Eigenschaften eines Echtzeitsystems

Was ist ein Echtzeitsystem ?

Echtzeitsystem
Def. nach DIN 443000

Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, daß die Verarbeitungs-ergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.

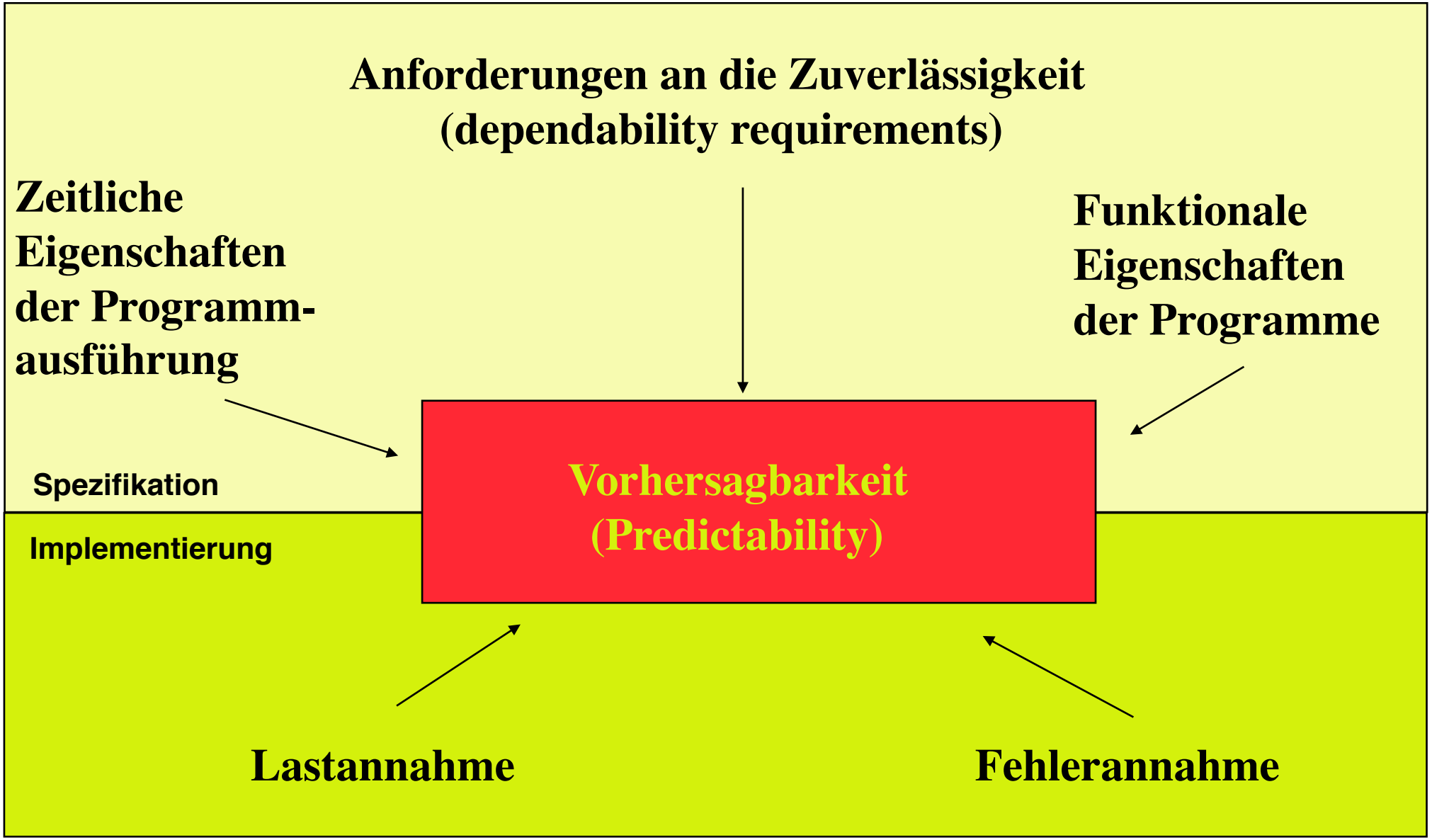
Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.

Was ist ein Echtzeitsystem ?

Lastannahme:
definiert die angenommene Spitzenbelastung, die durch die reale Umgebung, in die das System eingebettet ist, erzeugt wird.
!! Statistische Argumente bzgl. einer geringen Wahrscheinlichkeit des gleichzeitigen Auftretens unabhängiger Ereignisse sind nicht zulässig. Beisp. In besonders kritischen Situationen treten meist sehr viele zeitlich korrelierte Ereignisse ein.

Fehlerannahme:
definiert die Art und die Anzahl der Fehler und welche Funktionalität unter dieser Annahme aufrecht erhalten wird.
!! Das System muß im schlimmsten Fall (worst case) Spitzenlast und maximale Anzahl von Fehlern handhaben.

Geltungsbereich der Annahmen:
(Assumption Coverage)
Wahrscheinlichkeit, daß die Annahmen mit der Wirklichkeit, d.h. dem Verhalten der Umgebung, übereinstimmen

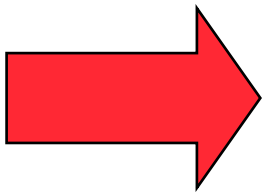

Anforderungen an die Zuverlässigkeit
(dependability requirements)
Zeitliche
Eigenschaften
der Programm-
ausführung
Funktionale
Eigenschaften
der Programme
Spezifikation
Vorhersagbarkeit
(Predictability)
Implementierung
Lastannahme
Fehlerannahme

Was ist ein Echtzeitsystem ?

Echtzeitsystem:

Ein Echtzeitsystem hat unter einer gegebenen Lastannahme und Fehlerannahme ein funktional und zeitlich ***vorhersagbares*** Verhalten.

Exakte zeitliche Vorhersagbarkeit **für individuelle Services ist die Eigenschaft, die ein RTS von einem "Nicht-RTS" unterscheidet !**



Zeit ist ein expliziter Entwurfsparameter

Zeitliche Anforderungen

- **Zeitliche Gültigkeit der Echtzeitdaten.**

  Wenn Daten verarbeitet oder angezeigt werden, müssen sie zeitliche Gültigkeit besitzen.
  ⇒ Echtzeitdaten altern!

- **Maximale Antwortzeiten.**

  Die maximalen Ausführungszeiten für individuelle Berechnungen (eingeschlossen Stimulus-Berechnung-Antwort-Zyklen) müssen beschränkt und bekannt sei.

- **Zeitliche Vorhersagbarkeit**

  Das zeitliche Verhalten des Gesamtsystems muß in allen Situationen exakt vorhersagbar sein, insbesondere auch unter Bedingungen, die sehr selten auftreten.
  ⇒ Planung

Garantien und “Best Effort”

Eine zeitliche Garantie bedeutet, daß für eine bestimmte Realisierung eines RT-Systems unter gegebenen Last- und Fehlerannahmen seine zeitliche Korrektheit durch analytische Methoden erwiesen wurde.

Garantie für individuelle Tasks - Jede garantierte Deadline wird eingehalten. Zeitfehler werden im Rahmen der Garantien behandelt.

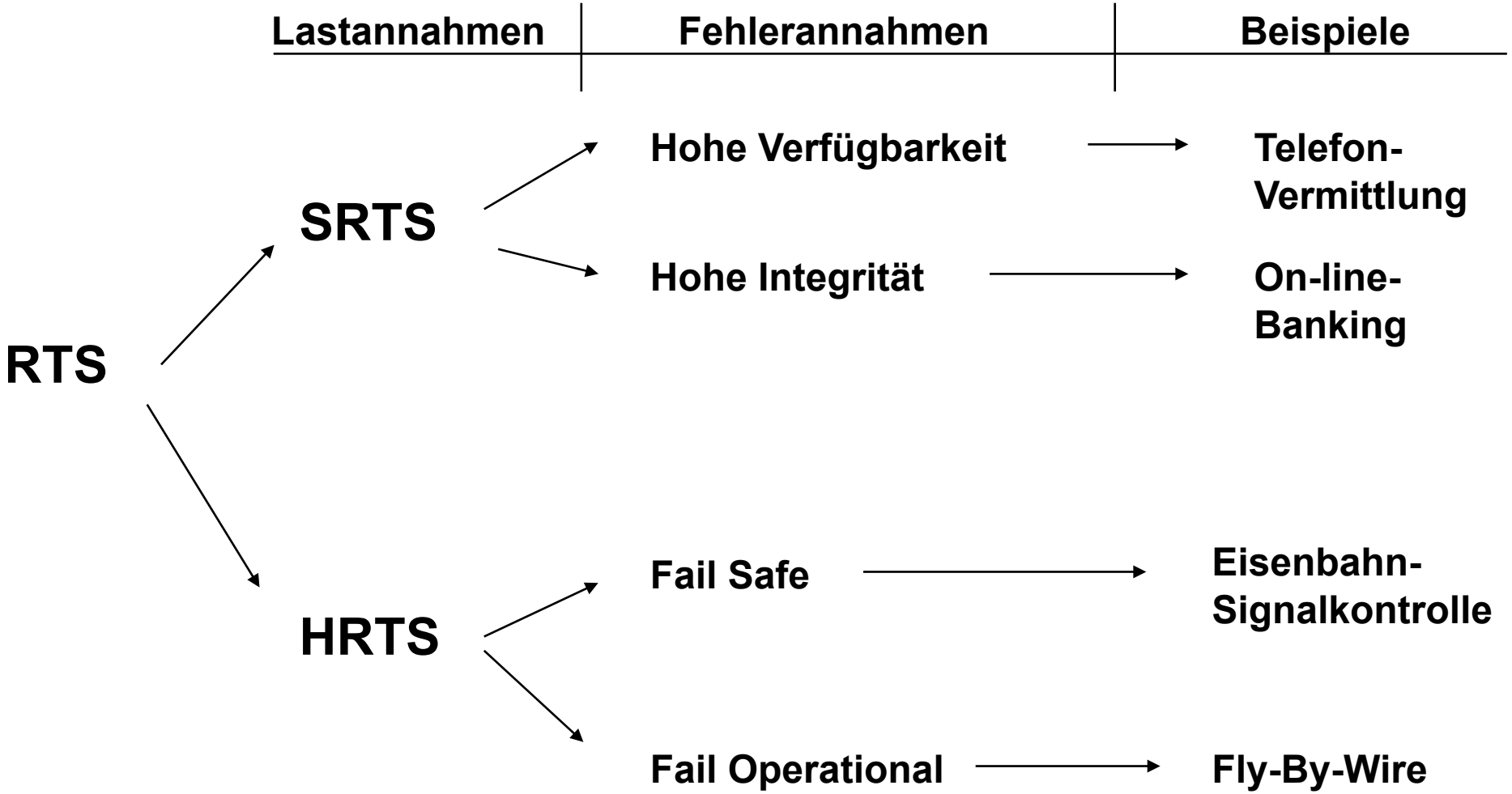
In einem Best Effort System werden keine Garantien gegeben. Die zeitliche Verifikation beruht auf probabilistischen Aussagen auch im Rahmen der angenommenen Last- und Fehlerannahmen.

Globale Garantien für Konsistenz oder maximale Zeitfehler. Verfahren zur Behandlung von Zeitfehlern vorhanden. Meist “gracefully degradable” Funktionalität.

Klassifikation von Echtzeitsystemen

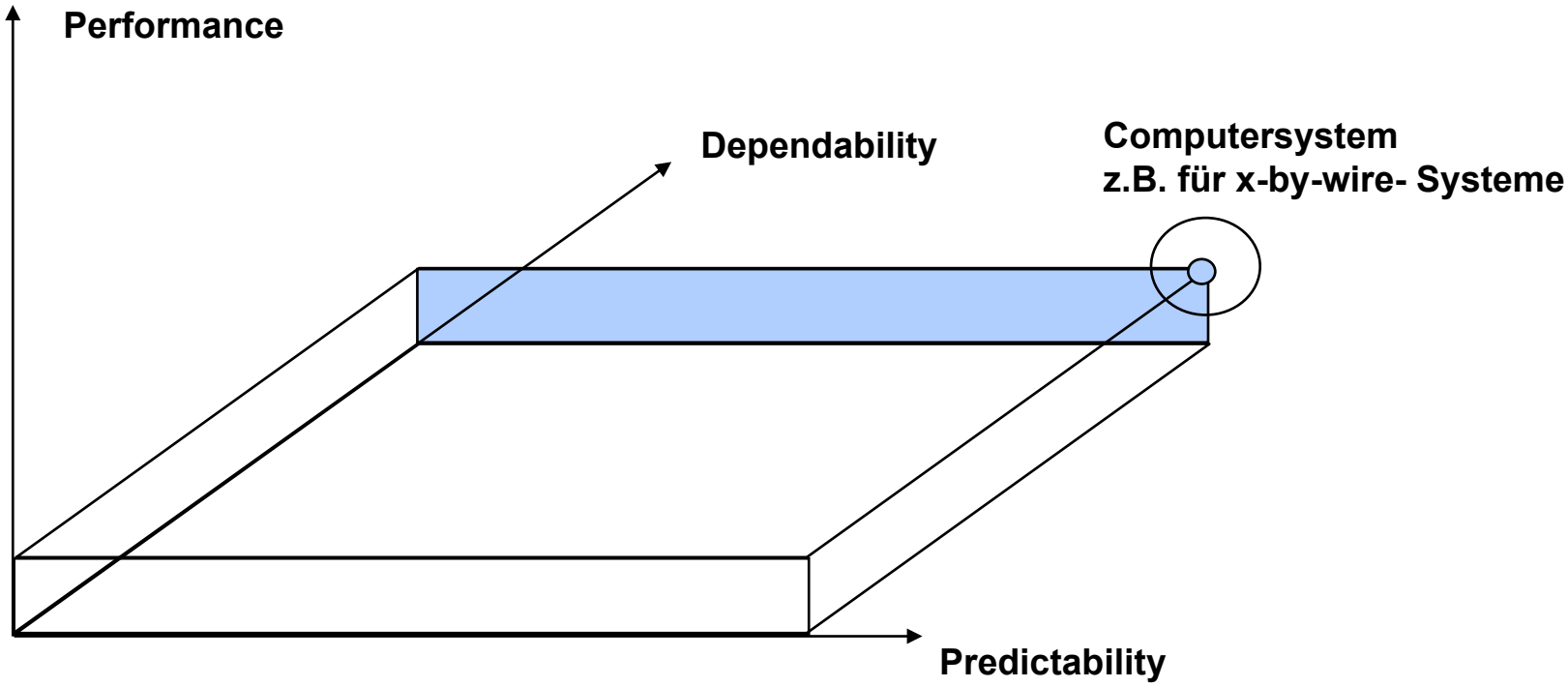
	Hard RT	Soft RT
Garantien	**Steuer- und Überwachungs-Systeme**	**Telefon-Vermittlungs-Systeme**
Best Effort	**nicht sinnvoll**	**Multi-Media Systeme**

Klassifikation von Echtzeitsystemen



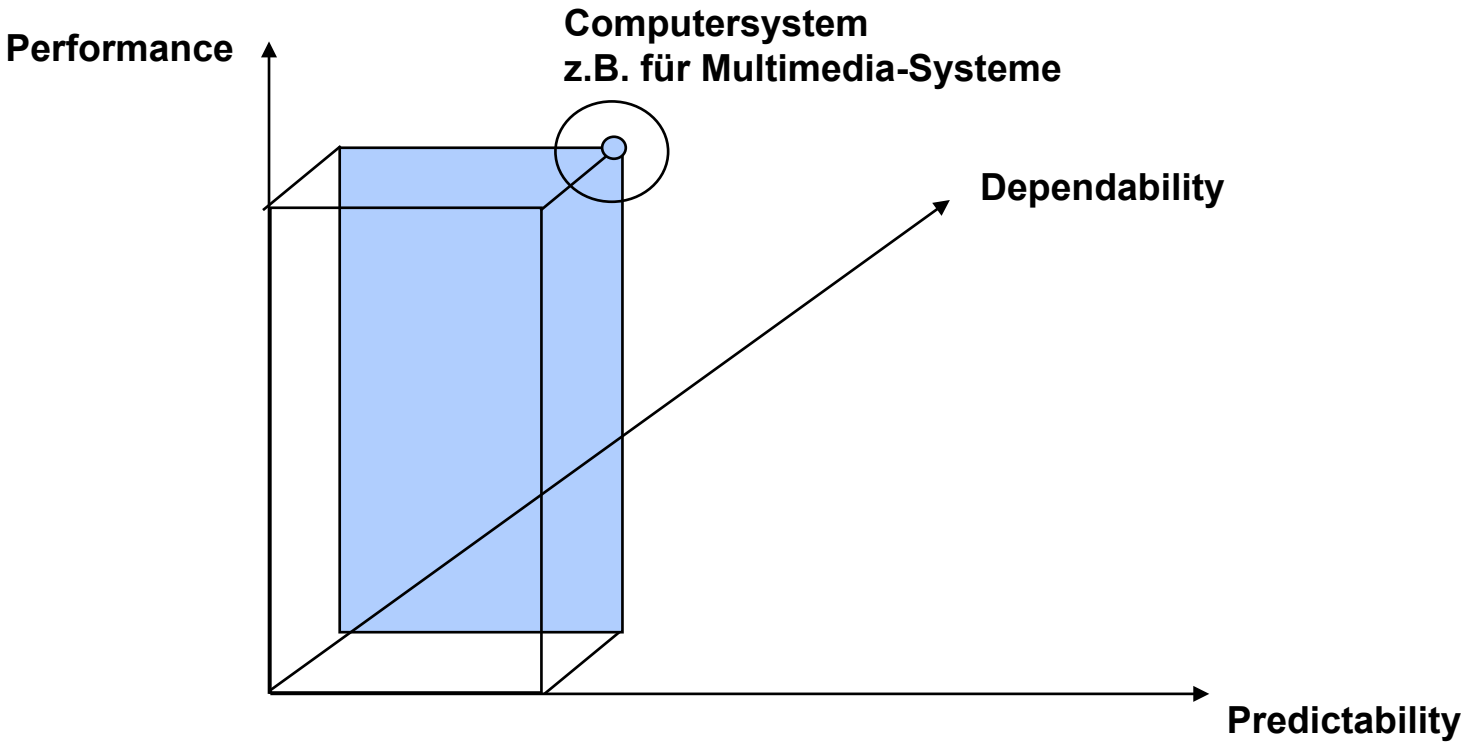
Anforderungen an die Architektur

- **hohe Zuverlässigkeit (Aspekte der Ausfallsicherheit und Prozeßsicherheit im Vordergrund)**
- **hohe Vorhersagbarkeit (individuelle Garantien unter Spitzenlastannahme)**
- **adäquate Leistung**



Anforderungen an die Architektur

- höchste Leistung
- Vorhersagbarkeit unter statistischen Annahmen
- Zuverlässigkeit: Aspekte der Wartbarkeit und Verfügbarkeit stehen im Vordergrund



Wie programmieren?

Spezifikation der Kontrolle

Beispiel: **Hinderniserkennung in einem Roboter**

d_1 und d_2 sind Meßwerte von einem Sensor

s_1 und s_2 sind kritische Abstandsschranken

IF (($d_1 < s_1$) AND ($d_2 < s_2$) THEN " alles o.k"
ELSE " raise alarm"

Ist diese Spezifikation eines Kontrollstatements ausreichend für eine Realzeitumgebung ?

Zusätzliche Angaben:

- In welchen Abständen wird die Distanz gemessen ?
- Wird die Alarmsituation mit jeder Messung überprüft ?
- Welcher maximale Zeitabstand ist zwischen Distanzänderungen und Alarm zulässig?

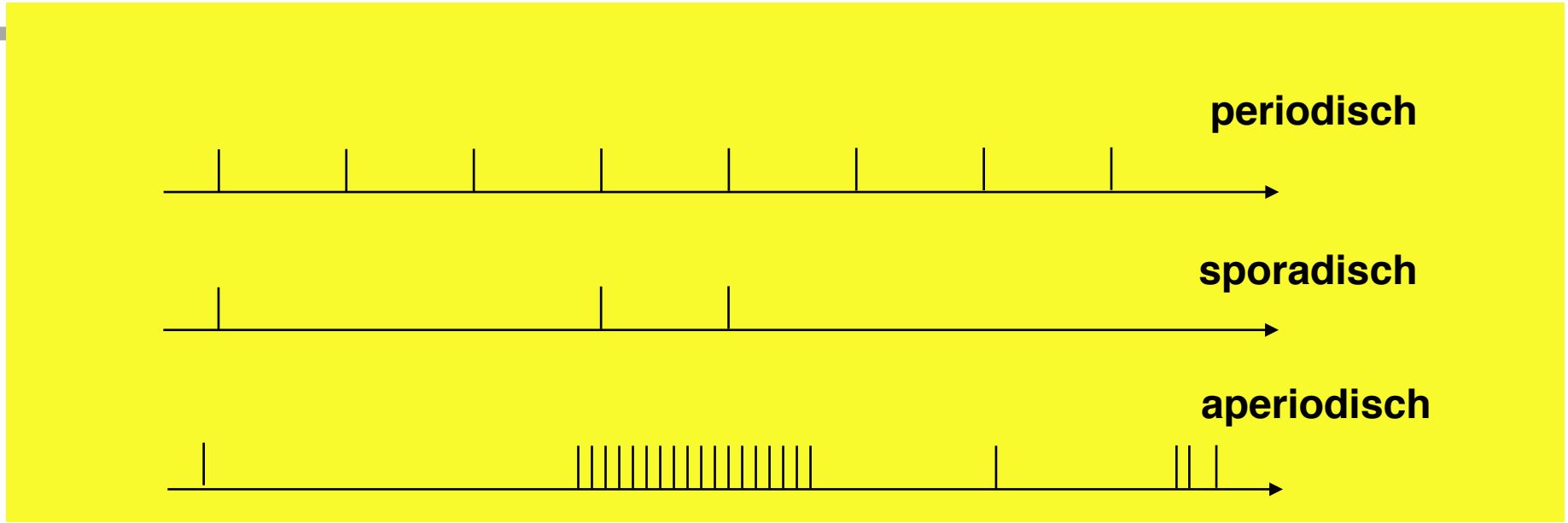
Logische und zeitliche Kontrolle:

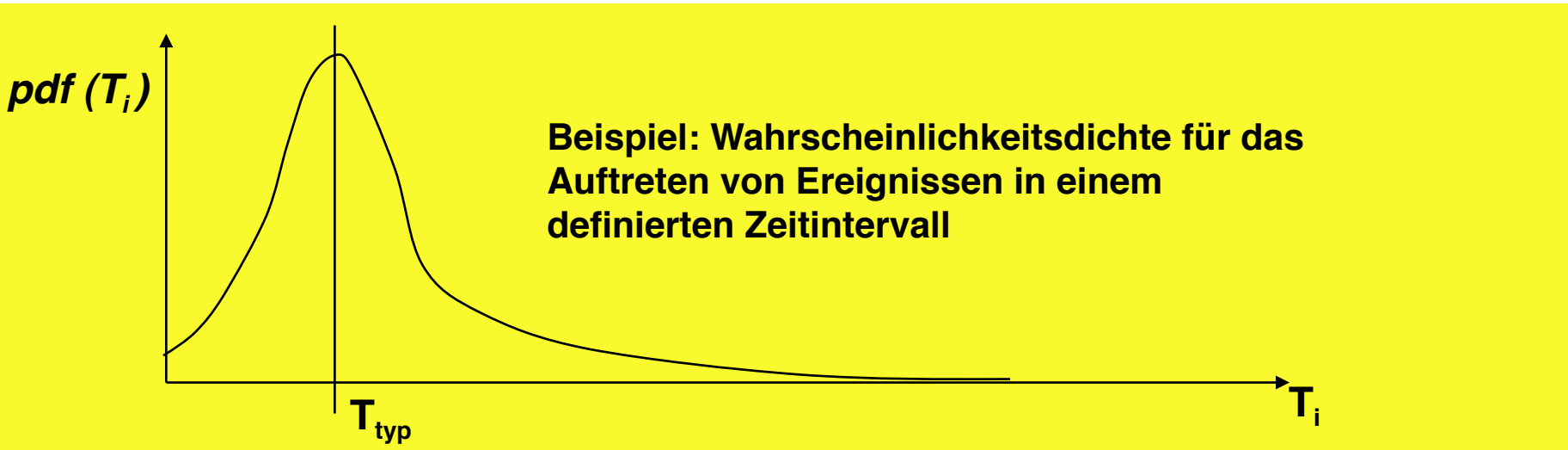
Kontrolle bestimmt den Zeitpunkt an dem eine ausgewählte Aktion startet. In einem Realzeitsystem muß unterschieden werden zwischen:

- **Logischer Kontrolle. Sie definiert den Kontrollfluß in einem Programm um die spezifizierte Datentransformation zu realisieren.**

- **Zeitlicher Kontrolle. Sie bestimmt den Zeitpunkt an dem eine Task gestartet oder von einer wichtigeren Task unterbrochen wird. Zeitliche Kontrolle ist eng verbunden mit der Planung des zeitlichen Ablaufs (Scheduling).**

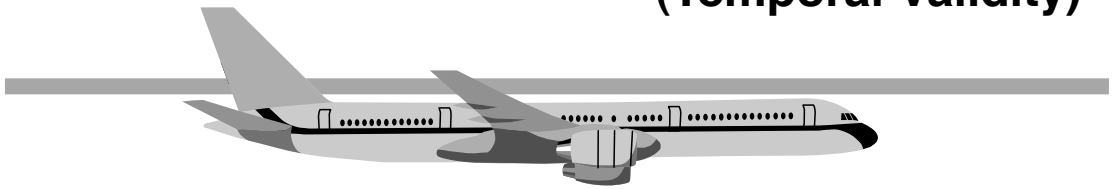
Zeitliche Verteilung der Ereignisse





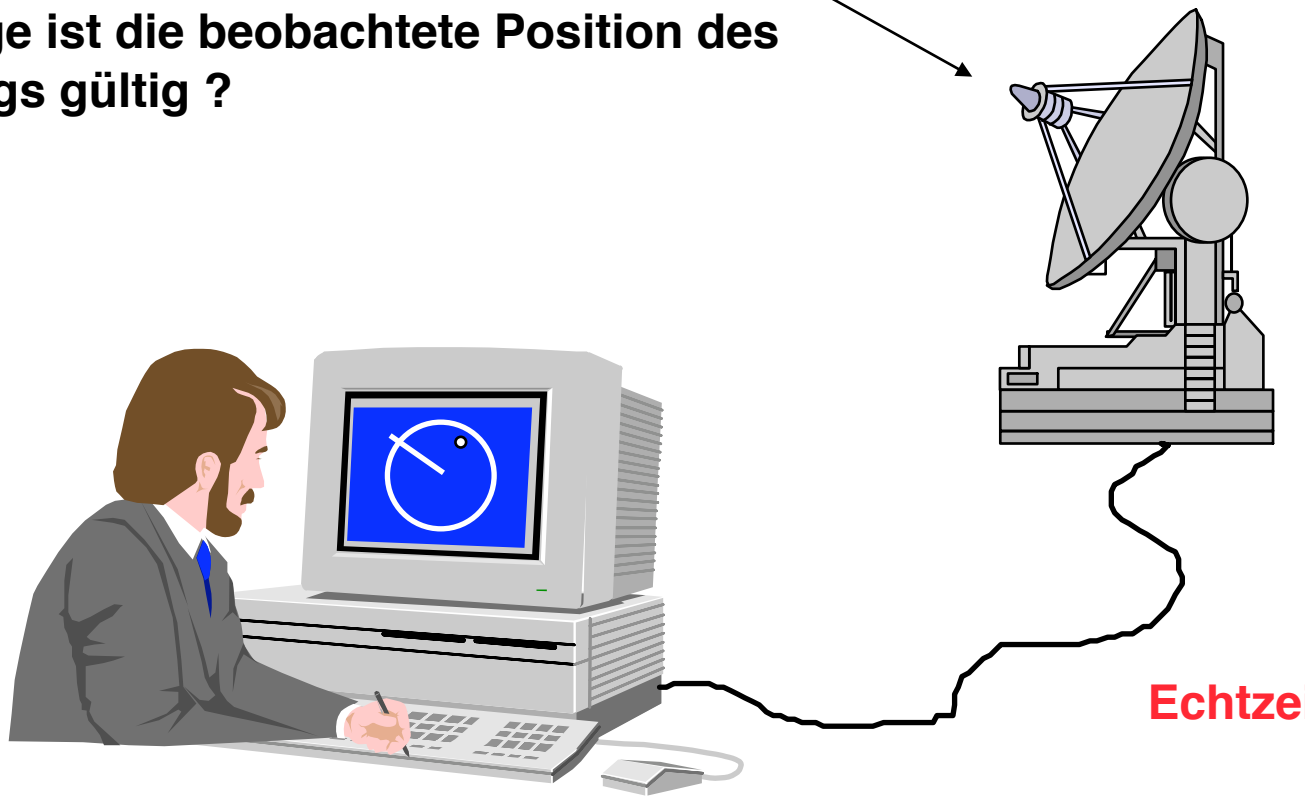
Beispiel: Wahrscheinlichkeitsdichte für das Auftreten von Ereignissen in einem definierten Zeitintervall

Zeitliche Gültigkeit von Echtzeitinformationen (Temporal Validity)



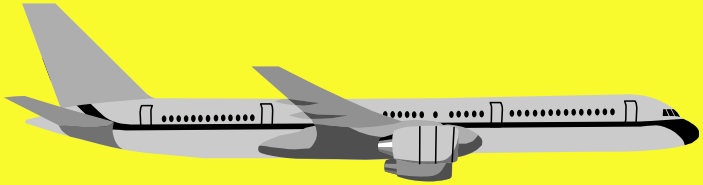
Zeit-Wert-Einheiten (Time-Value Entities):
V = E(t)
V: Beobachteter Wert
E: Echtzeitkomponente

Wie lange ist die beobachtete Position des Flugzeugs gültig ?



Echtzeitdaten altern !

Abschätzung von Ungenauigkeiten



aktuelle Position

?

gemessene Position

re-t

v

i

t

Zeit für die Aufbereitung der Meßdaten

Ungenauigkeiten (Jitter):

re-t : reading error
zeitlicher (!) Meßfehler
i: imprecision
zeitliche Varianz der Verarbeitung
v: variance
Unbestimmtheit des Berechnungsbeginns

Gesamtfehler U = re-v + ce + re-t + i + v

re-v : Meßfehler des Positionswertes
ce: Ungenauigkeiten der Positionsberechnung

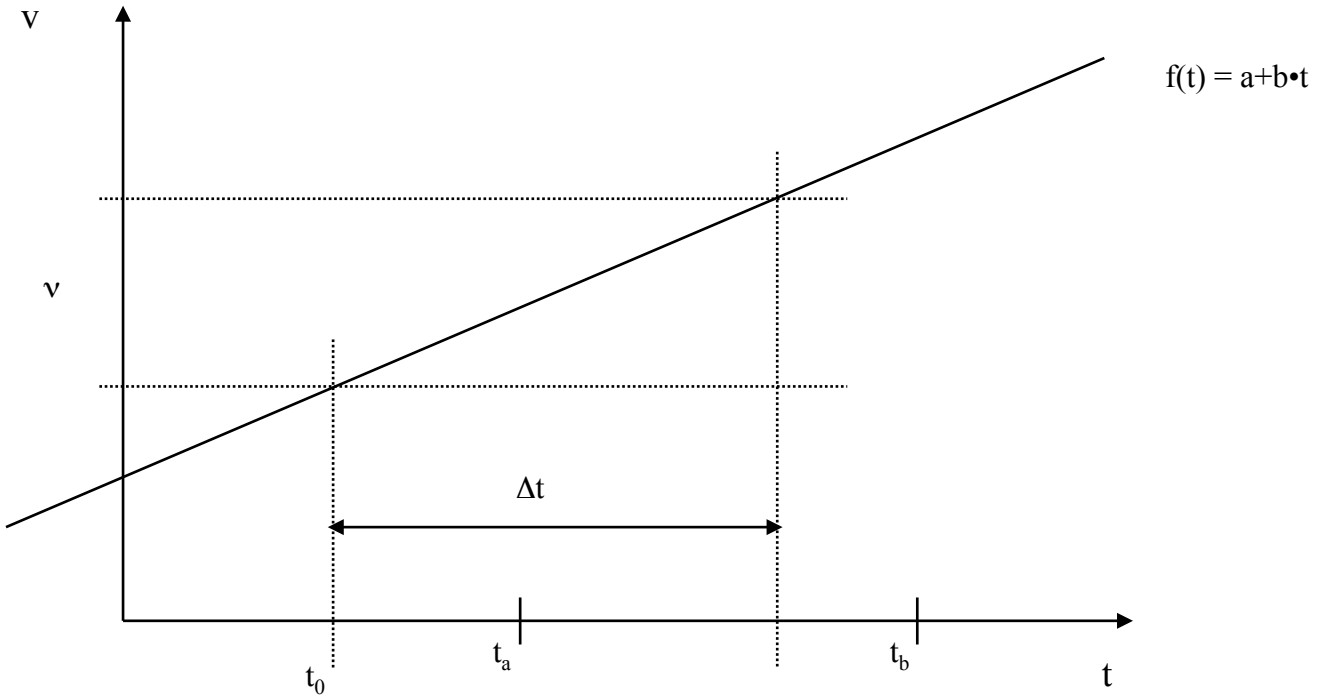
Daraus ergeben sich die Aspekte:

1. **Time of a value:**
   Wie genau kann man den Zeitpunkt einer Beobachtung bestimmen?

   Wie genau kann man eine Beobachtung zu einem bestimmten Zeitpunkt initiieren?

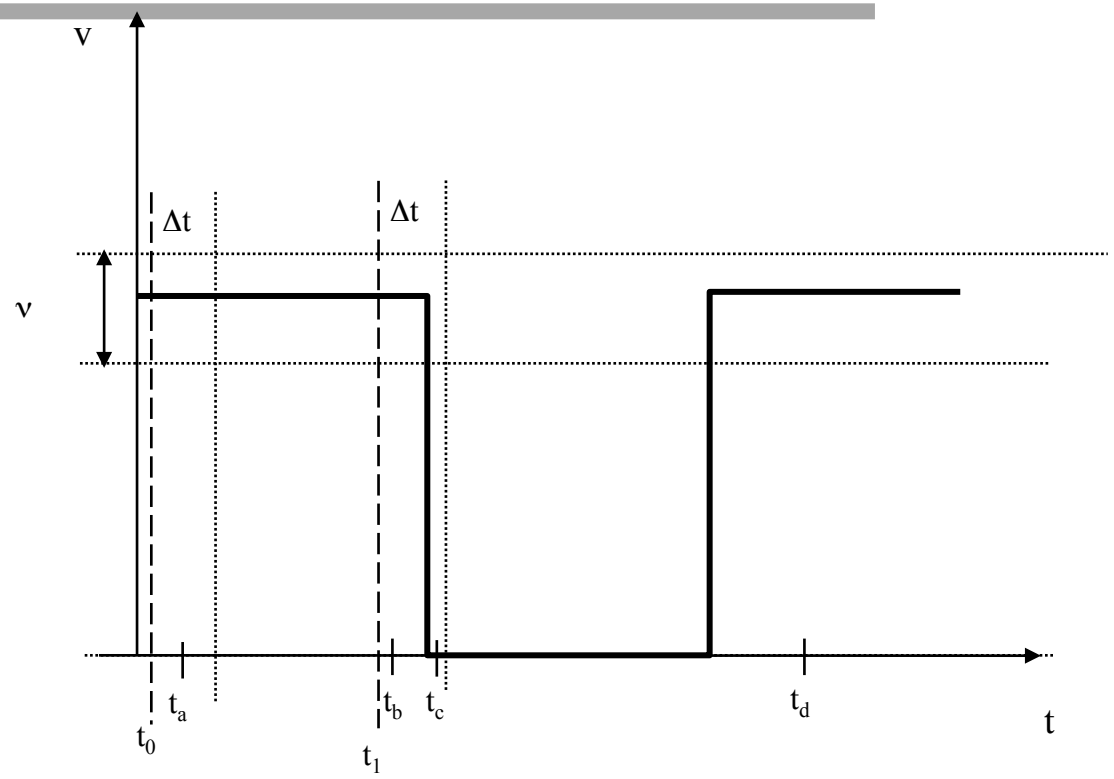
2. **Value over time: Wie ändert sich der Wert während der Behandlung im Rechner?**

Zeitliche Gültigkeit von Sensordaten



t_0 : point of observation
Δt: temporal validity interval
t_a : temporally consistent
t_b : temporally not consistent

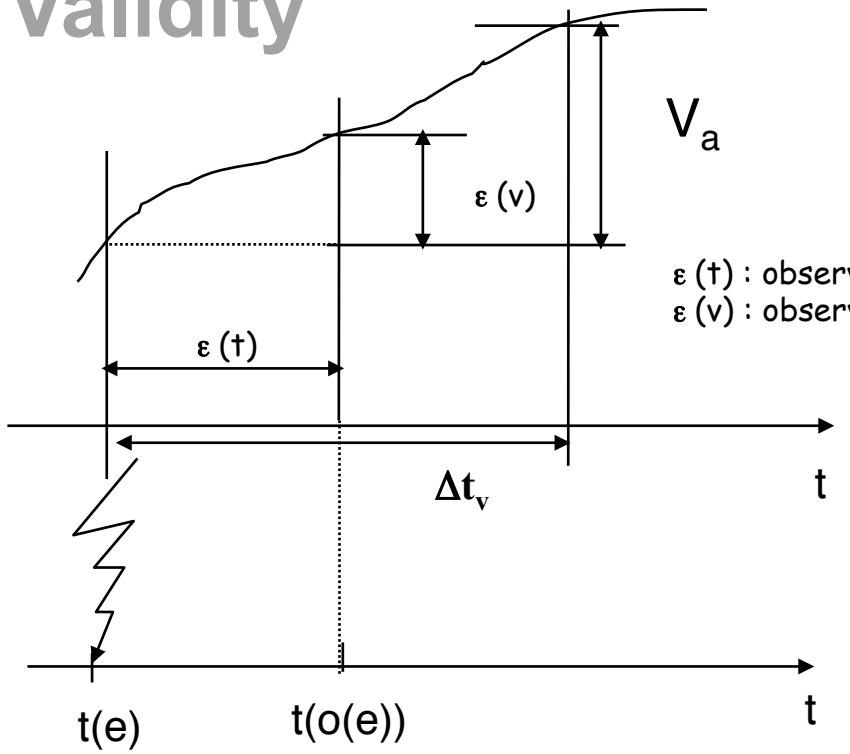
Zeitliche Gültigkeit von Sensordaten



f(t) = { 1 if door is open
0 if door is closed

t_0,t_1 : points of observation
Δt: temporal validity intervals
t_a, t_b, t_d.: temporally consistent
t_c : temporally not consistent

Temporal Validity



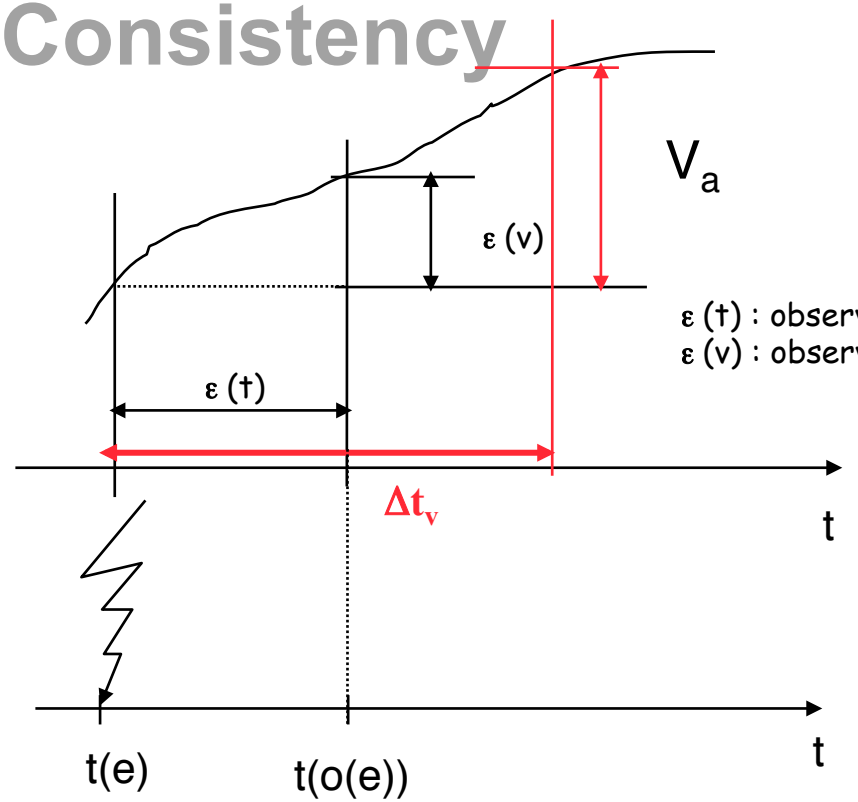
ε (t) : observation error (jitter) in the temporal domain
ε (v) : observation error in the value domain

given a max. acceptable error in the value domain V_a, an observation is valid for all t(o(e)) in the time interval Δt_v iff:

$$|v_{t(e)} - v_{t(o(e))}| \leq V_a$$

Temporal Validity denotes a Time Interval

Temporal Consistency



ε (t) : observation error (jitter) in the temporal domain
ε (v) : observation error in the value domain

given a max. acceptable error in the value domain V_a, an observation at $t(o(e)) \geq t(e)$ is consistent iff:

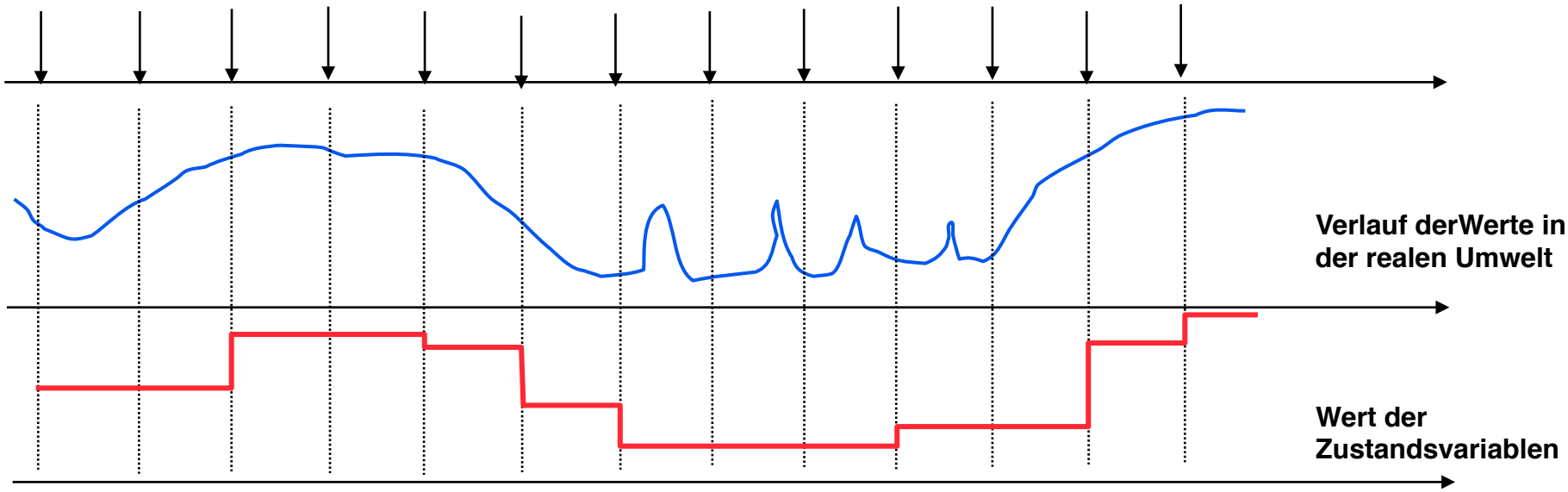
$$|v_{t(e)} - v_{t(o(e))}| \leq V_a$$

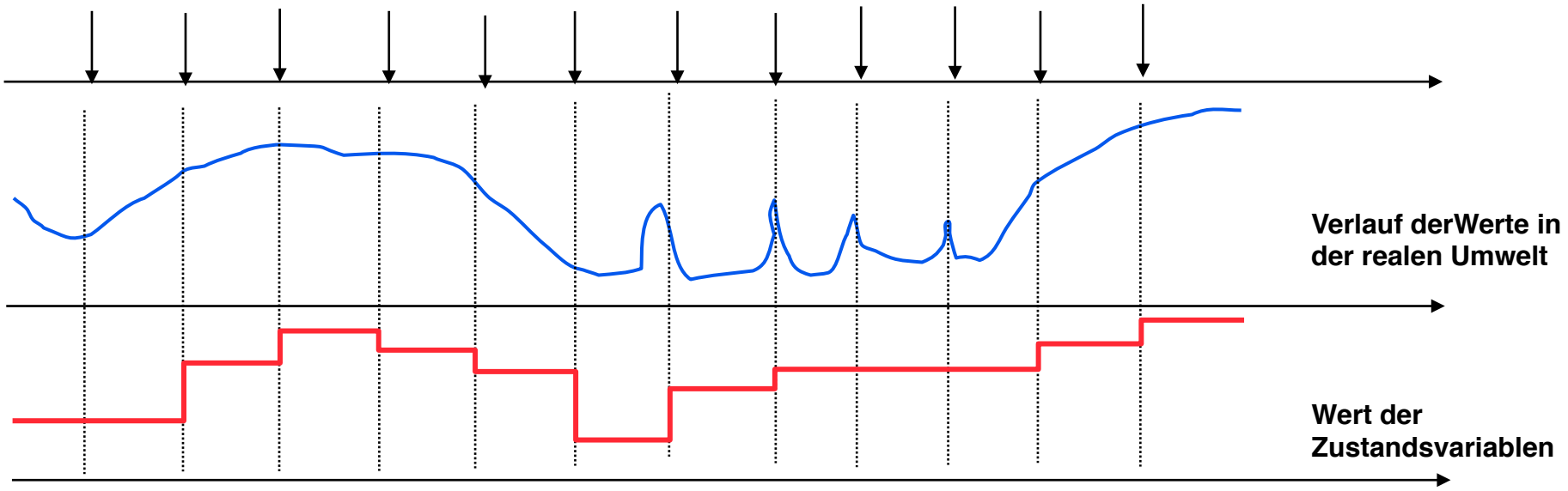
In the temporal validity interval all observations are temporally consistent.

Wie können Objekte der realen Welt im Rechner dargestellt werden ?

Unterscheidung zwischen:

- **Wert, der eine Bedeutung in der physischen Welt hat,**
- **Repräsentation des Wertes, die eine Bedeutung im Rechner hat,**
- **Struktur, in der der Wert oder die Repräsentation gespeichert und verarbeitet wird.**


Verlauf derWerte in
der realen Umwelt
Wert der
Zustandsvariablen


Verlauf derWerte in
der realen Umwelt
Wert der
Zustandsvariablen

Realzeitkomponente und Realzeitrepräsentation

Def.: **Eine Realzeitkomponente (Real-Time Entity RTE) ist eine Größe der „realen Welt“, d.h. im weitesten Sinne, des technische Prozesses. Die Änderung einer RTE ist eine Funktion der Zeit.**

Eigenschaften: **kontinuierlich**
diskret

Def.: **Eine Realzeitrepräsentation (Real-Time Image RTI) ist die Repräsentation einer RTE über die der Zustand einer RTE erfasst wird oder über die eine Zustandsänderung der RTE erwirkt wird.**

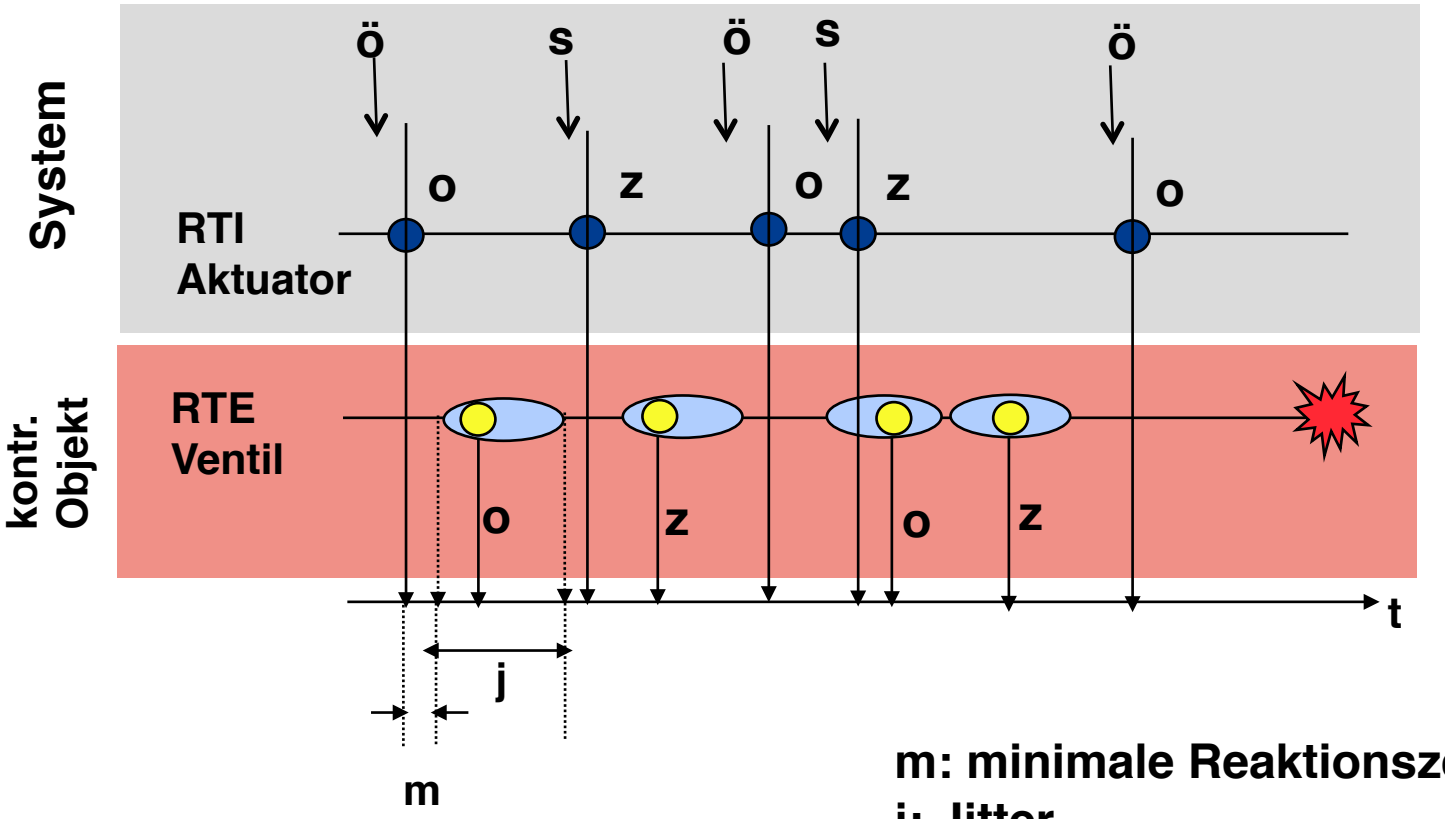
Eigenschaften:

- **Ein RTI ist zu einem bestimmten Zeitpunkt gültig, wenn es sowohl in der Werte- als auch in Zeitdomäne eine adäquate Repräsentation der RTE darstellt.**
- **RTIs sind nur in einem spezifizierten Zeitintervall gültig**
- **RTIs basieren 1. auf einer Beobachtung oder 2. der Annahme über einen Zustand (Beisp. 1. Sensordaten, 2. Annahme über die Stellung eines Motors)**
- **RTIs können im Rechner als RTO gespeichert werden oder außerhalb des Rechners in einem Aktuator.**

Beispiel für den Zusammenhang zwischen RTE und RTI

Ein RTI ist entweder nur beschreibbar (write-only) oder nur lesbar (read-only) aber nicht beides !!

Beispiel: Ventil



Kommando:
ö: öffnen
s: schließen

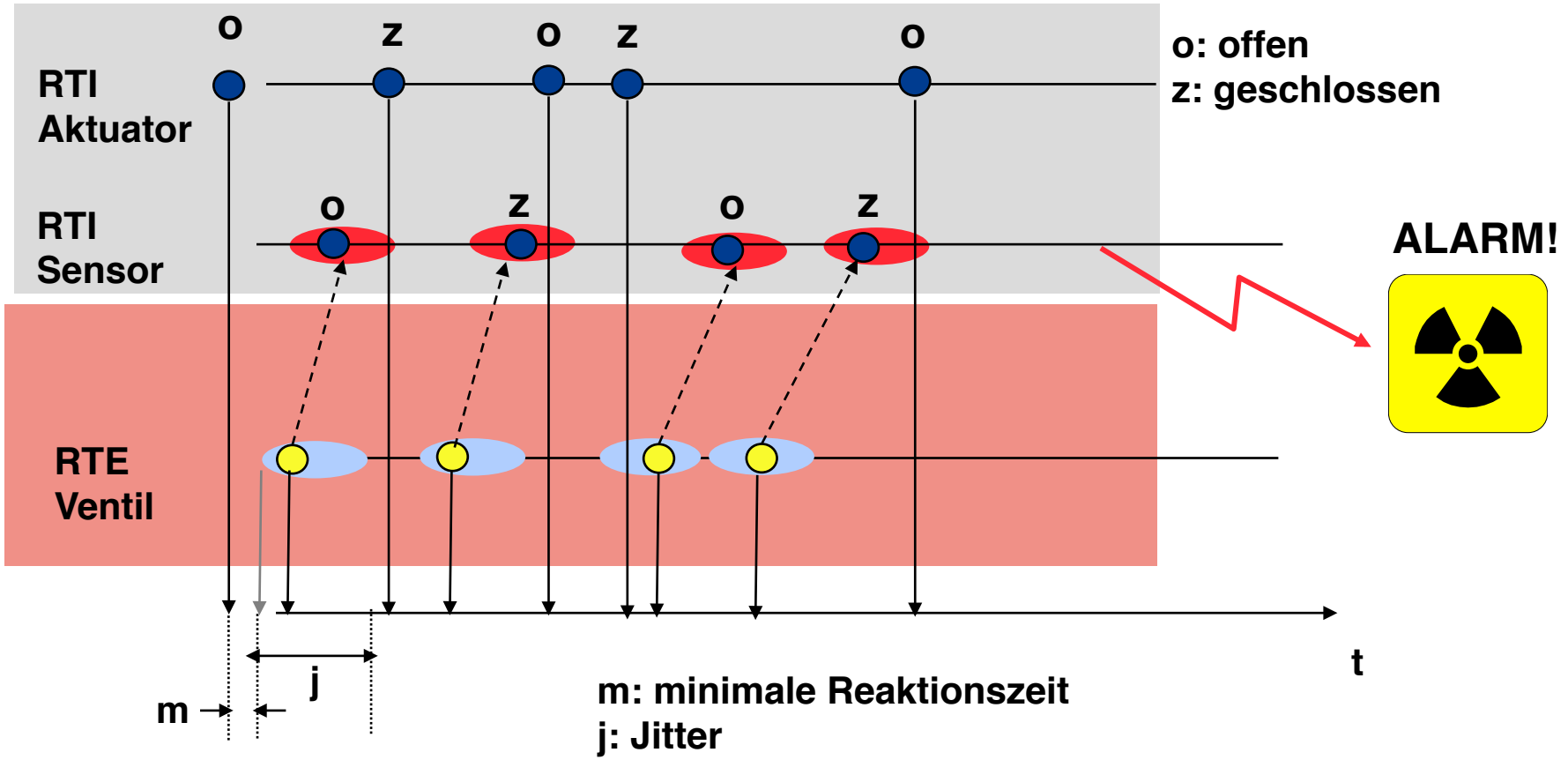
Zustand RTI:
o: offen
z: geschlossen

Zustand RTE:
o: offen
z: geschlossen

m: minimale Reaktionszeit
j: Jitter

Sensorische Rückkopplung erlaubt Überprüfung

! **Die RTE "Ventil" wird durch zwei RTIs repräsentiert, einmal auf der sensorischen und auf der aktorischen Seite.**



Modellierung der Umwelt: Ereignisse und Zustände

Ereignisse: **Ereignisse repräsentieren das singuläre Auftreten von Änderungen der Umwelt in der Zeit. Sie werden charakterisiert durch: < L, T, ∆v > wobei L den Ort, T den Zeitpunkt des Auftretens und ∆v den Wert oder die Wertänderung bezeichnet. Ereignisse haben keine zeitliche Dauer.**

Zustände: **Zustände repräsentieren Eigenschaften der Umgebung. Zustände können eine zeitliche Dauer haben. Sie werden charakterisiert durch einen Wert und den Zeitpunkt der Beobachtung.**

Ereignisse werden dem System signalisiert. Sle müssen vollständig und in der richtigen Reihenfolge gespeichert werden, um einen externen Zustand korrekt rekonstruieren zu können.

Zustände müssen aktiv vom System abgefragt werden. Es wird nur der jeweils aktuelle Zustand zum Zeitpunkt der Abfrage beobachtet. Zwischenzustände, falls vorhanden, gehen verloren.

Zeitliche Kontrolle:

Prinzipiell zwei unterschiedliche Bedingungen:

- **Zeitgesteuert: Ein periodisches Kontrollsignal wird von dem Fortgang einer Uhr gesteuert. *TTA* Time Triggered Architecture**

- **Ereignisgesteuert: Ein Kontrollsignal wird beim Eintreten eines internen oder externen Ereignisses generiert. *ETA* Event Triggered Architecture.**

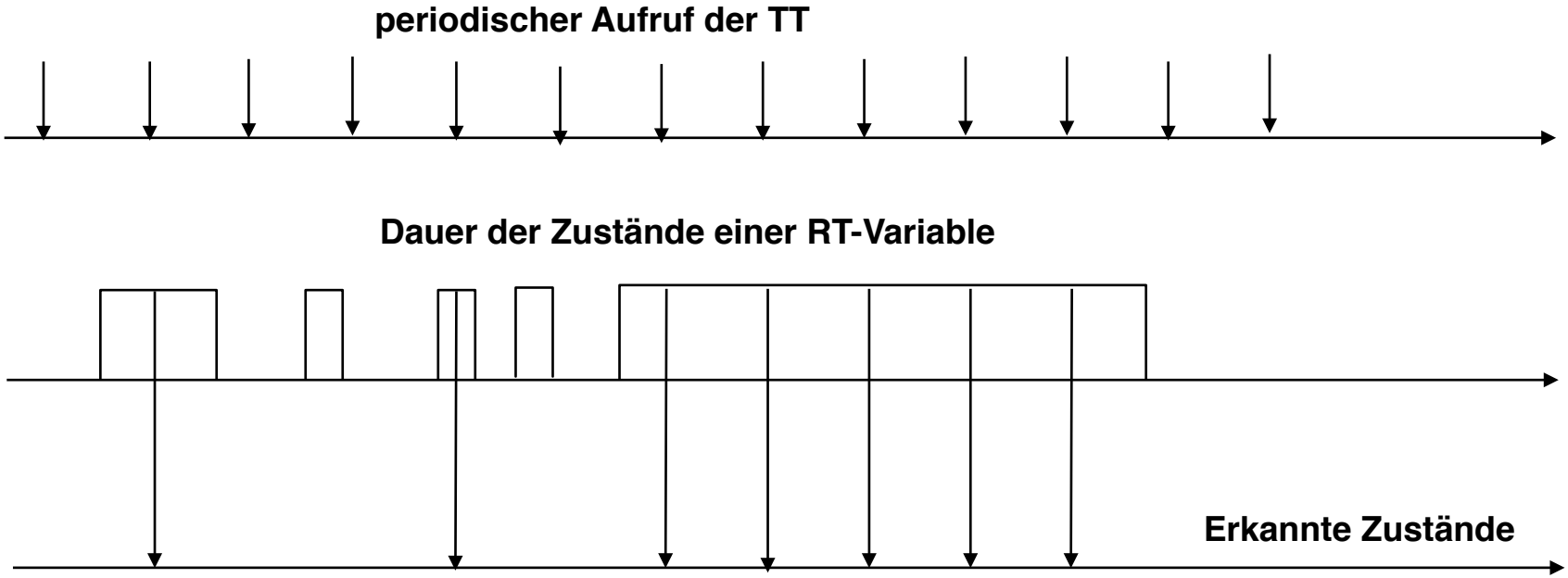
Quellen für Kontrollsignale:

- **Realzeituhr**
- **Eine vorhergehende Task (Precedence Task)**
- **Ein externes Ereignis (Interrupt)**

Zeitgesteuertes Verfahren: Trigger Task (TT)

Eine periodische TT wertet ein Prädikat (Triggerbedingung) auf einer Menge temporär gültiger RT-Variablen aus. Das Ergebnis einer TT ist:

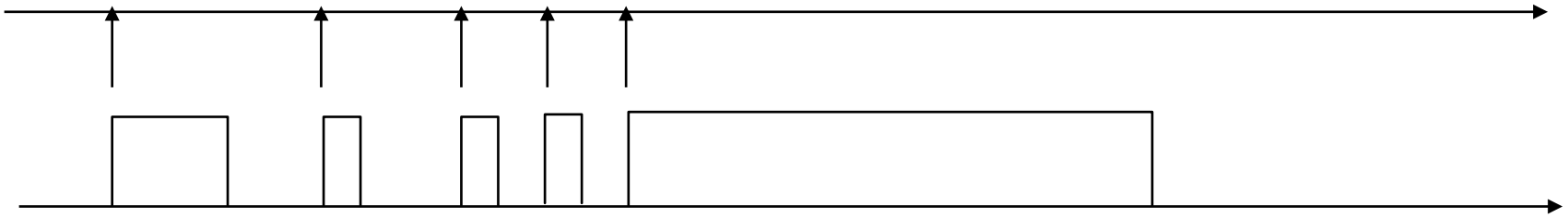
- **ein Kontrollsignal, das eine andere Task aktiviert**
  oder
- **eine neue RT-Variable**



Es können nur Zustände zuverlässig beobachtet werden, deren Dauer länger ist, als die Granularität der TT. Kurzlebige Zustände müssen zwischengespeichert werden.

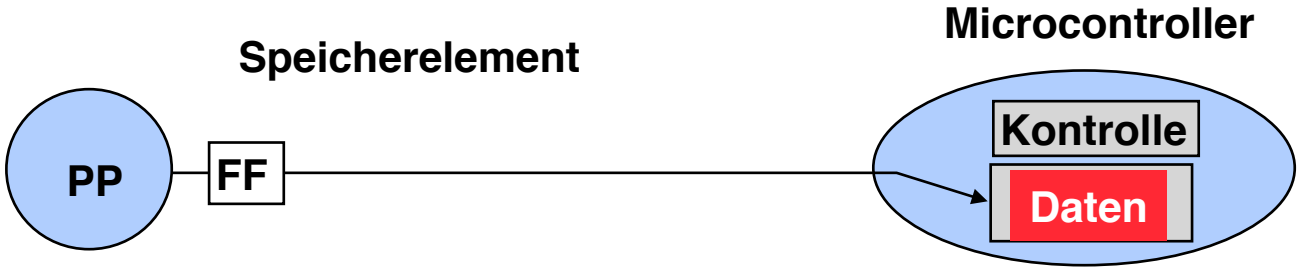
Ereignisgesteuerte Verfahren: Interrupts

Def.:
Ein Interrupt ist eine asynchron zur Berechnung auftretende Anforderung für eine bestimmte Task Aktivierung, die durch ein exterenes Ereignis hervorgerufen wird. (Diese Definition schließt synchrone Software Interrupts, Traps und Exceptions aus.)



Interne und externe Zustandsspeicherung

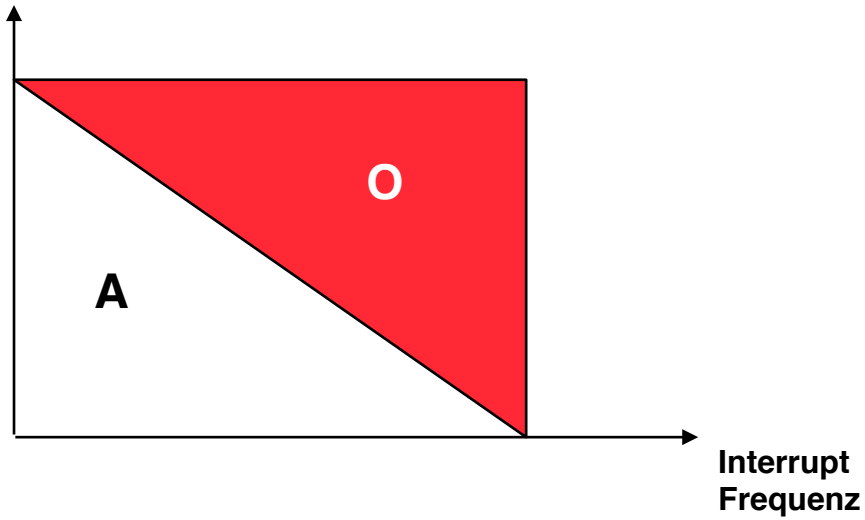
PP: physischer Prozeß



Wie verhalten sich die unterschiedlichen Verfahren bei Störungen ?

Software Overhead für die Interruptverarbeitung

O: benötigte CPU-Kapazität für den Verwaltungsaufwand bei der Interruptverarbeitung
A: verfügbare Zeit für anwendungsspezifische Aufgaben



Folgerungen:

- **Anzahl der Interrupts muß beschränkt werden**
- **Zeit für den Verwaltungsaufwand muß möglichst klein sein**

Vergleich von ereignisgesteuerten und zeitgesteuerten Verfahren

	Interrupt	Trigger Task
Verzögerung	Int. Antwortzeit	Periode
Kontrolle	extern	intern
Aufwand Abfrage	null	1/Periode * TT
Aufwand Verarbeitung	variabel	konstant
Unterbrechung	jederzeit	vorhersagbar
Speicherelement	intern	extern
Triggerbedingung	einfach	komplex
Vorhersagbarkeit	gering	hoch

Example: Ignition Control

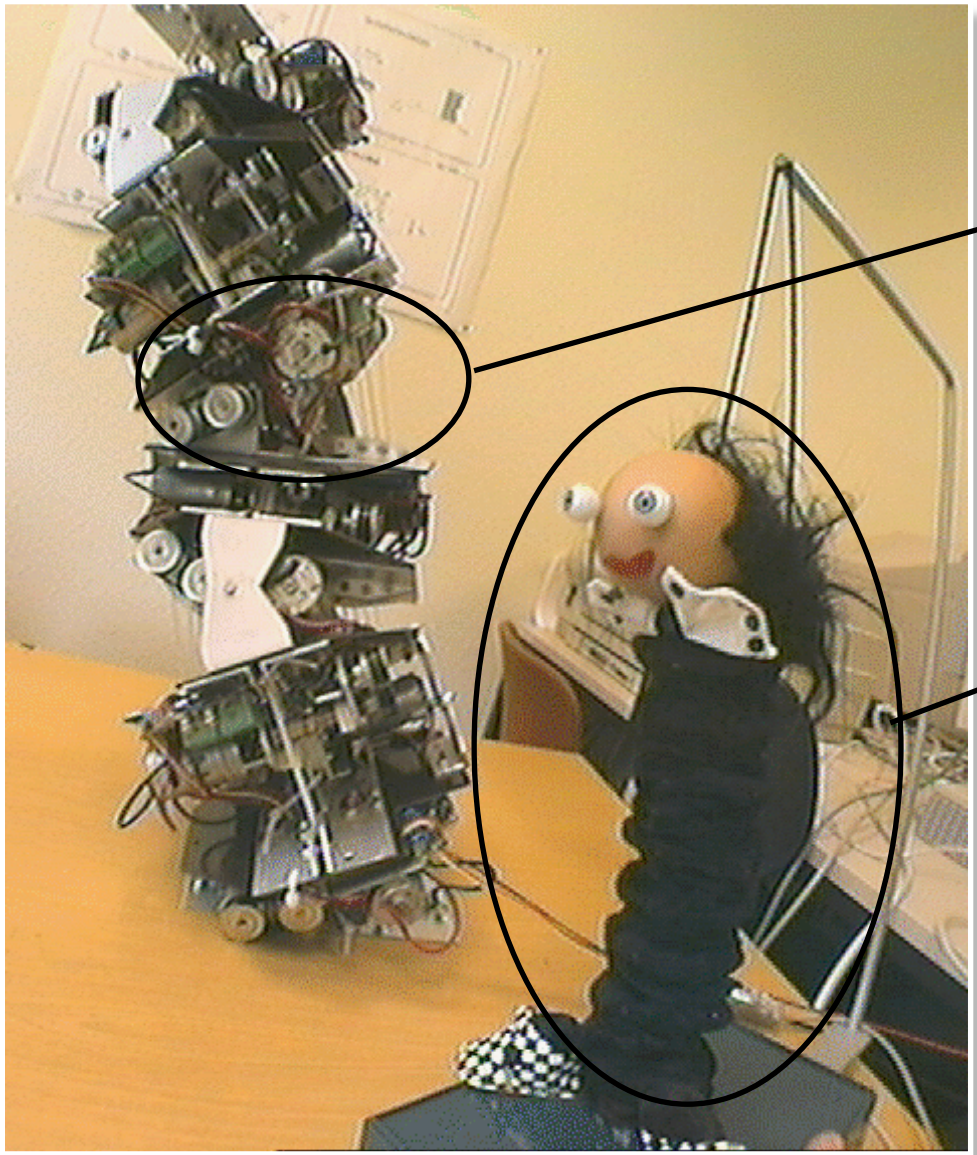
Consider the control of the ignition point by continuously sampling the position of the crank shaft. At 10.000 rpm/min we have 166 rpm/sec. If 1° of position accuracy is required, we need 166 x 360 = 60.000 sampling points/second, meaning that the sampling interval is below 17 µsec.

If there would be a sensor which detects the position of the crank shaft with a sufficient accuracy , the system would have to process 166 events/second, i.e. ~ 1 event/6 millisecond. The problem is that the system must be able to respond to an event within 17 µsec to keep the position point sufficiently precise.

If the motor runs at 1000 rpm, the sampling rate is still 60.000/sec in the TT approach. In the event triggered approach ist goes down to ~ 17 events/sec = 1 event/58 ms which now have to be serviced within 166 *µ*sec.

Overhead: Service time is 10*µ*s,
TT: 58% constant load
ET; at 10000 rpm the load is 0,16%, at 1000 rpm the load is 0,016%

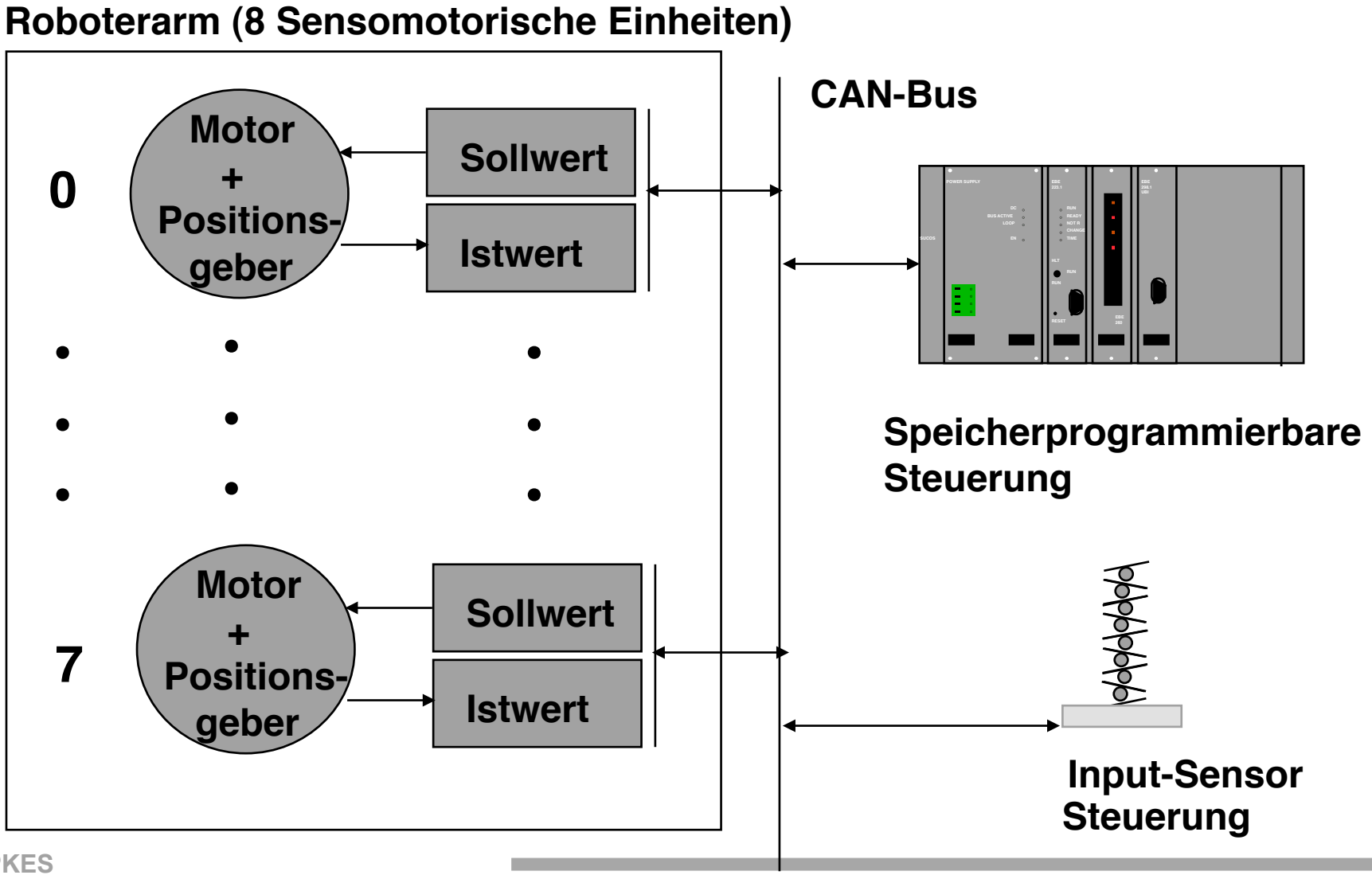
Example: Robot Control



single CAN-connected robot-element
(the prototype consists currently of 8 elements)

Input-Sensor
Remote Control

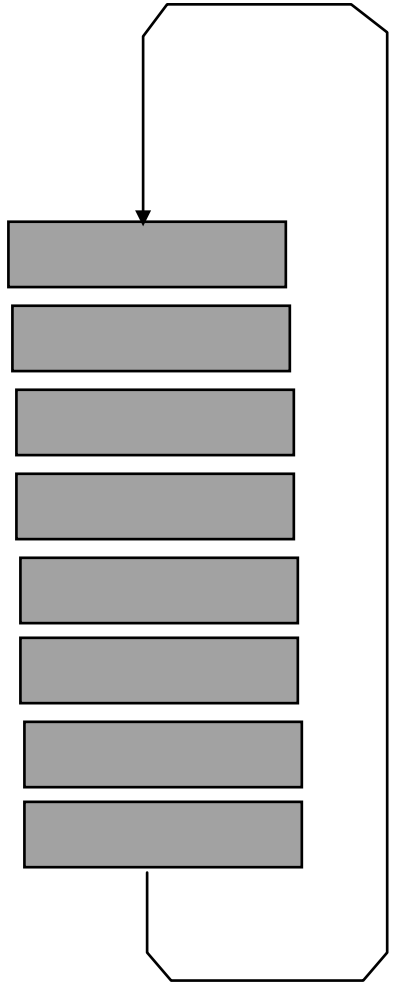
Konfiguration der Steuerungsanwendung



Einfache Kontrollschleife

```
LOOP {
  for( i = 0; i <= 7; i++) {
    NomPos = Read ( InputSensor [ i ] );
    RealPos   = Read ( RobEl [ i ] );

    if( NomPos == RealPos) StopMotor(RobEl [ i ]); else
    if(NomPos <RealPos)     StartMoveLeft(RobEl [ i ]); else
                            StartMoveRight(RobEl [ i ]);
  }
}
```

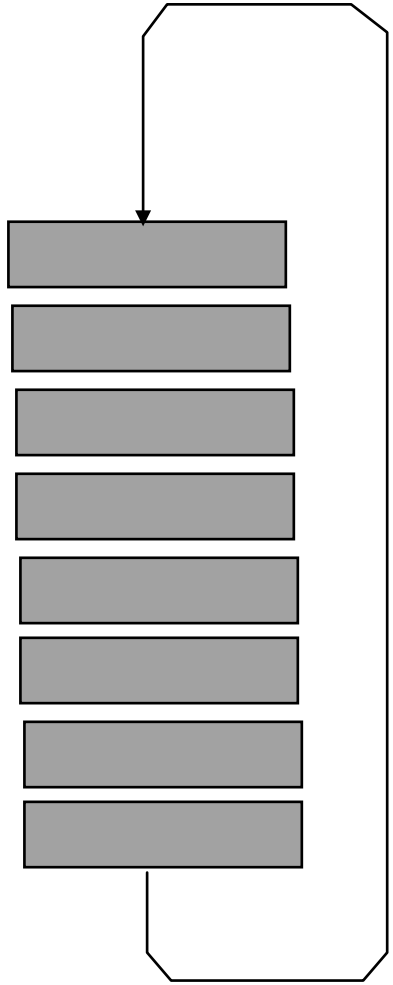


Kontrollprogramm für eine sensomotorische Einheit

Die Gesamtschleife ist zeitkritisch daraus folgt, daß jeder Befehl der Schleife zeitkritisch ist.

Einfache Kontrollschleife

8 Robot-Elements => 8 Sensor-Elements
typical duration of a movement: > 200 ms, < 3 s
(dependent on Motor Characteristics)



Single Loop:

shortest duration of movement (cycle time) ~ 200 ms
=> WCET for single Sensor/RobEl-Control: ~25 ms
WCET for StopMove: 16 ms
=> polling the input sensor is time critical:
Read(InputSensor) < 10 ms

shortest cycle: 25ms x 8 = 200 ms

Dynamische Kontrolle

```
class RobEl { ...
  Task InputSens ( ... );
  Task RobContr ( ... );
}
```

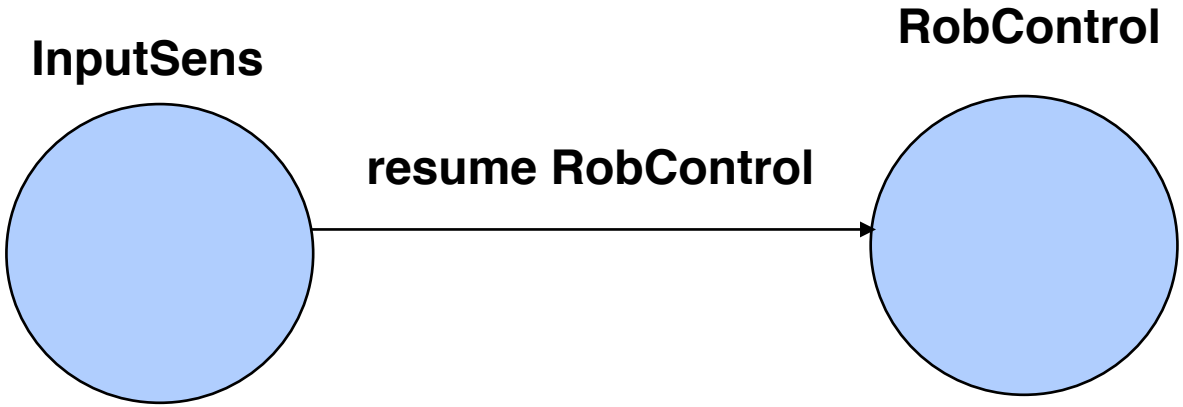
```
InputSens {
  LOOP {
    NomPos = Read ( MyInputSensor );
    RealPos = Read ( MyRobEl);

    if( NomPos has changed)
        resume RobControl;
    delay ( 200 );
  }
}
```

```
RobControl
{

  LOOP {
    ComputeMoveDurationAndDirection ( );
    S_TIMER = NOW + MoveDuration;
      MOVE_MOTOR
            StartMove( Direction );
            suspend ( );
      CONTROL
            StopMove ( )
      END;
   suspend ( );
            }
}
```

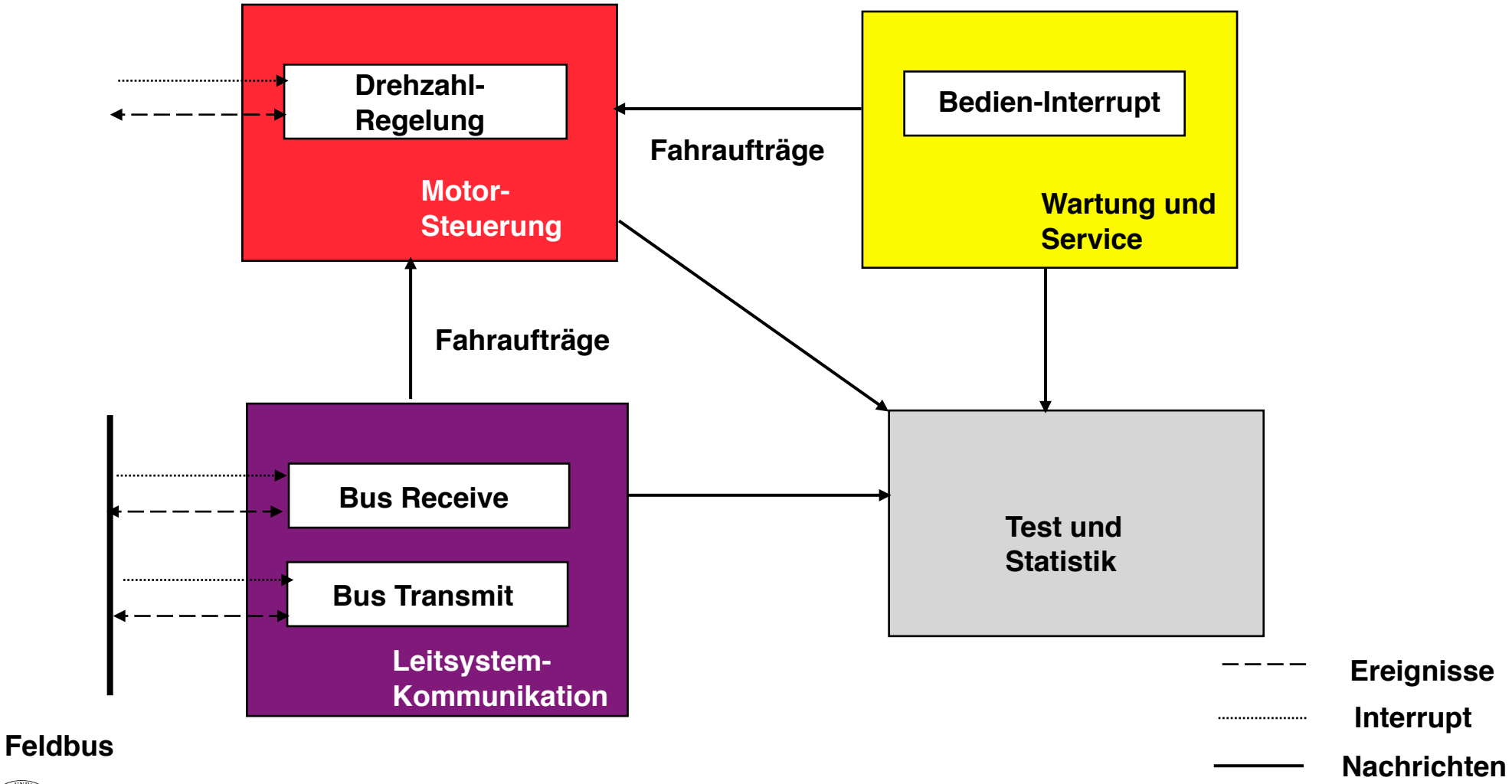
8 Robot-Elements => 8 Sensor-Elements
typical duration of a movement: > 200 ms, < 3 s
(dependent on Motor Characteristics)



Multi-Tasking:

Only StopMove ( ) is time critical (16 ms)
=> shortest possible duration of movement: < 30 ms = 25 + ε
Position Accuracy: ~10 times better than in Single Loop System

Architektur einer Beispiel - Anwendung



Tasks und Handler