Application Architecture - HS2021 joel.schaltegger@ost.ch

1 Einführung Softwarearchitektur

1.1 Was ist Softwarearchitektur?

Definiert die grundlegenden **Prinzipien und Regeln** für die Organisation eines Systems sowie dessen **Strukturierung in Bausteinen und Schnittstellen** und deren Beziehungen zueinander wie auch zur Umgebung. → Beschreibt grobgranulare Strukturen wie: Komponenten mit Schnittstellen, Assoziationen/Interaktionen, Prinzipien & Muster (Pattern)

1.1.1 Bausteine

Beschreiben statische Struktur des Systems. Bestehen aus:

- Pakete: Fassen untergeordnete Pakete bzw. Komponenten zu einer Einheit zusammen
- Komponenten: Klar definiertes Verhalten. Können Schnittstellen anbieten/konsumieren

1.1.2 Architektur vs Entwurf (Design)

Architektur:

- Legt grundlegende Strukturen fest
  - B sp.: Mobile Native-App mit REST-Zugriff auf Web-Server
- Darstellung oft mit UML Komponentendiagramm

Entwurf:

- Legt feingranulare Struktur fest
  - B sp.: Aufbau der Native-App: GUI – Funktionsebenen – Zugriff auf Server
- Darstellung oft mit UML-Klassendiagramm

1.1.3 Rolle von Softwarearchitekt

Arbeitet eng mit dem Requirements Engineering zusammen (Schwerpunkt auf NFA). Ist verantwortlich für:

- Grobgranularen Strukturen
- Querschnittskonzepte (z.B. Persistenz, Kommunikation, GUI)
- Erkennen/Aufzeigen von Konsequenzen bei Architekturentscheidungen

→ Erarbeitet/bearbeitet die Technical User Stories des Product-Backlog

1.2 Ziele von Softwarearchitektur

Ziel der Softwarearchitektur ist die Sicherstellung einer adäquaten Softwareproduktequalität.

- Nützlichkeit: Applikation erfüllt ihre Funktion
- Festigkeit: Software ist stabil und langlebig
- Ästhetik: User Experience ist ansprechend

1.2.1 ISO 25010 Softwareproduktequalitätsmodell

Benutzer:

- Funktionalität
- Performanz und Effizienz
- Kompatibilität und Interoperabilität
- Benutzbarkeit
- Verfügbarkeit

Betreiber & Entwickler:

- IT-Sicherheit
- Wartbarkeit
- Portierbarkeit

1.3 Modellierung

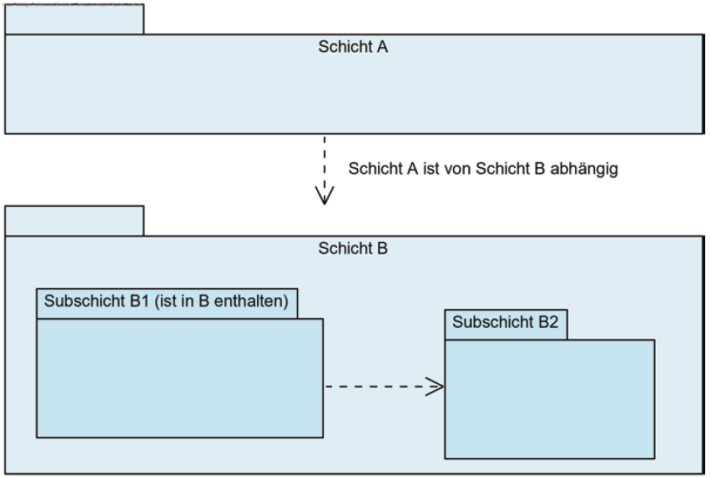
4 Sichten: Kontext, Baustein, Laufzeit und Verteilungssicht.

1.3.1 Kontextsicht

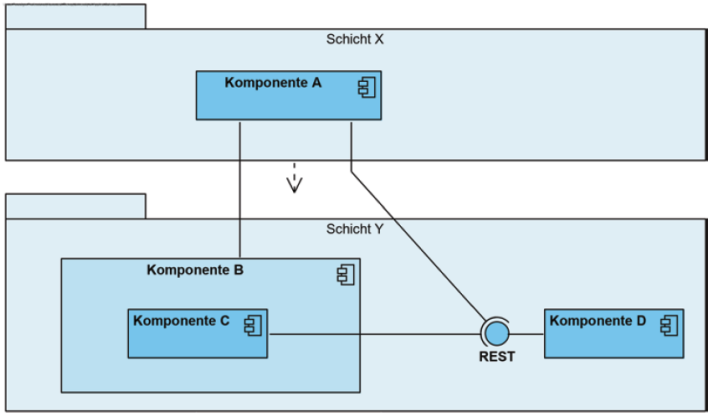
Das System steht im Zentrum als Black-Box. Alle Umsysteme und relevanten Stakeholder sind als externe Agenten dargestellt. Die Verbindungen von und zum System bilden Datenflüsse ab.
Diagramme: Kontext- oder UML-Diagramm

1.3.2 Bausteinsicht

Struktureller Aufbau des Systems
Paketdiagramm:



Komponentendiagramm:



1.3.3 Laufzeitsicht

Fokussiert auf die dynamischen Aspekte wie Prozessabläufe, Interaktionen zwischen Bausteinen und Verhalten.
Diagramme: Sequenz-, Aktivitäts- und Zustands-Diagramm

1.3.4 Verteilungssicht

Ermöglicht die Abbildung von Hardware-Devices, Netzwerkverbindungen und Laufzeitumgebungen. **Darstellen:** Komponenten, welche in einer spezifischen Laufzeitumgebung installiert werden.

1.3.5 Werkzeuge

Textbasierte Dokumentation: Wiki für kollaborales Erarbeiten der Inhalte, plus Versionierung. Bsp: Confluence
Modellierungswerkzeuge: Unterstützung von Diagrammsprachen wie UML, SysMl. Bsp: Visual Paradigm
Code-Analysetool: Statische Analyse und Bewertung der technischen Qualität von Sourcecode. Bsp: SonarQube

2 Week01

3 Week01

4 Week04

5 Week05

6 Week06

7 Week07

8 Week08

9 Week09

10 Week10

11 Week11

12 Week12

13 Week13