

Softwarearchitektur und Qualitätsszenarien

Mechanismen, um Qualitätsmerkmale zu erreichen

Burkhardt Renz

Fachbereich MNI
Technische Hochschule Mittelhessen

Wintersemester 2020/21

Übersicht

- Qualitätsmerkmale
 - Wichtige Qualitätsmerkmale von Software
 - Funktionalität, Qualität und Architektur
- Szenarien für Qualitätsmerkmale
- Mechanismen
- Fazit

Qualitätsmerkmale

Wichtige Qualitätsmerkmale von Software

Qualitätsmerkmale

Was ist Qualität?

Die Qualität eines Softwaresystems ist seine Eignung für den vorgesehenen Zweck (IEEE).

Qualitätsmerkmale?

- Funktionalität
- Benutzbarkeit (*usability*)
- Verfügbarkeit (*availability*)
- Leistungsfähigkeit (*performance*)
- Sicherheit (*security*)
- Änderbarkeit (*modifiability*)
- Testbarkeit (*testability*)

Qualitätsmerkmale

Funktionalität, Qualität und Architektur

Funktionalität und Architektur

- Funktionalität kann durch ganz unterschiedliche Architekturen erreicht werden (Beispiel KWIC)
- Architektonische Struktur aus dem Gesichtspunkt Funktionalität?
- Funktionalität erzwingt aber nicht bestimmte Qualitätsmerkmale
- Funktionalität und Architektur sind orthogonal

Qualitätsmerkmale und Architektur

- Architektur bestimmt das Erreichen gewünschter Qualitätsmerkmale
- Aber: Architektur alleine erzeugt die Qualität nicht – dazu muss Design und Implementierung entsprechend sein.

Beispiel Leistungsfähigkeit

bestimmt durch

- Art der Kommunikation zwischen Komponenten (Architektur)
- Nutzung von gemeinsamen Ressourcen (Architektur)
- Wahl von Algorithmen (Design)
- Codierung von Algorithmen (Implementierung)

Übersicht

- Qualitätsmerkmale
- Szenarien für Qualitätsmerkmale
 - Charakterisierung von Qualität
 - Qualitätsszenarien
- Mechanismen
- Fazit

Szenarien für Qualitätsmerkmale

Charakterisierung von Qualität

Charakterisierung von Qualität

Qualitätsmerkmale als solche sind keine geeignete Basis für den Entwurf einer Architektur:

- Änderbarkeit? Jedes System ist „änderbar“ – die Frage ist, inwiefern!
- Sicherheit? in Bezug worauf? in Bezug auf welche Gefahr!
- Verfügbarkeit? 24h jeden Tag? Zu Bürozeiten? 110 Jahre?

Ferner:

- Qualitätsmerkmale konkurrieren
- Qualitätsmerkmale schwer abgrenzbar

Szenarien für Qualitätsmerkmale

Qualitätsszenarien

Qualität „festnageln“

Qualitätsszenarien

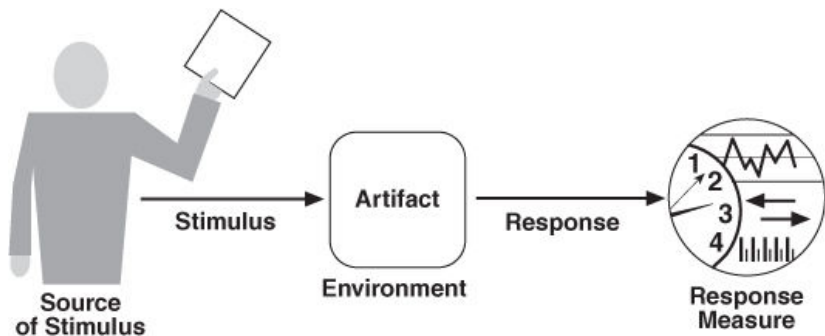
Charakterisierung von Qualitätsmerkmalen durch Szenarien, die beschreiben, wie sich das System bei bestimmten Stimuli verhalten soll.

Beispiel Änderbarkeit

Ein Szenario zum Thema Änderbarkeit könnte so aussehen

- Der Wunsch, die Benutzeroberfläche zu ändern tritt auf (Anwender soll verschiedene Farbschemata wählen können)
– Stimulus
- Der Wunsch kommt von der Kundenbetreuung
– Quelle des Stimulus
- Die Änderung wird während der Implementierung gemacht
– Kontext
- Die Änderung wird am Code durchgeführt – Artefakt
- Die Änderung soll keine Seiteneffekte haben – Antwort
- Die Änderung soll innerhalb von drei Tagen durchgeführt und getestet sein – Maß

Szenarien für Qualitätsmerkmale



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 4.4

Szenarien für Qualitätsmerkmale

Qualitätsanforderungen werden formuliert durch Szenarien mit

- Quelle des Stimulus
- Stimulus
- Kontext (*environment*)
- betroffenes Artefakt
- Antwort
- Maß für die Antwort

Für solche Qualitätsanforderungen gibt es **architektonische Mechanismen**, sie zu erreichen.

Architektonische Entscheidungen für Qualitätsmerkmale

Um Qualitätsmerkmale zu erreichen, wie sie in Qualitätsszenarien formuliert sind, werden architektonische Entscheidungen in folgenden Bereichen bedacht:

- 1 Verantwortlichkeit – welche Elemente sind betroffen?
- 2 Koordination – wie müssen sie zusammenarbeiten?
- 3 Datenmodell – welche Daten werden dafür benötigt?
- 4 Ressourcen – welche Infrastruktur braucht man?
- 5 Zuordnung von Komponenten – welche Aufgabe für welche Komponente?
- 6 Bindungszeit – zu welchem Zeitpunkt wird diese Zuordnung vorgenommen?
- 7 Technologie – welche Technologien setzt man ein?

Übersicht

- Qualitätsmerkmale
- Szenarien für Qualitätsmerkmale
- **Mechanismen**
 - ... für Benutzbarkeit
 - ... für Verfügbarkeit
 - ... für Leistungsfähigkeit
 - ... für Sicherheit
 - ... für Änderbarkeit
 - ... für Testbarkeit
- Fazit

Mechanismen

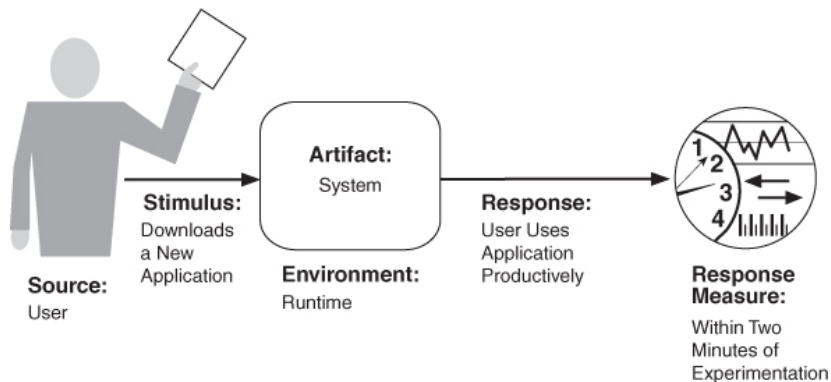
... für Benutzbarkeit

Benutzbarkeit

Benutzbarkeit bedeutet wie leicht ein Anwender eine beabsichtigte Lösung erreicht und wie gut ihn das System darin unterstützt.

- Angemessenheit
- Erlernbarkeit
- Ergonomie
- Fehlerverhalten
- Adaption an den Anwender

Qualitätsszenario für Benutzbarkeit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 11.1

Mechanismen für Benutzbarkeit

- Trenne Benutzeroberfläche von der Anwendung
- Unterstützung von Aktionen des Anwenders
 - Entscheidungsfreiheit: Abbrechen, Undo etc
 - Information: Wo bin ich? Wo war ich?
 - Konzeptionelle Integrität
- System denkt mit
 - Modell der Aufgabe
 - Modell des Anwenders
 - Modell des Systems

Mechanismen

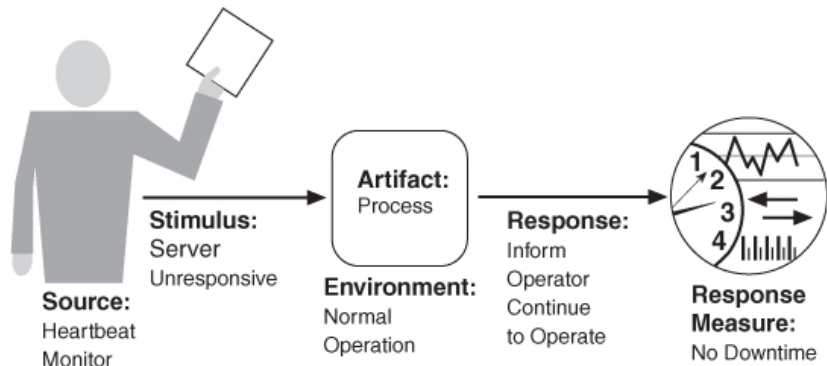
... für Verfügbarkeit

Verfügbarkeit

Verfügbarkeit betrifft die Frage, ob ein System ausfällt, und welche Konsequenzen Ausfälle haben.

- Wie wirkt sich ein Ausfall aus?
- Wie oft kann ein Ausfall auftreten?
- Wie lange dauert ein Ausfall?
- Wie werden Ausfälle entdeckt?
- Wie werden Ausfälle überwunden?

Qualitätsszenario für Verfügbarkeit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 5.1

Mechanismen für Verfügbarkeit

- Fehlererkennung
 - Ping/Echo
 - Wachsignal
 - Ausnahmebehandlung
- Reaktion auf Fehler
 - Redundante Komponenten
 - Recovery
 - Backup-System
- Fehlervermeidung
 - Antizipatorische Maßnahmen
 - Transaktionen
 - Monitoring

Mechanismen

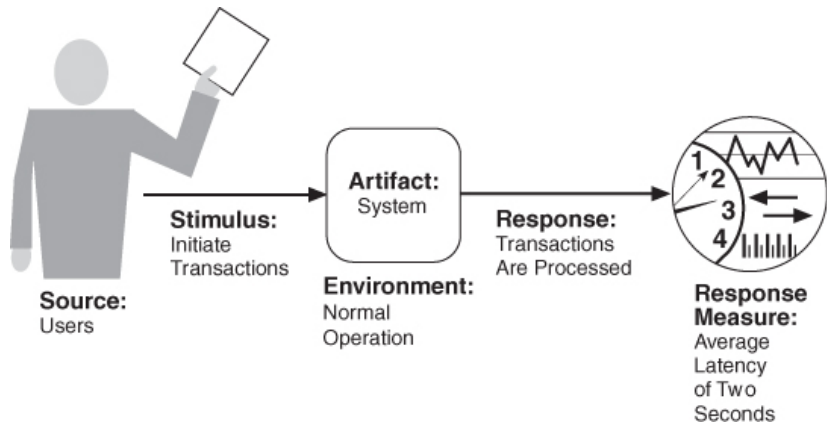
... für Leistungsfähigkeit

Leistungsfähigkeit

Leistungsfähigkeit betrifft Antwortzeit und Durchsatz eines Systems unter Last.

- Latenzzeit
- Antwortzeit
- Deadlines
- Durchsatz
- Priorisierung, Abweisen von Anfragen
- Datenverluste durch Überlast

Qualitätsszenario für Leistungsfähigkeit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 8.1

Mechanismen für Leistungsfähigkeit

- Ressourcenverbrauch
 - Verbesserung der Effizienz
 - Verminderung von Overhead
 - Verwalten von Frequenzen der Ereignisreaktion
- Ressourcenmanagement
 - Nebenläufigkeit
 - Redundanz
 - Skalierung
- Ressourcenzuteilung
 - Priorisierung
 - Warteschlangen

Mechanismen

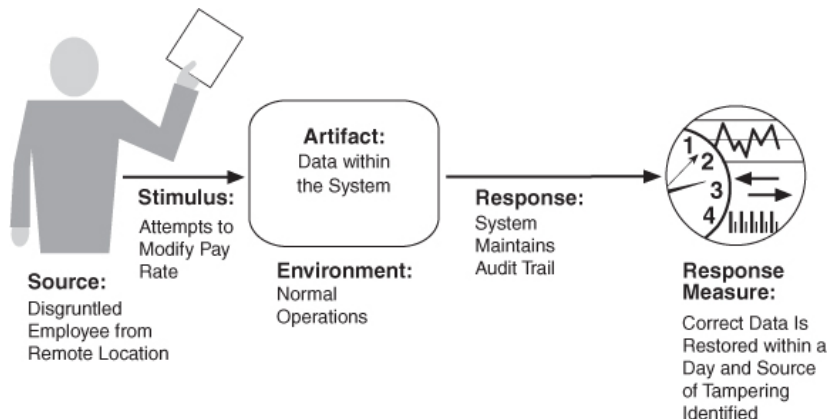
... für Sicherheit

Sicherheit

Sicherheit bedeutet die Fähigkeit unzulässige Zugriffe zu verwehren und zugleich die Leistungsfähigkeit des Systems zu erhalten

- Vertraulichkeit von Informationen
- Berechtigungen
- Integrität
- Nichtabstreitbarkeit
- Revisionsfähigkeit
- Verfügbarkeit

Qualitätsszenario für Sicherheit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 9.1

Mechanismen für Sicherheit

- Abwehr von Angriffen
 - Authentifizierung von Anwendern
 - Berechtigungskonzept
 - Datenverschlüsselung
 - Digitale Signaturen u.ä. Konzepte
 - Einschränkungen im Zugriff
 - Einschränkungen in der Offenlegung von Zugängen
- Erkennen von Angriffen
 - Vergleich signifikanter Systemlastparameter
 - Messen von Netzlasten
 - Digitale Signaturen für Softwarekomponenten
- Reaktion auf Angriffe
 - Wiederherstellen eines sicheren Zustands
 - Identifizieren von Angreifern
 - Spurverfolgung

Mechanismen

... für Änderbarkeit

Änderbarkeit

Änderbarkeit bedeutet, dass die Software verändert, erweitert, adaptiert, portiert und wiederverwendet werden kann.

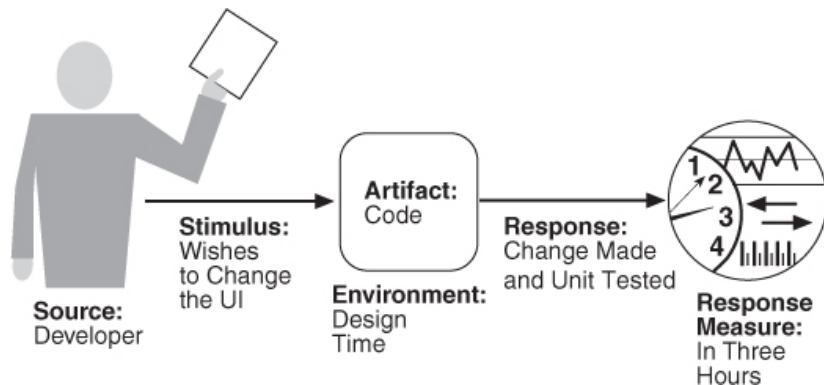
Aspekte der Veränderung

- Funktionen des Systems
- Infrastruktur, Plattform des Systems
- Qualitätsmerkmale

Zeitpunkt der Variation

- bei Entwurf, Codierung
- bei Installation (Konfiguration)
- zur Laufzeit

Qualitätsszenario für Änderbarkeit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 7.1

Mechanismen für Änderbarkeit

- Änderungen lokalisieren
 - Semantische Kohärenz von Komponenten
 - Antizipation von Änderungen
 - Adaptierbarkeit einplanen
 - Mögliche Optionen begrenzen
- Folgeeffekte vermeiden
 - Kapselung (*Information hiding*)
 - Schnittstelle von Implementierung trennen
 - Intermediär verwenden
- Festlegung hinausschieben
 - Dynamischer Polymorphismus
 - Konfiguration beim Systemstart
 - Adaptierbarkeit zur Laufzeit
 - Komponenten zur Laufzeit laden (Plugins)

Mechanismen

... für Testbarkeit

Testbarkeit

Testbarkeit betrifft die Frage, wie leicht Fehler an einer Software erkannt und lokalisiert werden können.

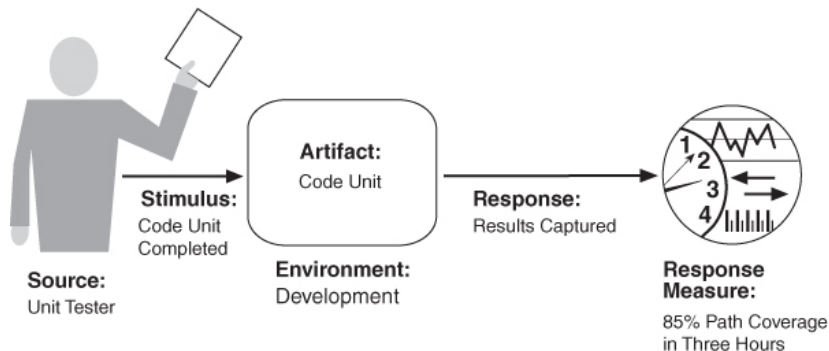
Was wird getestet?

- Einzelne Komponenten (*Unit Test*)
- Zusammenspiel von Komponenten (Integrationstest)
- Gesamtsystem (Systemtest)

Konzepte

- Steuerbarkeit
- Beobachtbarkeit

Qualitätsszenario für Testbarkeit – Beispiel



Quelle: Rick Kazman, Paul Clements, Len Bass *Software Architecture in Practice, Third Edition*, Kap. 10.1

Mechanismen für Testbarkeit

- Schnittstellentest
 - Testfälle abspielen
 - Implementierungen für Testzwecke ersetzen
 - Spezielle Testschnittstellen
 - Testgetriebene Vorgehensweise
- Interne Überwachung
 - Aufzeichnen des Kontrollflusses
 - Instrumentierung für Testzwecke
- Komplexität limitieren
 - Strukturelle Komplexität
 - Komplexität durch Nebenläufigkeit

Übersicht

- Qualitätsmerkmale
- Szenarien für Qualitätsmerkmale
- Mechanismen
- **Fazit**
 - Mechanismen, Stile & Muster





Fazit

Mechanismen, Stile & Muster

Mechanismen, Stile & Muster

- Mechanismus** Vorgehensweisen, um eine Architektur zu finden, die Qualitätsszenarien erfüllt
- Architekturstil** Paradigma der Struktur eines Systems: Elemente, Struktur und Interaktionsmechanismen.
- Architekturmuster** Problemorientierte beispielhafte Lösung für den Aufbau eines Systems: wie erreicht man gewünschte Eigenschaften

Literatur

-  Rick Kazman, Paul Clements, Len Bass
Software Architecture in Practice Part Two
Boston: Addison-Wesley, Third Edition 2012.
-  Cervantes Humberto, Rick Kazman
Designing Software Architectures: A Practical Approach
Boston: Addison-Wesley, 2016.
-  Ian Gorton
Essential Software Architecture Chap. 3
Harlow, UK: Addison-Wesley, 2000.
-  Jan Bosch
Design and Use of Software Architectures Chap. 4-6
Harlow, UK: Addison-Wesley, 2000.