

# Softwarequalität – 6 Hauptkriterien

*Qualität in der Softwareentwicklung ist messbar und definierbar. Diese sechs Hauptkriterien bilden das Fundament für nachhaltige, professionelle Software-Engineering-Prozesse und dienen als Orientierung für QA-Teams, Entwickler und technische Führungskräfte. Jedes Kriterium adressiert einen spezifischen Aspekt der Softwarequalität und erfordert unterschiedliche Prüfmethoden und Optimierungsstrategien.*

# Die sechs Qualitätskriterien im Überblick

## Funktionalität

*Liefert die Software die geforderten Funktionen korrekt und vollständig gemäß den Spezifikationen?*

## Zuverlässigkeit

*Läuft sie stabil, fehlerarm und reproduzierbar – auch unter Last und in Extremsituationen?*

## Benutzbarkeit

*Können Zielnutzer die Software intuitiv, schnell und sicher bedienen?*

## Effizienz

*Wie gut nutzt die Software verfügbare Ressourcen wie Zeit, CPU, RAM und Netzwerkbandbreite?*

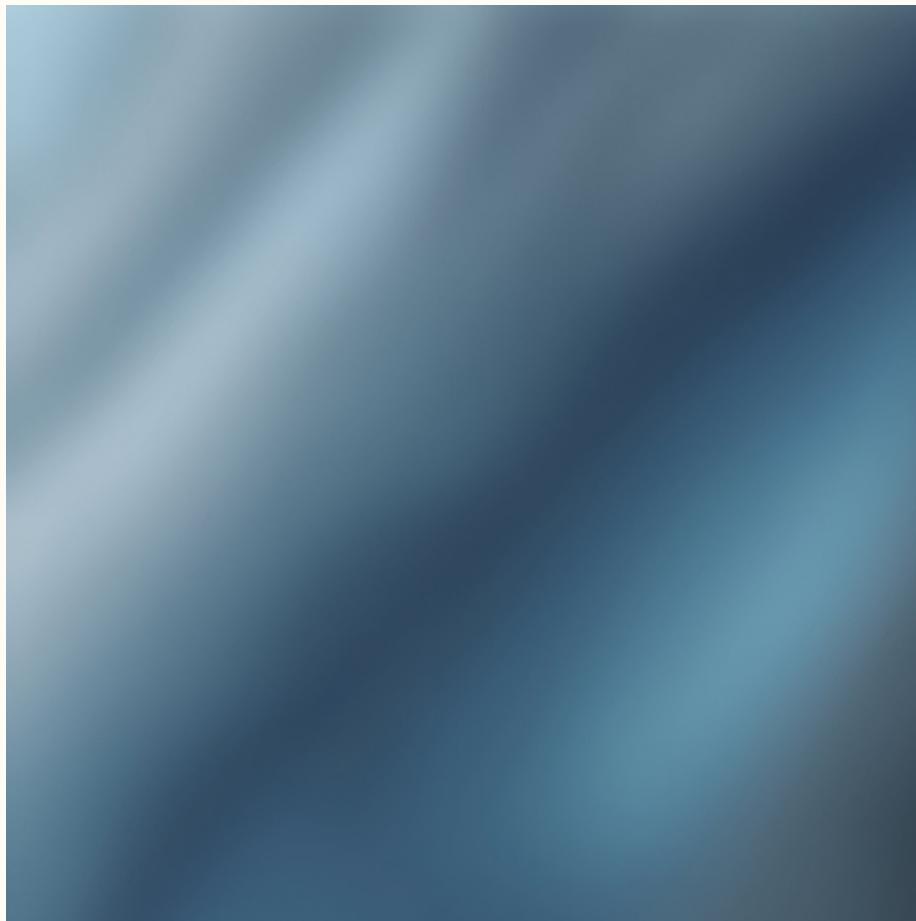
## Änderbarkeit

*Wie leicht können Fehler behoben, Features erweitert oder Anpassungen vorgenommen werden?*

## Übertragbarkeit

*Wie problemlos läuft die Software auf verschiedenen Hardware-Plattformen, Betriebssystemen und Umgebungen?*

## Quiz: Verständnis der Grundkriterien



Testen Sie Ihr Verständnis der sechs Qualitätskriterien mit diesen praxisnahen Fragestellungen:

1

Funktionalität ohne Absturz

Geben Sie ein Praxisbeispiel, wie ein Fehler in der Funktionalität aussieht – ohne dass die Applikation abstürzt.

2

Zuverlässigkeit messen

Woran würden Sie Zuverlässigkeit konkret messen oder nachweisen?

3

Benutzbarkeit steigern

Nennen Sie drei Maßnahmen, die Benutzbarkeit verbessern, ohne neue Funktionen zu entwickeln.

4

Effizienz-Relevanz

Wann ist Effizienz wichtig, auch wenn die Software grundsätzlich funktioniert?

# Funktionalität vs. Zuverlässigkeit vs. Benutzbarkeit

## Funktionalität

„Kann die Software X ausführen?“  
(und wird X korrekt durchgeführt?)

## Zuverlässigkeit

„Kann die Software X immer wieder ausführen, ohne Ausfälle oder Inkonsistenzen?“

## Benutzbarkeit

„Können Menschen X ohne Frust, Fehler oder umständliche Workarounds ausführen?“

## Typische Abgrenzungsfälle in der Praxis

**Prüfungsszenario:** Software erfüllt alle geforderten Funktionen technisch korrekt, aber die Bedienung führt regelmäßig zu Nutzerfehlern → Funktion erfüllt, Usability mangelhaft.

**Produktivszenario:** Seltene Crashes bei ungewöhnlichen Eingaben oder Randfällen → Funktion grundsätzlich erfüllt, aber Zuverlässigkeit unzureichend.

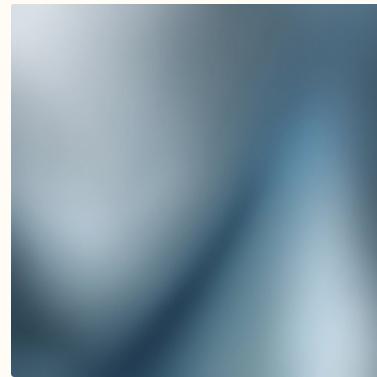
Diese Unterscheidung ist entscheidend für gezielte Qualitätsverbesserung und effektive Ressourcenallokation im QA-Prozess.

# Effizienz, Änderbarkeit, Übertragbarkeit – typische Hebel



## Effizienz optimieren

- *Algorithmen und Datenbank-Queries optimieren*
- *Intelligentes Caching implementieren*
- *I/O-Operationen minimieren*



## Änderbarkeit fördern

- *Klare Architektur mit geringer Kopplung*
- *Umfassende Test-Coverage*
- *Saubere, dokumentierte Schnittstellen*



## Übertragbarkeit sichern

- *Industriestandards konsequent nutzen*
- *Externe Abhängigkeiten reduzieren*
- *Konfiguration statt Hardcoding*

**Achtung Zielkonflikte:** Extreme Performance-Optimierung kann die Änderbarkeit verschlechtern. Übertragbarkeit erfordert manchmal Effizienz-Kompromisse. Bewusste Trade-off-Entscheidungen sind Teil professioneller Softwareentwicklung.

# Quiz: Vertiefung und praktische Anwendung

## 1 Zielkonflikt identifizieren

*Nennen Sie ein konkretes Beispiel für einen Zielkonflikt zwischen Effizienz und Änderbarkeit aus Ihrer Praxis.*

## 2 Übertragbarkeit steigern

*Welche Maßnahmen erhöhen die Übertragbarkeit, ohne die eigentliche Fachlogik anzufassen?*

## 3 Code-Qeruch erkennen

*Woran erkennen Sie im Code oder Projektstruktur, dass die Änderbarkeit wahrscheinlich problematisch ist?*

## 4 Effizienz-Probleme belegen

*Welche Kennzahlen oder Beobachtungen würden Sie nutzen, um Effizienz-Probleme objektiv zu belegen?*

