

# Echtzeitsysteme

## Anhang zur Einleitung

19. Oktober 2009

# Überblick

## Anhang zur Einleitung

Funktionale Anforderungen

Nicht-Funktionale Anforderungen

Klassifikation von Echtzeitsystemen

Eingebettete Systeme

Zusammenfassung

Bibliographie

# Datenaufzeichnung

## Zustandsvariablen

Zustandsänderung des kontrollierten Objektes ist eine Funktion der Zeit

- ▶ der Zustand bildet sich aus einem Satz von Zustandsvariablen
  - ▶ im Falle des kontrollierten Objekts „Automobil“, z.B.:
    - ▶ Geschwindigkeit
    - ▶ Rotation eines Rads
    - ▶ Stellung des Blinkschalters
    - ▶ Öffnung des linken Seitenfensters
    - ▶ Position des Kolbens in einem Zylinder
  - ▶ ein komplexes Gebilde mit Hunderten von Zustandsvariablen
- ▶ erfasst wird er durch Aufzeichnung der Werte der Zustandsvariablen
  - ▶ sofern beim Aufzeichnen die Zeit eingefroren worden ist

Echtzeitinstanz (engl. *real-time entity*) die für einen bestimmten Zweck  
bedeutsame Zustandsvariable aus der Menge aller Zustandsvariablen

# Datenaufzeichnung (Forts.)

## Sichtbarkeit von Zustandsvariablen

Kontrollbereich (engl. *sphere of control*) einer Echtzeitinstanz ist das Subsystem, zu dem die betreffende Echtzeitinstanz gehört

- ▶ besitzt Autorität zur Änderung des Wertes einer Echtzeitinstanz

Echtzeitinstanzen können außerhalb ihrer Kontrollbereiche nur beobachtet, nicht jedoch verändert werden

- ▶ im Falle des kontrollierten Objekts „Automobil“, z.B.:
  - ▶ die aktuelle Kolbenposition in einem Zylinder des Motors, die zum Kontrollbereich des Automobils gehört
  - ▶ sie kann (durch Aufzeichnung) von außen nur observiert werden
- ▶ jedes kontrollierte Objekt definiert einen eigenen Kontrollbereich

# Datenaufzeichnung (Forts.)

Abbilder von Zustandsvariablen

**Echtzeitabbild** (engl. *real-time image*) Repräsentation der Beobachtung einer Echtzeitinstanz innerhalb des Echtzeitrechensystems

- ▶ ist immer nur für ein begrenztes Zeitintervall akkurat (d.h. gültig)
- ▶ die Intervalllänge hängt ab von der Dynamik des kontr. Objekts

**Echtzeitdatenbasis** (engl. *real-time data base*) die Menge aller zeitlich akkurate Echtzeitabbilder des kontrollierten Objekts

- ▶ ist mit jeder Werteänderung einer Echtzeitinstanz zu aktualisieren
- ▶ die Aktualisierung erfolgt entweder periodisch oder spontan:
  - zeitgesteuert** (engl. *time-triggered*)  $\leadsto$  periodisch
    - ▶ zu festen Zeitpunkten des Echtzeitrechensystems
  - ereignisgesteuert** (engl. *event-triggered*)  $\leadsto$  spontan
    - ▶ zum Änderungszeitpunkt des kontrollierten Objekts

# Datenvergütung

Beurteilung von Messwerten

Sensoren liefern Rohdaten, die bei Abtastung über mehrere Schritte erst vergütet werden müssen, um ein korrektes Echtzeitabbild zu erhalten:

## 1. Signalangleichung (engl. *signal conditioning*)

- ▶ Rohdatensequenz einlesen und mitteln, um Messfehler zu verkleinern
- ▶ geglätteten Wert kalibrieren und in interne Maßeinheit umwandeln

## 2. Plausibilitätsprüfung des gemessenen Werts

- ▶ den Wert mit anderen Messwerten in Korrelation setzen
- ▶ dadurch auch versuchen, einen ggf. fehlerhaften Sensor zu erkennen

bestätigtes Datenelement (engl. *agreed data element*)

- ▶ signalbereinigter und plausibler Messwert
- ▶ korrektes Echtzeitabbild der korrespondierenden Echtzeitinstanz

# Überwachung von Echtzeitinstanzen

Abnormales Prozessverhalten erkennen

Ausfall nur einer Anlagenkomponente kann Abweichungen/Störungen im Normalbetrieb vieler Echtzeitinstanzen verursachen:

- ▶ Grenzwertverletzungen können die Folge sein und Alarne auslösen
- ▶ Schneeballprinzip gleich werden ggf. korrelierte Alarne generiert
- ▶ ein **Alarmschauer** (engl. *alarm shower*) ist mögliche Konsequenz

Echtzeitrechensysteme müssen Alarne erkennen, anzeigen und Operateure bei der Ursachenfindung eines Alarmschauers assistieren

- ▶ Buchführung aller Alarne, mit exaktem Zeitstempel pro Alarm
  - ▶ über die Zeitordnung lässt sich der Erstalarm identifizieren
- ▶ ggf. wissensbasierte Systeme zur **Alarmanalyse** einsetzen

# Überwachung von Echtzeitinstanzen (Forts.)

Alarmsituationen mit Spitzenlast und von Seltenheit

**Alarmspitze** (engl. *peak-load alarm situation*)  $\mapsto$  **Alarmschauer**

- ▶ darf die **Vorhersagbarkeit** (engl. *predictability*) des Verhaltens eines Echtzeitrechensystems nicht beeinträchtigen
- ▶ eine höchst bedeutsame Eigenschaft für viele Echtzeitanwendungen

**seltenes Ereignis** (engl. *rare-event situation*)

- ▶ ein Ereignis, das zwar höchst selten auftritt, aber höchst relevant ist
- ▶ Validierung der Leistungsfähigkeit eines Echtzeitrechensystems im Falle eines solchen Ereignisses ist enorm schwierig

*Einziger Zweck des Systems zur Überwachung und zum Herunterfahren einer Kernkraftanlage ist zuverlässige Leistung in einer seltenen Alarmsituation mit Spitzenlast — die hoffentlich nie eintreten wird. [1, S. 5]*

# Digitale Kontrolle

Regelung und Steuerung eines kontrollierten Objekts

Echtzeitrechensysteme berechnen Stellwerte für Aktoren oft ohne ein unterliegendes konventionelles Kontrollsysteem

- ▶ das kontrollierte Objekt erfährt dann eine **direkte digitale Kontrolle**
- ▶ die Kontrollanwendungen zeigen dabei eine hohe **Regelmäßigkeit**
  - ▶ eine meist endlose Sequenz von Kontrollperioden

## Rückgekoppelte Kontrollschleife (engl. *feedback control loop*)

initialisiere Stellwert;

initialisiere Zeitgeber und Unterbrecher;

bei Zeitgeberunterbrechung **erledige** /\* abtasten, regeln, steuern \*/

A/D-Wandlung der Echtzeitinstanz, Echtzeitabbild ziehen;

Echtzeitdatenbasis aktualisieren, neuen Stellwert berechnen;

D/A-Wandlung des Stellwerts, Echtzeitinstanz verändern;

**basta.**

# Digitale Kontrolle (Forts.)

Systeme mit mehreren Abtastraten (engl. *multirate systems*)

Anlagen mit zig oder gar hunderten rückgekoppelter Kontrollschleifen, betrieben von einem Echtzeitrechensystem, sind keine Seltenheit

- ▶ eine solche Anlage hat mehr als einen Freiheitsgrad
  - ▶ der Zustand ist definiert durch mehrere Echtzeitinstanzen
  - ▶ pro Echtzeitinstanz mindestens ein Paar Sensor/Aktor
- ▶ jede Echtzeitinstanz kann eine andere Dynamik zeigen
  - ▶ verschiedene Abtastraten werden die Konsequenz sein

Abtastraten solcher Systeme sind häufig **harmonisch**: längere Perioden bilden ein ganzzahliges Vielfaches kürzerer Perioden

- ▶ kleinster gemeinsamer Nenner ist die kürzeste Periode, die damit auch den „Arbeitstakt“ des Echtzeitrechensystems vorgibt
- ▶ jede Kontrollschleife kann kurz nach Beginn ihrer Periode beginnen
  - ▶ sollte vor Beginn ihrer nächsten Periode komplett durchgelaufen sein

# Interaktion Mensch/Maschine

Kritische Funktion von großer Bedeutung

Echtzeitrechensysteme müssen das Bedienpersonal (Operateure) ...  
**informieren** über den aktuellen Zustand des kontrollierten Objekts  
**assistieren** bei der Überwachung/Steuerung einer Maschine bzw. Anlage

- viele Unfälle basieren auf Fehler der Mensch/Maschine-Schnittstelle

**Buchführung** und **langfristige Speicherung** von Prozessdaten besitzt je nach Anwendungsfall einen sehr hohen Stellenwert:

*Zum Beispiel ist in manchen Ländern die pharmazeutische Industrie per Gesetz verpflichtet, alle relevanten Prozessparameter eines jeden Produktionsstoßes in ein Archiv aufzuzeichnen und zu speichern, so dass die zum Zeitpunkt des Produktionslaufs vorherrschenden Prozessbedingungen nachträglich untersucht werden können, wenn zum späteren Zeitpunkt ein schadhaftes Produkt am Markt identifiziert worden ist. [1, S. 5]*

# Rechtzeitigkeit (engl. *timeliness*)

Latenzen minimieren und determinieren

$d^{cpu}$  klein, konstant

- ▶ Ausführungszeit des schlimmsten Falls bestimmen
  - ▶ engl. *worst-case execution time*, WCET
- ▶ Komplexität der Alg. auf  $O(1)$  beschränken

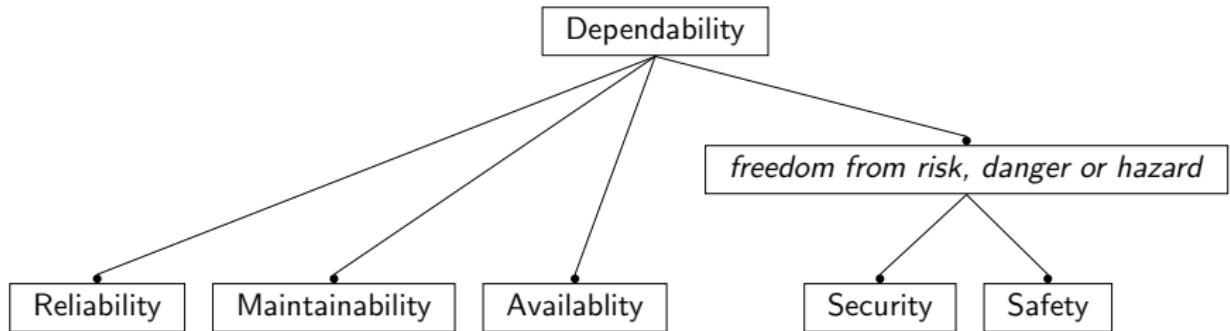
$\Delta d^{cpu}$  (engl. *delay jitter*) sehr klein, konstant

- ▶ schlimmsten Fall (max. Differenz) bestimmen
- ▶ **schwankungsfreies Echtzeitrechensystem** bauen

Fehlererkennung in kurzer Zeit, mit sehr hoher Wahrscheinlichkeit

- ▶ Größenordnung der Abtastfrequenz der schnellsten Kontrollsleife  $\leadsto$  **Latenz**
  - ▶ Fehler kompensieren, -ausbreitung eingrenzen
- ▶ korrektive Schritte sind so noch durchführ- bzw. ein sicherer Systemzustand ist so noch erreichbar

# Verlässlichkeit (engl. *dependability*)



*The trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers. [2]*

# Zuverlässigkeit (engl. *reliability*)

## Mittlere Betriebsdauer

$R(t)$  die Wahrscheinlichkeit, dass ein System seinen Dienst bis zum Zeitpunkt  $t$  leisten wird, sofern es bei  $t = t_0$  betriebsbereit war

- ▶ Annahme: eine **konstante Fehlerrate** von  $\lambda$  Fehler/Stunde
- ▶ Zuverlässigkeit zum Zeitpunkt  $t$ :  $R(t) = \exp(-\lambda(t - t_0))$ 
  - ▶ mit  $t - t_0$  gegeben in Stunden
- ▶ Inverse der Fehlerrate  $1/\lambda$  ist die *mean time to failure* (MTTF)

ultra-hohe **Zuverlässigkeit** wenn  $\lambda \leq 10^{-9}$  Fehler/Stunde gefordert ist

- ▶ Beispiel: elektronisch gesteuerte Bremsanlage im Automobil
  - ▶ das Kfz sei durchschnittlich eine Stunde täglich in Betrieb
  - ▶ dann darf jährlich nur ein Fehler pro eine Million Kfz auftreten
- ▶ andere Beispiele sind Eisenbahnsignalanlagen, Kernkraftwerk- und Flugüberwachungssysteme

# Wartbarkeit (engl. *Maintainability*)

Mittlere Reparaturdauer

- $M(d)$  die Wahrscheinlichkeit, dass das System innerhalb Zeitspanne  $d$  nach einem reparierbaren (gutartigen) Fehler wieder hergestellt ist
- ▶ Ansatz: **konstante Reparaturrate** von  $\mu$  Reparaturen/Stunde
  - ▶ die Inverse  $1/\mu$  ist dann die *mean time to repair* (MTTR)

**Fundamentaler Konflikt** zwischen Zuverlässigkeit und Wartbarkeit:

- ▶ ein wartbares System erfordert einen modularen Aufbau
  - ▶ kleinste ersetzbare Einheit (engl. *smallest replaceable unit*, SDU)
  - ▶ über Steckverbindungen lose gekoppelt mit anderen SDUs
  - ▶ dadurch ist jedoch eine höhere (physikalische) Fehlerrate gegeben
  - ▶ darüberhinaus verbuchen sich höhere Herstellungskosten
- ▶ ein zuverlässiges System ist aus einem Guss gefertigt...

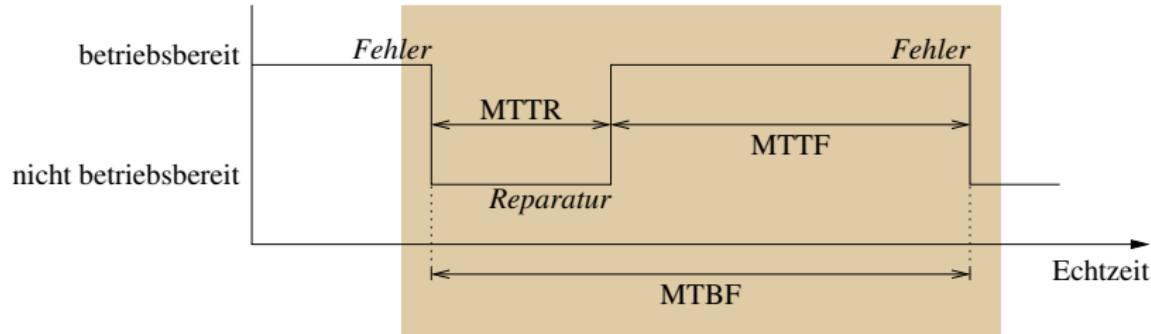
*Beim Entwurf von Produkten für den Massenmarkt geht die Zuverlässigkeit meist auf Kosten von Wartbarkeit.*

# Verfügbarkeit (engl. *availability*)

MTTF und MTTR im Zusammenhang

Maß zur Bereitstellung einer Funktion vor dem Hintergrund eines abwechselnd korrekt und fehlerhaft arbeitenden Systems

- Zeitanteil der **Betriebsbereitschaft**:  $A = \text{MTTF}/(\text{MTTF} + \text{MTTR})$
- $\text{MTTF} + \text{MTTR}$  auch kurz: *mean time between failures* (MTBF)



- ☞ hohe Verfügbarkeit bedeutet kurze MTTR und/oder lange MTTF

# Sicherheit $\mapsto$ Security und Safety

Robustheit des Echtzeitrechensystems stärken

**security** Schutz von Informationen und Informationsverarbeitung vor „intelligenten“ Angreifern

- ▶ allgemein in Bezug auf **Datenbasen** des Echtzeitystems
  - ▶ Vertraulichkeit (engl. *confidentiality*)
  - ▶ Datenschutz (engl. *privacy*)
  - ▶ Glaubwürdigkeit (engl. *authenticity*)
- ▶ speziell z.B. Diebstahlsicherung: Zündungssperre im Kfz
  - ▶ Kryptographie (engl. *cryptography*)

**safety** Schutz von Menschen und Sachwerten vor dem Versagen technischer Systeme

- ▶ Zuverlässigkeit trotz **bösartigen** (engl. *malign*) Fehlerfall
  - ▶ Kosten liegen um Größenordnungen über den Normalbetrieb
- ▶ Abgrenzung von unkrit., gutartigen (engl. *benign*) Fehlern
- ▶ oft ist Zertifizierung (engl. *certification*) erforderlich

# Verlässlichkeit ↔ Komplexität

Automobil eine Bestandsaufnahme vom Jahr 2005 ...

- ▶ etwa 90 % der Innovationen im Auto bringt die Elektronik ein
  - ▶ gut 80 % davon sind Software
- ▶ etwa ein Drittel aller Pannen liegen an fehlerhafte Elektronik
  - ▶ gut 80 % davon sind Softwarefehler

*Everything should be made as simple as possible, but no simpler. (Albert Einstein)*

*You know you have achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away. (Antoine de Saint Exupery)*

# Merkmale von Echtzeitsystemen

Klassifikation nach äußeren Faktoren  $\mapsto$  Charakteristik der Anwendung

- ▶ *hard/soft real-time*
- ▶ *fail safe/operational*

Klassifikation nach inneren Faktoren  $\mapsto$  Charakteristik des Systems<sup>1</sup>

- ▶ *guaranteed response/best effort*
- ▶ *resource adequate/inadequate*
- ▶ *event/time triggered*

---

<sup>1</sup>Entwurf und Implementierung

## hard real-time vs. soft real-time

*hard real-time* mindestens ein strikt einzuhaltender Termin liegt vor

- ▶ dass das System seine Zeitschranken einhält, ist zu validieren
  - ▶ auf Basis einer nachweislich korrekten, effizienten Vorgehensweise
  - ▶ oder durch ausgiebige Simulation und intensives Testen
- ▶ Validierung und Ausführung können gekoppelt (engl. *online*) sein
  - ▶ dynamische Einplanung ↪ **Akzeptanztest** (engl. *acceptance test*)

*soft real-time* ein strikt einzuhaltender Termin ist nicht gegeben

- ▶ die im Mittel von einem System zu erwartende Leistung bzw. **Dienstgüte** (engl. *quality of service*) steht im Vordergrund
  - ▶ gelegentlicher Leistungsabfall kann geduldet werden

*Wenn man die beste Dienstgüte fordert, die das System bieten kann, jedoch erlaubt, dass die gelieferte Dienstgüte schlechter sein darf, als durch die Zeitschranken definiert ist, dann sind die Zeitschranken weich.* [3, S. 29]

# *fail-safe* vs. *fail-operational*

## *fail-safe* Charakteristik des kontrollierten Objekts

- ▶ sicherer Zustand ist im Fehlerfall identifizier- und einnehmbar
- ▶ z.B. der Wächter bzw. *Zerberus*<sup>2</sup> (engl. *watchdog*) ...
  - ▶ überwacht den laufenden Betrieb des Echtzeitrechensystems
  - ▶ zwingt sein kontrolliertes Objekt in einen sicheren Zustand wenn das periodische Lebenszeichen des Echtzeitrechensystems ausbleibt

*In solch einem System ist Rechtzeitigkeit nur erforderlich, um hohe Verfügbarkeit zu erreichen, nicht jedoch, um Sicherheit  $\mapsto$  Safety zu garantieren, da der Watchdog das kontrollierte Objekt bei einer Zeitverletzung in einen sicheren Zustand zwingt. [1, S. 14]*

## *fail-operational* sicherer Zustand ist nicht identifizierbar

- ▶ z.B. ein Flugkontrollsysteem (engl. *flight control system*) ...
  - ▶ muss auch im Fehlerfall ein Minimum an Funktionalität garantieren

---

<sup>2</sup>In der gr. Mythologie der Höllenhund, der den Eingang zur Unterwelt bewacht.

## *guaranteed response vs. best effort*

### *guaranteed response* analytische Antwortgarantie

- ▶ Ausgangspunkt: die Spezifikation einer **Fehler- und Lasthypothese**
- ▶ Schlüsse über einen adequaten Entwurf sind rein logischer Natur
- ▶ sorgfältige Planung/umfassende Analyse des Entwurfs: ein Muss

*Die Fehlerwahrscheinlichkeit eines perfekten Systems mit Antwortgarantie reduziert sich auf die Wahrscheinlichkeit, dass Annahmen über Spitzenlast und Fehleranzahl/-typen in Realität gelten (engl. assumption coverage).*

[1, S. 14]

### *best effort* keine analytische Antwortgarantie

- ▶ rigorose Spezifikation der Fehler-/Lasthypothese ist nicht gefordert
- ▶ Test-/Integrationsphase bestätigt den ausreichenden Entwurf
- ▶ korrektes Verhalten trotz *rare-event situation* ist kaum nachweisbar

## *resource adequate vs. resource inadequate*

*resource adequate* für Spitzenlast- und Fehlerszenarien stehen gemäß Spezifikation genügend Betriebsmittel zur Behandlung bereit

- ▶ notwendige Eigenschaft, um Antwortgarantien geben zu können
- ▶ unbedingte Voraussetzung für strikte Echtzeitsysteme
  - ▶ insbesondere für sicherheitskritische Systeme

*resource inadequate* probabilistische Aussagen über die erwarteten Spitzenlast- und Fehlerszenarien sind akzeptabel

*Es wird davon ausgegangen, dass ein ausreichendes Betriebsmittelangebot zur Behandlung aller möglichen Situation ökonomisch nicht sinnvoll ist und stattdessen dynamische Strategien der Betriebsmittelzuteilung auf Teilhaberschaftsbasis (engl. resource sharing) genügen. [1, S. 15]*

- ▶ typisch für Systeme/Produkte des Massenmarkts: Automobile...

## event-triggered vs. time-triggered

Auslöser (engl. *trigger*) ist allgemein ein Ereignis, das den Start einer vordefinierten Aktion zur Folge haben soll:

- ▶ Ausführung eines Prozesses, einer Prozedur  $\mapsto$  Aufgabe (engl. *task*)
- ▶ Übertragung eines Signals und/oder einer Nachricht

Auslösungsmechanismus ist je nach Ansatz implementiert als Funktion des kontrollierten Objekts oder des Echtzeitrechensystems:

*event-triggered* Zustandswechsel einer Echtzeitinstanz, kontr. Objekt

- ▶ sporadisch auftretender *Interrupt*
- ▶ dynamische Einplanung (engl. *online scheduling*)

*time-triggered* Voranschreiten der Zeit, Echtzeitrechensystem

- ▶ periodische *Zeitgeberunterbrechung*
- ▶ statische Einplanung (engl. *offline scheduling*)

# Eingebettete Echtzeitsysteme

eingebettetes System (engl. *embedded system*)

- ▶ jedes **in einem Produkt verborgenes** jedoch von einem Rechner verschiedenes Rechensystem
  - ▶ Thermostate, Spielzeuge, TV/Audio/Video-, Haushaltsgeräte, ...
  - ▶ Kraftfahrzeuge, Flugzeuge, Raumschiffe, ..., Waffen
- ▶ ein **rechnerbasiertes System**, das eine spezielle Funktion oder einen speziellen Funktionsbereich abdeckt
  - ▶ Ölraffinerie, Walzstraße, Hochofen, Kernkraftanlage, ...
  - ▶ Anlagen für wissenschaftliche, technische und industrielle Zwecke
- ▶ **für eine spezielle Aufgabe entworfen**, dennoch versehen mit Wahlmöglichkeiten und Optionen

*Embedded systems often have several things to do at once. They must respond to external events (e.g., someone pushes the elevator button). They must cope with all unusual conditions without human intervention. Their work is subject to deadlines. [4, S. 1]*

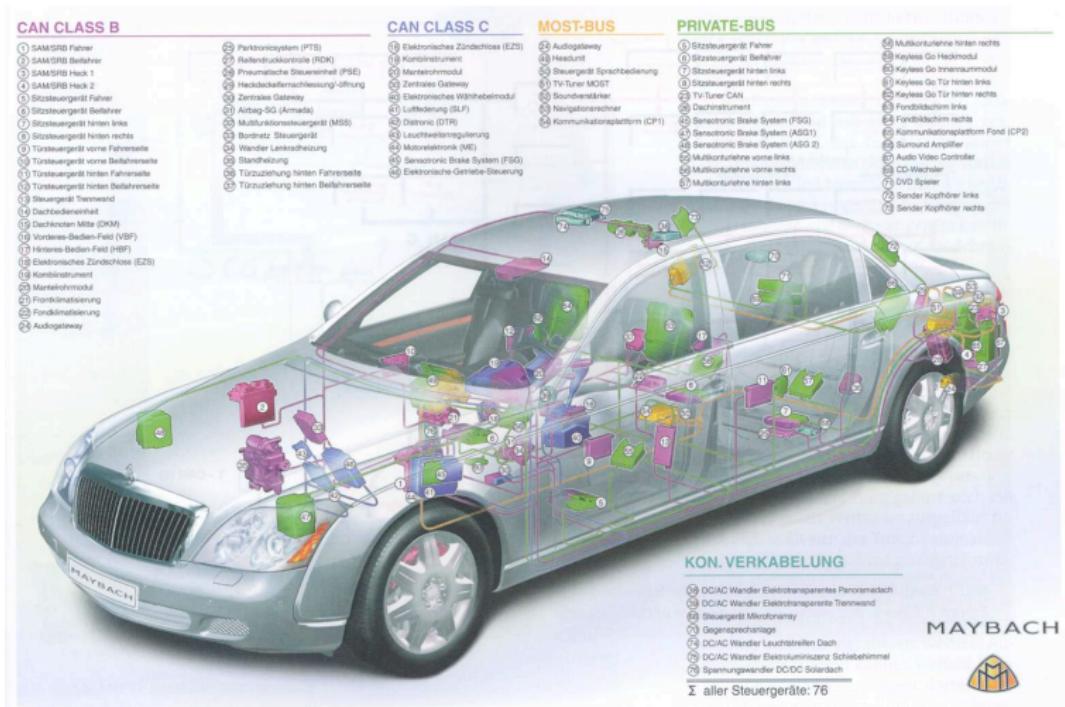
# Spezialzwecksysteme

Wenn Kompromisslösungen impraktikabel sind



# Spezialzwecksysteme (Forts.)

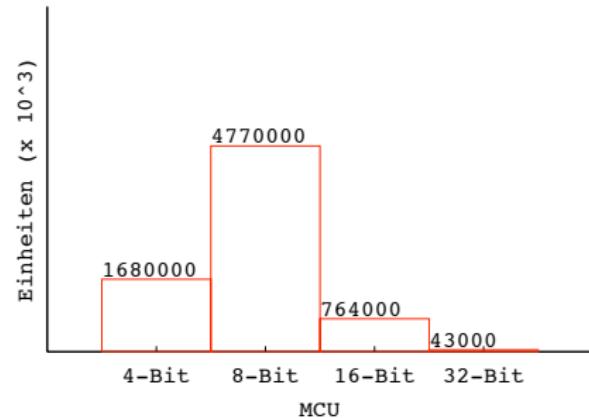
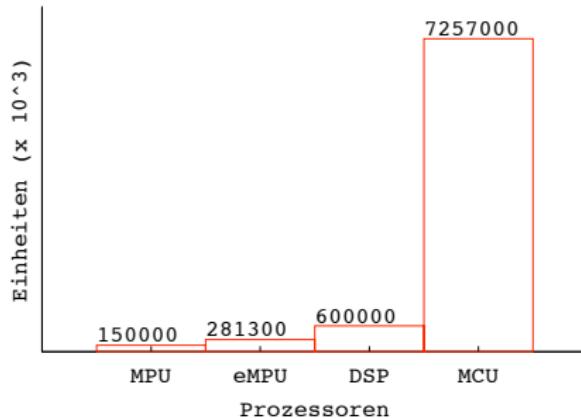
## Verteiltes System auf Rädern



(Quelle: DaimlerChrysler [5])

# Y2K Prozessorproduktion

„Where have all the processors gone?“



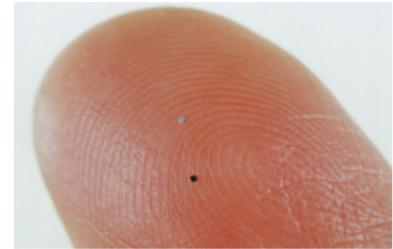
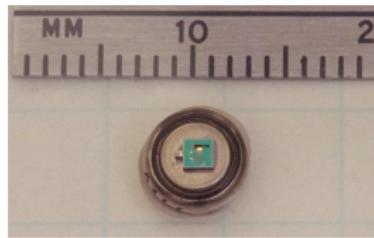
über 8 Mrd. Prozessoren [6]

1.8 % (MPU)  
98.2 % (eMPU, DSP, MCU)

Server, Desk-/Laptops, ...  
eingebettete Systeme

# Drahtlose Sensornetze

„Intelligenter Staub“ (engl. *smart dust*)

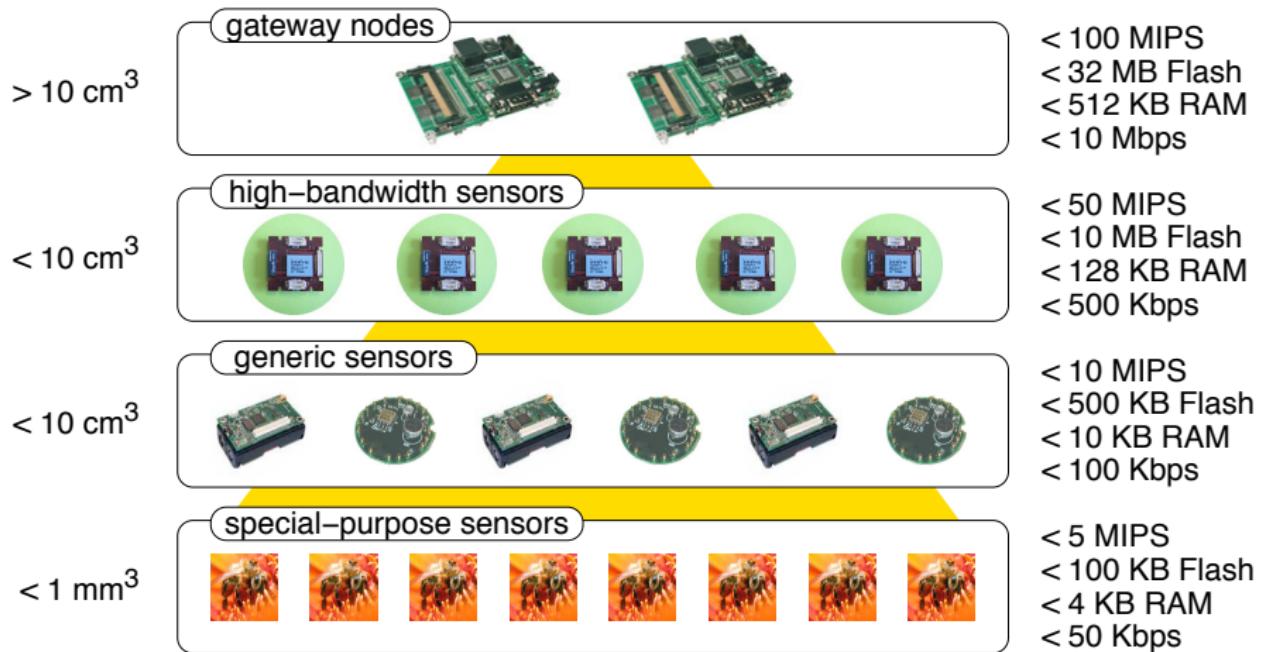


$\mu$ Controller von Sand-/Staubkorngröße, die über Radiofrequenztechnik miteinander kommunizieren [7]:

- ▶ jeder einzelne Kleinstrechner bildet einen kubischen Sensor (*mote*)
- ▶ u.A. gedacht zur Überwachung menschenfeindlicher Umgebungen

# Drahtlose Sensornetze (Forts.)

## Referenzarchitektur und Technologie



# Drahtlose Sensornetze (Forts.)

„Welt am Draht“

$\left\{ \begin{array}{ll} \text{[verteiltes]} & \textit{grid} \\ \text{[durchdringendes]} & \textit{pervasive} \\ \text{[allgegenwärtiges]} & \textit{ubiquitous} \end{array} \right\} \textit{computing} \implies \textcolor{red}{\textit{ambient intelligence}}$

- ▶ nahezu jedes „Gerät“ ist mit Kleinstrechnern (Sensoren, Aktoren) bestückt, die die unbegrenzte globale Vernetzung ermöglichen
- ▶ die Gerätenetze sind in einer Art und Weise in die Umgebung eingebettet, dass ihre Konnektivität jederzeit verfügbar und höchst unaufdringlich ist

Fiktion? Ja, noch ...

# Einbettbare Echtzeitbetriebssysteme

„Kleinvieh macht Mist“

{BlueCat, HardHat} Linux, Embedix, Windows {CE, NT Embedded}, ...

- ▶ Probleme bereiten u.a. Skalierbarkeit und Betriebsmittelbedarf
  - ▶ insbesondere der Bedarf an RAM und Energie...

..., BOSS, C{51, 166, 251}, CMX RTOS, C-Smart/Raven, eCos, eRTOS, Embos, Ercos, Euros Plus, Hi Ross, HyNet-OS, **ITRON**, LynxOS, MicroX/OS-II, Nucleus, OS-9, OSE, **OSEK** {Flex, Plus, Turbo, time}, Precise/{MQX, RTCS}, proOSEK, pSOS, **PURE**, PXROS, **QNX**, Realos, RTMOSxx, Real Time Architect, RTA, RTOS-UH, RTX{51, 166, 251}, RTXC, Softune, SSXS RTOS, ThreadX, TinyOS, VRTX, **VxWorks**, ...

- ▶ der Anteil proprietärer Lösungen liegt bei über 50 % [8]
  - ▶ in vielen Fällen wird das Rad neu erfunden...

# Resümee

Anforderungen funktionaler und nicht-funktionaler Art

- ▶ Aufzeichnung, Vergütung, Überwachung (dig. Kontrolle)
- ▶ Interaktion Mensch/Maschine
- ▶ Rechtzeitigkeit und Verlässlichkeit

Klassifikation nach äußeren und inneren Faktoren

- ▶ *hard/soft real-time, fail safe/operational*
- ▶ *guaranteed response/best effort, resource adequate/inadequate*
- ▶ *event/time triggered*

eingebettete Systeme sind spezielle Echtzeitsysteme

# Literaturverzeichnis

- [1] Hermann Kopetz.  
*Real-Time Systems: Design Principles for Distributed Embedded Applications.*  
Kluwer Academic Publishers, 1997.
- [2] IFIP.  
Working Group 10.4 on Dependable Computing and Fault Tolerance.  
<http://www.dependability.org/wg10.4>, 2003.
- [3] Jane W. S. Liu.  
*Real-Time Systems.*  
Prentice-Hall, Inc., 2000.
- [4] David E. Simon.  
*An Embedded Software Primer.*  
Addison-Wesley, 1999.

## Literaturverzeichnis (Forts.)

- [5] DaimlerChrysler AG.  
Der neue Maybach.  
*ATZ/MTZ Sonderheft*, page 125, September 2002.
- [6] David Tennenhouse.  
Proactive computing.  
*Communications of the ACM*, 43(5):43–50, May 2000.
- [7] David E. Culler and Wei Hong.  
Wireless sensor networks — introduction.  
*Communications of the ACM*, 47(6):30–33, June 2004.
- [8] Collin Walls.  
The perfect RTOS.  
In *Proceedings of the embedded world 2004*, Nürnberg, 2004.