Hausaufgabe

- Bitte stellen Sie Ihre Lösungen vor

- Prozedural:
- Funktional:
- Objekt-orientiert:
- Pipe-and-Filter:

Funktional

Mögliche Lösung in Haskell:

```
import Data.List


main = do
  file <- readFile "kwic.txt"
  mapM_ putStrLn (sort $ map (intercalate " ") (concat $ map rot  (map words (lines file))))

rotate 0 _ = []
rotate n (x:xs) = (xs ++ [x]) : rotate (n-1) (xs++[x])

rot l = rotate (Data.List.length l) l
```

Vor- und Nachteile

- Diskutieren Sie bitte in Breakout-Gruppen die Vor- und Nachteile der 4 Architekturen

- Prozedural
  + : Effiziente Datennutzung, Funktionalität kann in Prozeduren verteilt werden
  - : Ändert sich die Datenstruktur, ändern sich vermutlich alle Funktionen
- Funktional
  + : Kurze Lösung, änderbar durch neue Funktionen
  - : Daten werden oft dupliziert
- Objekt-orientiert
  + : Leicht änderbar; Verarbeitung kann zeilenweise geschehen (bis auf Sortierung!)
  - : Schwieriger testbar. Parallelisierung mit Bordmitteln schwierig
- Pipe-and-Filter
  + : Verständlich: Kontrollfluss = Datenfluss, Filter parallelisierbar (bis auf Sortierung!)
  - : Boilerplate Code für Pipes notwendig.

Architectural Styles

Dom zu Worms
(Romanik)

Notre-Dame
(Gotik)

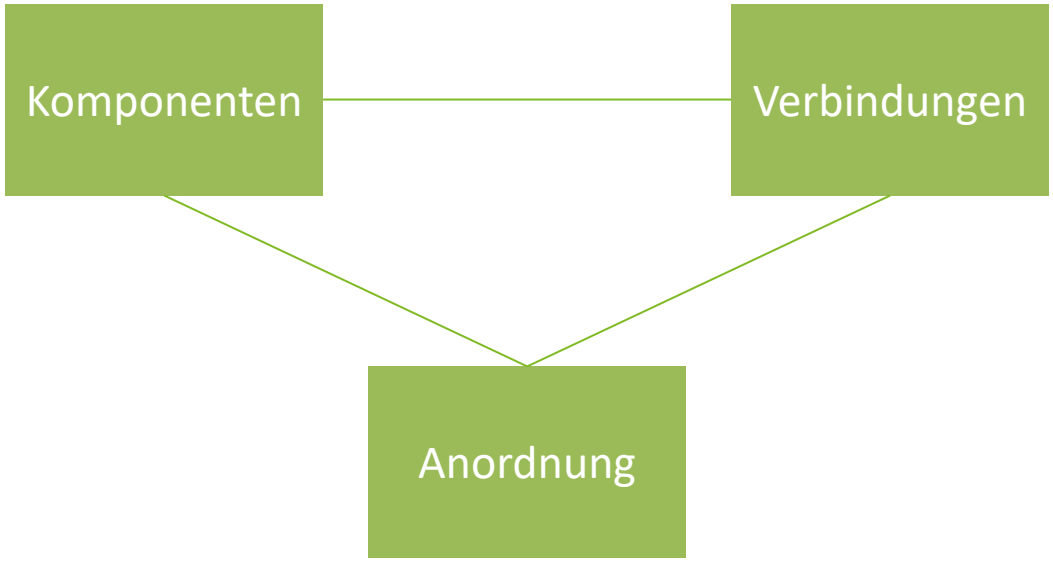
Dom zu Florenz
(Renaissance)

Chrysler Building
(Art Deco)

Bauhaus Dessau

Nationaltheater Weimar
(Neoklassizismus)

Was macht die Stile aus?



Beispiel - Gothik

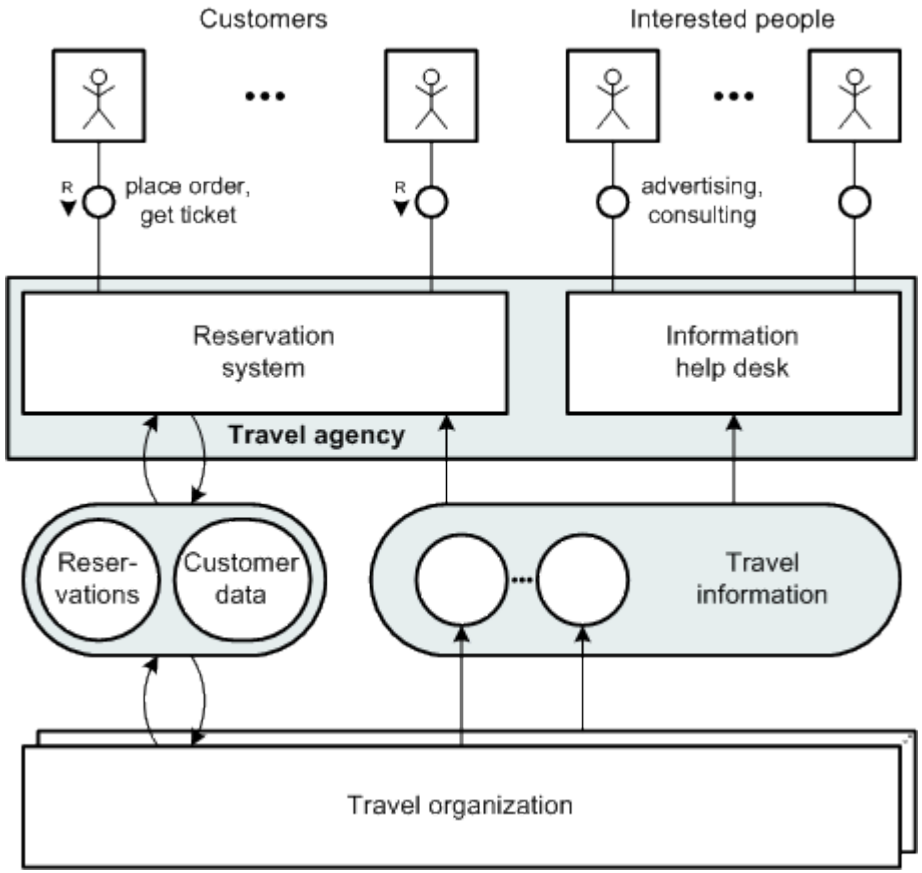
„Offene“ Fassade durch Fenster.

Verbindungen: Rippengewölbe, Spitzbogen, Zug- und Ringanker

Anordnung: Betonung der Vertikalen.

Bildquelle: https://commons.wikimedia.org/wiki/File:Cath%C3%A9drale_Notre-Dame_de_Paris,_20_March_2014.jpg

Beispiel - Software



Frage: Was sind die Komponenten und Verbindungen?

Komponenten: Akteure, Module, Speicher, …

Verbindungen: Schnittstellen

Anordnung: 4 Schichten

Quelle: http://www.fmc-modeling.org/quick-intro

Frage

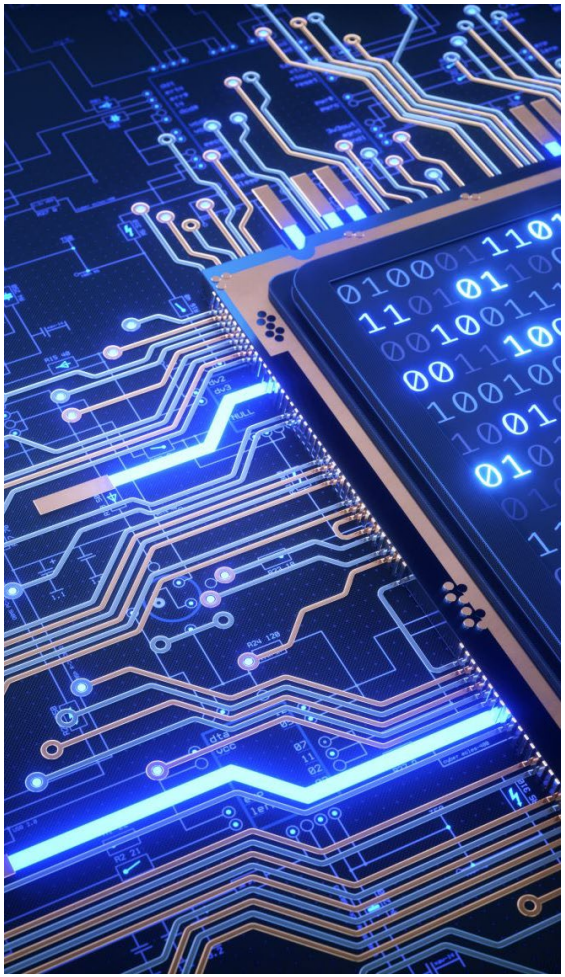
Welche Software-Architekturstile kennen Sie?

distributed: service-oriented, microsvcs, eventbasiert, peer-to-peer, client-server,

monolithisch: layered, pipe-and-filter, microkernel

Pattern:

mvc, mvvm, singleton, factory, …



Definition „Architektur“

“Software architecture is the fundamental **organization** of a system, embodied in its **components**, their **relationships** to one another and the environment, and the **principles** governing its design and evolution.”

-- IEEE 1471

Komponenten: Orte, an denen Daten gespeichert oder verarbeitet werden.

Verbindungen: Aufrufe, Filter, Queues, …

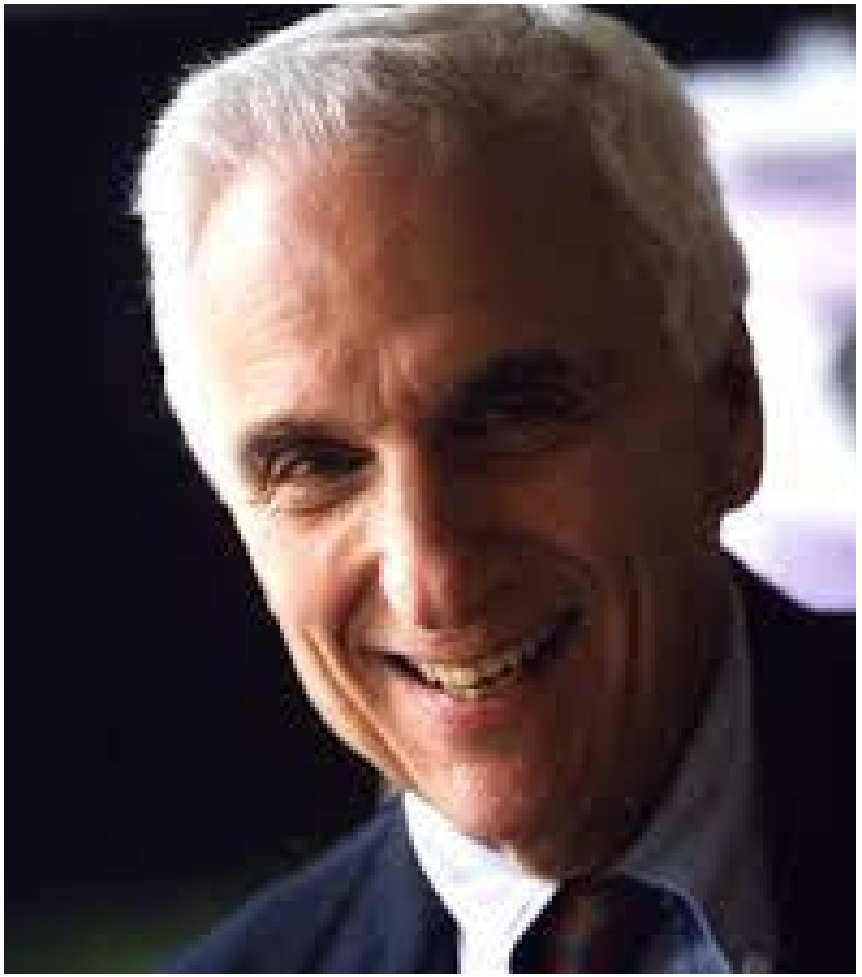
Prinzipien: Wie passt das hier rein?

Conway‘s Law

“Organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.”

Melvin E. Conway, 1967

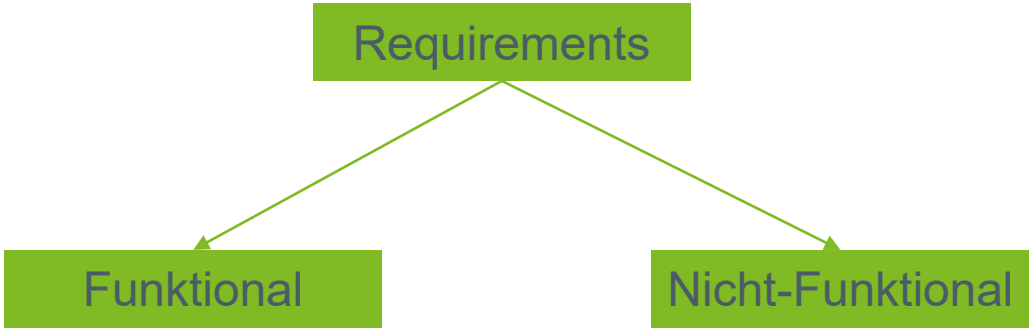
Frage: Welche Auswirkungen hat Conway‘s Law auf Softwarearchitektur?



Überblick

1. Was ist Architektur?
2. Anforderungen & -ilities
3. Viewpoints
4. Architektur Prozess
5. Architekturmuster (inkl. SOLID)
6. Architekturprinzipien
7. Architekturstile
8. Die Rolle des Architekten
9. Governance und Architekturevaluation

Requirements und Architektur



Keine direkten Anforderungen an die Architektur.

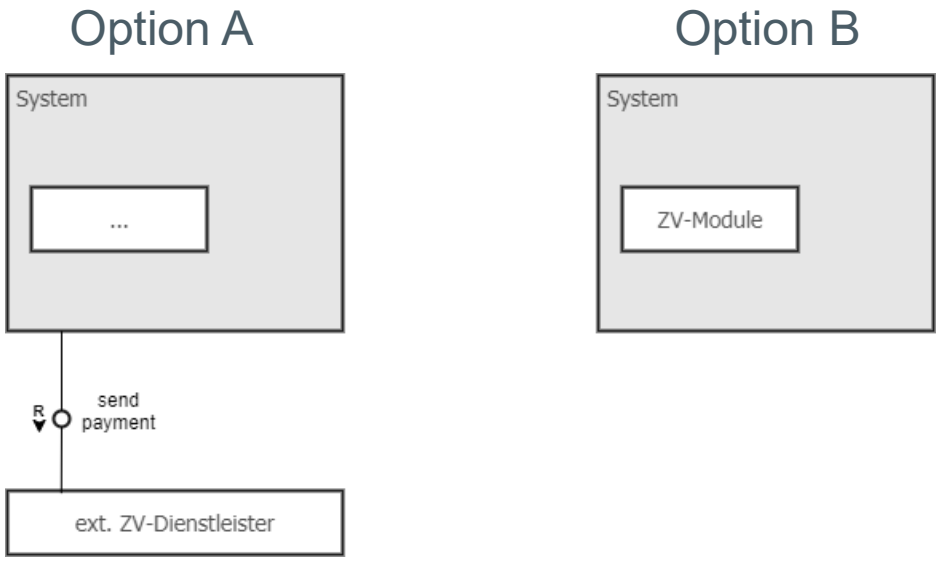
Achtung: Keine „Anweisungen“ bzgl. Architektur in den funktionalen Requirements.

Diese werden durch die **architektonischen Eigenschaften** der Lösung umgesetzt.

Trotzdem beeinflussen die funktionalen Rqmnts die Architektur

Beispiel

R47: Das System muss ausgehende Zahlungsanweisungen versenden können.



Welche Anforderungen an die Architektur ergeben sich bei A oder B?
A: Die Verbindung muss gesichert werden (Verschlüsselung, …)
B: Der Zugang zu dem ZV-Modul muss gesichert werden (Autorisierung, Logging, …)