

# Softwarearchitektur und Anwendungsentwicklung

## Was ist Softwarearchitektur?

Prof. Dr. Burkhardt Renz

Fachbereich MNI  
Fachhochschule Gießen-Friedberg

Sommersemester 2007



„Architektur ist die Kombination von *utilitas*, *firmitas* und *venustas*.“

frei nach Vitruvius (Römischer Architekt 90-20 v. Chr.)

Softwarearchitektur will erreichen, dass das Anwendungssystem die Anforderungen erfüllt (utilitas), robust ist gegenüber Änderungen (firmitas) und eine gewisse Schönheit hat (venustas).

—

Fragt sich nur: wie?!

- Was ist Softwarearchitektur?
  - Definition von Softwarearchitektur
  - Diskussion der Definition
  - Wozu Softwarearchitektur?
  
- Darstellung der Softwarearchitektur
  - Sichten
  - Notationen

## Was ist Softwarearchitektur? – Definition

*The software architecture of a program or computing system is the structure or the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.*

— Len Bass et al.

# Elemente der Definition

## Elemente der Softwarearchitektur

- Elemente, *Komponenten* – definieren den Ort von Berechnungen oder auch Speicher, z.B. Filter, Datenbanken, Objekte, Server, Klienten usw.
- Beziehungen, *Konnektoren* – definieren die Interaktion der Komponenten, z.B. Funktionsaufruf, Pipe, Event usw.
- Eigenschaften – definieren Vorgaben und Beschränkungen für Komponenten, z.B. Schnittstellen, Qualitätsmerkmale usw.

## Diskussion

- Abstraktion: *Wesentliche* Elemente und ihre Beziehungen
- Struktur: Zerlegung und Zusammenbau
- Strukturen: Verschiedene Sichten auf das Wesentliche des Systems
- Beziehungen: Organisierende Prinzipien des Zusammenwirken der Elemente
- Plan: *Bauplan* im Unterschied zu Dokumentation und Projektplan

## Welche Strukturen?

Software ist die Beschreibung des gewollten Verhaltens einer universellen, abstrakten Maschine (Computer).

Beschreibung – Beschriebenes – Umgebung

- 1 Struktur der Beschreibung: Codestruktur
- 2 Struktur der Umgebung: Infrastruktur
- 3 Struktur des Beschriebenen: konzeptionelle Struktur des (laufenden) Systems

## Aufgabe und Ziele

- Mittel der Kommunikation und Diskussion aller Interessengruppen (*Stakeholder*)
- Festlegung der ersten und grundlegenden Entscheidungen für Design und Implementierung (*Constraints*)
- Grundlage für das Erreichen der funktionalen und qualitativen Merkmale des zu konstruierenden Systems
- Architektur fungiert als „mentaler Prototyp“
- Architektur(-stile, -muster) sind wiederverwendbar, auf andere Systeme übertragbar

## Erwartungen an eine Softwarearchitektur

- Klare Entwurfsabsichten, damit die intendierte Architektur erhalten und verständlich bleibt
- Designzentren festlegen
- Basis für Design, damit die wesentlichen Fragen gestellt werden
- Verbesserung der Änderbarkeit, indem die Designzentren erkennbar bleiben und erhalten werden

# Grundlegende Entscheidungen in der Architektur

- Funktionalität** Fähigkeit des Systems, die intendierte Aufgabe zu erfüllen (kann durch verschiedene Architekturen erreicht werden)
- Qualitätsmerkmale** Insbesondere nicht-funktionale Merkmale, die in hohem Maße von der Architektur beeinflusst werden
- Organisation** Architektur bestimmt auch die Organisation des Entwicklungsprozesses
- Eckpfeiler** Architektur legt den Rahmen fest, in dem sich Design und Implementierung bewegen (z.B. Framework)

## Fazit: Was ist Softwarearchitektur?

*To be architectural is to be the most abstract depiction of the system that enables reasoning about critical requirements and constrains all subsequent refinements*  
— Mark Klein

*A mental prototype is a model of the system which outlines a potential system solution. It describes functionality, but leaves open most of the implementation details.*  
— Andreas Knöpfel, Bernhard Gröne, Peter Tabeling

## Sichten der Softwarearchitektur

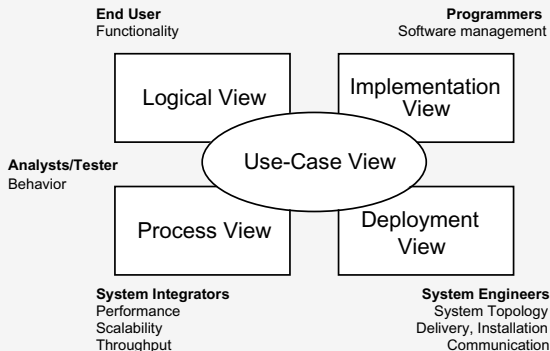
Das Wesentliche der Architektur lässt sich in der Regel nicht in *einer* Sicht allein darstellen. Man unterscheidet deshalb verschiedene Sichten.

Im konkreten Fall wählt man die „passenden“ Sichten, um die Architektur darzustellen.

Es gibt verschiedene Methoden, Softwarearchitektur darzustellen:

- Das 4+1-Sichten-Modell von Phillippe Kruchten, verwendet im (Rational) Unified Process
- 4 Sichten von Hofmeister, Nord und Soni
- Viewtypes und Styles des SEI
- FMC Fundamental Modeling Concepts, gelehrt am Hasso-Plattner-Institut

# Kruchten & UML



The "4+1" view model (Phillipe Kruchten 1995)

## Charakteristik der 4+1-Sichten

Für jede Sicht wird angegeben:

**I** Inhalt, **K** Komponenten, **B** Beziehungen und **S** Stakeholder.

- Use Case Sicht
  - I: Verhalten des Systems
  - K: Akteure, Anwendungsfälle
  - B: Interaktion, Verwendung, Vererbung
  - S: Anwender, Analytiker, Tester
- Logische Sicht
  - I: Vokabular des Gebietes, Funktionalität
  - K: Klassen, Verantwortlichkeiten, Kollaborationen
  - B: Assoziation, Vererbung, Abhängigkeit, Steuerung
  - S: Anwender, Analytiker, Designer, Bereichsexperte

## Charakteristik der 4+1-Sichten

- Prozess-Sicht
  - I: Performanz, Skalierbarkeit, Verfügbarkeit
  - K: Prozesse, Threads
  - B: Aktivierungssteuerung, (gemeinsame) Ressourcen
  - S: Designer, Deployer
- Implementierungs-Sicht
  - I: Systembestandteile, Konfigurationsmanagement
  - K: Dateien, Repositories
  - B: Enthaltensein, Abhängigkeit
  - S: Designer, Entwickler, Konfigurationsmanager
- Physische Sicht
  - I: Hardwaretopologie
  - K: Hardwareresourcen
  - B: Kommunikationskanäle, Abhängigkeit
  - S: Hardwareingenieur, Deployer.

## Hofmeister, Nord und Soni

- Konzeptionelle Sicht** beschreibt Komponenten und Konnektoren und wie sie zusammenarbeiten
- Modulsicht** beschreibt Subsysteme, bestehend aus Modulen mit ihren Schnittstellen, eventuell angeordnet in Schichten
- Ausführungssicht** beschreibt Ausführungseinheiten (z.B. Prozesse), die auf einer bestimmten Plattform laufen und kommunizieren
- Codesicht** beschreibt Quelldateien, binäre Komponenten, Bibliotheken, ausführbare Programme und weitere Dateien

# Viewtypes des SEI

## Viewtype = Perspektive auf das System

A viewtype defines the element types and relationship types used to describe the architecture of a software system from a particular perspective

## Drei Viewtypes

- 1 Module viewtype
- 2 Component-and-connector viewtype
- 3 Allocation viewtype

# Styles

## Style = Spezielle Ausprägung/Muster der Perspektive

A style guide is the description of an architectural style that specifies the vocabulary of design (set of element and relationship types) and the rules (sets of topological and semantic constraints) for how that vocabulary can be used.

## Beispiel

Pipes & Filters ist ein Style des Component-and-Connector Viewtypes. (Wir werden weitere Architekturstile kennenlernen.)

## Fundamental Modeling Concepts

Ausgehend von einer Klassifikation dynamischer Systeme schlägt Siegfried Wendt eine Darstellung der fundamentalen Strukturen eines Softwaresystems vor, die die grundlegende konzeptionelle Architektur des Systems verständlich machen

Drei fundamentale Strukturen in informationellen dynamischen Systemen:

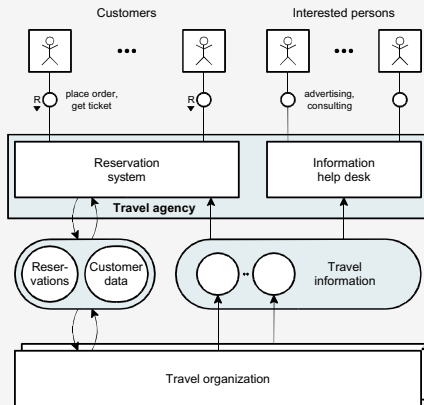
- Aufbaustrukturen
- Wertebereichsstrukturen
- Ablaufstrukturen

# Notation von FMC

## Bipartite Graphen



- Aufbaustrukturen bestehen aus
  - aktiven, verarbeitenden Komponenten (Agenten)
  - passiven, datenhaltenden Komponenten (Kanälen und Speichern)
  - Struktur: Agenten verarbeiten Daten, Ergebnisse werden an Kanälen oder in Speichern beobachtbar
- Darstellung der Wertebereichsstrukturen durch Mengen und Relationen, optisch ähnlich den Venn-Diagrammen, inhaltlich Entity-Relationship-Modelle
- Ablaufstrukturen durch Petri-Netze, genauer Stellen-Transitions-Netze

## Beispiel Aufbaustruktur in FMC



© FMC Research Staff – <http://fmc.hpi.uni-potsdam.de>

# Literatur

-  **Peter Tabeling**  
Softwaresysteme und ihre Modellierung Kap. 3, 12  
Berlin: Springer, 2006.
-  **Jochen Ludewig, Horst Lichter**  
Software Engineering Kap. 17  
Heidelberg: dpunkt.Verlag, 2007.