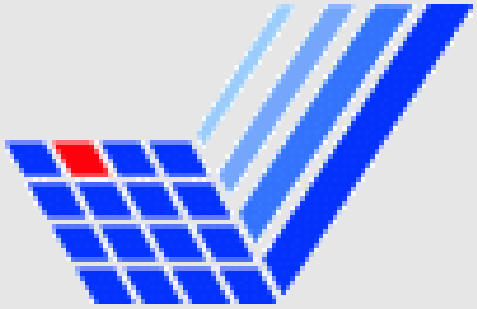
Zeit- und ereignisgesteuerte Echtzeitsysteme

Stephan Braun

Stephan.Braun.Hagen@t-online.de

PG AutoLab
Seminarwochenende 21.-23. Oktober 2007

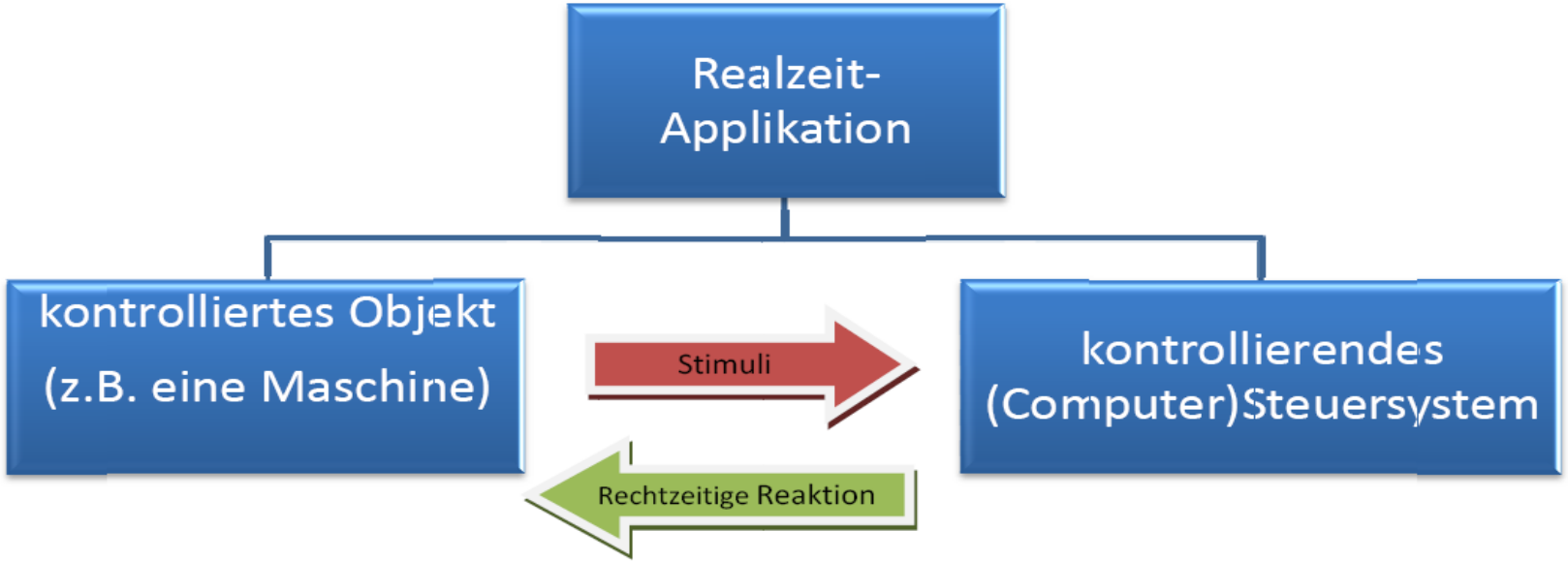


Überblick

- Echtzeitsystemmodell
- Einführung Ereignis- und zeitgesteuerte Systeme
- Ereignisgesteuerte Systeme
- Zeitgesteuerte Systeme
- Zeit- und ereignisgesteuerte Systeme im Vergleich
- Literatur

Echtzeitsystemmodell

- Echtzeitsysteme werden in Applikationen zur Kontrolle realer technischer Systeme verwendet

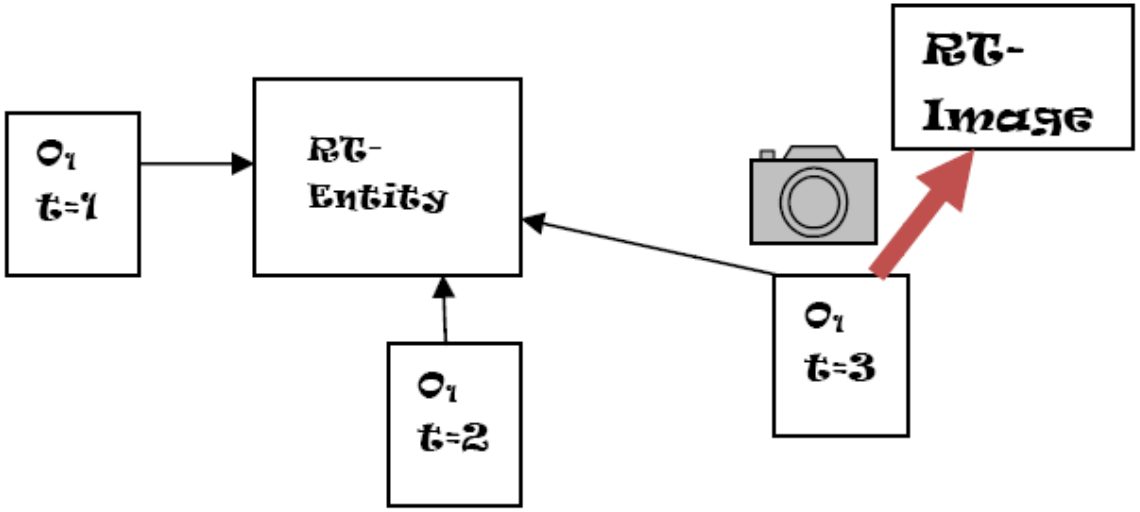


Echtzeitsystemmodell

- **Verhalten der Echtzeitappliaktion:**
  - Modellierung des Verhaltens mit Hilfe von Prozessen
  - Berechnungen anhand von zeitabhängigen, signifikanten Zustandsvariablen (= RT-Entities)
- Auto: Position, Geschwindigkeit, Stellung der Scheibenwischer, Zylinderstellung...
  - *statische* Attribute: Name, Typ, Wertebereich
  - *dynamische* Attribute: Wertebelegung
- Jede RT-Entity ist im Einflussbereich (sphere of control, SOC), eines Subsystems, dort werden Werte festgesetzt.
- Außerhalb: RT-Entities nur observierbar, nicht modifizierbar

Echtzeitsystemmodell

- Funktionale Anforderung: Beobachten der RT-Entitites und Sammeln dieser Beobachtungen (Observation)
- Observation: atomare Datenstruktur O=<Name, t, Wert>
- Repräsentation durch RT-Image:
  - hat beschränkte Gültigkeit
  - muss nach Ablauf dieser ersetzt werden



Echtzeitsystemmodell

- RT-Database:
  - Menge zeitlich gültiger RT-Images
  - RT-Entity ändert Wert → RT-Database aktualisieren

- Zwei Ansätze:
  - *periodisches* Update (zeitgesteuert/time-triggered)
  - Update *nach Zustandsänderung* (ereignisgesteuert/event triggered)

- RT-Objekte: Container für RT-Images innerhalb eines Computerknotens
- RT-Images werden dort manipuliert und gespeichert

Ereignis- und zeitgesteuerte Systeme

- Abhängig von den **Auslösemechanismen** für Kommunikations- und Verarbeitungsaktivitäten, zwei unterschiedliche Ansätze:

1.) **ereignisgesteuert** (event-triggert):

Alle Aktivitäten werden gestartet, sobald eine signifikante Zustandsänderung (z.B. ein Ereignis in einer RT-Entity) bemerkt wird

2.) **zeitgesteuert** (time-triggered):

Alle Aktivitäten werden periodisch zu bestimmten Zeitpunkten gestartet

Ereignisgesteuerte Systeme

- Echtzeitsystem ist **ereignisgesteuert** (event-triggered), wenn Steuerungssignale vom Auftreten von Ereignissen abhängen:
  - Empfang einer Nachricht
  - Beendigung eines Tasks
  - Externer Interrupt

→ alle Aktivitäten werden durch ein Ereignis angestoßen

Ereignisgesteuerte Systeme

- Zwei unterschiedliche Ereignisarten:

1.) vorhersagbare Ereignisse (*predictable events*; P-events):

Grundlage z.B. physikalische Gesetze: Ereignis ist Konsequenz eines anderen Ereignisses

→ Auftreten ist **deterministisch**; Ressourcen frühzeitig reservierbar

2.) Zufällige Ereignisse (*chance events*; C-events):

Auftreten hängt von zufälligem Prozess ab

→ **nicht deterministisch**; statistische Hilfsmittel nötig, um diese im Systementwurf zu berücksichtigen.

Ereignisgesteuerte Systeme

- Scheduling-Strategie:
  - Dienste sind nachfrage-orientiert, reagieren auf plötzlich auftretende Ereignisse

    → dynamische (*online*) Scheduling Strategie:

    - Task-Organisation zur Laufzeit → flexibel
    - Problem: Tasks i.d.R. durch Abhängigkeitsrelationen gekennzeichnet, z.B.:
      - Gegenseitiger Ausschluss
      - Ressourcenabhängigkeiten

Ereignisgesteuerte Systeme

- Dynamische Algorithmen:
  - Earliest Deadline First (EDF)
  - Least Laxity First (LLF)

  - Deadline Monotonic (DM), Rate Monotonic (RM):

    → *Statisches Prioritätsscheduling*:

    Prozesse bekommen statische Prioritäten zugewiesen:
    - wichtige Prozesse: hohe Priorität
    - unkritische Prozesse: niedrige Priorität

    → Prozess darf nur von anderem mit höherer Priorität unterbrochen werden

Ereignisgesteuerte Systeme

- *Fehlertoleranz* → Fehlfunktionen so abfangen, dass das gewünschte Verhalten gesichert bleibt

  - Hier: durch aktive Redundanz: Bei Fehlern übernimmt sein Replikat, der ständig aktiviert ist, seine Aufgaben
  - Replikation der (Funktionen) von Knoten erfordert **replica determinism**: korrekt replizierte Systeme verhalten sich komplett identisch

  - **Problem**: Quittungen, Timeouts und wiederholte Übermittlungen → Verhalten kann sich unterscheiden

(1 Replikat hat Quittung vor Timeout erhalten, ein anderer aber nicht)

→ Abstimmung, um Zustandssynchronität zu erlangen

Zeitgesteuerte Systeme

- Echtzeitsysteme sind **zeitgesteuert** (time-triggered), wenn die Steuerungssignale vom Fortschritt einer (globalen) Zeitnotation ausgehen.

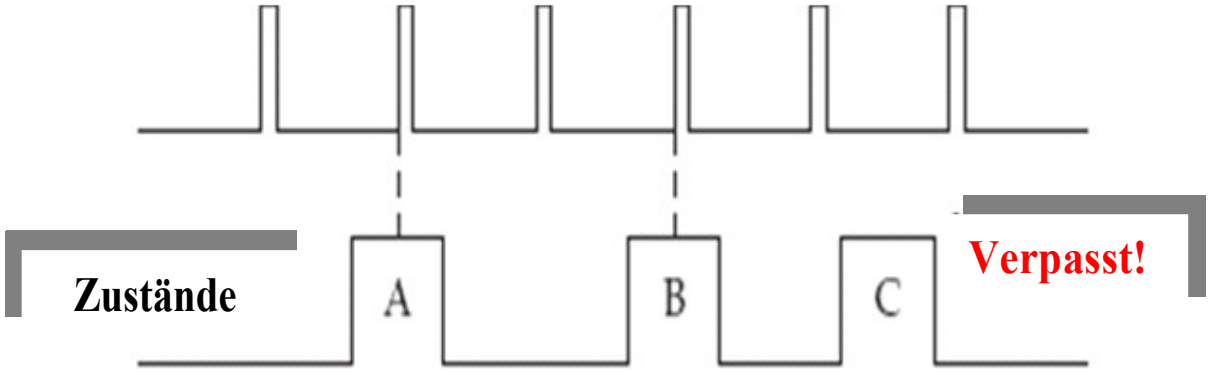
  → **Ziel**: schnelle Übermittlung der Zustandswerte

- Zustandsübermittlung wird *periodisch* vorgenommen; die Periode wird auf das dynamische Verhalten der RT-Entities abgestimmt.

  → Beobachtungsgitter (*observation lattice*):
  - bestimmt Eckpunkte der Übermittlung;
  - RT-Entities nur zu diesen Zeitpunkten überwacht

Zeitgesteuerte Systeme

- Wahl der Gitterpunkte problematisch:
  - Punkte eng genug wählen, so dass kurzlebige Zustände entdeckt werden:



→ Zwischenspeicherung der Zustände

  - Punkte nicht zu eng wählen: Zustand muss sich u.U. erst stabilisieren

Zeitgesteuerte Systeme

- ***Scheduling-Strategie***:
  - *statische*, vorgegebene Schedules
  - Berücksichtigung aller Abhängigkeiten
  - in Entwicklungsphase erstellt (*offline*)

  →zur Laufzeit: „table-look-up“ → ausführbarer Task
  - alle Input-/Outputaktivitäten vorgeplant

Zeitgesteuerte Systeme

- ***Fehlertoleranz***:
  - alle Taskwechsel, Modiwechsel und Kommunikations-aktivitäten global synchronisiert

  → Vermeidung nichtdeterministischer Entscheidungen
  - keine zusätzliche Kommunikation unter den Replikaten nötig

Zeit- und ereignisgesteuerte Systeme im Vergleich

1.) *Vorhersagbarkeit*:

- Zeitgesteuerte Systeme:
  - gründliche Planung der Abläufe in Entwicklungsphase:
    - Festlegung des observation lattice
    - Schätzung der maximalen Ausführungszeit aller zeitkritischen Tasks
    - Schedules offline erstellen

    → komplettes zeitliches Systemverhalten präzise vorhersagbar
- Ereignisgesteuerte Systeme:
  - Pläne dynamisch aufgrund expliziter Nachfrage erzeugt - verschiedene dynamsiche Ablaufpläne

    → Verhalten nicht deterministisch vorhersagbar

Zeit- und ereignisgesteuerte Systeme im Vergleich

2.) *Testbarkeit*: zeitliche Performanz ist Hauptaspekt

- Ereignisgesteuerte Systeme:
  - Test unter simulierter Spitzenauslastung, da seltene C-Events u.U. sonst nicht häufig genug auftreten
  - Änderung eines Tasks kann Konsequenzen für andere Tasks haben

→ aufwändige Tests nötig; kritischer Punkt: simulierte Auslastung = Realität?

- Zeitgesteuerte Systeme:
  - Vergleich der Testergebnisse mit detaillierten Entwurfsplänen
  - diskrete Abfrageraster → ~~*endliche*~~ Menge: jede Reaktion auf Input observierbar, wertmäßig erfassbar und vergleichbar

→ systematischer Test

Zeit- und ereignisgesteuerte Systeme im Vergleich

3.) *Ressourcennutzung*:

- Zeitgesteuerte Systeme:
  - alle Schedules fest vorgegeben;
  - Zeitfenster für jeweiligen Task mind. so groß wie seine max. Ausführungszeit;
  - ist die Differenz zw. durchschnittlicher und max. Ausführungszeit groß: nur kleiner Teil der Berechnungszeit benötigt → feste maximale Ausführungszeit = u.U. „Zeitverschwendung“
- Ereignisgesteuerte Systeme:
  - Schedules nur für die Tasks, die durch die jeweiligen Umstände angestoßen werden (dynamisch) → CPU bereits nach der tatsächlichen und nicht der maximalen Ausführungszeit wieder zur Verfügung

Zeit- und ereignisgesteuerte Systeme im Vergleich

- zusätzliche Laufzeitressourcen für dynamische Scheduling-Algorithmen, Synchronisation und Speichermanagement

- *Niedrige* oder *durchschnittliche* Systemauslastung:

→ Ressourcennutzung bei ereignisgesteuerten Systemen besser (tatsächliche Ausführungszeit besser als maximale)

- Spitzenauslastung:

→ Situation kann sich umkehren, da mehr Zeit für Abwickeln der Interrupts, das Speichermanagement, die Synchronisation und die Scheduling-Algorithmen benötigt wird

Zeit- und ereignisgesteuerte Systeme im Vergleich

4.) *Erweiterbarkeit:*

= Kosten, um bestehende Funktionalität zu ändern und neue Funktionen hinzuzufügen

- Ereignisgesteuerte Systeme:
  - **Ändern** bereits bestehender Tasks/Knoten:

    alle Schedulingentscheidungen erst zur Laufzeit → relativ einfach
  - **Hinzufügen** eines neuen Tasks/Knotens:

    Modifikation einiger Protokollparameter; lokale Änderung der Ausführungszeit kann Auswirkungen auf andere Tasks/Knoten haben → komplettes System erneut testen

Zeit- und ereignisgesteuerte Systeme im Vergleich

- Zeitgesteuerte Systeme:
  - **Ändern** bereits bestehender Tasks/Knoten:

jeder Task eigenständig → ist Ausführungszeit des geänderten Tasks nicht größer als maximale Ausführungszeit: keine Auswirkungen

  - **Hinzufügen** eines neuen Tasks/Knotens:

Aufwand abhängig von Informationsfluss zum/vom Knoten:

- Knoten *passiv* (z.B. Display) → keine Modifikation nötig
- Knoten *aktiv* → Zuweisung eines Kommunikationsfensters

→ Schedules neu berechnen

Zeit- und ereignisgesteuerte Systeme im Vergleich

Eigenschaft	*zeitgesteuertes System*	*ereignisgesteuertes System*
zeitliche Vorhersagbarkeit	+	-
dynamische Ressourcenzuweisung	-	+
Determinismus	+	-
Flexibilität	-	+

Fazit:

→ zeitgesteuerte Systeme verlangen sehr detaillierte Entwicklungsphase (Festlegung der maximalen Ausführungszeit, Erstellung der Schedules); Zeitverhalten gut vorhersagbar

→ ereignisgesteuerte Systeme verlangen keine detaillierten Pläne, allerdings bei Verifikation umfangreiche Systemtests nötig

Literatur

- Weiterführende Literatur:

- H. Kopetz, „*Real-Time Systems: Design Principles for Distributed Embedded Applications*“
- H. Kopetz, „*Event-Triggered versus Time-Triggered Real-Time Systems*“
- H. Kopetz, „*Should Responsive Systems be Event-Triggered or Time-Triggered?*“
- R. Williams, „*Real-time Systems Development*“, Elsevier 2006
- H. Wörn, U. Brinkschulte, „*Echtzeitsysteme*“, Springer 2005
- M. Oetken, Seminar „*Echtzeitbetriebssystemkerne*“, 1997