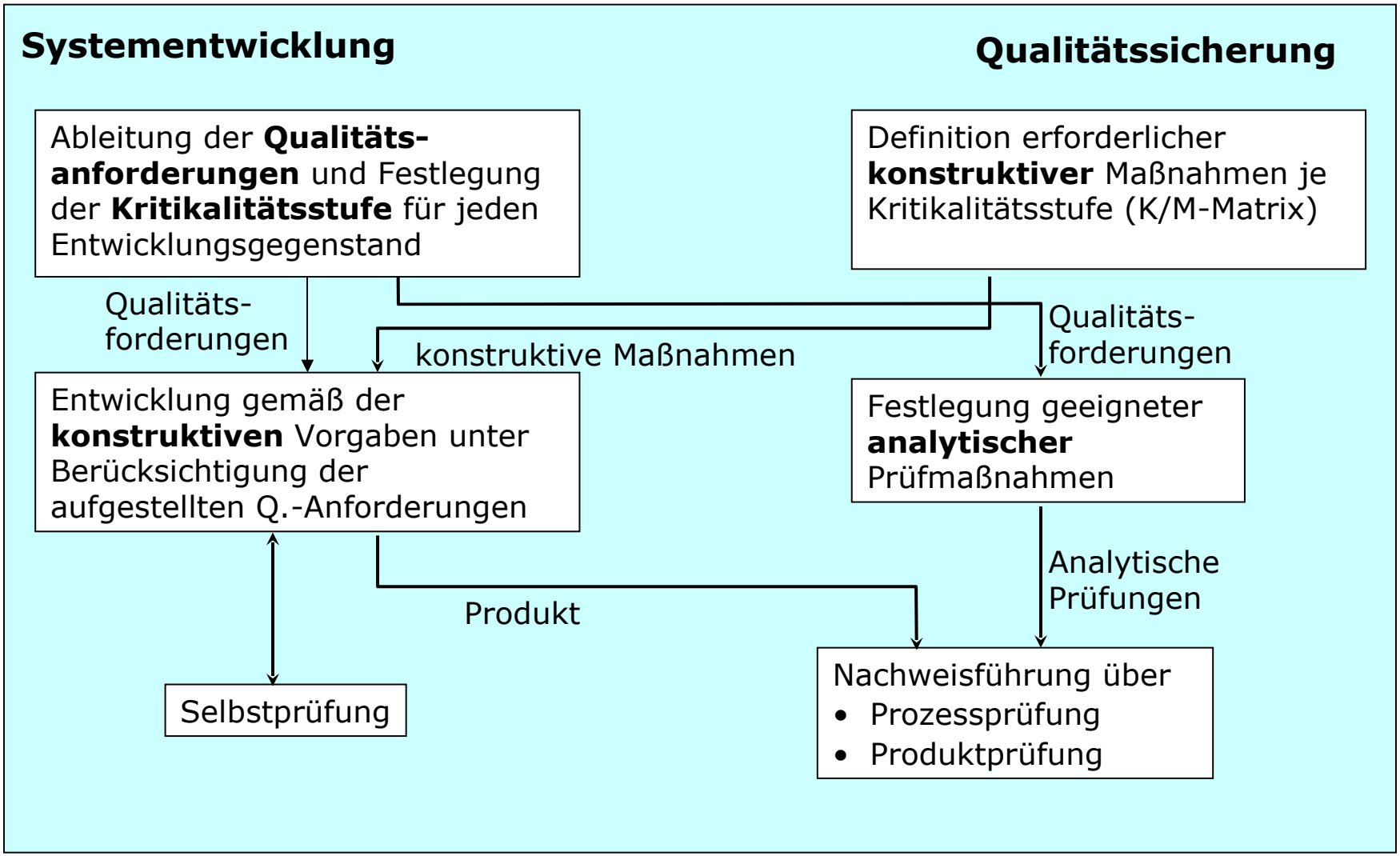
Software-Qualitätsmanagement

Vorlesung im Modul 10-202-2319 Software-Management

Sommersemester 2009

Prof. Dr. Hans-Gert Gräbe

http://bis.informatik.uni-leipzig.de/HansGertGraebe


Systementwicklung
Qualitätssicherung
Ableitung der **Qualitäts-anforderungen** und Festlegung der **Kritikalitätsstufe** für jeden Entwicklungsgegenstand
Definition erforderlicher **konstruktiver** Maßnahmen je Kritikalitätsstufe (K/M-Matrix)
Qualitäts-forderungen
konstruktive Maßnahmen
Qualitäts-forderungen
Entwicklung gemäß der **konstruktiven** Vorgaben unter Berücksichtigung der aufgestellten Q.-Anforderungen
Festlegung geeigneter **analytischer** Prüfmaßnahmen
Produkt
Analytische Prüfungen
Selbstprüfung
Nachweisführung über
• Prozessprüfung
• Produktprüfung

Qualitätssicherung und Systementwicklung

- konstruktive Maßnahmen sind im Rahmen der Anforderungsanalyse und vor Beginn des Entwicklungsprozesses zu fixieren
- analytische Maßnahmen werden Entwicklungsprozess begleitend oder zu Meilensteinen, mit denen Entwicklungsprozess-Etappen abgeschlossen werden, wirksam
- analytische Maßnahmen können konstruktive Vorgaben erfordern, um die Produktion der zu analysierenden Daten zu initiieren.

Analytische und konstruktive QM-Maßnahmen beeinflussen sich gegenseitig: Vorausschauende konstruktive Planung erspart analytischen Aufwand

Qualitätsplanung und Qualitätssicherung

Qualitätsmanagement besteht aus den Phasen

- **Qualitätsplanung**
  - vorbereitende Aktivitäten zur Festlegung von Standards, zu erreichender Parameter und von Verantwortlichkeiten
  - Ergebnis: Qualitätssicherungsplan und Prüfplan

- **Qualitätssicherung**
  - Systementwicklung begleitende Aktivitäten zur Umsetzung und Dokumentierung der erreichten Qualitätsparameter
  - Ergebnis: Protokolle, Zertifikate, Vorschläge für die Projektsteuerung
  - **Qualitätslenkung** und **Qualitätsprüfung**

Qualitätsplanung

- **Qualitätszielbestimmung**: Festlegung von Qualitätsanforderungen an den Prozess und an das Produkt in überprüfbarer Form.
- **Planung der Qualitätslenkung**: Planung der Umsetzung, Steuerung, Überwachung und Korrektur des Entwicklungsprozesses mit dem Ziel, die vorgegebenen Anforderungen zu erfüllen.
- **Planung der Qualitätsprüfung**: Planung der Durchführung der im Rahmen der Qualitätsplanung festgelegten Maßnahmen zur
  - Erfassung von Istwerten der Qualitäts-Indikatoren
  - Überwachung der Umsetzung der konstruktiven Maßnahmen
  - Tests, Reviews, Audits, Inspektionen
- **Planung der Qualitätsverbesserung**: Planung der Auswertung der Qualitätssicherungs-Ergebnisse und Prozessverbesserung.
  - Mängel- und Fehleranalyse (Verbesserung der Prozessqualität)

Qualitätssicherungsplan und Prüfplan

- Ergebnisse der Qualitätsplanung werden in einem **Qualitäts-Sicherungsplan** dokumentiert (prozess-orientiert), die begleitenden Maßnahmen in einem **Prüfplan** festgelegt (produkt-orientiert).
  - **Festlegung der Aufgaben**
    - Was ist zu tun?
    - Identifizierung der zu sichernden *Produkte*
    - Identifizierung der relevanten Qualitäts*merkmale*, ihre relative *Bedeutung* und ihre *Quantifizierung* in Form von Metriken
  - **Festlegung der Vorgaben und Hilfsmittel**
    - Wie ist es zu tun?
    - Auswahl der zur Datenerfassung und Qualitätsprüfung geeigneten *Techniken* und *Methoden*
    - konstruktive Vorgaben (etwa Richtlinien, Vorlagen)
    - analytische Vorgaben (Verfahren, Werkzeuge)

Qualitätssicherungsplan und Prüfplan

- **Festlegung der Termine**
  - (Bis) wann ist es zu tun?
  - Festlegung der *Zeitpunkte* für die den gesamten Entwicklungsprozess begleitende *Datenerfassung*
  - Einordnung des Prüfplans in den Projektplan
- **Festlegung der Verantwortlichkeiten**
  - Wer hat es zu tun?
  - Festlegung der *Verantwortlichkeiten* für die Qualitätsprüfung und -lenkung.
  - Definition und Besetzung von *Rollen* (Q-Manager, Prüfer, Autor, Gutachter)

- Der IEEE-Standard 730 für den *Software Quality Assurance Plan* beschreibt den Aufbau eines solchen Qualitätssicherungsplans.

Grundsätze für die Qualitätssicherung in der Software-Entwicklung

- produkt- und prozessabhängige Qualitätszielbestimmung
- quantitative Qualitätssicherung
- maximale konstruktive Qualitätssicherung
- frühzeitige Fehlerentdeckung und -behebung
- entwicklungsbegleitende, integrierte Qualitätssicherung
- unabhängige Qualitätssicherung

Prinzip der produkt- und prozessabhängigen Qualitätszielbestimmung

- Nur 50% der Betriebe legen Qualitätsmerkmale fest [Spillner et al. 94].
- explizite und transparente Qualitätszielbestimmung ist vor Beginn des Entwicklungsprozesses äußerst hilfreich.
  - Die in der Qualitätszielbestimmung festgelegten Qualitätsanforderungen werden vom Auftraggeber für den Abnahmetest verwendet.
  - Für den Software-Lieferanten ergeben sich aus den Qualitäts-anforderungen die Maßnahmen für den Entwicklungsprozess und die Qualitätsprüfung.
- **Vorteil**: Prinzip der produkt- und prozessabhängigen Qualitätszielbestimmung vor Beginn des Entwicklungsprozesses bringt Planungs- und Kalkulations-sicherheit.
- **Nachteil**: Qualitätssicherung beansprucht zusätzliche Ressourcen

Prinzip der quantitativen Qualitätssicherung

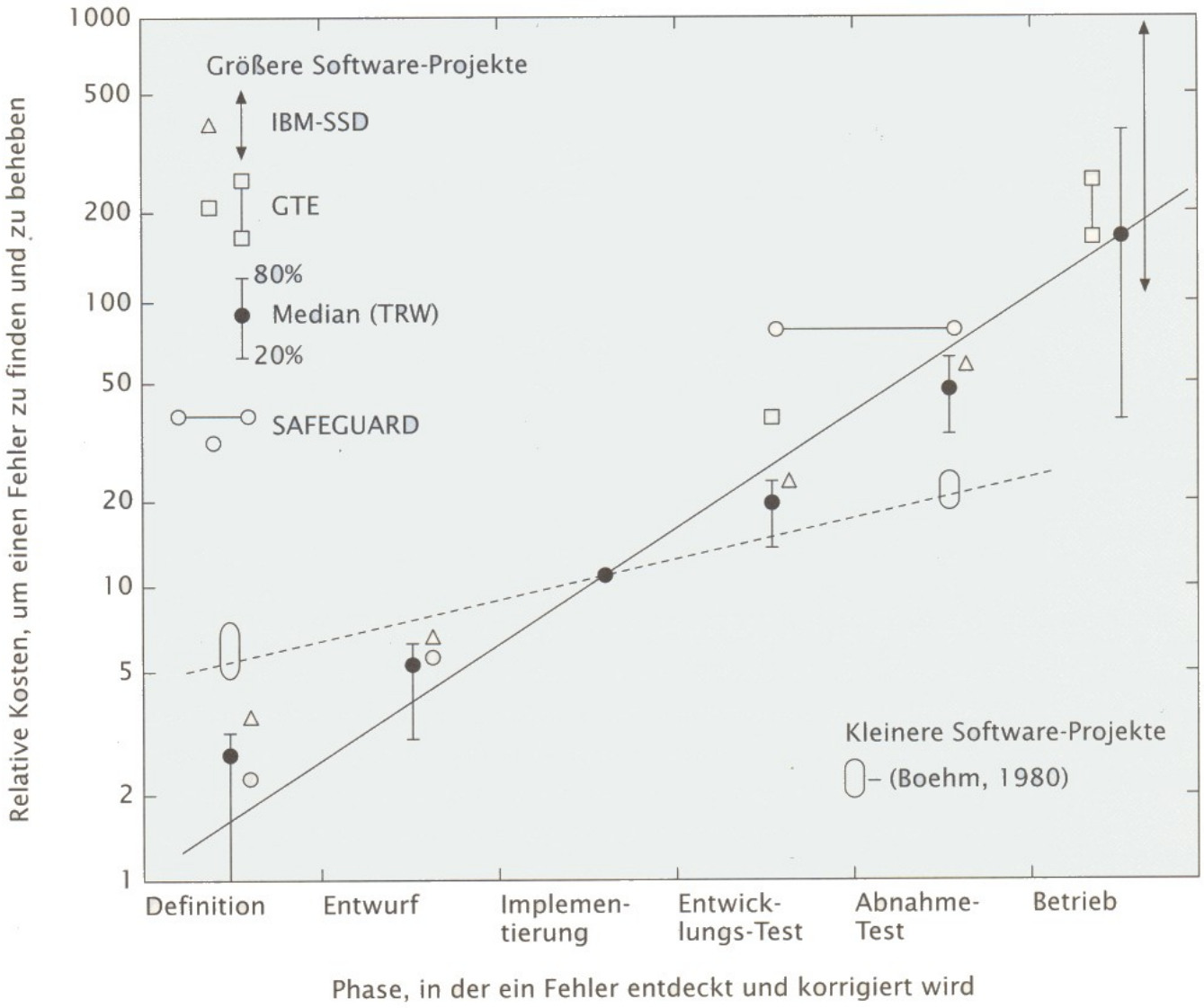
- **Nachteile**:
  - Schwierigkeiten bei der Quantifizierung von Soll- und Istwerten:
  - Metriken sind ziel- und kontextabhängig
  - Anzahl der Variationsparameter ist um ein Vielfaches höher als bei traditionellen Produktionsprozessen
  - kreativer Charakter vieler Aspekte der Software-Entwicklung
  - unkontrollierte Variabilität von Entwicklungsprozessen
- **Vorteile:**
  - Messen ist geeignet
    - zum besseren Verständnis unterschiedlicher Qualitätsmerkmale
    - zur besseren Planung und Sicherung von Qualitätsmerkmalen
    - zur Verbesserung von Entwicklungsansätzen
  - Methoden und Werkzeuge zur Planung und Durchführung der Datenerfassung sowie zur Auswertung und Präsentation von Messdaten sind oft schon vorhanden.

Prinzip der maximalen konstruktiven Qualitätssicherung

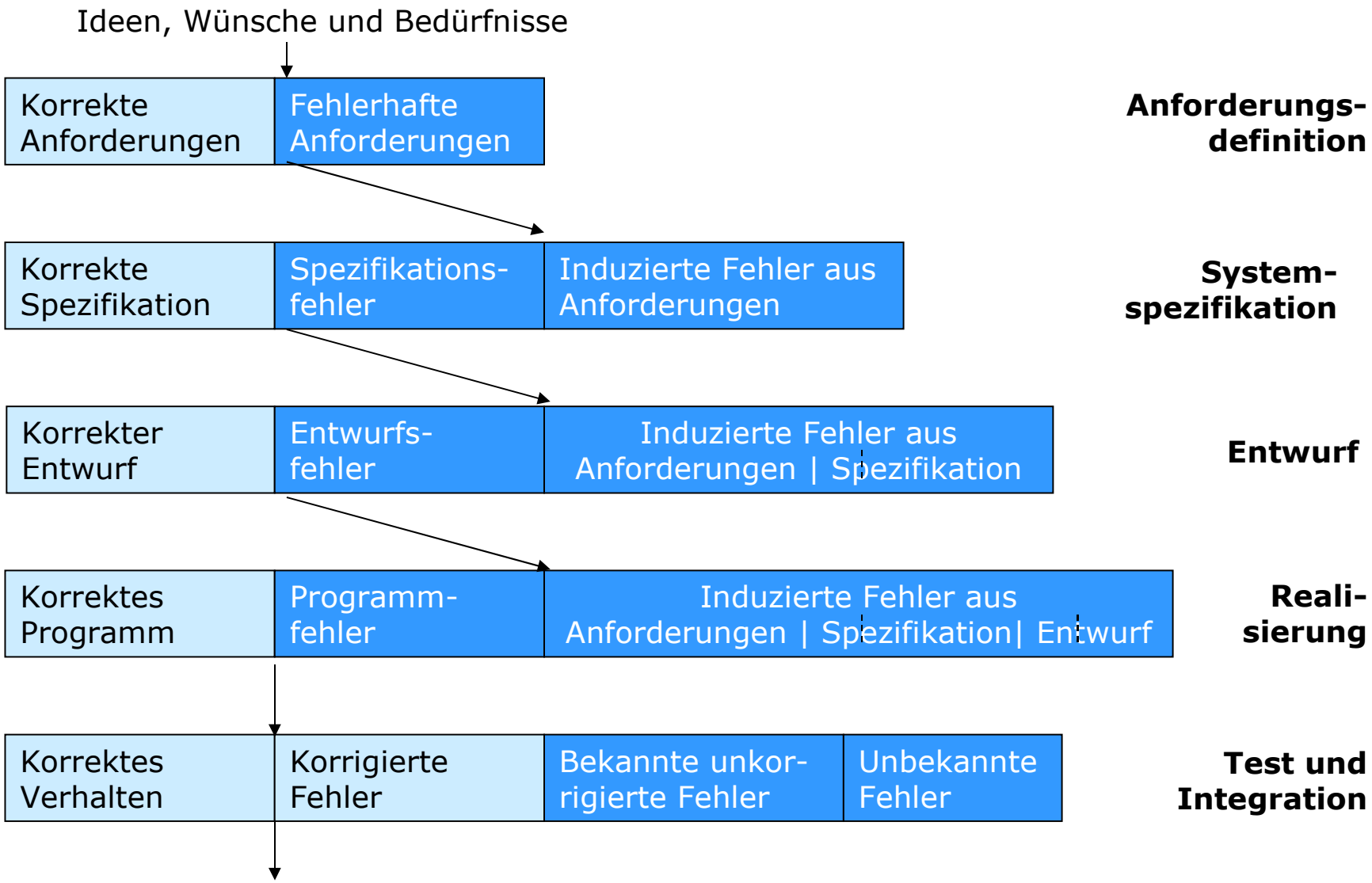
- „Fehler die nicht gemacht werden, brauchen auch nicht behoben werden“ ist das Ziel, das von der maximalen konstruktiven Qualitätssicherung verfolgt wird.
- Reduzierung der analytischen Maßnahmen über Einschränkung der Variationsbreite durch vorausblickende konstruktive Maßnahmen.
- **Vorteile:**
  - direkte Verbesserung der Produktivität
  - Reduktion analytischer Maßnahmen
  - Voraussetzung für analytische Maßnahmen
  - Vermeidung von Fehlern
- **Nachteile**:
  - Konstruktive Maßnahmen schränken die Variabilität und damit die Kreativität ein.

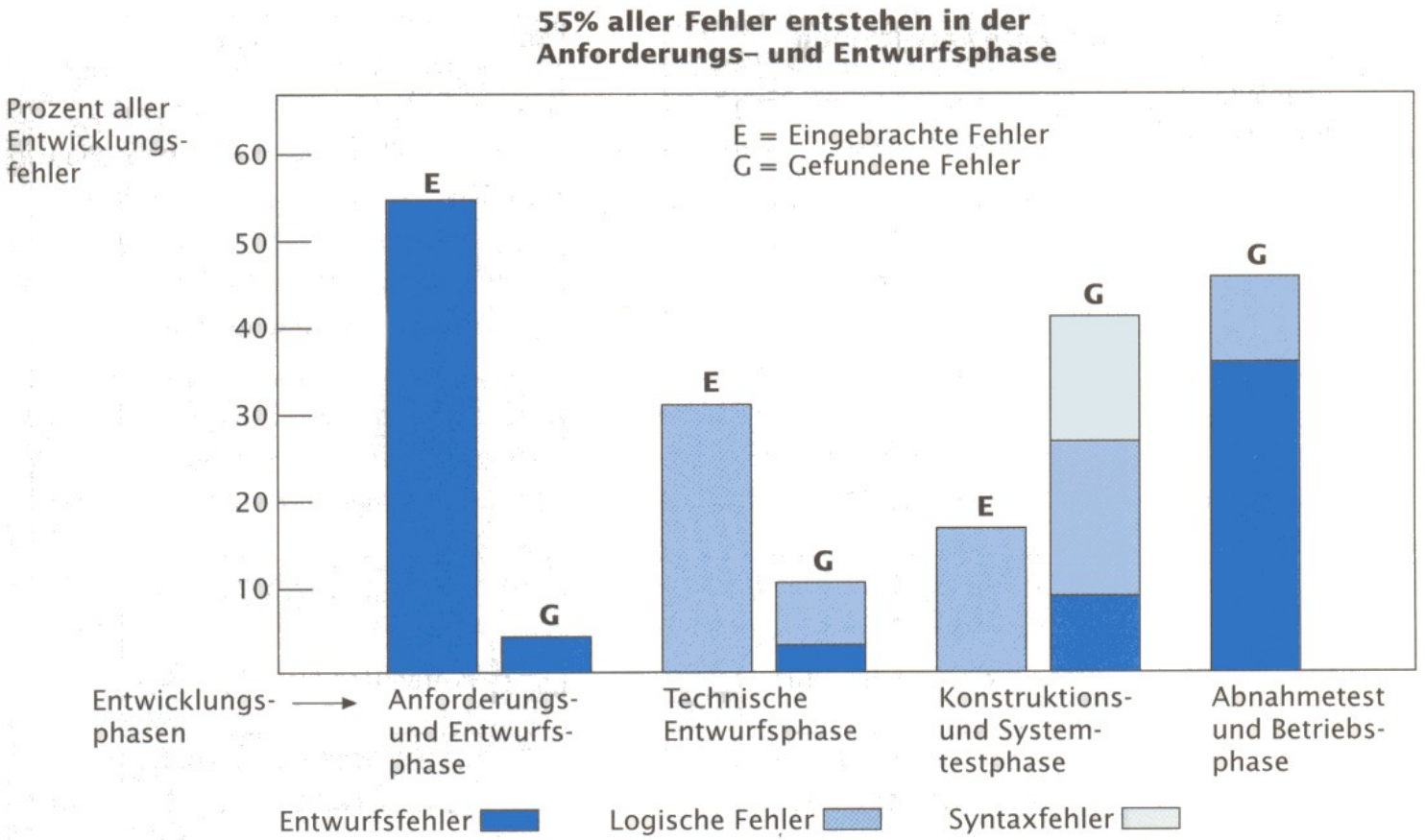
Prinzip der frühzeitigen Fehlerentdeckung und –behebung

- Fehler ist:
  - Abweichung von den Anforderungen des Auftraggebers
  - Inkonsistenz in den Anforderungen
- Ziel ist es, Fehler gar nicht erst zu machen (konstr. Maßnahmen) oder zum frühestmöglichen Zeitpunkt zu erkennen und zu beheben
- **Vorteile:**
  - Fehler in späteren Phasen werden vermieden
  - Kosten werden reduziert
  - mit höherer Wahrscheinlichkeit werden Fehler richtig korrigiert
  - die Fehlerfortpflanzung wird reduziert
- **Nachteil**: Langsamerer Projektfortschritt durch Fehlersuche bereits in frühen Projektphasen
  - Wird aber durch die Vorteile mehr als aufgewogen
- **Folgerung**: Viel Aufmerksamkeit den frühen Projektphasen



Kosten einer verzögerten Fehlerentdeckung [Boehm 76]


Ideen, Wünsche und Bedürfnisse
Korrekte Anforderungen
Fehlerhafte Anforderungen
Anforderungs-definition
Korrekte Spezifikation
Spezifikations-fehler
Induzierte Fehler aus Anforderungen
System-spezifikation
Korrekter Entwurf
Entwurfs-fehler
Induzierte Fehler aus Anforderungen | Spezifikation
Entwurf
Korrektes Programm
Programm-fehler
Induzierte Fehler aus Anforderungen | Spezifikation| Entwurf
Reali-sierung
Korrektes Verhalten
Korrigierte Fehler
Bekannte unkor-rigierte Fehler
Unbekannte Fehler
Test und Integration
Software mit bekannten und unbekannten Fehlern und Mängeln

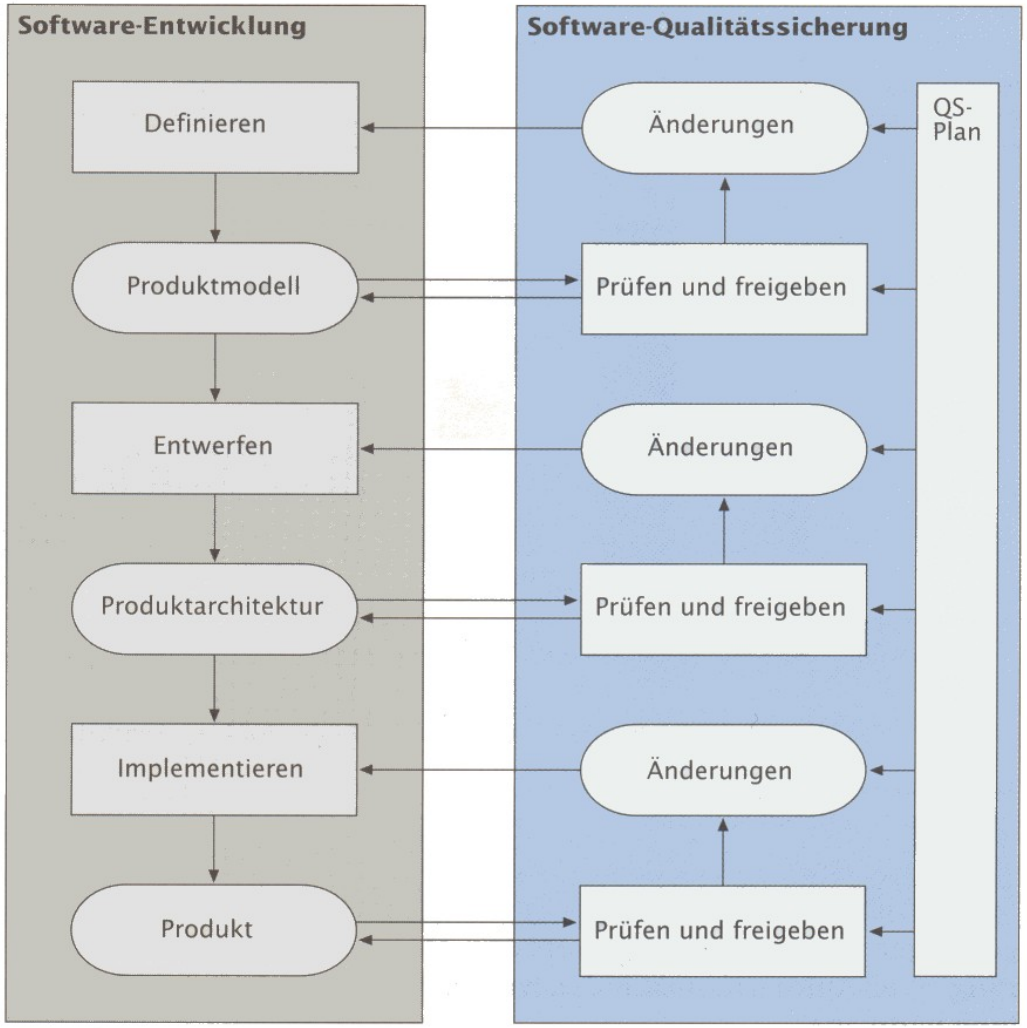


Fehlerbeseitigungskosten (abgeleitet von Alberts 1985)

Prinzip der entwicklungsbegleitenden, integrierten Qualitätssicherung

- Um das Prinzip der frühzeitigen Fehlerentdeckung zu realisieren, ist Softwareentwicklung begleitende und in den Entwicklungsprozess integrierte Qualitätssicherung nötig.

- **Gründe:**
  - Einbettung der Qualitätssicherung in das organisatorische Ablaufmodell der Software-Entwicklung
  - Qualitätssicherung findet zu dem Zeitpunkt statt, zu dem sie im Entwicklungsprozess angebracht ist
  - Qualitätssicherung wird nicht als Fremdkörper empfunden, sondern gehört per se zur Software-Erstellung


Software-Entwicklung
Software-Qualitätssicherung
Definieren
Produktmodell
Entwerfen
Produktarchitektur
Implementieren
Produkt
Änderungen
Prüfen und freigeben
Änderungen
Prüfen und freigeben
Änderungen
Prüfen und freigeben
QS-Plan

Prinzip der unabhängigen Qualitätssicherung

*„... Testing is a **destructive** process, even a sadistic process ...“* [Myers 79]

- Der Entwickler eines Produkts ist am schlechtesten geeignet, um durch Anwendung analytischer QS-Maßnahmen die Ergebnisse seiner Tätigkeit zu betrachten.
- Entwickler darf aber seine Aufgaben im Bereich QS nicht vernachlässigen.
- Zwei organisatorische Alternativen:
  - Qualitätssicherung als organisatorisch unabhängiger Teil von der Gestaltung
  - Qualitätssicherung als Teil der Entwicklung

Prinzip der unabhängigen Qualitätssicherung - Qualitätssicherung als organisatorisch unabhängiger Teil

- **Vorteile:**
  - Entwicklung übt keinen „Druck" auf die Qualitätssicherung aus
  - Neutralität
  - klare Budgetaufteilung
  - Betonung der Qualitätssicherung

- **Nachteile:**
  - Gefahr der Isolierung der Qualitätssicherung von der Entwicklung
  - gleichmäßige Personalauslastung ist unter Umständen sicherzustellen

Prinzip der unabhängigen Qualitätssicherung - Qualitätssicherung als Teil der Entwicklung

- **Vorteile:**
  - flexiblere Einsetzung des Personals
  - Qualitätssicherung „bekommt alles mit"
  - Erleichterung der Teamarbeit
  - vertrauensvolle Zusammenarbeit

- **Nachteile:**
  - Entwicklungsmanagement kann „Druck" auf die Qualitätssicherung ausüben
  - Budgetmittel können zugunsten der Entwicklung umverteilt werden

Prinzip der unabhängigen Qualitätssicherung
Personalalternativen

3 Möglichkeiten für die Personalausstattung der Qualitätssicherung:

- Personal arbeitet nur in der Qualitätssicherung
  - Ermöglicht die Einstellung von Mitarbeitern mit einem hohen Spezialisierungsgrad, aber diese bekommen nie Erfahrung mit der Entwicklung von Software.
- Jeder Mitarbeiter rotiert in festgelegten Abständen zwischen der Qualitätssicherung und der Entwicklung
  - Ermöglicht einen systematischen Wissenstransfer, aber Mitarbeiter müssen Tätigkeiten durchführen, zu denen sie keine „Lust" haben.
- Jeder Mitarbeiter arbeitet sowohl an der Qualitätssicherung als auch an den Entwicklung (in der Praxis üblich)
  - Ermöglicht einen flexiblen Personaleinsatz, aber die Vermischung der Entwicklung und Qualitätssicherung könnte dazu führen, dass keine dieser Arbeiten richtig durchgeführt werden.

Prinzip der unabhängigen Qualitätssicherung - Verhältnis zwischen Entwicklung und Qualitätssicherung

- Strikte Trennung zwischen Entwicklung und Qualitätssicherung:
  Die Entwicklung erstellt Produkte und die Qualitätssicherung ist für die Überprüfung zuständig.
  - Entwickler kann sich auf die konstruktiven Aspekte konzentrieren
  - Wird aber nicht zu Sorgfalt angehalten
  - Realisierung im Ansatz des Pair Programming

- Operative Qualitätssicherung wird in die Entwicklung integriert:
  Die Entwicklung ist für einen definierten Qualitätszustand ihrer Produkte selbst zuständig. Erst danach setzt die externe Q.-Sicherung ein.
  - Klar definierte, transparente Verantwortlichkeiten
  - Eigenverantwortlichkeit der Entwickler
  - Erfordert messbare Qualitätsstufen und Nachweis, dass sie erreicht wurden

Prinzip der unabhängigen Qualitätssicherung im Lichte der quantitativen Qualitätssicherung

- Wird das Prinzip der quantitativen Qualitätssicherung befolgt, so werden viele Prüfparameter instrumentell erfasst.
- Personal der Qualitätssicherung kann sich auf Fragen der Interpretation der Messwerte konzentrieren.
- In der Qualitätssicherung sind folgende zusätzliche Maßnahmen erforderlich:
  - Sammlung von Daten,
  - Validierung dieser Daten und
  - Einrichten einer quantitativ orientierten Datenbank.
- Vorteile einer unabhängigen Qualitätssicherung sind:
  - objektive, unabhängige Qualitätssicherung,
  - heilsame Wirkung auf Entwicklung und
  - Qualitätsvergleiche werden möglich.

Das V-Modell [Boehm 81, 84]

- Erweiterung des Wasserfallmodells um ein integriertes **Qualitäts-Sicherungssystems** zur Durchführung des Qualitätsmanagements.
  - genaue Festlegungen zu Verifikation und Validierung von Teilprodukten
  - Verifikation: Überprüfung auf Übereinstimmung zwischen Spezifikation und Produkt (Wird ein korrektes Produkt entwickelt?)
  - Validierung: Überprüfung der Eignung eines Produkts hinsichtlich seines Einsatzzwecks (Wird das richtige Produkt entwickelt?)

- **V-Modell** ist ein **Vorgehensmodell**. Es gliedert sich in 4 Submodelle:
  - System-Entwicklung (SE)
  - Qualitätssicherung (QS)
  - Konfigurationsmanagement (KM) und
  - Projektmanagement (PM)

Für uns ist der Teil **Qualitätssicherung** wichtig**.**

5. Beispiel: Qualitätssicherung im V-Modell