# Architekturmodellierung: Die Rolle der Architektur im Entwicklungsprozess

Modellbildung in der Entwicklung

Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München

SS 2007



## Inhalte

Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
3. Anwendung
   1. Ontologien
   2. Anforderungsanalyse
   3. Architekturmodellierung
      1. Grundlagen
      2. Modelle und Sichten
      3. Beispielanwendung
   4. Qualitätssicherung
   5. Modellqualität
4. Modellgetriebene Entwicklung
5. Domänenspezifische Modellierung



## Grundlagen: Definition

> *Architecture is defined [...] as the **fundamental organization** of a system, embodied in its **components**, their **relationships** to each other and the environment, and the **principles governing ist design and evolution**.*
>
> IEEE Std. 1471-2000

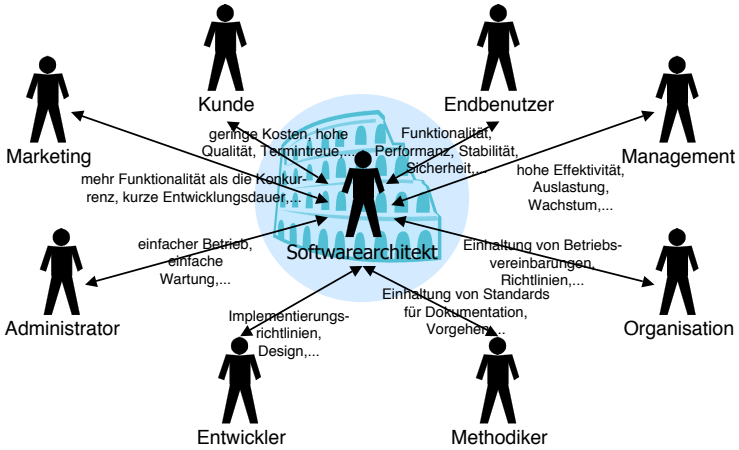
- Softwarearchitektur
  - unterteilt das Anwendungssystem in Bestandteile mit definierten Verantwortlichkeiten und einem festgelegten Zusammenspiel
  - ermöglicht die Beurteilung der Qualität des Gesamtsystems anhand der Qualitäten der Einzelteile
  - definiert Schnittstellen für zukünftige Erweiterungen
- Die Softwarearchitektur beeinflusst maßgeblich Entwicklungs- und Weiterentwicklungsprozesse



## Grundlagen: Ziel des Architekturentwurfs

- Ziel des Architekturentwurfs ist:
  - Den Bauplan des Systems zu erstellen, nach dem sich dann Erstellung, Integration, Wartung, Pflege und Weiterentwicklung ausrichten
  - Eine vollständige und prägnante Architekturbeschreibung erstellen
  - Dabei die geforderten funktionalen und nicht funktionalen Anforderungen gewährleisten
  - Eine Zerlegung des Systems in handhabbare, in sich abgeschlossene, unabhängig entwickelbare, wiederverwendbare Einheiten
- Die Architekturbeschreibung dient als
  - Kommunikations- und Diskussionsplattform
  - Design- und Implementierungsplan
  - Grundlage für Projektplanung und Überwachung

# Grundlagen: Spannungsfeld Architektur






# Modelle und Softwarearchitektur

- Ein Architekturmodell besteht aus:
  - Menge von elementaren Komponenten
  - Export- und Import-Schnittstellen der Komponenten
  - Komposition der Bausteine zu einer geeigneten Softwarearchitektur
- Ein Architekturmodell soll präzise und vollständig sein:
  - Interaktion und Kommunikation
  - Verbindungen und Strukturen
  - Zustandsraum und Datenstrukturen
- Trotzdem innere Struktur der Bausteine verbergen!



# Sichten und Modelle

> A view is a representation of a whole system from the perspective of a related set of concerns
>
> IEEE STD 1471-2000

Sicht: Beschreibung des Systems aus einem spezifischen Blickwinkel zur

- Reduktion der Komplexität der Beschreibung des Gesamtsystems
- Spezialisierung für unterschiedliche Zielgruppen (z.B. Nutzer, Kunde, Entwickler, Tester)
- Fokusierung auf spezifische Aufgaben (z.B. Entwurf, Implementierung, Verteilung, Qualitätssicherung)



# Sichten im Architekturentwurf

Aufgabenspezifische Trennung von Sichten:

- Orientiert am Entwicklungsprozess: Spezifikationssichten und Erstellungssichten
- Spezifikationssichten (Analyse und Entwurf eines Systems)
  - Fachliche Sicht
  - Technische Sicht
  - Verteilungssicht
- Erstellungssichten (Entwicklung und Deployment des Systems)
  - Entwicklungssicht
  - Testsicht
  - Bereitstellungssicht

## Fachliche Sicht

- Zweck
  - Darstellung der Konzepte des Anwendungsgebiets möglichst unabhängig von einer softwaretechnischen oder plattform-spezifischen Lösung
- Inhalte
  - Fachliche Komponenten inklusive Schnittstellen und Verhalten
  - Beziehungen zwischen fachlichen Komponenten
  - Systemgrenze und Anbindung existierender fachlicher Komponenten
- Beschreibung
  - Überblick durch CRC-Karten, Box-and-Line-Diagramme
  - Objektorientierte Modellierungstechniken der UML
  - in Spezialfällen formale Techniken (OCL, ADLs, etc.)



## Technische Sicht

- Zweck
  - Darstellung der softwaretechnischen Lösung für das System inklusive der Umsetzung der fachlichen Sicht
- Inhalte
  - Technische Komponenten des Systems und deren Schnittstellen
  - Beziehungen zwischen den Komponenten, inklusive der Abbildung der fachlichen Komponenten in die technische Sicht
  - Beziehung zu Basissoftware und Abbildung auf die Hardware
- Beschreibung
  - Überblick möglich durch Box-and-Line-Diagramme
  - Komponentendiagramm der UML
  - Interface Definition Languages (IDLs)
  - Quellcode-Fragmente für Details von Mechanismen
  - in Spezialfällen formale Techniken (OCL, ADLs, etc.)



## Verteilungssicht

- Zweck
  - Darstellung der Verteilung der Komponenten zur Ausführungszeit auf die Prozessoren und Rechner
- Inhalte
  - Abbildung der Komponenten des Systems auf die Elemente der Systemarchitektur zur Laufzeit
  - Darstellung des Client-/Server-Schnitts
  - Darstellung der ausführungsnahen Qualitätseigenschaften, wie zum Beispiel Performance
- Beschreibungen
  - Textuelle Beschreibung und informelle Box-and-Line-Diagramme
  - Verteilungsdiagramm der UML



## Erstellungssichten

- Entwicklungsicht
  - Darstellung der erstellten Beschreibungen des Systems und der Vorgänge bei Entwicklung und Integration des Systems
  - Beschreibungen zum System, insbesondere die unterschiedlichen Modelle und der Quellcode
  - Entwicklungsumgebung und zugehörige Abläufe
- Testsicht
  - Entwurf und Realisierung einer umfassenden Test-Suite für das System und seine Komponenten
  - Entwurf einer Testumgebung mit aufeinander abgestimmten Testwerkzeugen und der dazugehörigen Testprozesse
- Bereitsstellungssicht
  - Darstellung der auszuliefernden Bestandteile des Systems
  - ausführbare Beschreibungen des Systems (insbesondere Binärcode)
  - Ablage der Beschreibungen auf Hardware-Komponente
  - Prozesse bei der Konfiguration und Installation

## Beispielanwendung: Workflowsystem

Fachliche Sicht: Branchenspezifische fachliche Komponenten

- Daten: Fachliche Objekte des Anwendungsgebietes
  z.B. Klassendiagramm mit Bearbeiter, Vorgang, Arbeitsschritt
- Funktionen: Fachliche Verwendung der Objekte
  z.B. Zustandsdiagramme von Anlegen, Löschen, Bearbeiten
- Prozesse: Fachliche Vorgänge des Anwendungsgebietes
  z.B. Aktivitätsdiagramme von Planung, Abwicklung, Prüfung
- Aktivitäten: Fachliche Vorgangsschritte
  z.B. Sequenzdiagramm von Datenerfassung, Vorgangsauswertung, Schrittfreigabe



## Beispielanwendung: Workflowsystem

Technische Sicht: Domänenspezifische Referenzarchitektur (4-Schichten-Architektur)

- Datenhaltungsschicht: Steuert die dauerhafte Haltung von Informationen des Anwendungssystems
- Anwendungsschicht: Setzt die einzelnen Funktionen des Anwendungssystems um
- Steuerungsschicht: Koordiniert die Abläufe des Anwendungssystems
- Präsentationsschicht: Organisiert die Nutzungsein- und Ausgaben des Anwendungssystems



## Beispielanwendung: Workflowsystem

Kopplung: Fachliche Sicht und technischer Sicht

- Aktivitäten: Präsentationsschicht
  z.B. Abbildung von Aktivitäten auf Benutzerinteraktionen
- Prozesse: Steuerungsschicht
  z.B. Abbildung von Ereignisse auf Methodenaufrufe
- Funktionen: Anwendungsschicht
  z.B. Abbildung von Operationen auf Objektmethoden
- Daten: Datenhaltungsschicht
  z.B. Abbildung von Klassenstrukturen auf Datenbankschemata

Häufige Variante: Kopplung von Steuerungs- und Anwendungsschicht



## Beispielanwendung: Workflowsystem

Verteilungssicht: Trennung von Schichten

- Horizontale Trennung zwischen Schichten
  - Datenhaltung + Anwendung vs. Präsentation: Thin Client
  - Datenhaltung vs. Anwendung + Präsentation: Fat Client
- Horizontale Trennung innerhalb von Schichten:
  - Verteilte Präsentationsschicht: Ultra-Thin Client
  - Verteilte Anwendungsschicht: Funktionale Stubs
  - Verteilte Datenhaltungsschicht: Datenbank-Cache
- Vertikale Trennung von Schichten: Replikationen

## Architektur und Qualität

Die Architektur bestimmt wesentlich die Qualität eines Systems

- Zuverlässigkeit
- Sicherheit (Security)
- Performanz
  - Antwortzeiten
  - Ressourcenausnutzung
  - Verhalten unter Last
- Wartbarkeit
  - Portierbarkeit
  - Änderbarkeit
- Wiederverwendbarkeit



## Model-Driven Architecture (MDA™) der OMG

The MDA approach

- defines system functionality using a platform-independent model (PIM) using an appropriate Domain Specific Language.
- Given a Platform Definition Model (PDM) corresponding to CORBA, DotNet, the Web, etc., the PIM is translated to one or more
- platform-specific models (PSMs) that computers can run, using different Domain Specific Languages, or a General Purpose Language like Java, C#, Python, etc.
- Automated tools generally perform these translations, for example tools compliant to the new OMG standard named QVT (QVT (Queries/Views/Transformations) is a standard for model transformation).



## Umfassende Architektur

- Kundenfunktionsordnungsstruktur: Nutzungsebene - Kundenfunktionen (KF)
  - Multi-funktionale Systeme (K.Funktionshierarchien - feature hierarchies)
  - Abhängigkeiten/Beziehungen zwischen KFs: Feature interaction
- Logische Systemarchitektur (Funktionsnetz)
  - Hierarchische Zerlegung des Systems in logische Komponenten

Konzeptionelle Architektur

- Software-Architektur
  - Design Time Software Architektur
    - Applikationssoftware
    - Software Plattform (OSEK, Treiber für Bussysteme)
  - Laufzeit Software-Architektur
    - Tasks
    - Scheduling
- Hardware Architektur
  - Steuergeräte
  - Kommunikationseinrichtungen
  - Sensoren und Aktuatoren
- Deployment

Technische Architektur



## Architekturmodelle: Aspekte

- Modelltheorie/Sichten
  - Data
  - Struktur (Architektur)
  - Schnittstellen
  - Zustand
  - Prozess
- Formalisierungsgrad
- Abstraktionsgrad
  - Logische vs. Realisierungssicht
    - Logische Sicht: Logisches Ereignis (Input: Fenster Schließen)
    - Technische Sicht: Technisches Ereignis (Fensterheberknopf spricht an)
  - Black Box vs. Glass Box Sicht

## Was ist eine Funktion?

Wir unterscheiden

- Kundenfunktion
- Bordnetzfunktion

Ein System bietet Funktionen an!



## Nutzungssicht: Systemschnittstellenmodel

- Beschreibt Systeme durch ihre Funktionalität über die Systemgrenzen
  - Syntaktische Schnittstelle
    - Logische Kanäle
    - Typen von Nachrichten
  - Semantische „Verhalten“-Schnittstelle
    - Diskrete Zeit - Modellierung von Zeitverhalten
    - Kausalität modelliert Zeitfluss
    - Verhalten als Relation zwischen Ein- und Ausgabe in Zeitrahmen
  - Verhalten kann nichtdeterministisch/unterspezifiziert sein
- Logischer Rahmen durch Systemzusicherungen
- Technischere Sicht
  - System Scope: Systemgrenzen (Aktuatoren/Sensoren oder Systemreaktionen)
  - Welche Signale
  - MMI
  - Sensoren/Aktuatoren



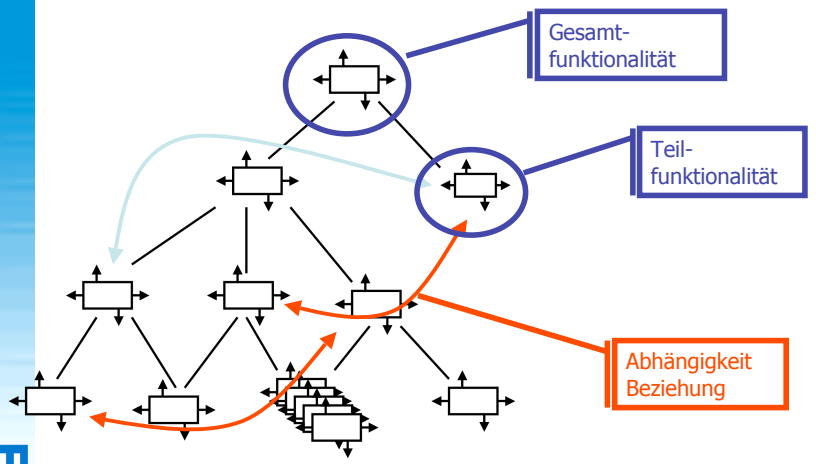
## Kundenfunktionalität: Nutzungssicht

Nutzungssicht multi-funktionaler Systeme:

- Kundenfunktionshierarchien
  - Teilfunktionsrelation als partielle Ordnung
  - Atomare Kundenfunktionen als „Blätter“ in der Hierarchie
- Abhängigkeiten zwischen Kundenfunktionen
  - Formen von Abhängigkeiten
    - Controls
    - Uses
    - Enables
    - Is_independent
  - Feature interaction
- Modellierung atomarer Funktionen
  - Interaktionsmuster
  - partielle Verhaltensfunktionen/partielle Zustandmaschines
- Kombination der Kundenfunktion zu multi-funktionalen Systemen



## Kundenfunktionshierarchie

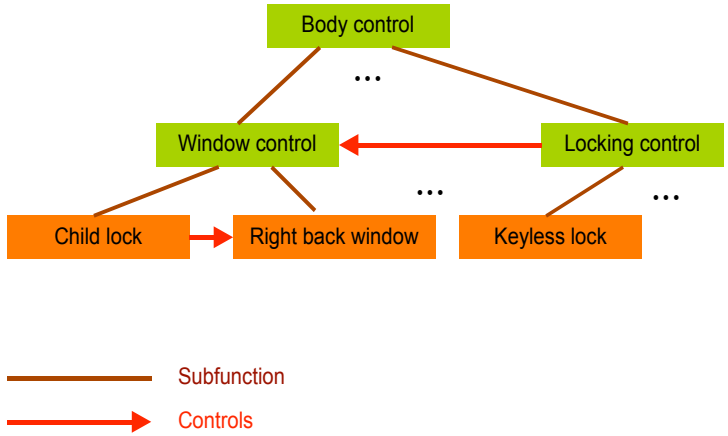


## Kundenfunktionshierarchie - Fazit

- Jede Kundenfunktion hat ein Verhalten - modelliert durch
  - Syntaktische Schnittstelle - Signale/Nachrichten
  - Verhalten
    - Partielle Ein/Ausgaberelation
    - Partielle Zustandsmaschine
    - Menge von Interaktionen (Interaktionsdiagramm)
- Die Kundenfunktionen bilden Hierarchie
  - Entspricht der Teilfunktionsbeziehung
- Es existieren Relationen/Beziehungen/Abhängigkeiten zwischen Kundenfunktionen
  - Ist_Teilfunktion
  - Ist_unabhängig
  - Kontrolliert
  - ....
- Elementare Kundenfunktionen werden zu zusammengesetzten Kundenfunktionen kombiniert
- Auf der Kundenfunktionshierarchie
  - können Abhängigkeiten analysiert und verifiziert werden
  - Testfälle generiert werden



## Beispiel: Service/feature hierarchy






## Was ist eine Funktion?

Wir unterscheiden

- Kundenfunktion (Nutzungssicht - Black Box)
- Bordnetzfunktion (Architektursicht - logisch)

Konsequenzen aus dem Ansatz:

- Ein System hat ein Schnittstellenverhalten!
- Ein System bietet Familie von Teilfunktionen an!
- Logische Architektur besteht aus Netz von logischen Komponenten (den Teil-Systemen)!
- Eine Komponente hat ein Schnittstellenverhalten!
- Ein Komponente bietet Familie von Teilfunktionen an!

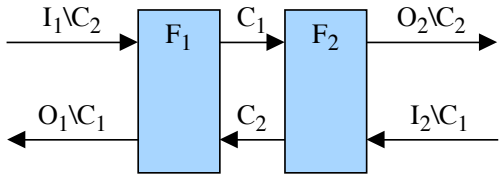


## Logische Systemarchitektur

- Systeme in logische Komponenten zergliedern
  - Totale Verhaltensfunktionen
  - Zustandmaschinen mit Ein- und Ausgabe
- Kommunikation über idealisierte „logische“ Kanäle
  - Addressierung über individuelle Kannäle (point to point connection)
  - Kommunikation erfordert keine Zeit
- Komposition als logische Konjunktion
- Globales Zeitmodell
- Systeme als Hierarchie von Sub-systemen

## Komposition von Systemen

$F_1 \in IF[I_1 \blacktriangleright O_1]$
$F_2 \in IF[I_2 \blacktriangleright O_2]$




$C_1 = O_1 \cap I_2$
$C_2 = O_2 \cap I_1$
$I = I_1 \backslash C_2 \cup I_2 \backslash C_1$
$O = O_1 \backslash C_1 \cup O_2 \backslash C_2$

$F_1 \otimes F_2 \in IF[I \blacktriangleright O],$

$(F_1 \otimes F_2).x = \{z|O: x = z|I \wedge z|O_1 \in F_1(z|I_1) \wedge z|O_2 \in F_2(z|I_2)\}$

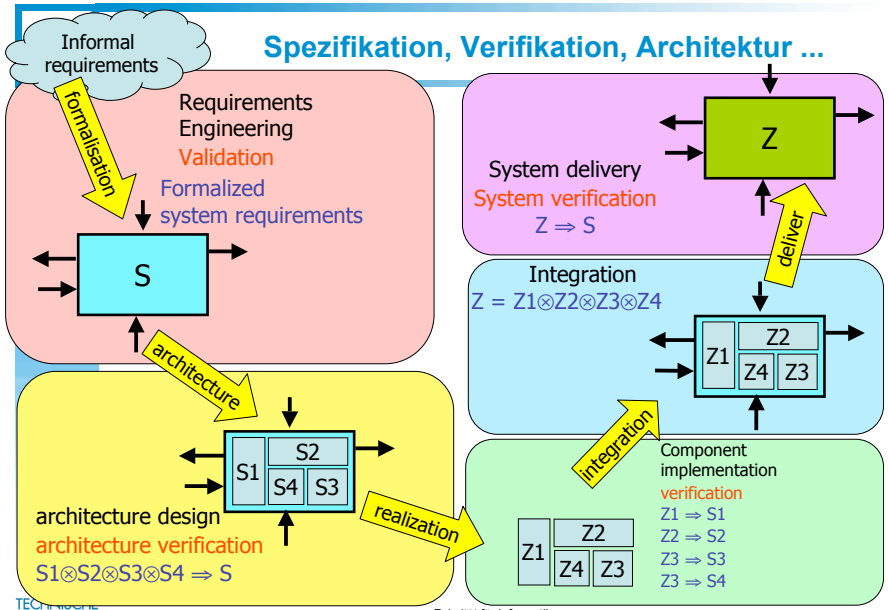


## Allgemeines logisches Systemmodel

- Ein System stützt sich auf ein Datenmodell
  - Nachrichten/Signale
  - Zustandsattribute
- Jedes System und Teilsystem hat ein Schnittstellenverhalten
  - Jedes System ist auch mögliches Teilsystem und Komponente
- Ein System ist eine Hierarchie von Teilsystemen - die logische Systemarchitektur
- Jedes System und Teilsystem
  - hat ein Schnittstellenverhalten (Ein/Ausgaberelation)
  - Kann repräsentiert werden durch
    - Schnittstellenspezifikation
    - Zustandsmaschine
    - Architektur von Teilsystemen

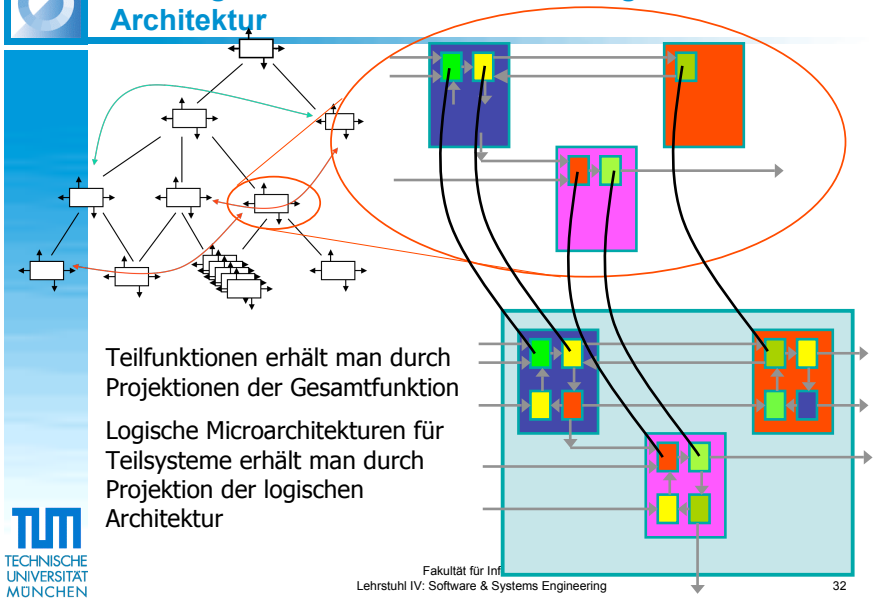


## Spezifikation, Verifikation, Architektur ...





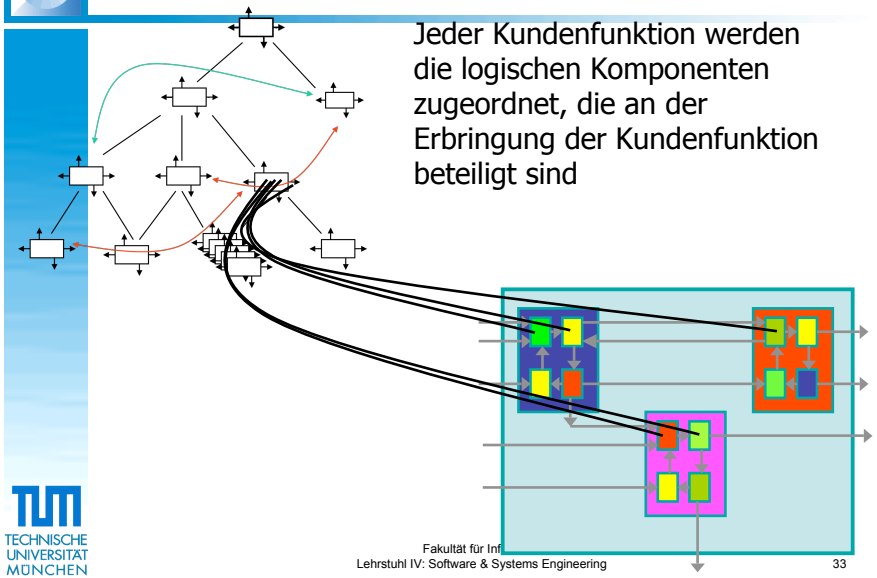

## Nutzungssicht - Microarchitektur - logische Architektur



Teilfunktionen erhält man durch Projektionen der Gesamtfunktion

Logische Microarchitekturen für Teilsysteme erhält man durch Projektion der logischen Architektur

## Zusammenhang Nutzungssicht - logische Architektur



Jeder Kundenfunktion werden die logischen Komponenten zugeordnet, die an der Erbringung der Kundenfunktion beteiligt sind



## Software Architektur

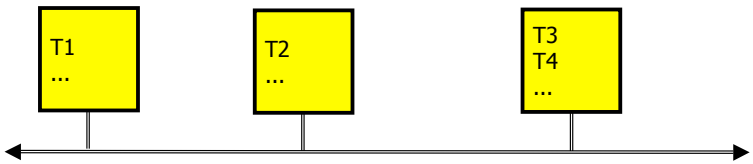
- Applikationssoftware
- Softwareplattform (vgl. AUTOSAR)
  - Real time Operating System OSEK
  - Device drivers
  - Bussysteme
- Sichten auf die Software
  - Entwurfszeitpunkt: Code Architektur
    Strukturierung der Software in Klasses und Module
  - Laufzeit: Task Architektur
    - Tasks, Threads, Processes
    - Scheduling



## Hardware Architektur

Die Hardware Architektur umfasst:

- Controllers
- Communication devices
- Sensor und actuators
- MMI






## Deployment

- Abbildung zwischen Software und Hardware

Bestimmt:

- Welche Software auf welchem Steuergerät läuft

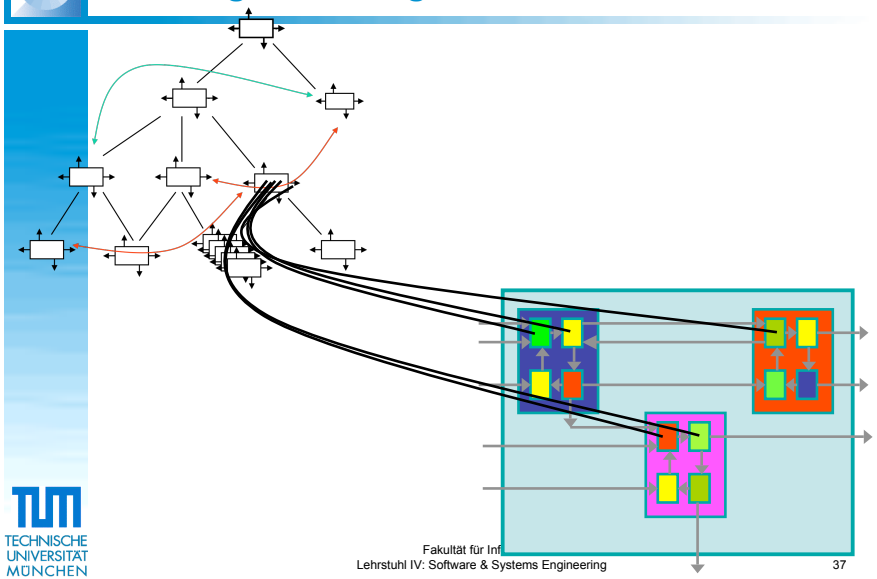
Wichtige Gesichtspunkte:

- Scheduling im Steuergerät
- Scheduling der Bussysteme (globaler Schedule bei FlexRay)

Ziel:

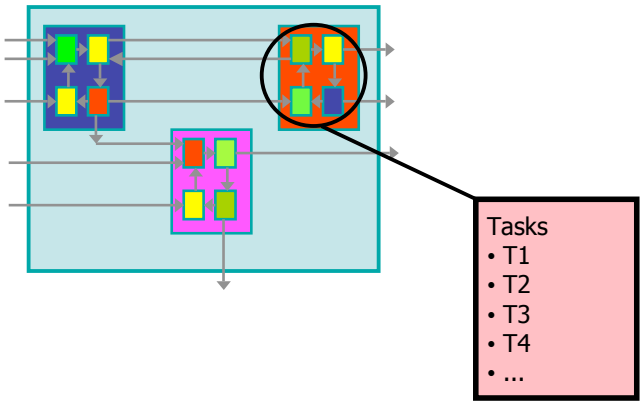
- Flexibles Deloyment

## Nutzungssicht - logische Architektur



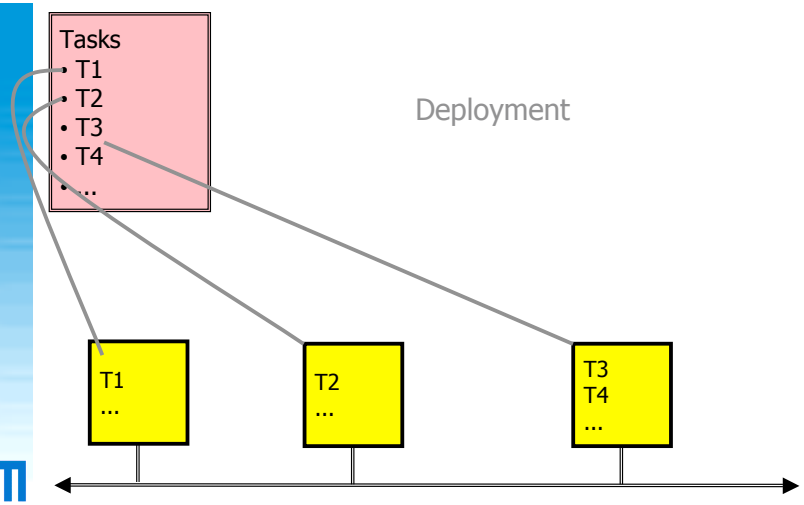


## Von der logische Architektur zur Softwarearchitektur



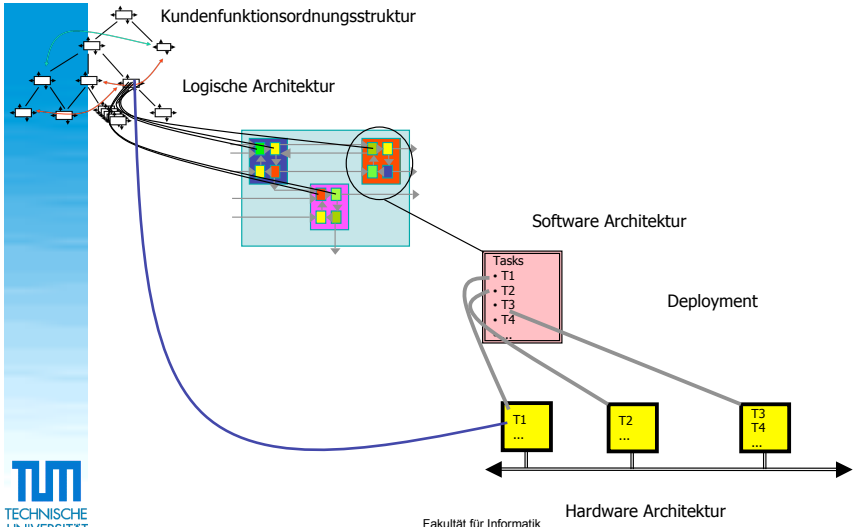



## Von der logische Architektur zur Softwarearchitektur

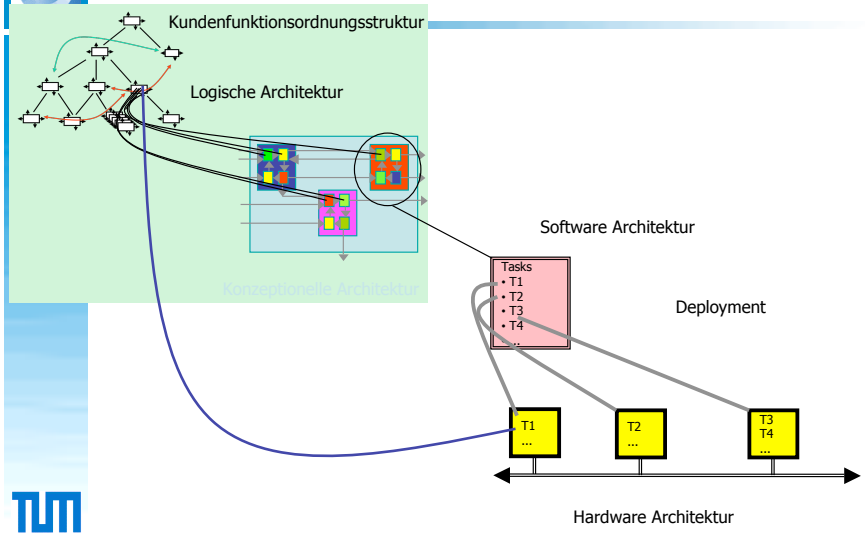





## Das umfassende Modell

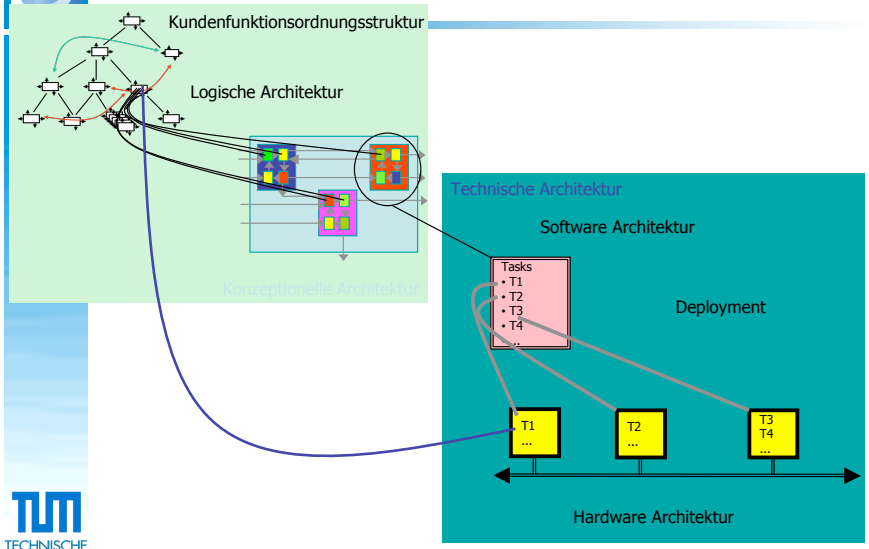


# Das umfassende Modell






# Das umfassende Modell



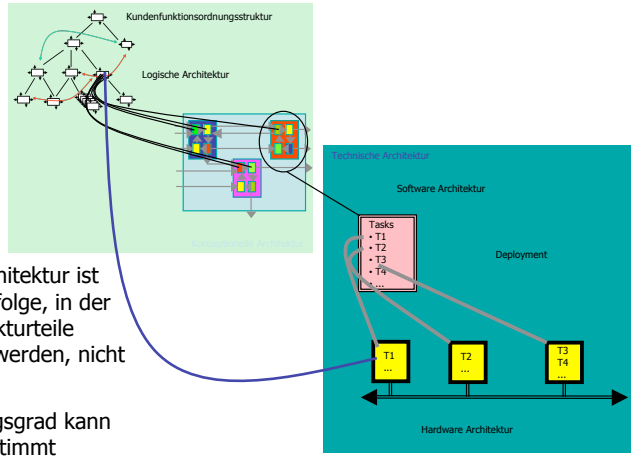



# Darstellung der Zusammenhänge im Modellbaukasten

- Eine Kundenfunktion wird durch eine Mikroarchitektur erbracht
- Eine Mikroarchitektur besteht aus einer Architektur von partiellen logischen Komponenten die Teilfunktion der Komponenten der logischen Architektur sind
- Jede logische Komponente realisiert ein Bündel von Teilfunktionalitäten
- Auf einer hinreichend feingranularer logischen Architektur werden Teilsysteme in Tasks überführt
- Ein Task ist ein „schedulbares“ Verhalten
- Das Deployment bestimmt welche Familien von Tasks zu schedulbaren Einheiten zusammengefasst werden
- Ein Scheduler bildet eine Familie von Tasks auf ein Verhalten ab (Verhalten des Steuergeräts nach außen)
- Durch einen Scheduler wird das Verhalten einer Teilfunktion/logischen Komponente transformiert (zeitlich verzögert)
- Durch das Busscheduling wird die Kommunikation einer Teilfunktion/logischen Komponente transformiert (zeitlich verzögert)
- Damit findet sich für jede Kundenfunktion das Verhalten verfeinert auf der Technischen Ebene wieder



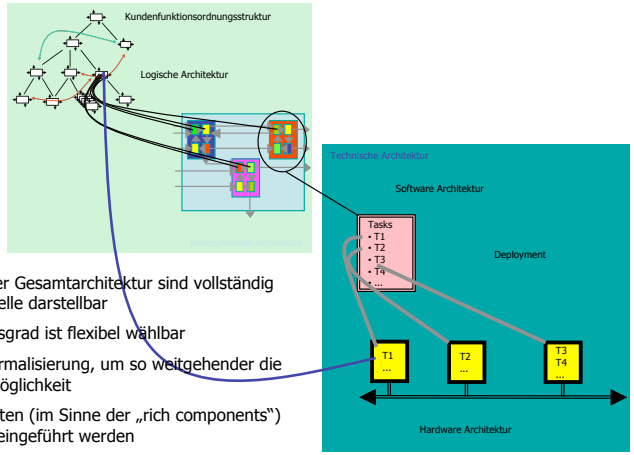
# Das Vorgehen: Top Down oder Bottom Up?




Durch die Gesamtarchitektur ist die Reihenfolge, in der die Architekturteile erarbeitet werden, nicht festgelegt

Der Füllungsgrad kann iterativ bestimmt werden

# Modellierungsdetail- und Formalisierungsgrad




Alle Bestandteile der Gesamtarchitektur sind vollständig durch formale Modelle darstellbar

Der Formalisierungsgrad ist flexibel wählbar

Umso höher die Formalisierung, um so weitgehender die Automatisierungsmöglichkeit

Weitere Eigenschaften (im Sinne der „rich components") können ins Modell eingeführt werden

Vollständige Verifikation der Übergänge zwischen Ebenen möglich



# Potential des Ansatzes

- Ableitung eines Meta-Modells
- Ausbau des Modells in Stufen
  - Formalisierungsgrad kalibrierbar
  - Verhaltensteilmodelle
- Anwendung im Prozess
  - Befüllung iterativ
  - Konsistenz prüfbar
  - Methodik ausbaubar
    - Testfälle generieren



# Literaturhinweise


- [AC98] Martín Abadi, Luca Cardelli: *A Theory of Objects – Monographs in Computer Science,* Springer Verlag 1998
- [BS01] Manfred Broy, Ketil Stølen. Specification and Development of Interactive Systems. Springer Verlag. 2001
- [BCK+98] Len Bass, Paul Clements, Rick Kazman, Ken Bass: *Software Architecture in Practice (Sei Series in Software Engineering)*, Addison-Wesley Publishing, 1998
- [BRS97] Klaus Bergner, Andreas Rausch, Marc Sihling: Using UML for Modeling a Distributed Java Application, Technischer Bericht der Universität München, TUM-I9735, http://wwwbib.informatik.tu-muenchen.de/infberichte/1997/TUM-I9735.ps.gz,1997
- [BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad und Michael Stal. Pattern-Oriented Software Architecture , Volume 1: A System of Patterns. John Wiley & Sons 1996.
- [CBB02] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford: *Documenting Software Architectures – Views and Beyond*, Addison-Wesley, 2002.




# Literaturhinweise


- [DW98] Desmond Francis D'Souza, Alan Cameron Wills. *Objects, Components, and Frameworks With UML: The Catalysis Approach*, Addison Wesley Publishing Company, 1998
- [GHJ+95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns – Elements of Reusable Object-Oriented Software. 1995.
- [HNS99] Christine Hofmeister, Robert Nord, Dilip Soni: *Applied Software Architecture*, Addison Wesley – Object Technology Series, 1999.
- [HS99] Peter Herzum, Oliver Sims. Business Component Factory: *A Comprehensive Overview of Component-Based Development for the Enterprise*, John Wiley & Sons, 1999.
- [JRL+00] Mehdi Jazayeri, Alexander Ran, Frank Van Der Linden, Philip Van Der Linden: *Software Architecture for Product Families: Principles and Practice*, Addison-Wesley Publishing, 2000
- [KWB03] Anneke Kleppe, Jos Warmer, Wim Bast: *MDA Explained*, Addison-Wesley, 2003.
- [SGW94] Bran Selic, Garth Gullekson, Paul T. Ward. Real-Time Object-Oriented Modeling. John Wiley & Sons. 1994