler pro Systemtest“ oder
  - die „Komplexität der Systemstruktur“

  als **Indikatoren** zur frühen Vorhersage der **erwarteten Zuverlässigkeit** des Softwareprodukts zu verwenden

- Die **Qualitätsplanung** muss mit dem **Auftraggeber abgestimmt** werden

1. Qualitätsplanung

- **Ergebnisse** aus Qualitätsplanung werden im **Qualitätssicherungsplan** dokumentiert, der folgende vier Fragen im Detail beantwortet:

1. **Was** muss gesichert werden?
   - Identifizierung der relevanten Qualitätsmerkmale, ihre relative Bedeutung und ihre Quantifizierung in Form von Maßen
2. **Wann** muss gesichert werden?
   - Festlegung der Zeitpunkte für die kontinuierliche Datenerfassung über den gesamten Entwicklungsprozess
3. **Wie** muss gesichert werden?
   - Auswahl der zur Datenerfassung und Qualitätsprüfung geeigneten Techniken und Methoden (Messwerkzeuge, Testverfahren, Reviews, Walkthroughs, Inspektionen, Audits etc.)
4. **Von wem** muss gesichert werden?
   - Festlegung der Rollen und Verantwortlichkeiten für die Qualitätsprüfung und -lenkung

2. Qualitätslenkung und -sicherung

3. Qualitätsprüfung

- **Führt** die im Rahmen der **Qualitätsplanung** festgelegten **Maßnahmen** zur Erfassung von **Qualitäts-Ist-Werten durch** UND
- **Überwacht**, ob die **konstruktiven Maßnahmen** umgesetzt werden
- Typische Aktivitäten sind:
  - die **Erfassung von Messdaten** über Messwerkzeuge oder Formblätter und Interviews (zur Erfassung von Fehler- oder Aufwandsdaten)
  - **Tests** (zur Erfassung dynamischer Produktmerkmale) oder Inspektionen
  - **Reviews, Walkthroughs** (zur Erfassung statischer Produktmerkmale) oder **Audits** (zur Erfassung der Prozesseigenschaften und -merkmale)

3. Qualitätsprüfung

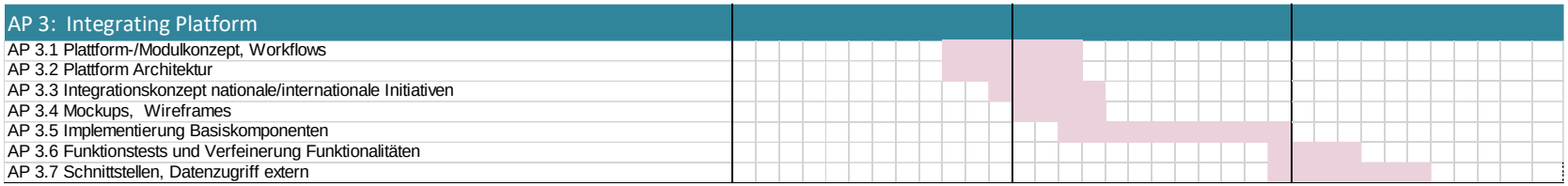
- Weitere Qualitätsprüfungen sind **Mängel- und Fehleranalysen**, die auf Mängelkatalogen und Problemberichten basieren

- Sie geben Antworten auf folgende Fragen:
  - In welcher Phase kommen welche Fehlertypen am häufigsten vor?
  - Wie viele bekannte, aber noch nicht behobene Fehler existieren noch im Softwareprodukt?

- Sind die Basis für weitere Verbesserungen des Entwicklungsprozesses

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Produkt- und prozessabhängige Qualitätszielbestimmung



- Wann ist bspw. AP 3.5 abgeschlossen?
- Welche Erwartung hat **AG** an die Abnahme?
  - Funktionalität <-> Qualität
- Welche Erwartung hat **AN** an die Abnahme?
  - Funktionalität <-> Qualität
- Wie kann sichergestellt werden, dass Erwartungshaltung beidseits zueinander passt?

Produkt- und prozessabhängige Qualitätszielbestimmung

- Ein **Softwareprodukt** sollte **nach der Fertigstellung** eine **bestimmte Qualität** besitzen
  - Problem: Oftmals wird die Qualität weder explizit noch implizit festgelegt
- Da gewünschte Qualität sowohl Kosten und Termine als auch konstruktive und analytische QM-Maßnahmen wesentlich beeinflusst, ist eine **explizite Qualitätszielbestimmung** für AG und AN sehr wichtig
  - So wird definiert, welche Qualität der AG benötigt und welche er erhält
  - Dadurch werden auch **pauschale** Forderungen nach „bester Qualität“ verhindert
  - Wichtig: Auftragnehmer sollte in der Lage sein anzugeben, wie sich die Kosten und Termine in Abhängigkeit von der gewünschten Qualitätsstufe verändern

Produkt- und prozessabhängige Qualitätszielbestimmung

- Die in der **Qualitätszielbestimmung** festgelegten **Qualitätsanforderungen** werden vom Auftraggeber beim **Abnahmetest** verwendet
  - Das Produkt wird hierbei überprüft, ob es die Qualitätsanforderungen umfassend erfüllt
- Für den Auftragnehmer ergeben sich aus den **Qualitätsanforderungen** die **Maßnahmen** für den **Entwicklungsprozess**
  1. Die Entwickler wissen, was von ihnen bzw. dem Produkt **erwartet** wird und können geeignete Methoden und Werkzeuge einsetzen
  2. Ohne festgelegte Qualitätsanforderungen tappen die SW-Entwickler im Dunkeln und wissen nicht, welche Qualitätsmerkmale zu fokussieren sind
- Fazit: Produkt- und prozessabhängige **Qualitätszielbestimmung** bringt sowohl für AG als auch AN die notwendige **Planungssicherheit**

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Quantitative Qualitätssicherung

- „Ingenieursmäßige Qualitätssicherung ist undenkbar ohne die Quantifizierung von Soll- und Ist-Werten“ (Rombach 1993, S. 270)

- **Quantifizierung** im Bereich Software-QS stößt auf **Probleme**:
  - Maße sind ziel- und kontextabhängig
  - Anzahl der Parameter ist um ein Vielfaches höher als bei konventionellen Produktions- und Fertigungsprozessen
  - Kreativer Charakter vieler Aspekte der SW-Entwicklung
  - Unkontrollierte Variabilität von Entwicklungsprozessen

Quantitative Qualitätssicherung

- Messen ist geeignet:
  1. Zum besseren Verständnis unterschiedlicher Qualitätsmerkmale durch die Erstellung deskriptiver Modelle
  2. Zur besseren Planung und Sicherung von Qualitätsmerkmalen durch die Entwicklung präskriptiver Modelle
  3. Zur Verbesserung von Entwicklungsansätzen durch die experimentelle Erprobung (Exploration) alternativer Methoden
  4. Methoden und Werkzeuge zur Datenerfassung, statistischen Auswertung und Präsentation von Messdaten sind vorhanden

- Dennoch: es gibt keinen allgemeingültigen Satz von Maßen, die unabhängig von Ziel und Kontext sind

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Maximale konstruktive Qualitätssicherung

- Das Prinzip der maximalen konstruktiven Qualitätssicherung verfolgt das allgemeine Ziel:
  - „Vorbeugen ist besser als heilen“ oder
  - „Fehler, die nicht gemacht wurden, müssen auch nicht behoben werden“
- **Ziel**: durch geeignete konstruktive Maßnahmen werden die notwendigen **analytischen Maßnahmen** auf ein **Minimum** reduziert

Maximale konstruktive Qualitätssicherung

- Beispiele
  - Eine vollständige Zweigüberdeckung ist ökonomisch nur dann sinnvoll zu erreichen, wenn der Quellcodeumfang klein und die Verzweigungslogik nicht zu komplex ist
  - Explizite Vereinbarung aller Variablen in einem Programm mit zugehöriger Typfestlegung ermöglicht erst folgende analytische Überprüfungen:
    - Sind alle verwendeten Variablen vereinbart?
    - Werden alle Variablen typgerecht angewandt?

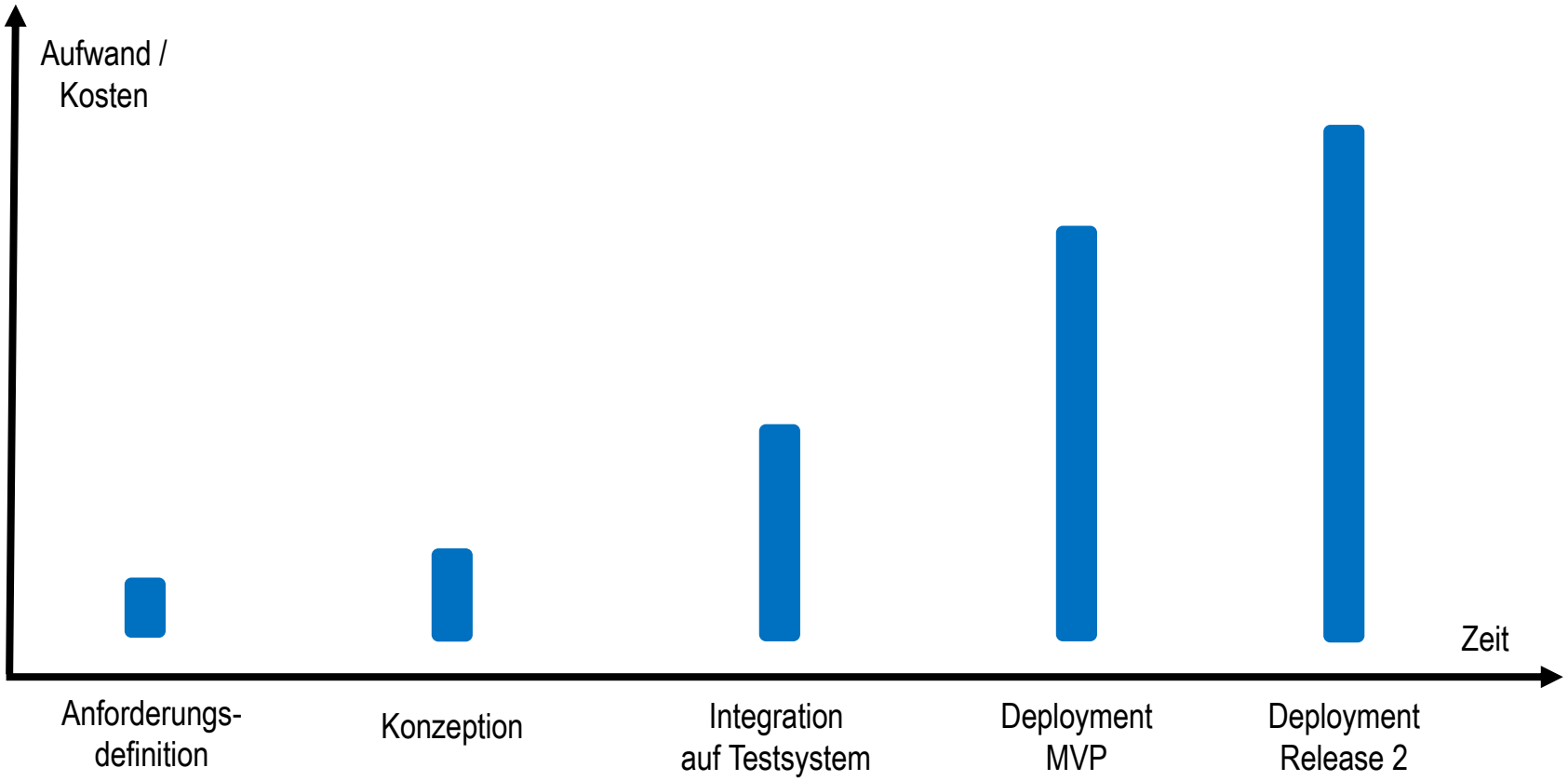
Maximale konstruktive Qualitätssicherung

- Fazit:
  - **Analytische** Maßnahmen werden oftmals erst durch **konstruktive** Maßnahmen **möglich** gemacht
  - Vorteile der maximalen konstruktiven QS:
    1. Direkte Verbesserung der Produktqualität durch Vermeidung von Fehlern
    2. Aufwandsreduktion im Rahmen der analytischen Maßnahmen

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Frühzeitige Fehlerentdeckung und -behebung

- Welche Aufwände und Kosten verursacht Fehlerbehebung üblicherweise im Verlauf der Softwareentwicklung?



Frühzeitige Fehlerentdeckung und -behebung

- Oftmals wird erst **nach** der Programmierung begonnen, ein Softwareprodukt auf die festgelegten Qualitätsanforderungen hin zu überprüfen
- Stellt man **nach** der Programmierung fest, dass das Softwareprodukt von den Anforderungen **abweicht**,
  - Muss nicht nur das Programm geändert werden,
  - Sondern auch der Entwurf.
- Noch aufwändiger werden Modifikationen, wenn Fehler **erst im Betrieb** des freigegebenen SW-Produkts festgestellt werden
  - Wenn also der Beta-Test beim Kunden stattfindet
- Dadurch schwindet auch schnell das **Vertrauen** zwischen Kunden und Auftragnehmern
  - Und verlorenes Vertrauen ist nur mühsam wieder aufzubauen

Frühzeitige Fehlerentdeckung und -behebung

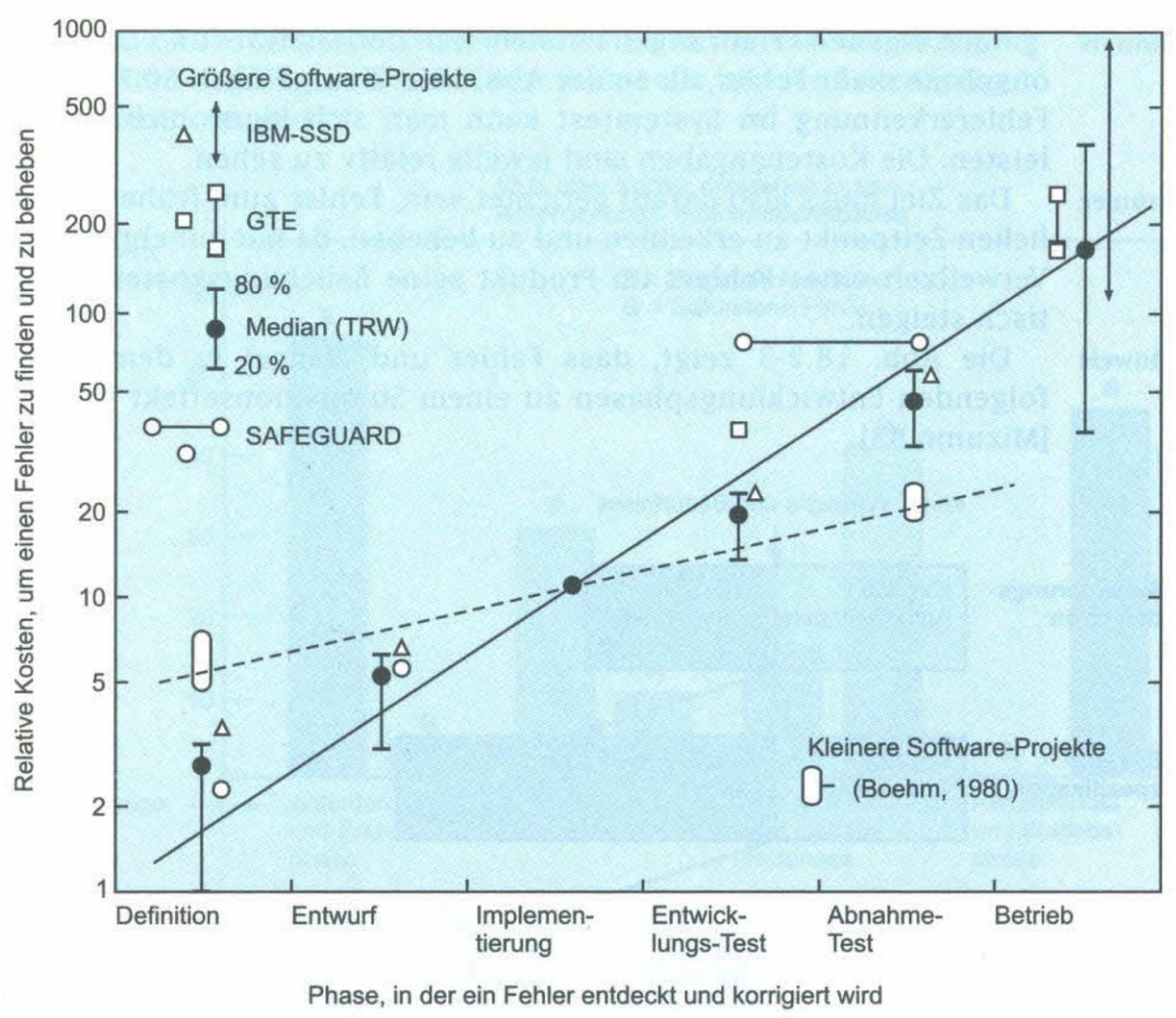
- Ein **Fehler** ist:
  - Jede **Abweichung** von den spezifizierten Anforderungen des Auftraggebers
  - Jede **Inkonsistenz** in den Anforderungen selbst

- Ein Fehler ist nicht nur eine **Abweichung** vom geforderten **funktionalen** Umfang, sondern auch von den gewählten **Qualitätsanforderungen**

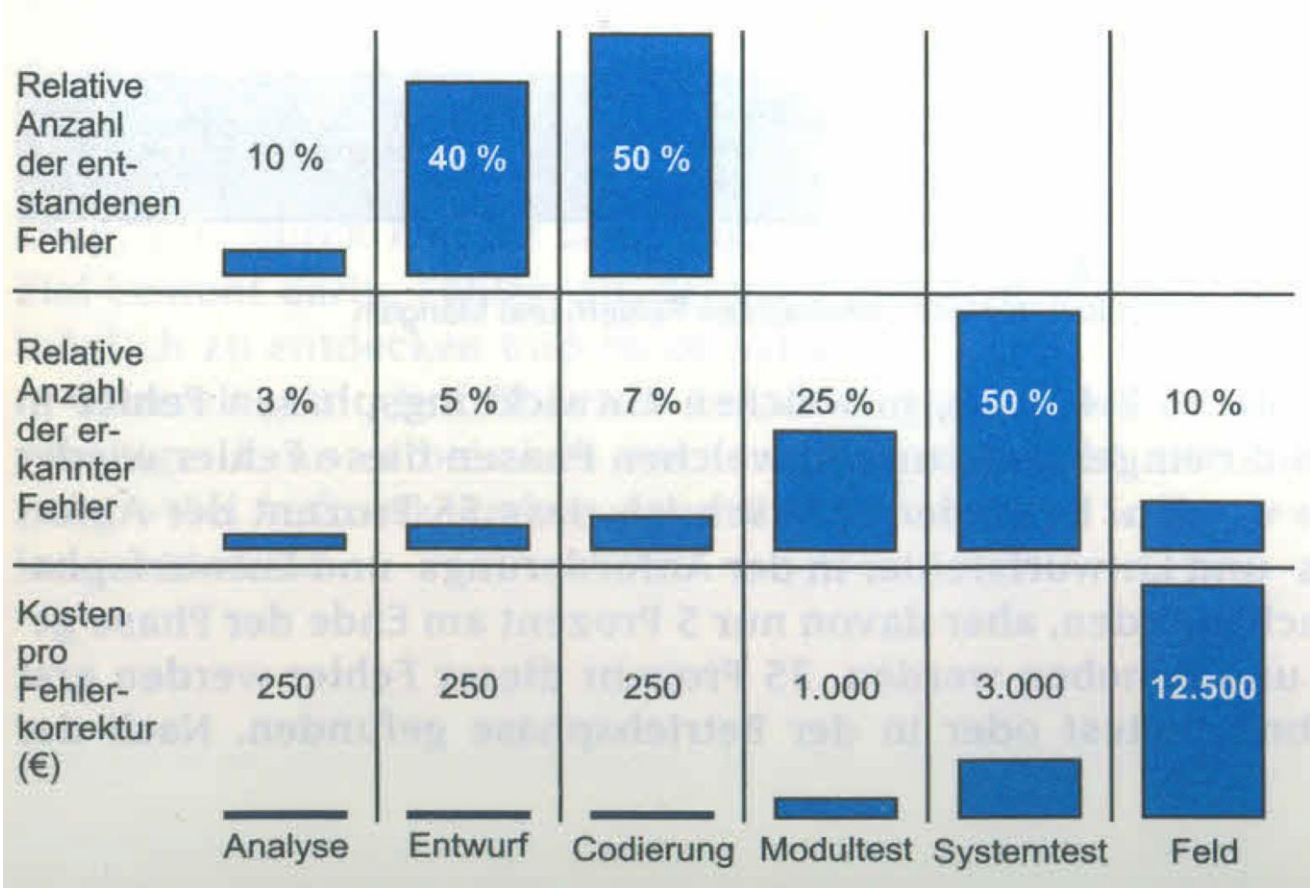
- Eine verzögerte Fehlerentdeckung führt zu **exponentiellem Kostenanstieg** des Software-Projekts

- Daher ist es sinnvoll, konstruktive und analytische QM-Maßnahmen **verstärkt zu Beginn einer Softwareentwicklung** einzusetzen

Frühzeitige Fehlerentdeckung und -behebung



Frühzeitige Fehlerentdeckung und -behebung



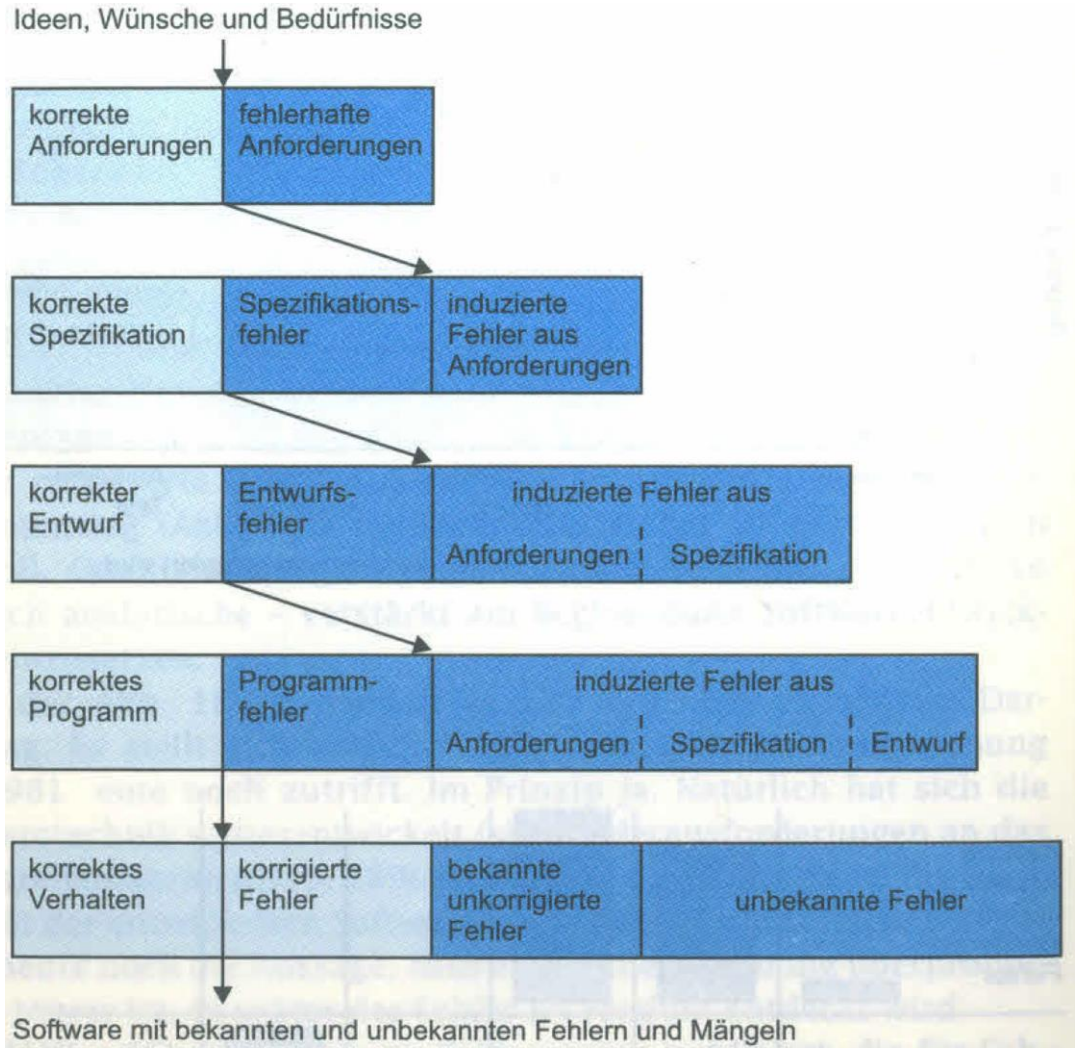
Fehleranzahl und Fehlerkorrekturkosten (vgl. Balzert 2008, S. 485)

Frühzeitige Fehlerentdeckung und -behebung

- Nach praktischer Erfahrung entstehen in der Analyse- und Definitionsphase mehr Fehler als in der vorherigen Abbildung zu sehen waren
  - Eine Rate von 50% Fehlererkennung im Systemtest kann man sich heute nicht mehr leisten

- **Ziel**: Fehler möglichst **frühzeitig** erkennen und beheben
  - Denn: die **Kosten steigen exponentiell** an, je später ein Fehler gefunden und behoben wird
  - Folgende Abbildung zeigt, wie sich Fehler und Mängel in aufeinander folgenden Entwicklungsphasen **summieren**

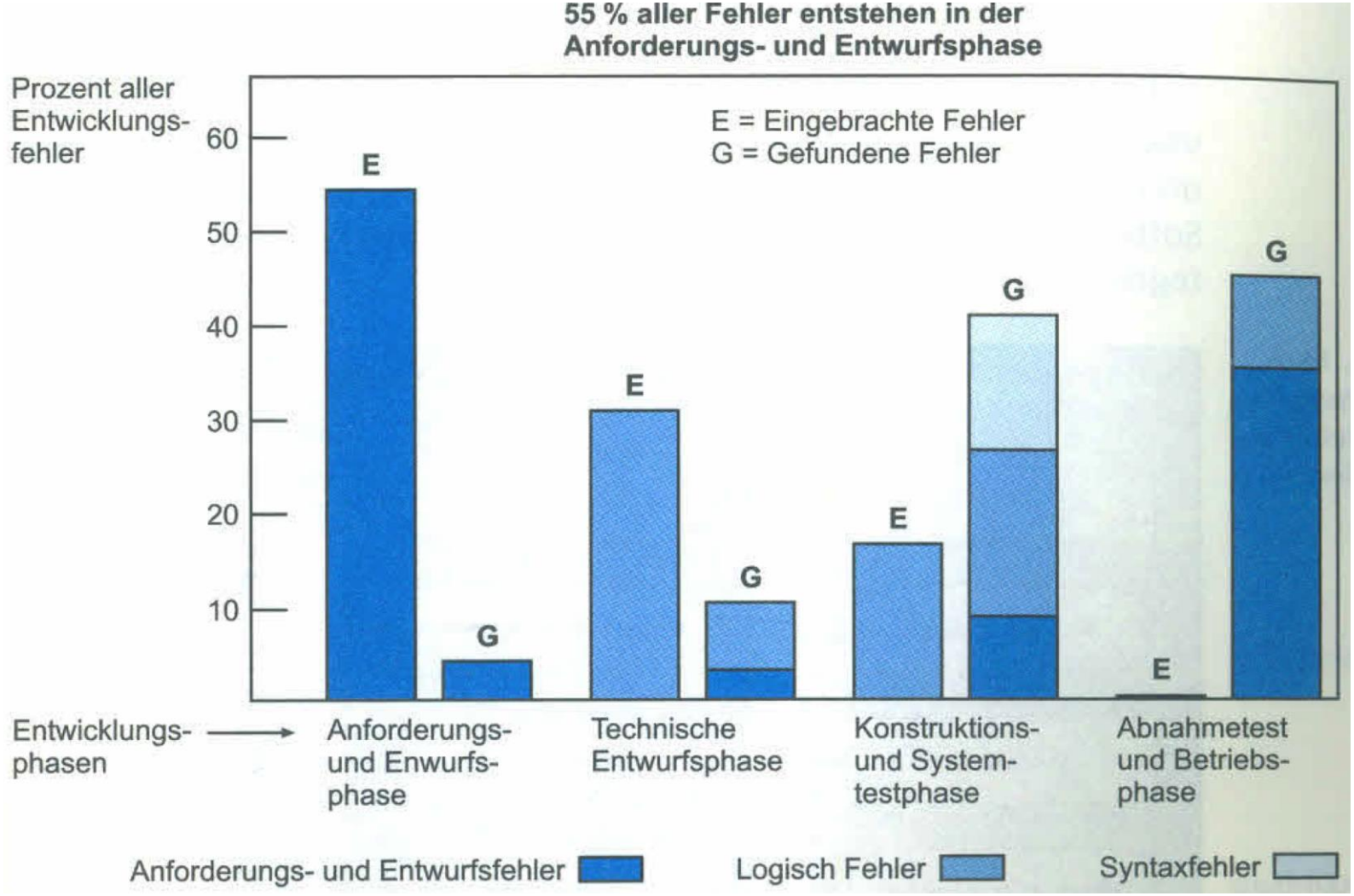
Frühzeitige Fehlerentdeckung und -behebung



Frühzeitige Fehlerentdeckung und -behebung

- Fehler werden in der Praxis
  - in **unterschiedlichen** Entwicklungsphasen in ein Produkt **eingebracht**
  - In **gleichen oder späteren** Phasen **behoben**
- Folgende Abbildung zeigt, wie sich dies in der Praxis verhält
  - Besonders kritisch ist, dass 55 % der Anforderungs- und Entwurfsfehler in der Anforderungs- und Entwurfsphase gemacht werden, aber davon **nur 5 % am Ende der Phase gefunden und behoben** werden
  - 35 % dieser Fehler werden erst beim **Abnahmetest** oder im **Betrieb** gefunden
    - Nach der Abbildung ist dies jedoch die teuerste Variante
  - die **Kosten** liegen um 100-mal höher als bei einer unmittelbaren Beseitigung

Frühzeitige Fehlerentdeckung und -behebung



Frühzeitige Fehlerentdeckung und -behebung

- Die Anwendung des **Prinzips der frühzeitigen Fehlerentdeckung und -behebung** bringt folgende Vorteile mit sich:
  - Fehler in **späteren Phasen** werden vermieden
  - **Kosten** werden reduziert
  - Mit höherer Wahrscheinlichkeit werden Fehler **richtig korrigiert**
  - Die **Fehlerfortpflanzung** wird reduziert
- **Primäres Ziel**: Fehler erst gar nicht machen => viele konstruktive Maßnahmen, um analytische Maßnahmen nicht zu benötigen
- **Sekundäres Ziel**: Fehler die dennoch gemacht wurden, so frühzeitig wie möglich zu entdecken und zu beheben

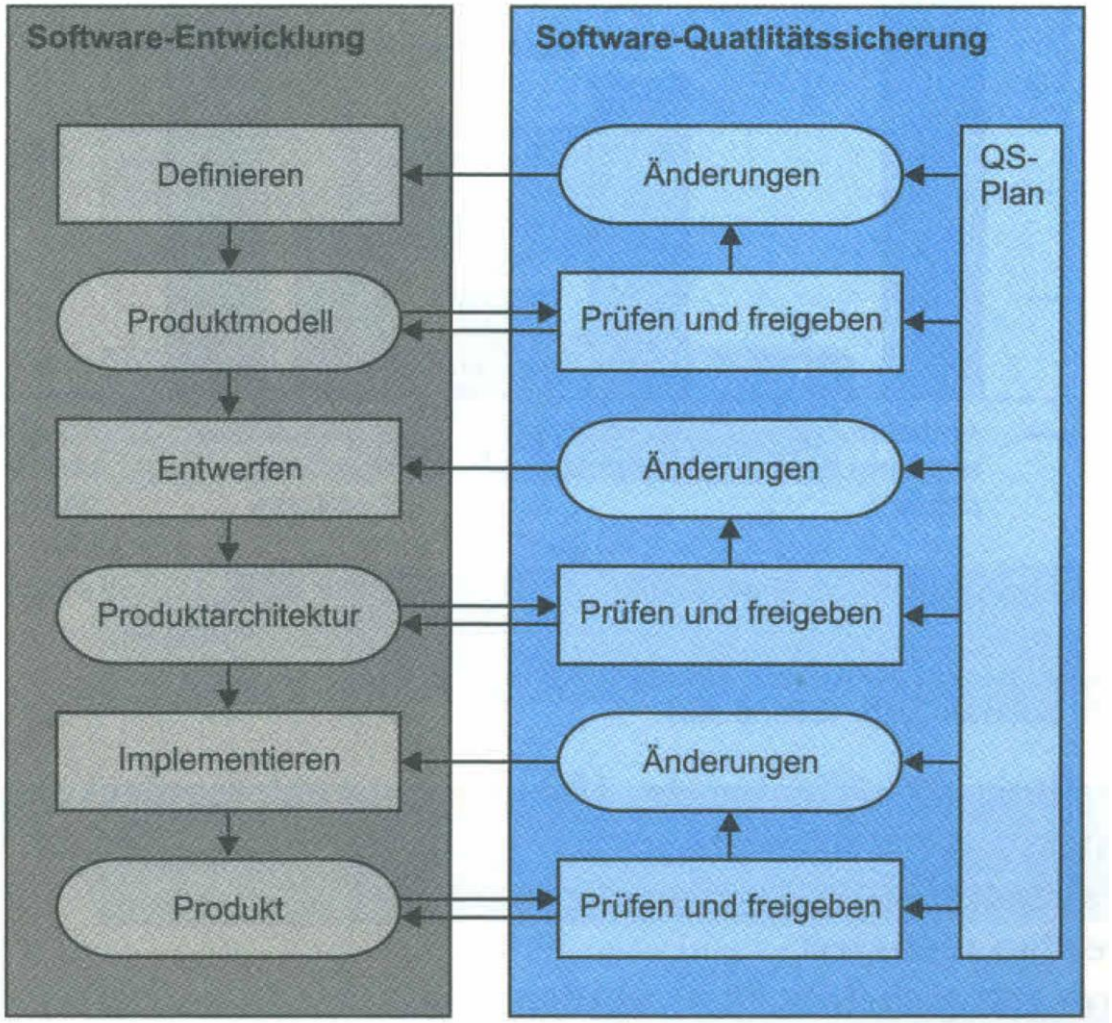
- **Fazit**: Viel Aufmerksamkeit in den frühen Phasen der Softwareentwicklung durch **entwicklungsbegleitende Qualitätssicherung**

- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Entwicklungsbegleitende Qualitätssicherung

- In **klassischen Vorgehensmodellen** setzt das analytische QM vielfach **erst am Ende** des Entwicklungsprozesses ein
  - Zum Beispiel mit Modul-, Integrations- und Systemtest
- Ziele:
  - **Prinzip der frühzeitigen Fehlerentdeckung umsetzen** und
  - zu einem systematischen, **vorgeplanten QM** zu gelangen
- Aufgabe:
  - eine Qualitätssicherung implementieren, die die SW-Entwicklung **kontinuierlich** begleitet und in den Entwicklungsprozess **integriert** ist

Entwicklungsbegleitende Qualitätssicherung



Entwicklungsbegleitende Qualitätssicherung

- Die Anwendung des Prinzips der **entwicklungsbegleitenden Qualitätssicherung** bringt folgende Vorteile mit sich:
  - Einbettung der QS in das organisatorische Ablaufmodell der Softwareentwicklung
  - QS findet jeweils zu dem Zeitpunkt statt, zu dem sie im Entwicklungsprozess angebracht ist
  - QS wird – ähnlich wie in Fertigungsprozessen – nicht als Fremdkörper empfunden; sie gehört direkt zum Softwareentwicklungsprozess
  - Ein Teilprodukt steht der nächsten Phase erst dann zur Verfügung, wenn eine bestimmte Qualität sichergestellt wurde
  - Das Qualitätsniveau ist zu jeder Zeit sichtbar
  - Der Entwicklungsfortschritt kann realistisch beurteilt werden

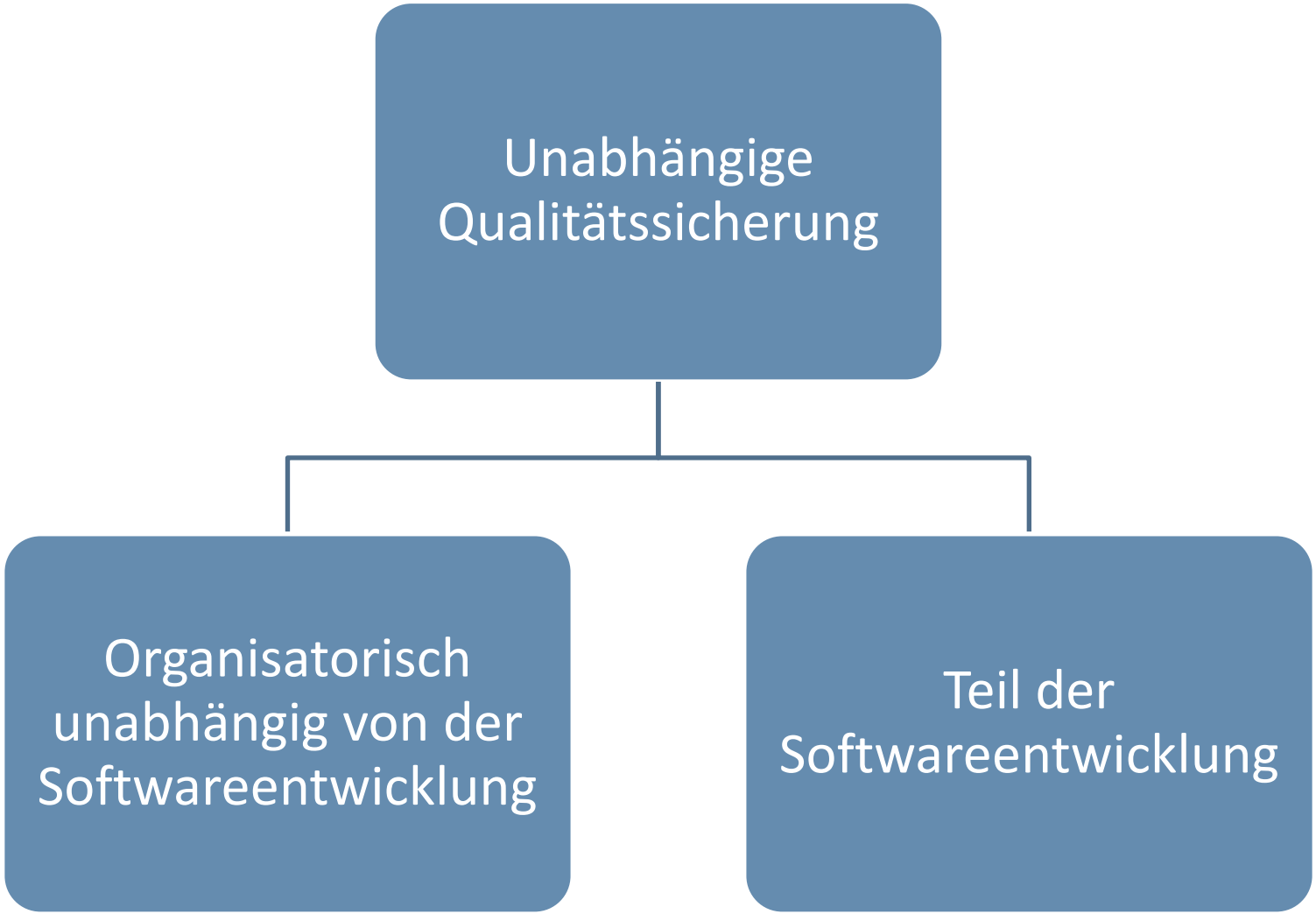
- Qualitätsmanagement
  - Einleitung
  - Produkt- und prozessabhängige Qualitätszielbestimmung
  - Quantitative Qualitätssicherung
  - Maximale konstruktive Qualitätssicherung
  - Frühzeitige Fehlerentdeckung und -behebung
  - Entwicklungsbegleitende Qualitätssicherung
  - Unabhängige Qualitätssicherung

Unabhängige Qualitätssicherung

- „... testing is a destructive process, even a sadistic process ...“ (Myers 1979, S. 5)

- Personen/Rollen, die ein Produkt definiert, entworfen und implementiert haben, sind **am wenigsten dazu geeignet** durch Anwendung analytischer QS-Maßnahmen die Ergebnisse der eigenen Tätigkeit **destruktiv zu überprüfen und zu beurteilen**
  - Konsequenz: Analytisches QM für das Teilprodukt X darf nicht von einem der Entwickler:innen des Teilprodukts X durchgeführt werden
  - Aber: Entwickler:in darf eigene Aufgaben auch nicht an die QS abschieben
  - So muss Entwickler:in ein selbst entwickeltes Programm zunächst selber testen, bevor es von der QS inspiziert und getestet wird

Unabhängige Qualitätssicherung



Organisatorisch unabhängige QS

Vorteile:

- SW-Entwicklung kann keinen Druck auf die QS ausüben
- Neutralität bleibt gewahrt
- Klare Budgetierung wird ermöglicht
- Bedeutung der QS wird betont => kein Anhängsel der SW-Entwicklung

Nachteile:

- Gefahr der Isolierung der QS von der SW-Entwicklung
- Eine gleichmäßige Personalauslastung ist unter Umständen schwierig sicherzustellen

QS als Teil der SWE

Vorteile:

- Personal kann flexibler eingesetzt werden
- QS steht nicht abseits, sondern bekommt alles von der SW-Entwicklung mit
- Gemeinsame Teamarbeit ist leichter möglich
- Vertrauensvollere Zusammenarbeit ist möglich

Nachteile:

- Das SW-Entwicklungsmanagement kann Druck auf die QS ausüben
- Budgetmittel können zugunsten der Entwicklung umverteilt werden

Unabhängige Qualitätssicherung

- QS und SW-Entwicklungsaufgaben sollten **parallel** durchgeführt werden
  - Mitarbeiter zu 80% SW-Entwickler und
    zu 20% Qualitätsüberprüfer anderer SW-Entwicklungen
  - Vorteile:
    - flexibler Personaleinsatz und kein zusätzlicher
      Overhead für die Qualitätssicherung
  - Nachteile:
    - Vermischung der Tätigkeiten kann dazu führen, dass keine der Arbeiten mehr zu 100% richtig durchgeführt werden,
    - dauernder Wechsel zwischen unterschiedlichen Tätigkeiten

Unabhängige Qualitätssicherung - Zusammenfassung

- Verantwortung der SW-Entwicklung vs. Verantwortung der QS
- hierbei gibt es zwei Modelle:
  1. Die SW-Entwicklung erstellt die Produkte. Die QS ist für die Überprüfung zuständig. Fehler werden der Entwicklung gemeldet und von ihr behoben
     - Entwickler kann sich auf die reine Entwicklung konzentrieren
     - Sorgfalt der Entwickler sinkt
  2. Die Software-Entwicklung ist für einen definierten Qualitätszustand ihrer Produkte selbst zuständig.
     - Erst wenn dieser Grad erreicht wurde, übernimmt die QS die Überprüfung der entwickelten SW-Produkte
     - Klar definierte Verantwortlichkeiten
     - Eigenverantwortung der Entwickler wird gefördert
     - Definition messbarer Qualitätsstufen ist notwendig

Herzlichen Dank für Ihre Aufmerksamkeit !