

# Überblick

## Architektur

- Erscheinungsform
- Zusammensetzbarkeit
- Skalierbarkeit
- Verlässlichkeit
- Zusammenfassung
- Bibliographie

# Echtzeitsysteme Architektur

17. Januar 2011

# Schönheit, Stabilität, Nützlichkeit

*Venustas, Firmitas, Utilitas*: Die drei Prinzipien von Architektur [1]



## Fieberthermometer

- ▶ 1 „elektr. Steuergerät“
- ▶ weiche Echtzeit



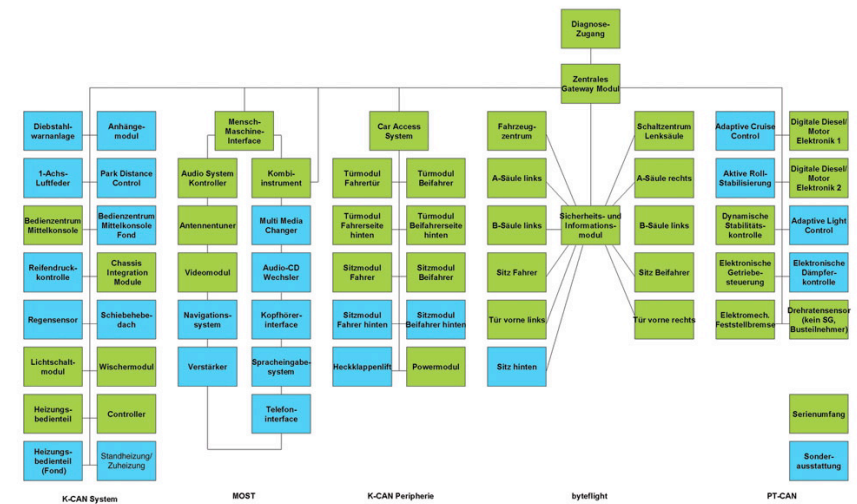
## Automobil (7er BMW, E65)

- ▶  $\geq 38 + 25$  elektr. Steuergeräte
- ▶ weiche bis harte Echtzeit

die Funktion eines Objektes bestimmt seine physikalische Form [1]

# Verteiltes System auf Rädern

Vernetzung beim 7er BMW, Baureihe E65



Quelle: BMW AG

## Abstraktion

Kapselung logischer Funktionen

**Erfassung** der wesentlichen funktionalen und nicht-funktionalen (insb. zeitlichen) Eigenschaften eines Objektes

- ▶ Automobil  $\mapsto$  **Mechatronik**  $\mapsto$  Steuergerätenetz  $\mapsto$  Steuergerät

**Verdeckung** aller irrelevanten Details der Implementierung hinter einer einfachen und stabilen Schnittstelle

- ▶ Modul, abstrakter Datentyp, Klasse, Objekt, ...
- ▶ Komponente (von lat. *componere*, zusammensetzen)
  - ▶ in der **Informatik** ein unabhängiges Softwaremodul, das in anderen Softwaresystemen wiederverwendet werden kann
  - ▶ im **Maschinenbau** eine aus verschiedenen Elementen bestehende Funktionseinheit eines übergeordneten Systems

## Abstraktion (Forts.)

Fehlertransparenz

Zuschnitt und Platzierung der Schnittstellen bilden wesentliche Aktivitäten des Entwurfs eines Echtzeitrechensystems

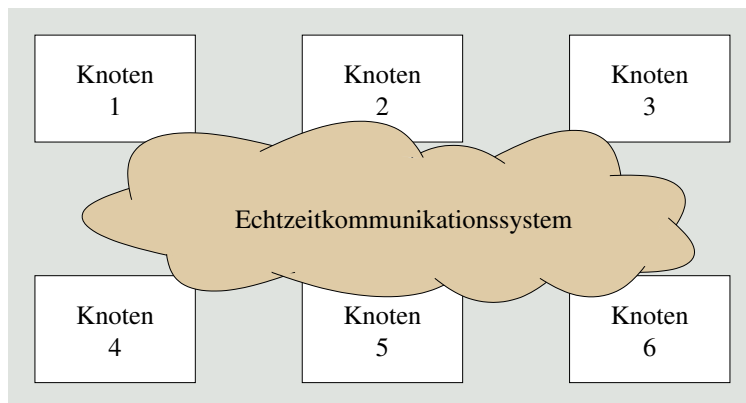
- ▶ bestimmen die Struktur und Eigenschaften des Systems als Ganzes

Herausforderung dabei ist, dass die Abstraktionen auch im **Fehlerfall** ihre Gültigkeit haben sollten

- ▶ das Ideal ist ein **verteiltes Rechensystem** mit einer 1-zu-1-Beziehung zwischen Funktion und Rechenknoten
  - Ursache bzw. Knoten einer Fehlfunktion ist „leicht“ identifizierbar
  - Auswirkungen eines fehlerhaften Knotens sind „gut“ vorhersehbar
- ▶ ein **zentralisiertes Rechensystem** erschwert die Fehlerdiagnose enorm, wie auch die Analyse von Fehlereffekten von Subsystemen

## Verteiltes Echtzeitrechensystem

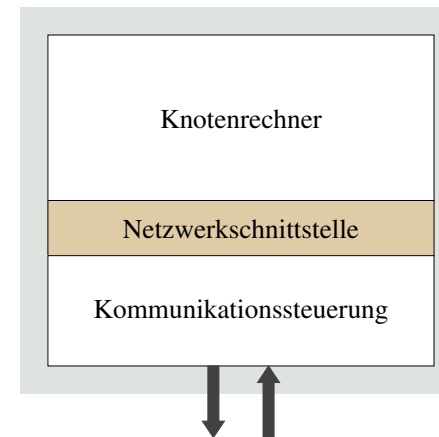
Rechenbetonte Gerätegruppe (engl. *computational cluster*)



- ▶ jeder Knoten erbringt eine Teilfunktion des Gesamtsystems
- ▶ ein **Kommunikationssystem** (KS) sorgt für die enge/lose Kopplung

## Grobstruktur eines Knotens

Partitionierung in zwei Subsysteme: Knotenrechner und Kommunikationssteuerung



### Echtzeitkommunikationssystem

- ▶ Menge der Subsysteme zur Kommunikationssteuerung der Knoten der Gerätegruppe
- ▶ zusammen mit dem jeweiligen phys. Verbindungsmedium

### Netzwerkschnittstelle

- ▶ **Transportschicht** des ISO OSI Referenzmodells [2]
- ▶ wichtigster Bestandteil des Echtzeit-KS

### Kommunikationssteuerung (engl. *communication controller*)

- ▶ Gerätetreiber und Netzsteuerung (Hardware und Software)

## Netzwerkschnittstelle

Semantik von Daten und Strategie der Steuerung

Abstraktion von den Details der Protokolllogik und der physikalischen Struktur des Kommunikationsnetzwerks. . .

- ▶ einerseits in Bezug auf die **Datensemantik**, die Nachrichteninhalte als Ereigniseintritt oder Zustandswert versteht
  - (a) da jedes Ereignis signifikant ist, sind alle Nachrichten entsprechend ihrer Ereigniszeitpunkte zwischenspeichern  $\mapsto$  sortierte Schlange
    - ▶ Nachrichtenverlust bedeutet ggf. Synchronisationsverlust
  - (b) da nur aktuelle Zustandswerte signifikant sind, ist immer nur die zuletzt empfangene Nachricht zu speichern  $\mapsto$  überschreiben
- ▶ andererseits in Bezug auf die **Steuerungsstrategie**, die zwischen zwei Kontrollbereichen differenziert
  - (a) **externe Kontrolle** im Knotenrechner, die vom Kommunikationssystem die Anzeige von Kontrollsignalen erfordert  $\leadsto$  **Ereignissteuerung**
  - (b) **autonome Kontrolle** im Kommunikationssystem, die Knotenrechner ununterbrochen weiter arbeiten lässt  $\leadsto$  **Taktsteuerung**

## Netzwerkschnittstelle (Forts.)

Entwurfsraum (engl. *design space*)

Datensemantik und Steuerungsstrategie jeweils in Bezug auf Sender und Empfänger resultieren in 16 **Kombinationsmöglichkeiten** [3, S. 32]:

Empfänger		Sender		Ereigniseintritt		Zustandswert	
		extern	autonom	extern	autonom	extern	autonom
Ereigniseintritt	ext.	<b>Ereignisnachricht</b>	kann sein	ja	ja	ja	ja
	aut.	nein	kann sein	ja	ja	ja	ja
Zustandswert	ext.	ja	kann sein	ja	ja	ja	ja
	aut.	nein	kann sein	ja	ja	ja	<b>Zustandsnachricht</b>

- ▶ die mit „nein“ markierten Kombinationen machen keinen Sinn:
  - ▶ die Sendezeitpunkte werden beliebig vom Empfangstakt abweichen
    - ▶ mehrere Sendeereignisse könnten in einen Empfangstakt fallen
  - ▶ Nachrichten, die Ereignisdaten beinhalten, werden ggf. verworfen
- ▶ besondere Bedeutung haben Ereignis- und/oder Zustandsnachrichten

## Nachrichten besonderer Bedeutung

Ereigniseintritt vs. Zustandswert

**Ereignisnachricht** (engl. *event message*)

- ▶ kombiniert Ereignissemantik mit externer Kontrolle:
  - ▶ jede eingehende Nachricht wird beim Empfänger gepuffert
  - ▶ Entsorgung erfolgt durch Konsumierung (d.h., bei Verarbeitung)
- ▶ erfordert 1-zu-1-Synchronisation zwischen Sender und Empfänger
  - ▶ zur Vermeidung von Pufferüberlauf bzw. Empfängerblockaden
- ▶ korrespondiert zum „klassischen“ Botschaftenaustausch  $\mapsto$  IPC

**Zustandsnachricht** (engl. *state message*)

- ▶ kombiniert Zustandswertsemantik mit autonomer Kontrolle
  - ▶ entspricht der Semantik globaler Variablen, jedoch. . .
    - (a) das Kommunikationssystem garantiert unteilbares schreiben
    - (b) es gibt nur 1 Schreiber (engl. *multiple reader, single writer, MRSW*)
  - ▶ gestattet eine losere Kopplung zwischen Sender und Empfänger
- ▶ korrespondiert zu den Anforderungen von Steuerungsanwendungen

## Interagierende Gerätegruppen

Netzübergang (engl. *gateway*)

**Netzübergangsknoten** (engl. *gateway nodes*) schlagen Brücken zwischen verschiedenen Gerätegruppen (engl. *cluster*)

- ▶ Netzübergänge kommen mit zwei Ausprägungen von Schnittstellen:
  1. eine Instrumenten- und eine Kommunikationsschnittstelle
    - ▶ auch als **Schnittstellenknoten** (engl. *interface node*) bezeichnet
  2. zwei Kommunikationsschnittstellen (d.h., Netzwerkschnittstellen)
- ▶ sie bilden einen „Umschlagplatz für relevante Informationen“
  - ▶ in nicht allen Gerätegruppen ist jede Information signifikant
  - ▶ Datenformate bzw. -repräsentationen können verschieden sein
  - ▶ Nachrichtenweiterleitung bedingt Transformationsvorgänge

Netzübergänge taktgesteuerter verteilter Systeme haben Schnittstellen zur gemeinsamen Benutzung von Daten (engl. *data-sharing interface*)

- ▶ durch die Netzübergangskomponenten verlaufen keine Steuersignale
- ▶ autonome Kontrolle der Gerätegruppen ist trotz Kopplung gesichert

## Netzübergangsknoten als Entwicklungshelfer

Altsysteme (engl. *legacy systems*)

Echtzeitrechnungssysteme ab einer gewissen Komplexität werden nicht von Grund auf (engl. *from scratch*) und komplett neu entwickelt

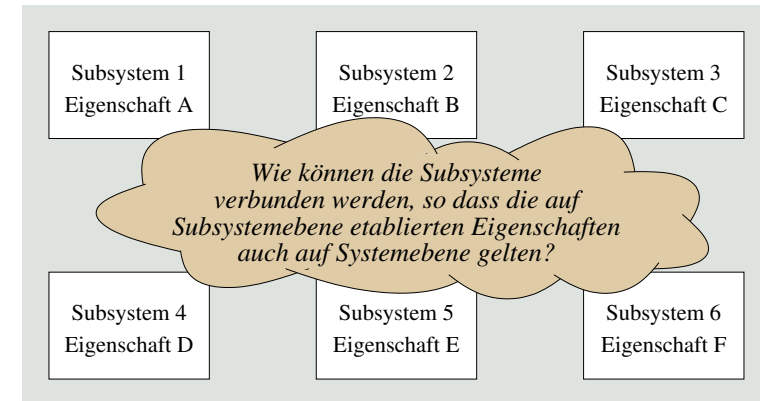
- ▶ der Produktionsplan ist ggf. vollständig, jedoch sieht er nicht selten auf mehrere Zeiträume aufgeteilte Ausbaustufen vor
- ▶ Technologiewandel bereits während der Produktionszeit ist typisch

Ausbaustufen wie auch Altsysteme bilden Gerätegruppen, die frühzeitig in Dienst gestellt und später erweitert/erneuert werden

- ▶ Netzübergänge fördern die Entwicklung **evolutionär stabiler Systeme**
  - ▶ neue Ausbaustufen können „leicht“ an bestehende angedockt werden
  - ▶ Eigenschaften von Altsystemen lassen sich „leicht“ nachbilden
- ▶ Netzübergangsknoten abstrahieren (teils) von Gerätegruppen

## Kompositionsproblem

Zentrale Rolle von Echtzeitkommunikationssystemen bzw. der Netzwerkschnittstelle



Architekturen sind zusammensetzbar (engl. *composable*) hinsichtlich einer spezifizierten Eigenschaft, wenn die Systemintegration diese vorher für ein Subsystem festgelegte Eigenschaft weiterhin aufrecht erhält

- ▶ **Rechtzeitigkeit** (engl. *timeliness*), Testbarkeit (engl. *testability*)

## Ereignisgesteuerte Kommunikationssysteme

Transport von Ereignisnachrichten

Rechtzeitigkeit bzw. zeitliche Kontrolle ist eine **globale Angelegenheit** des gesamten verteilten Rechensystems

- ▶ bei ereignisgesteuerten Protokollen ist zeitliche Kontrolle an der Netzwerkschnittstelle undefiniert; das bedeutet:
  - ▶ ein einziger gemeinsamer Transportkanal schürt **Zugriffskonflikte**
    - ▶ Lösungsansätze sind zufällige Zugriffe (Ethernet), vorgegebene Zugriffsreihenfolgen (*token ring*) und priorisierte Nachrichten (CAN)
    - ▶ ohne jedoch das Grundproblem vom Tisch zu bekommen. . .
  - ▶ bei getrennten Transportkanälen droht **Überlastung** des Empfängers
- ▶ diese Kontrolle ist weiter oberhalb (des KS) sicherzustellen
  - ▶ in der Diensteschicht (engl. *middleware*) bzw. verteilten Anwendung
  - ▶ sie muss **nicht deterministisches Systemverhalten** „kaschieren“
  - ▶ vergleichsweise leicht bei weicher Echtzeit, schwer bis unmöglich sonst

☞ die Architektur ist **nicht zusammensetzbar** bzgl. Rechtzeitigkeit

## Zeitgesteuerte Kommunikationssysteme

Transport von Zustandsnachrichten

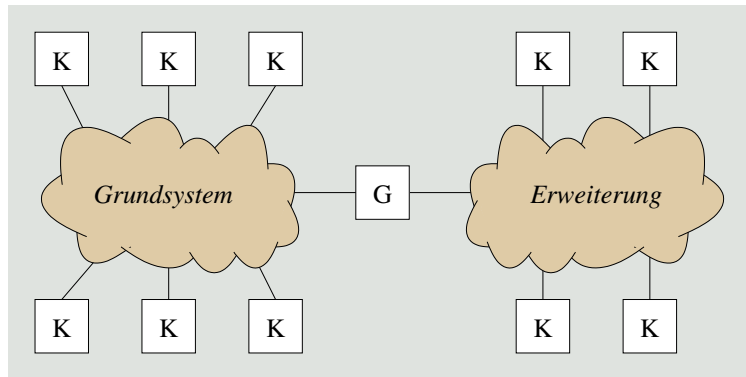
Rechtzeitigkeit bzw. zeitliche Kontrolle ist eine **lokale Angelegenheit** des Kommunikationssystems

- ▶ bei zeitgesteuerten Protokollen ist die zeitliche Kontrolle an der Netzwerkschnittstelle wohl definiert
  - ▶ Nachrichten werden zu festen, vorgegebenen Zeitpunkten transferiert
    - ▶ auf Basis einer **Ablaufabelle** in der Kommunikationssteuerung
  - ▶ Knotenrechner haben keinen Einfluss auf das Zeitverhalten des KS
    - ▶ die Netzwerkschnittstelle ist frei von Steuersignalen
    - ▶ sie hat eine **Daten teilende** (engl. *data sharing*) **Semantik**
    - ▶ **Steuerfehlerausbreitung** (*control-error propagation*) ist **unmöglich**
- ▶ sämtliche zeitlichen Eigenschaften wurden beim Entwurf festgelegt
  - ▶ Knoten sind unabhängig von der Netzwerkschnittstelle testbar
  - ▶ Systemintegration verändert nicht das Zeitverhalten der Schnittstelle

☞ die Architektur ist **zusammensetzbar** in Bezug auf Rechtzeitigkeit

## Transparenter Ausbau einer Gerätegruppe

Hinzunahme von Knoten  $\leadsto$  Andocken neuer Gerätegruppen



- ▶ neue Anforderungen sind keine Ausnahme, sondern die Regel
- ▶ eine **skalierbare Architektur** ist offen für Änderungen...

## Erweiterbarkeit

Graduelle Leistungszunahme — bzw. Leistungsabnahme, bei Schrumpfung

Architekturen dürfen **keine zentralen Flaschenhälse** aufweisen, um skalierbar zu sein in Bezug auf Rechen- und Kommunikationsleistung

- ▶ eine Hinzunahme von Knoten richtet sich nach der noch freien Kommunikationskapazität der Gerätegruppe
  - ▶ lediglich die Rechenleistung des Systems wird erhöht
- ▶ ist die Kommunikationskapazität einer Gerätegruppe erschöpft, so eröffnet der neue Knoten eine neue Gerätegruppe
  - ▶ ein Knoten der alten Gruppe „mutiert“ zum Netzübergangsknoten
  - ▶ der „geopferte“ und der neue Knoten bilden eine neue Gruppe
  - ▶ der Netzübergang ist transparent für andere Knoten (vergl. S. 10-12)
- ▶ die Zuordnung von Funktion zu Knoten muss weiterhin einer globalen Verteilungsdisziplin gehorchen
  - ▶ **Lastausgleich** (statisch, dynamisch)

☞ nur eine **verteilte Architektur** ermöglicht unbegrenztes Wachstum

## Komplexität

Komponentenanzahl, Anzahl und Art der Komponenteninteraktionen

*The partitioning of a system into subsystems, the encapsulation of the subsystem, the preservation of the abstractions in case of faults, and most importantly, a strict control over the interaction patterns among subsystems, are thus the key mechanisms for controlling the complexity of a large system. [3, S. 37]*

Komplexität eines großen Systems kann reduziert werden, wenn...

- ▶ das **innere Verhalten** der Subsysteme verborgen/gekapselt ist,
- ▶ zur Abkapselung **stabile Schnittstellen** Verwendung finden und
- ▶ diese Schnittstellen der Subsysteme „einfach und verständlich“ sind

## Abkapselung des Zeitverhaltens einer Gerätegruppe

Netzübergänge zeitgesteuerter Architekturen

Schnittstellen von Netzübergangsknoten in zeitgesteuerten Architekturen beschreiben die Eigenschaften der abgekapselten Gerätegruppen exakt

- ▶ sie enthalten das komplette Wissen, um Schlussfolgerungen über das zeitliche Verhalten einer Gerätegruppe führen zu können
  - ▶ Wissen über funktionale **und nicht-funktionale Eigenschaften**
- ▶ sie geben eine Daten teilende Semantik vor, haben keine Steuersignale und bilden eine „**zeitliche Brandmauer**“
  - ▶ eine Steuerfehlerausbreitung ist per Definition ausgeschlossen
- ▶ sie repräsentieren und spezifizieren ein Subsystem, dessen (inneres) Verhalten von autonomer Kontrolle geprägt ist

Architekturen, die für die kompositen Strukturen eines Systems derartige Schnittstellen vorgeben, sind skalierbar

- ▶ Schlussfolgerungen über das korrekte Verhalten einer Gerätegruppe sind unabhängig von der Anzahl der Gerätegruppen im System

## Reagierendes System

(engl. *responsive system*)  $\mapsto$  Verteilung + Echtzeitperformanz + Fehlertoleranz [5]

**Fehlereingrenzung** durch eine **Sicherheitshülle** (engl. *containment*)

- ▶ fehlertolerante Systeme sind in Partitionen strukturiert
  - ▶ in „fehlerdämmende Regionen“ (engl. *error-containment regions*)
- ▶ Fehler, die in einer Partition auftreten bleiben isoliert. . .
  - ▶ sie werden lokal erkannt und korrigiert oder maskiert
  - ▶ sie können das restliche System nicht beschädigen
- ▶ Fehlererkennung betrifft den Wert- wie auch den Zeitbereich

**Replikation** (der Funktionen) von Knoten

- ▶ erfordert **Replikdeterminismus** (engl. *replica determinism* [4])
  - ▶ aktiv replizierte Knoten sehen denselben Zustand *zur selben Zeit*
  - ▶ mit Zugeständnis zur endl. Genauigkeit der *Uhrensynchronisation*
- ▶ Maßnahme zur Fehlermaskierung: *hot/warm/cold stand-by*
  - ▶ ohne Replikdeterminismus ist Abstimmung (engl. *voting*) sinnlos. . .

## Unterstützung von Zertifizierung

Nachweis von Fehlereindämmung durch konstruktive Maßnahmen

Folge einer **Störung** (engl. *fault*) ist ein Fehler, der die Beschädigung des Systemzustands bedeutet

- ▶ sie ist katastrophal, gefährlich, bedeutend, undedeutend oder ohne Effekt in Bezug auf den **Sicherheitsspielraum** (engl. *safety margin*)
  - ▶ katastrophale/gefährliche Fehler müssen extrem unwahrscheinlich sein
  - ▶ die Ausbreitungsmöglichkeit von Fehlern ist stark einzuschränken
- ▶ eine Systemarchitektur, die fehlereindämmende Regionen vorsieht bzw. ermöglicht, ist dabei von zentraler Bedeutung
  - ▶ auch, um Zertifizierung auf wesentliche Komponenten zu beschränken

*If it is not possible to demonstrate that the error-containment coverage is very close to one, i.e., that the consequences of a particular error in a non safety-critical function do not have an adverse effect on a safety-critical system function, then, the error in the non safety-critical function must be classified as catastrophic. [3, S. 41]*

## Redundanz

**Primärsystem** (engl. *primary system*)  $\iff$  **Sekundärsystem** (engl. *secondary system*)

**hot standby** Primär- und Sekundärsystem laufen simultan

- ▶ Sekundärsystem *spiegelt* Primärsystemzustand
  - ▶ in **Echtzeit**
- ▶ Daten beider Systeme sind „jederzeit“ konsistent

**warm standby** Sekundärsystem läuft im Hintergrund

- ▶ Sekundärsystem *sichert* Primärsystemzustand
  - ▶ **regelmäßige Intervalle**, Fixpunkt (engl. *checkpoint*)
  - ▶ Fehlerfall  $\leadsto$  zum letzten Fixpunkt (engl. *recovery*)
- ▶ Daten beider Systeme sind zeitweise inkonsistent

**cold standby** Sekundärsystem startet bei Ausfall des Primärsystems

- ▶ auf Sekundärsystem gesicherte Primärsystemdaten
  - ▶ seltene und eher **unregelmäßige Intervalle**
  - ▶ planmäßige Sicherungskopie (engl. *backup*)
- ▶ Daten beider Systeme sind zeitweise inkonsistent

## Resümee

**Erscheinungsform**  $\mapsto$  **verteiltes Echtzeitrechensystem**

- ▶ Netzwerkschnittstelle, Kommunikationssteuerung
- ▶ externe vs. autonome Kontrolle; Ereignis- vs. Zustandsnachrichten

**Zusammensetzbarkeit** (engl. *composability*)

- ▶ interagierende Gerätegruppen, Kompositionsproblem
- ▶ ereignis- vs. zeitgesteuerte Kommunikationssysteme

**Skalierbarkeit** (engl. *scalability*)

- ▶ Erweiterbarkeit (aber auch Schrumpfung), Komplexität
- ▶ Abkapselung von Zeitverhalten, Eigenständigkeit

**Verlässlichkeit** (engl. *dependability*)

- ▶ Fehlereingrenzung, Replikation, Replikdeterminismus
- ▶ Sicherheitshüllen, fehlereindämmende Regionen, Zertifizierung

## Literaturverzeichnis

- [1] Vitruv Marcus Vitruvius Pollio.  
*De Architectura Libris Decem.*  
 27 v.Chr.  
 Übersetzung von Curt Fensterbusch, Vitruv — Zehn Bücher über  
 Architektur, Primus Verlag, 1996.
- [2] International Organization for Standardization.  
*Information technology – Open Systems Interconnection – Basic  
 Reference Model: The Basic Model.*  
 ISO/IEC 7498-1. ISO, 1994.
- [3] Hermann Kopetz.  
*Real-Time Systems: Design Principles for Distributed Embedded  
 Applications.*  
 Kluwer Academic Publishers, 1997.

## Literaturverzeichnis (Forts.)

- [4] Stefan Poledna.  
*Replica Determinism in Fault-Tolerant Distributed Real-Time  
 Systems.*  
 PhD thesis, Technical University of Vienna, Vienna, Austria, 1995.  
 Research Report 28/95.
- [5] Miroslav Malek.  
 Responsive computer systems.  
*Real-Time Systems*, 7(3), 1994.  
 Special Issue.