

# Grundlagen der Echtzeitplanung

---

## 1. Grundlegende Begriffe und Konzepte

## 2. Planungsverfahren ( Scheduling )

### 2.1 Planen aperiodischer Tasks

- Planen durch Suchen
- Planen nach Fristen
- Planen abhängiger Tasks

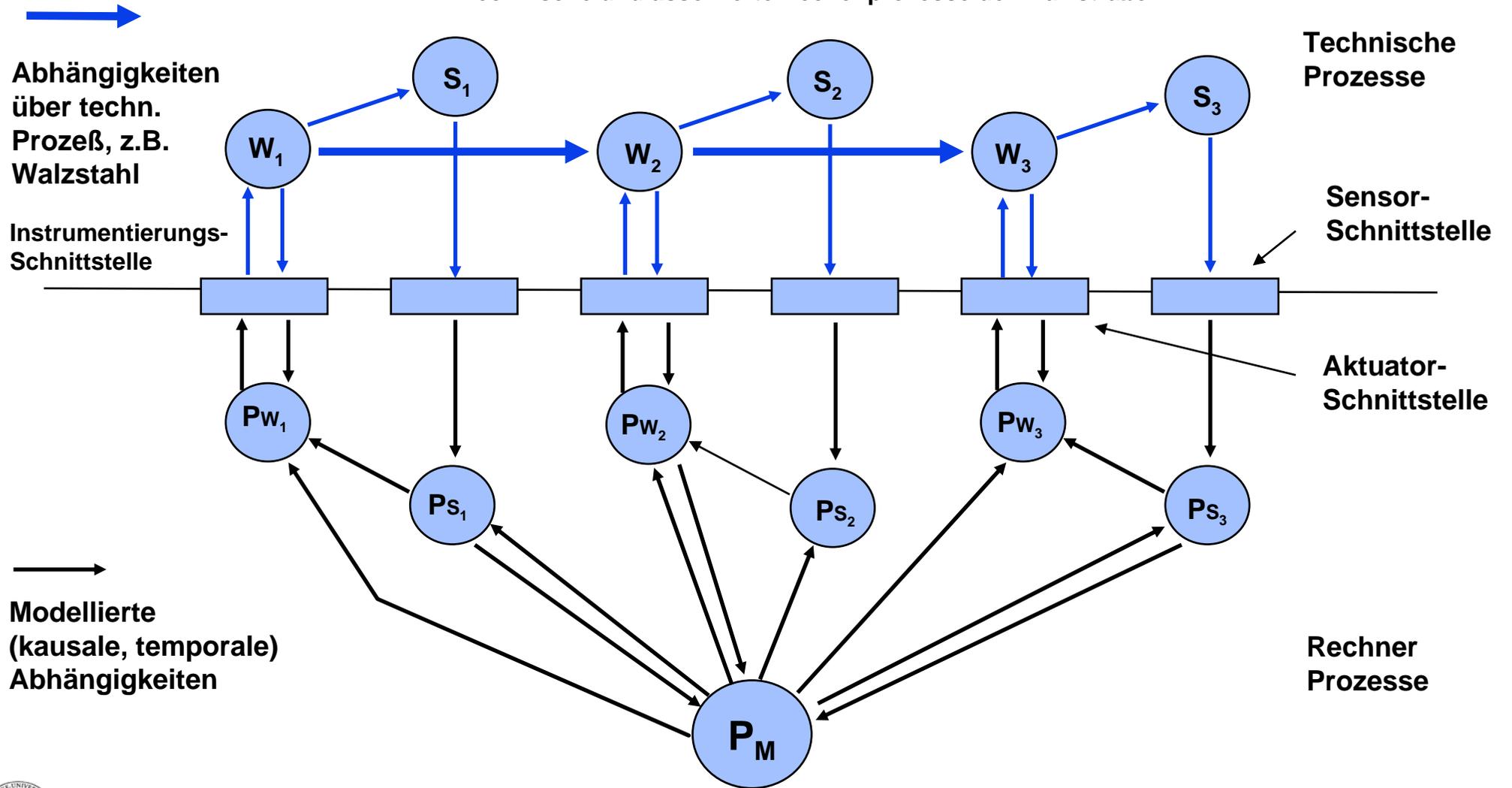
### 2.2 Planen periodischer Tasks

- Planen nach monotonen Raten
- EDF



# Modellierung

Technische und assoziierte Rechenprozesse der Walzstraße



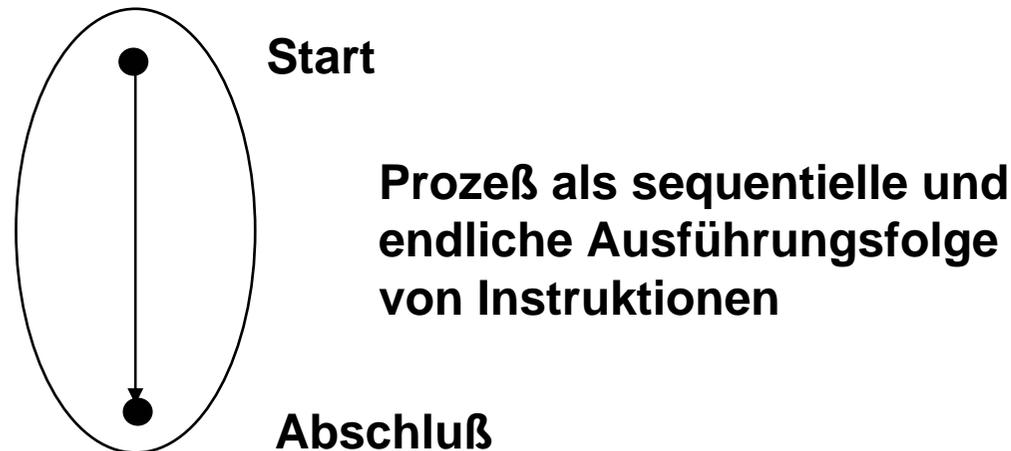
# Prozeßmodell

---

Ein Prozeß (oft auch als Task bezeichnet) ist die Ausführung eines sequentiellen Programms auf einem Prozessor in seiner spezifischen Umgebung (Kontext).

Ein Prozeß ist:

- **Träger der Aktivität**
- **die Abstraktion eines Prozessors**
- **die kleinste planbare Einheit**
- **erfüllt eine von Programm spezifizierte Aufgabe**



# Prozeßmodell (cont.)

---

**Der spezifische Kontext ist dem Prozeß zugeordnet, nicht dem Programm !**

**z.B. ein Programm zur Abfrage eines Temperatursensors kann durch Prozesse in individuellen Kontexten unterschiedlich ausgeführt werden.**

**Eine Prozeßausführung benötigt Ressourcen**

- **(Prozessor-) Zeit**
- **Speicher**
- **Peripherie**

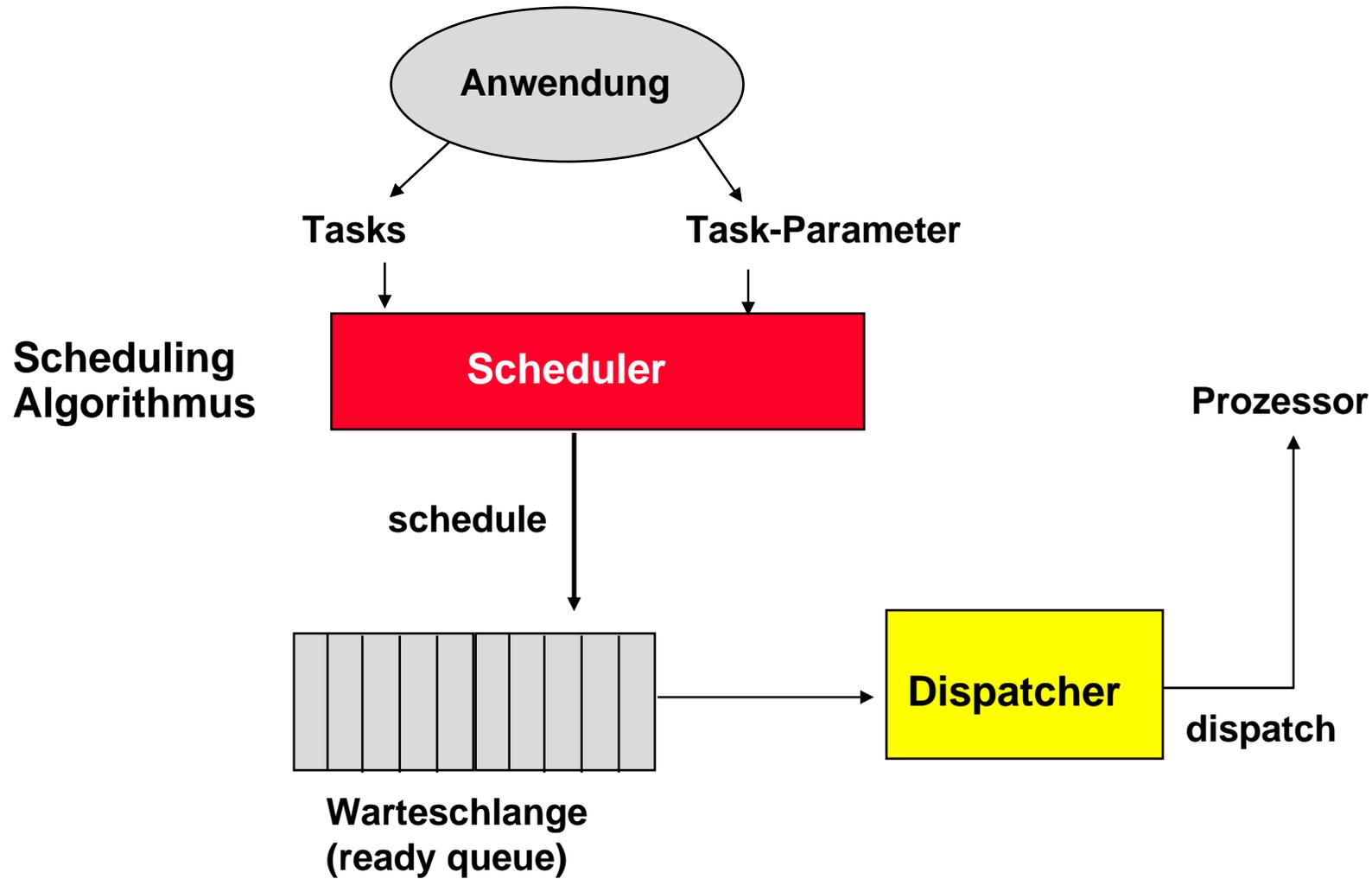
**Die Ausführungszeit ist abhängig von:**

- **der Leistungsfähigkeit des Prozessors**
- **der Verfügbarkeit der Betriebsmittel**
- **Peripherie des Prozessors**
- **in die Ausführung eingehenden Daten**
- **Verzögerungen durch Erledigung wichtigerer Aufgaben**



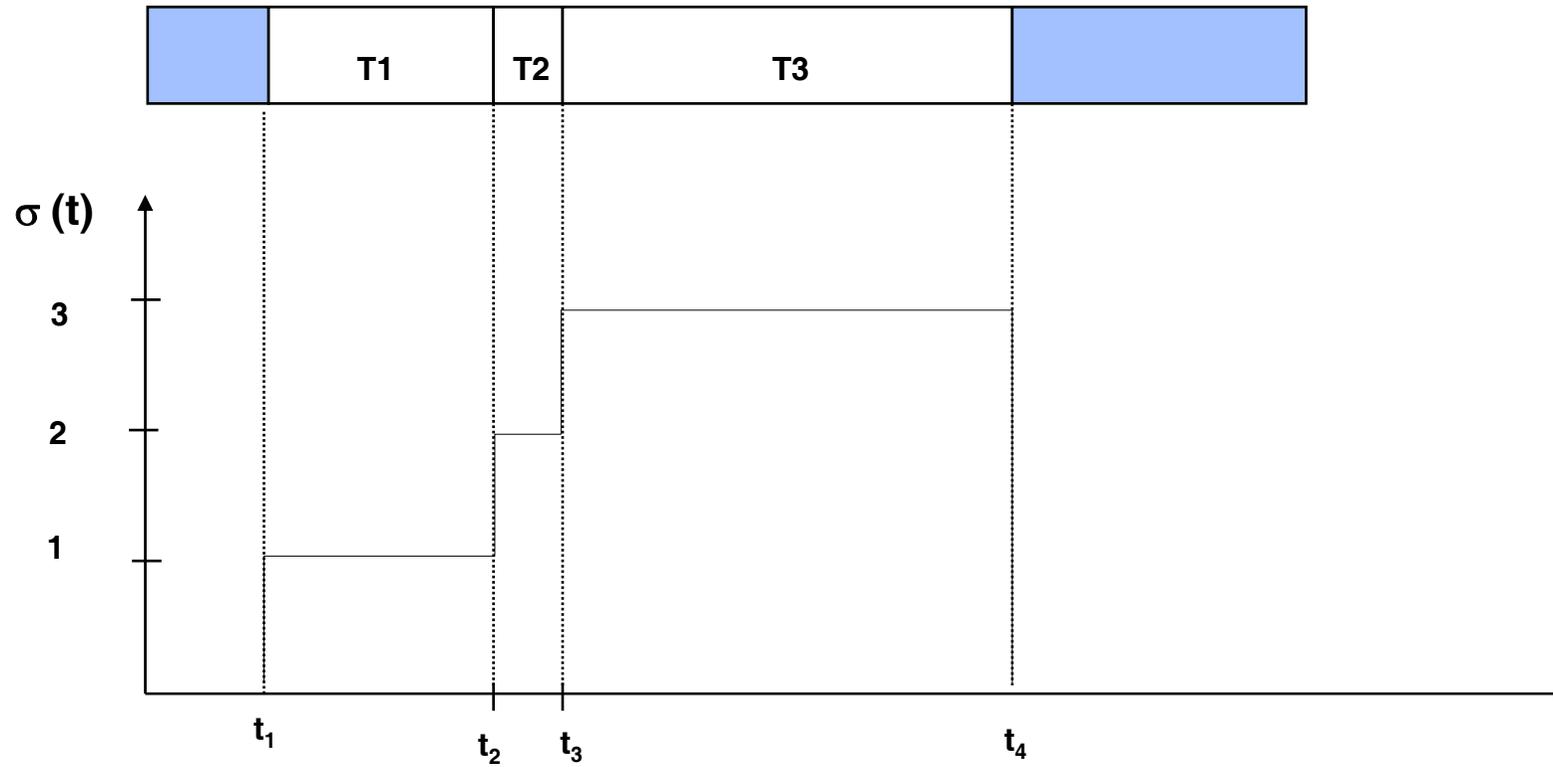
# Scheduling und Dispatching

---

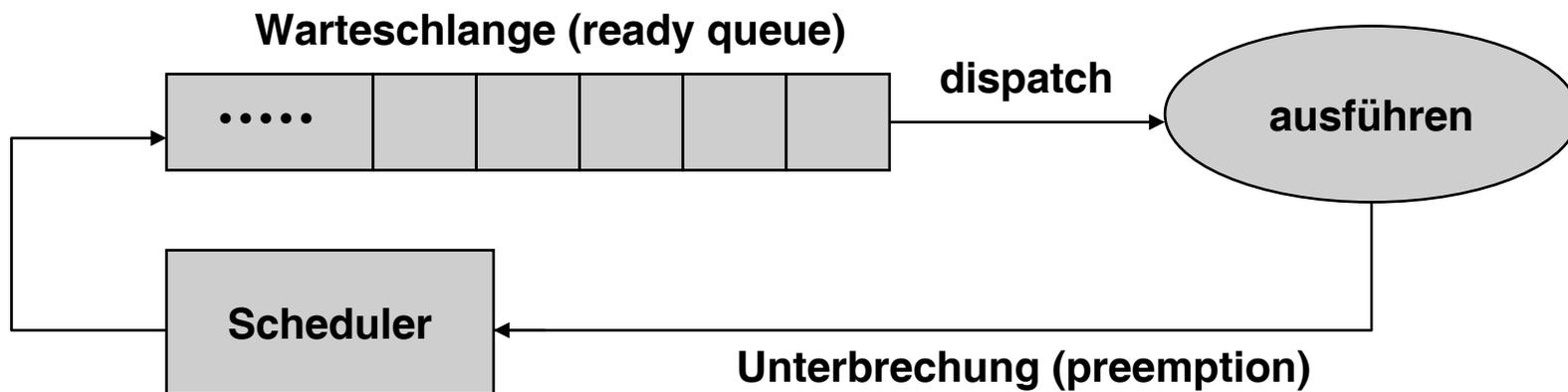
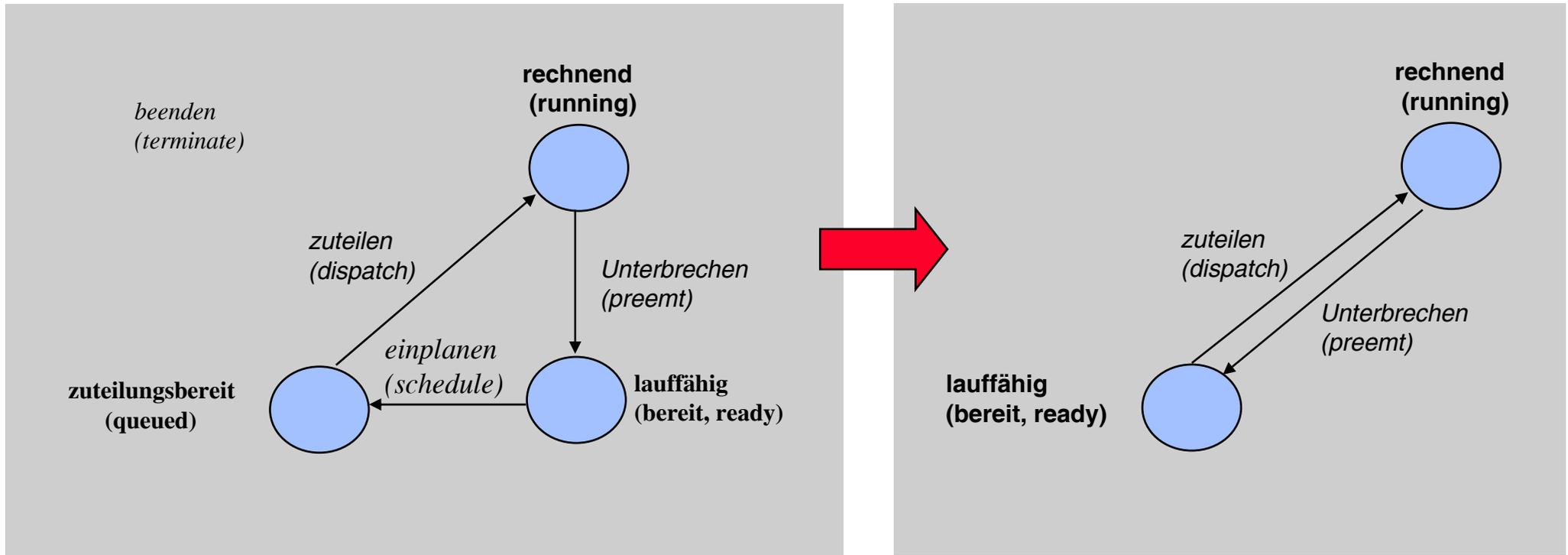


# Beispiel eines Plans für 3 Tasks (nicht unterbrechbar)

---

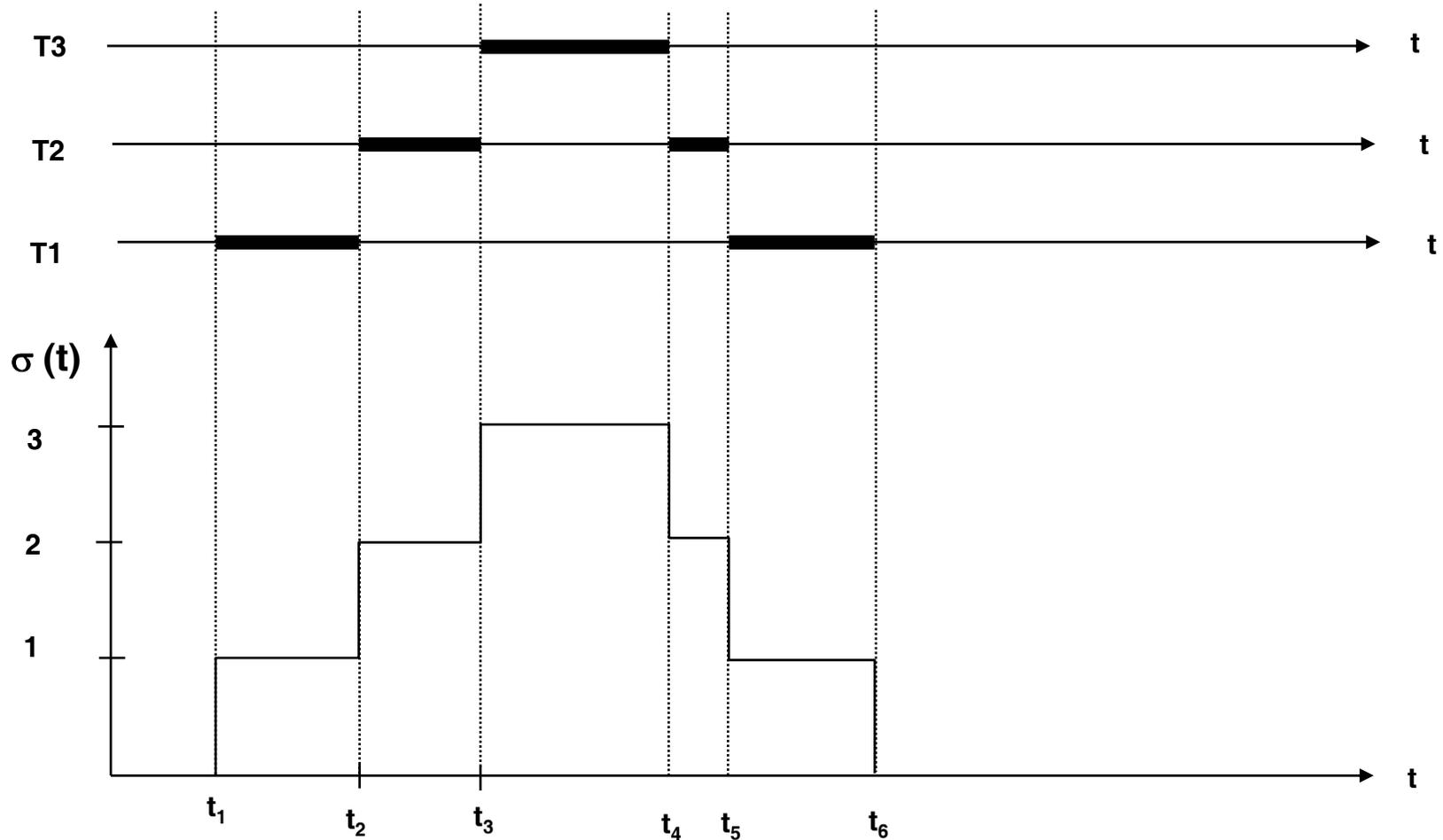


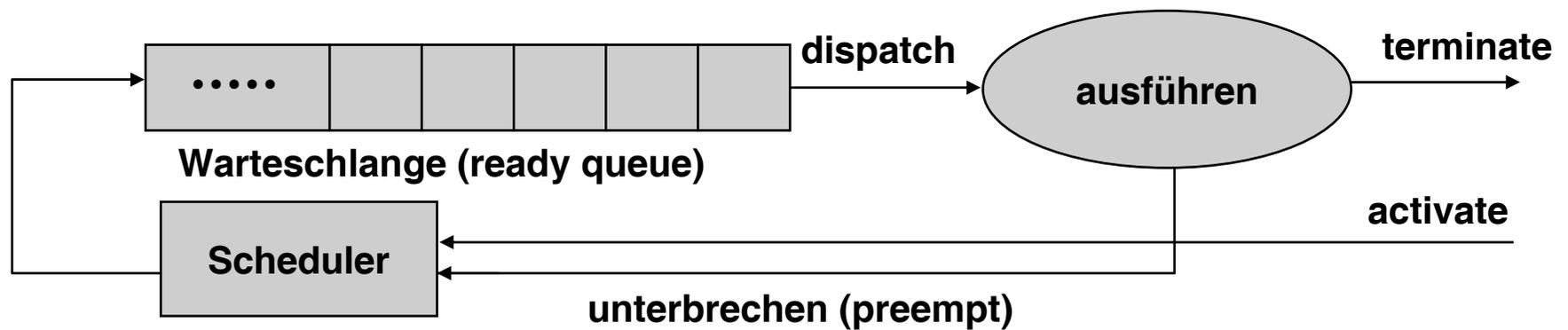
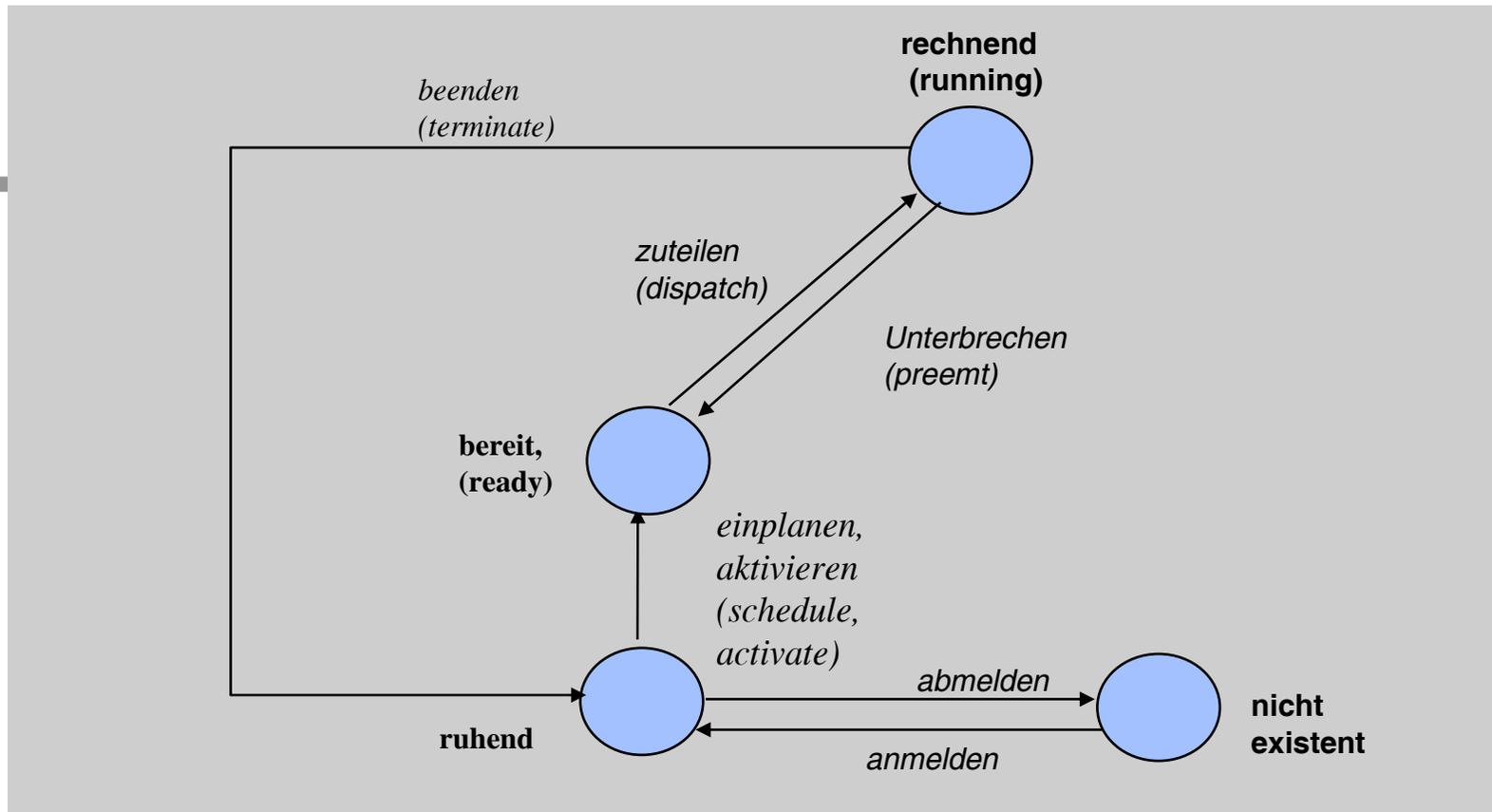
# Zustandsdiagramm einer Task



# Beispiel eines unterbrechbaren Schedules

---





# Faktoren, die bei der Planung berücksichtigt werden müssen:

---

 **Gemeinsame Nutzung von Ressourcen**

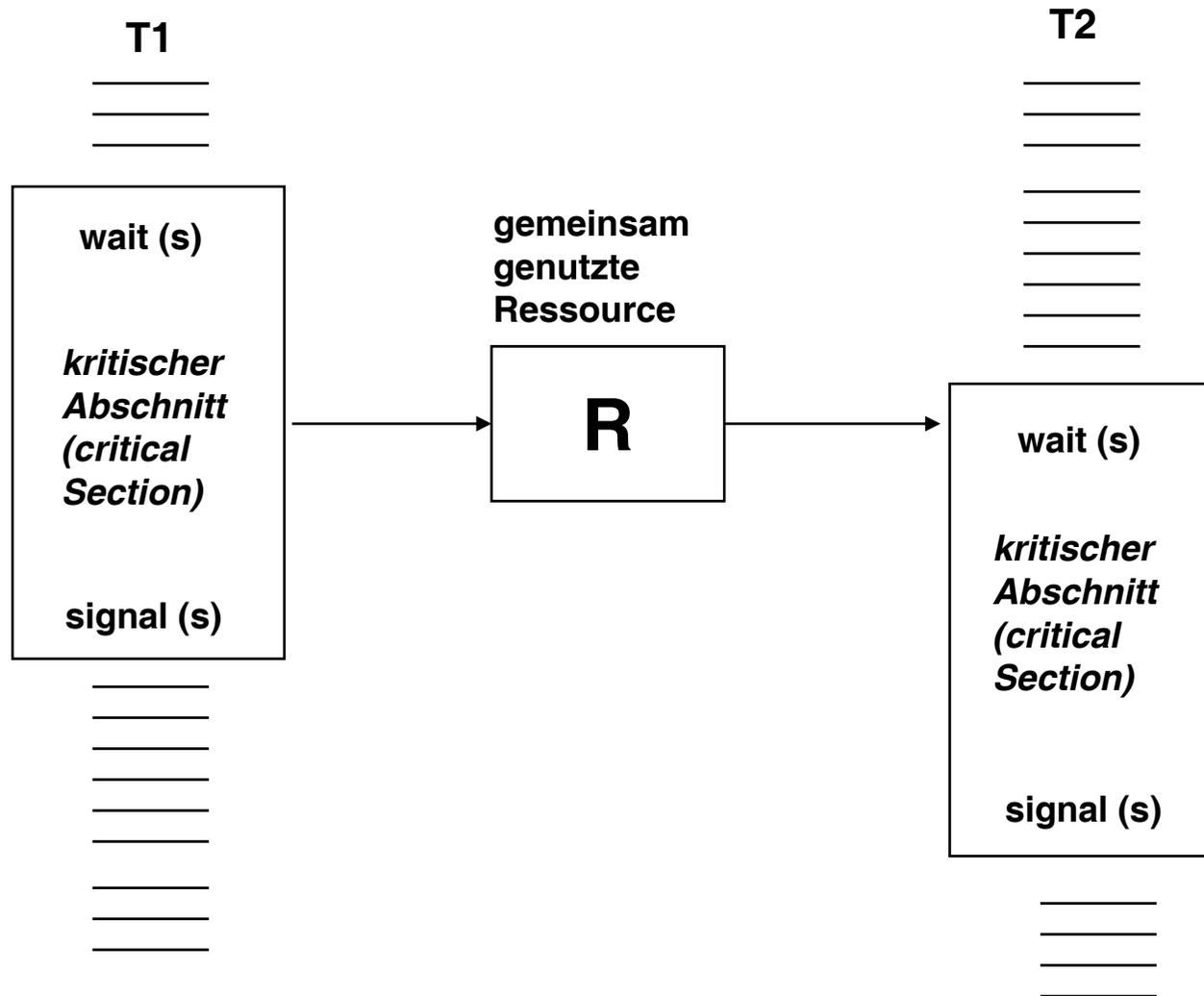
 **Vorrangrelation**

 **Zeitvorgaben**



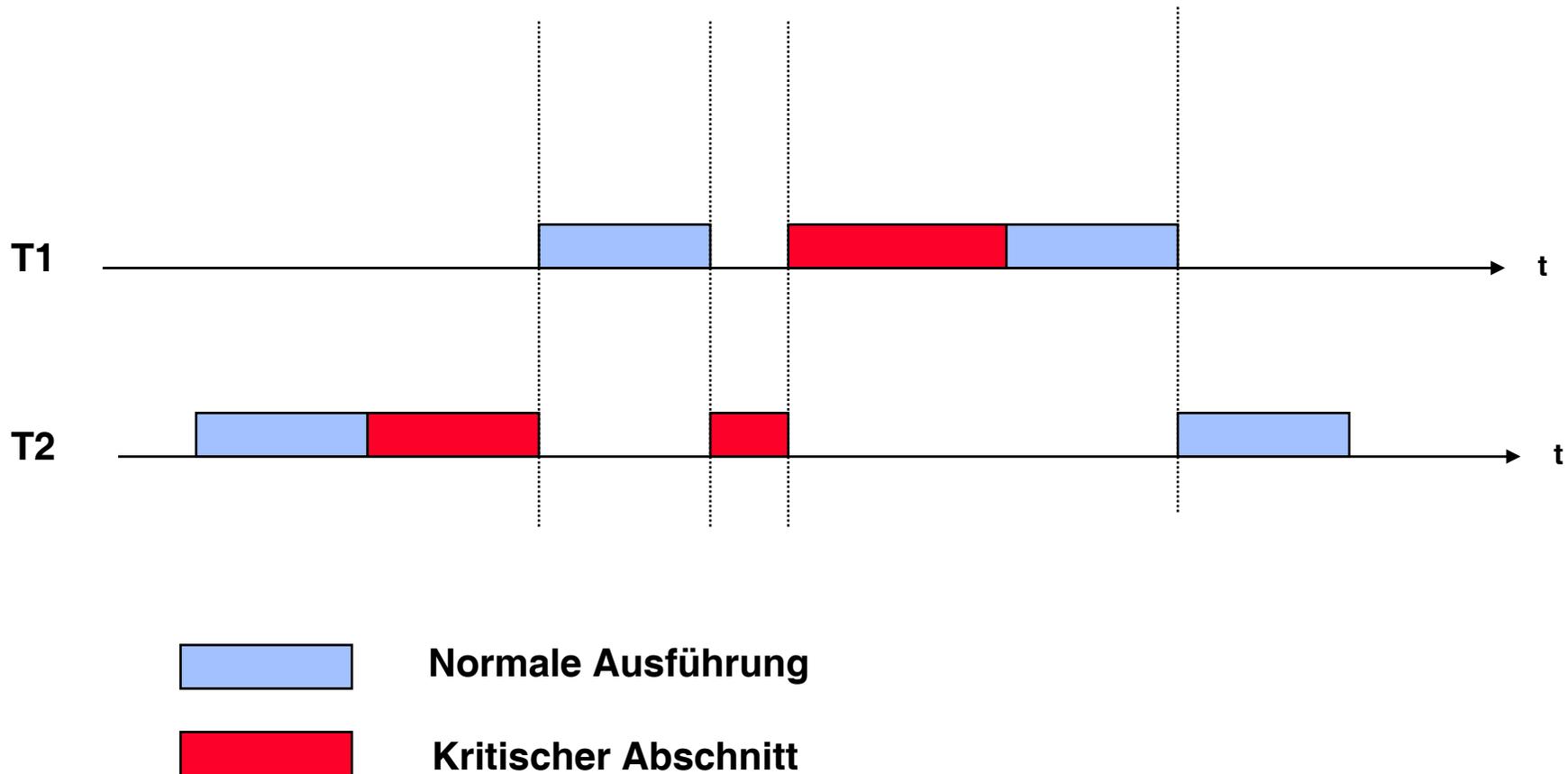
# Gemeinsam benutzte Ressourcen

---

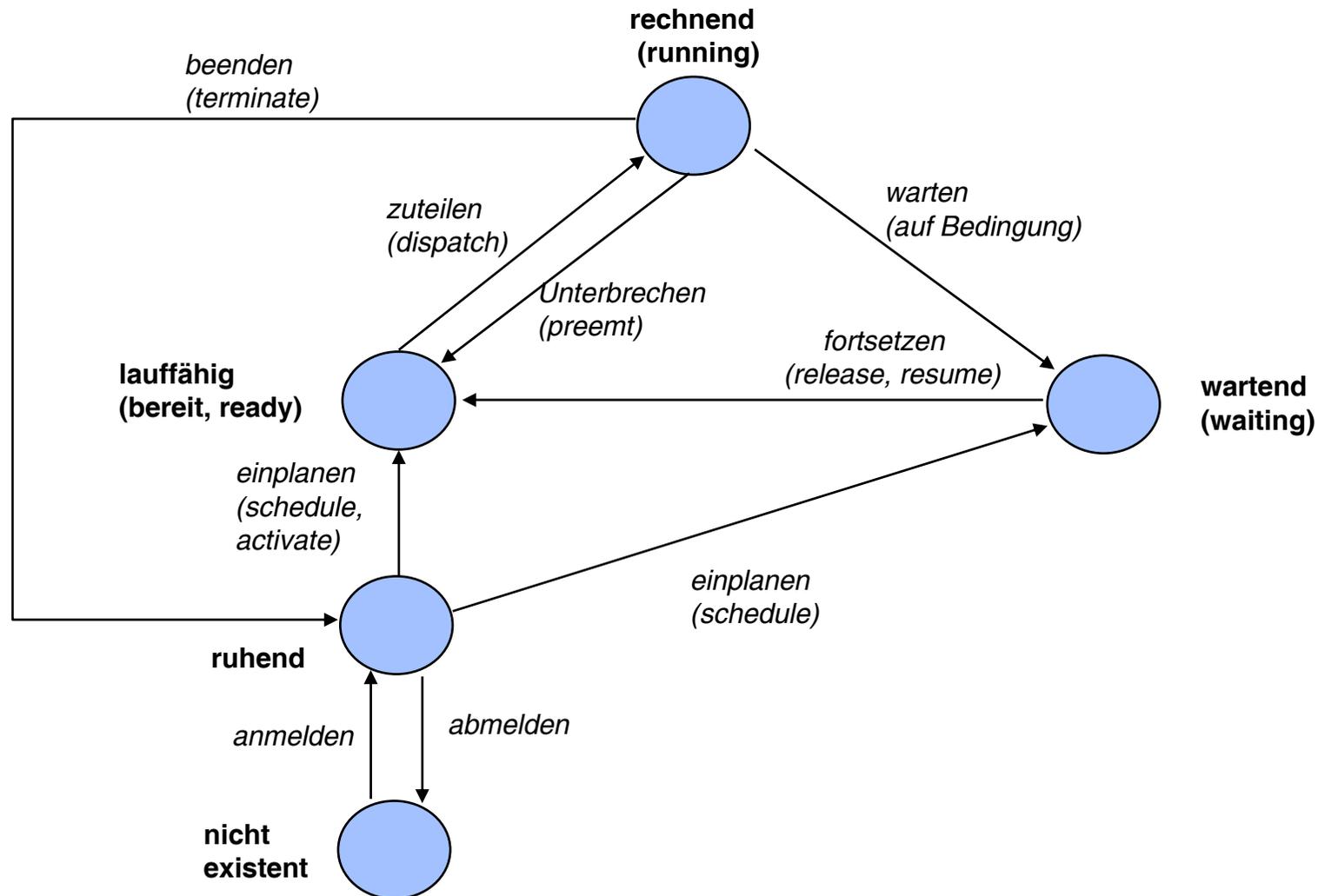


# Beispiel für die Blockierung an einer gemeinsam genutzten Ressource

---



# Zustandsdiagramm einer Task

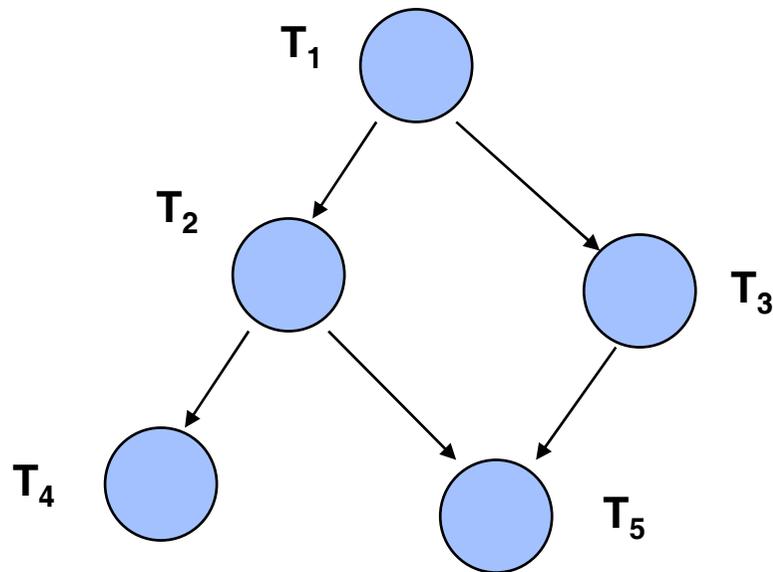


# Vorrangrelation

---

Tasks sind nicht unabhängig

➔ Zeitliche Bedingungen sind von der Vorrangrelation abhängig



$$T_1 < T_3$$

$$T_1 \rightarrow T_2$$

$$T_1 < T_4$$

$$T_1 \not\rightarrow T_4$$



# Zeitliche Bedingungen

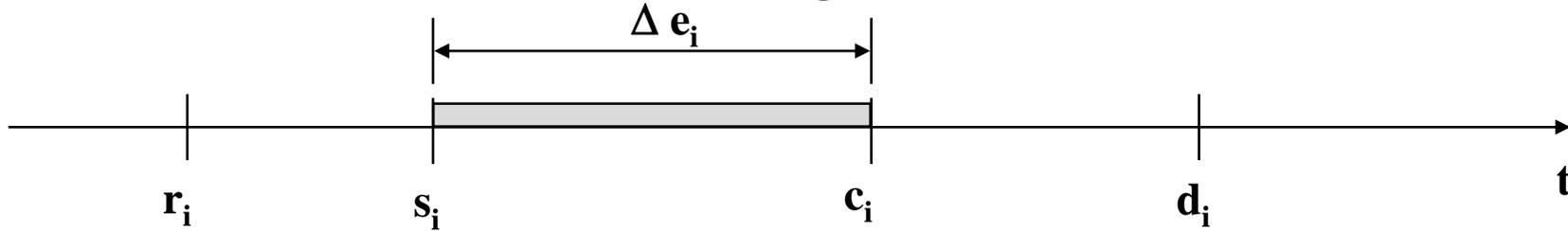
Größen, die beim Scheduling benötigt werden:

- $T_i$  bezeichnet den Prozeß (die Task)  $i$
- **Bereitzeit (ready time)  $r_i$  :**  
Frühster Zeitpunkt, an dem der Prozessor dem Prozeß  $T_i$  zugeteilt werden darf.
- **Ausführungszeit (execution time)  $\Delta e_i$  :**  
Reine Rechenzeit auf dem Prozessor. Zur Planung legt man meist die "worst case execution time" (WCET) zugrunde.
- **Startzeit (starting time)  $s_i$  :**  
Prozessor beginnt  $T_i$  auszuführen.
- **Abschlußzeit (completion time)  $c_i$  :**  
Prozessor beendet die Ausführung von  $T_i$ .
- **Frist (deadline)  $d_i$  :**  
Zu diesem Zeitpunkt muß die Ausführung von  $T_i$  abgeschlossen sein.

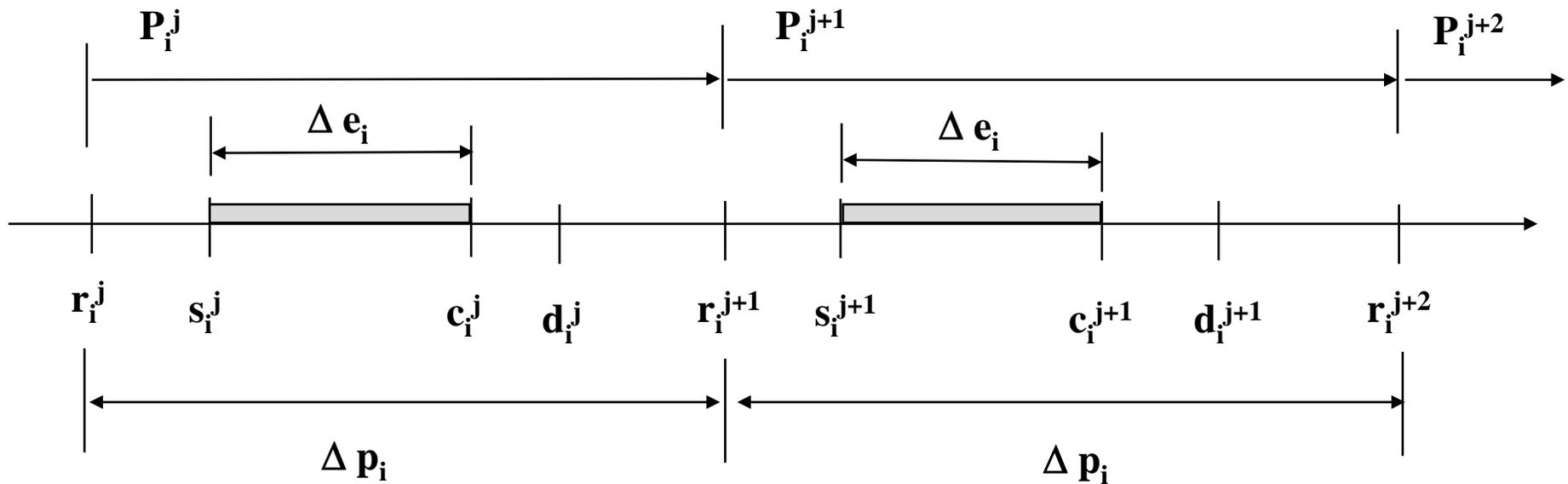


# Zeitpunkte und Zeitintervalle

a.) für eine individuelle Task-ausführung



b.) für periodische Prozesse



## Periodische Tasks:

**Periodische Tasks werden mit einer bestimmten Frequenz  $F$  regelmäßig aktiviert.**

**(Periode =  $1 / F$ )**

**Das Zeitintervall  $\Delta p_i$  definiert für eine periodische Task den Rahmen ihrer  $j$ -ten Ausführung  $P_i^j$ .**

**Bei periodischen Tasks sind alle Bereitzeiten  $r_i^j$  für  $P_i$  ausgehend von der ersten Bereitzeit  $r_i^1$  festgelegt durch:**

$$r_i^j = (j-1) \Delta p_i + r_i^1 \quad \text{für } j \geq 1$$

## Sporadische Tasks:

**Sporadische Tasks treten nicht regulär auf. Man nimmt aber eine obere Schranke bzgl. der Häufigkeit ihres Aufrufs an. (geht in die Lastannahme ein!)**

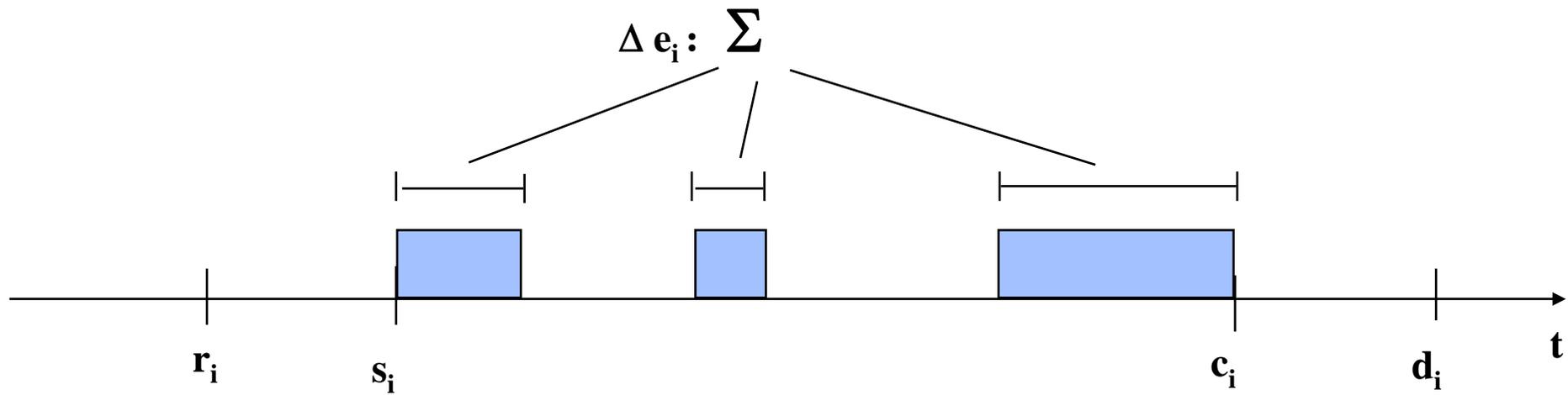
## Aperiodische Tasks:

**Keine obere Schranke bzgl. der Häufigkeit ihres Aufrufs.**

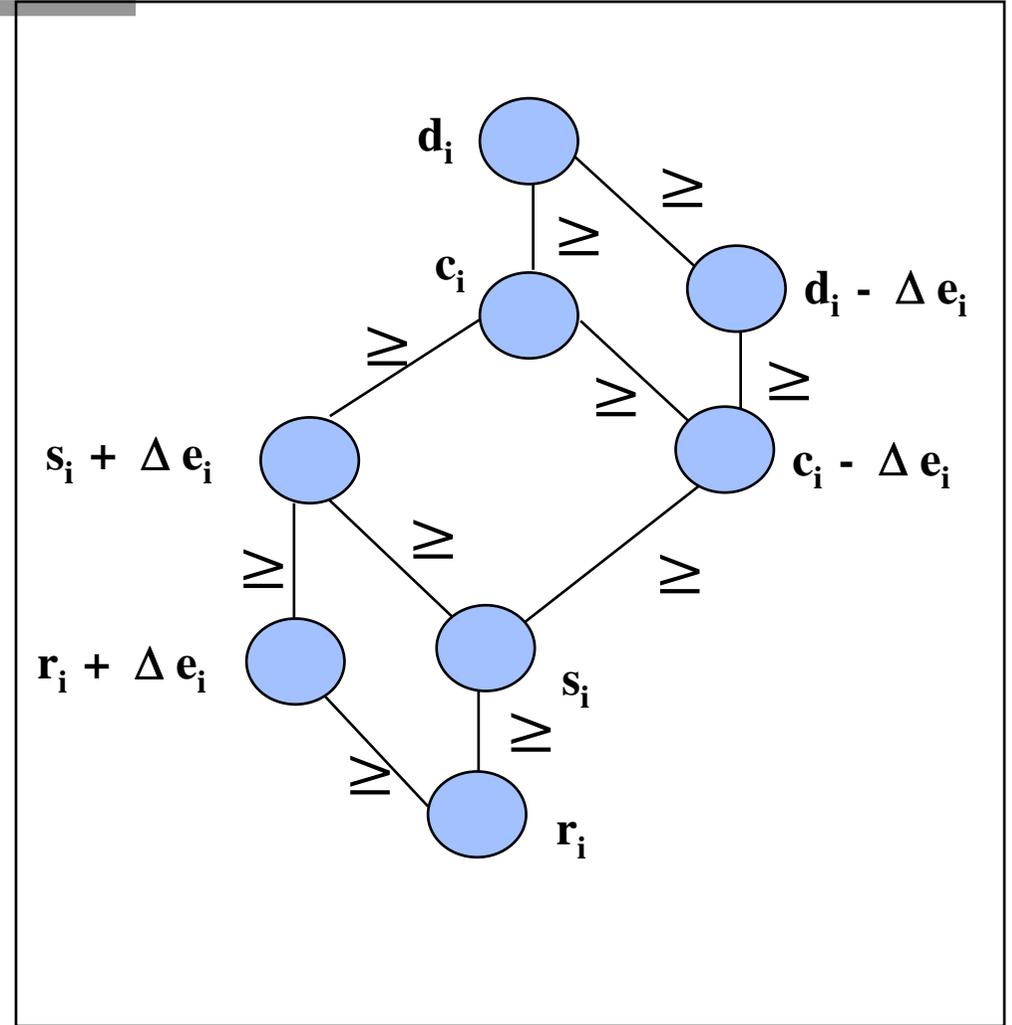
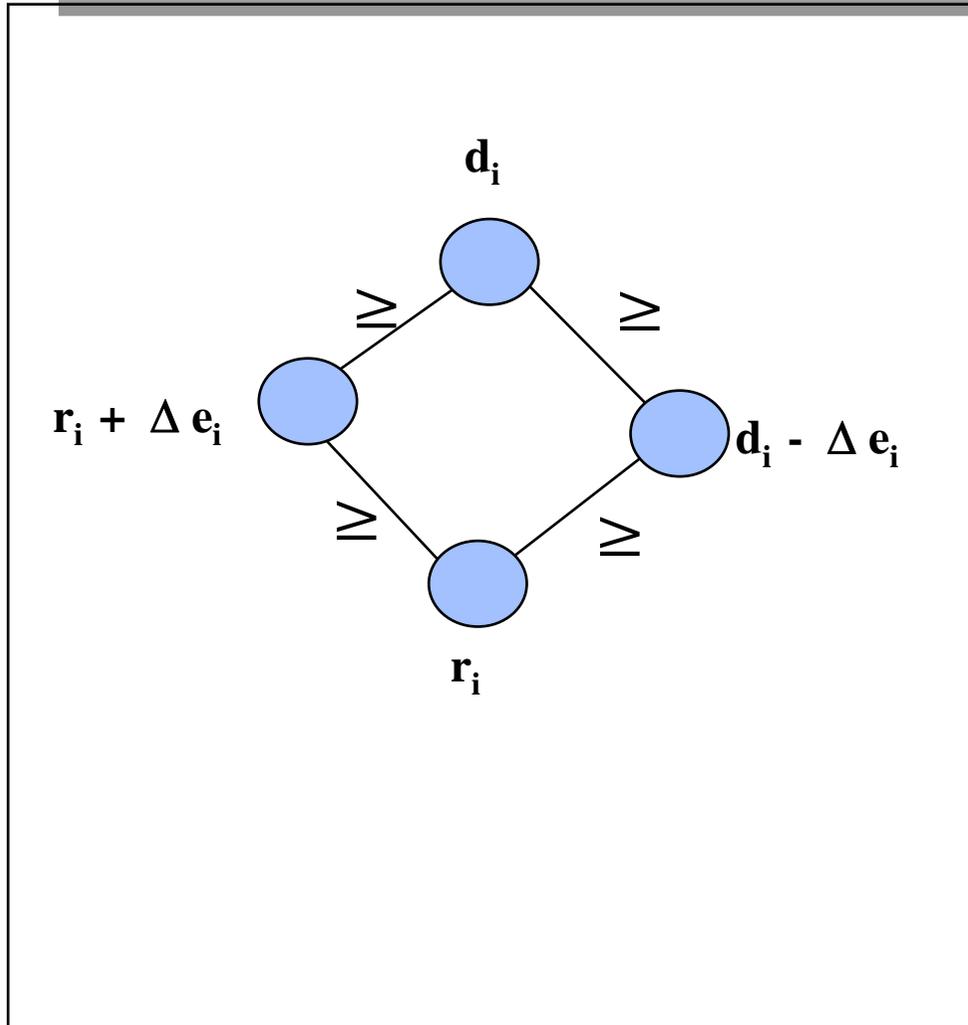


---

## Unterbrochene Ausführung von Tasks



## Beziehung zwischen den eingeführten Größen



# Scheduling

---

**Allgemeine Formulierung des Schedulingproblems:**

**Gegeben seien:**

Eine Menge von Tasks             $T = \{T_1, T_2, \dots, T_n\}$   
Eine Menge von Prozessoren     $P = \{P_1, P_2, \dots, P_m\}$   
Eine Menge von Ressourcen       $R = \{R_1, R_2, \dots, R_s\}$

**Def.:** Scheduling bedeutet die Zuordnung von Prozessoren und Ressourcen zu Tasks, so dass alle für individuelle Tasks definierten Beschränkungen eingehalten werden.

**In seiner allgemeinen Form ist das Scheduling-Problem NP-vollständig.**

→ **Einschränkungen:**      **Anzahl der betrachteten Prozessoren/Ressourcen, nur periodische Tasks, gleiche Bereitzeiten aller Tasks, keine Anhängigkeiten, nur feste Prioritäten, ...**



# Scheduling

---

**Was wird geplant ?**    **Prozessorressourcen**  
**Speicherressourcen**  
**Kommunikationsressourcen**

## **Planung von Prozessorressourcen:**

- Annahmen:**
- **Planbare Einheiten: unabhängige Tasks**
  - **Ressource: Ein einzelner Prozessor**
  - **Taks können zu jedem Zeitpunkt unterbrochen werden**
  - **Zeit für Systemaktivitäten wie Taskwechsel sind vernachlässigbar**

**Ausgangspunkte für die Planung sind *Planungsgrößen* wie:**  
**Bereitzeiten, Ausführungszeiten, Fristen (Deadlines), Perioden**



# Planungsphasen

---

**Einplanbarkeitsanalyse (Schedulability-Analysis)**

**Planerstellung (Schedule Construction)**

**Prozessorzuteilung (Dispatching)**



# Einplanbarkeitsanalyse (Schedulability-Analysis)

---

Unter Berücksichtigung der Planungsgrößen wird geprüft, ob ein Plan erstellt werden kann.

Eine Task wird charakterisiert durch :  $\langle T_i, r_i, \Delta e_i, f_i, d_i \rangle$

$T_i$  : Task - Nummer,

$r_i$  : Bereitzeit,

$\Delta e_i$  : Ausführungszeit (Worst Case Execution Time (WCET)),

$f_i$  : Frequenz bei periodischen Prozessen, bei sporadischen Prozessen die obere Schranke ihrer Häufigkeit,

$p_i$  : Periode bei periodischen Prozessen, bei sporadischen Prozessen die obere Schranke ihres zeitlichen Abstands :  $p_i = 1/f_i$

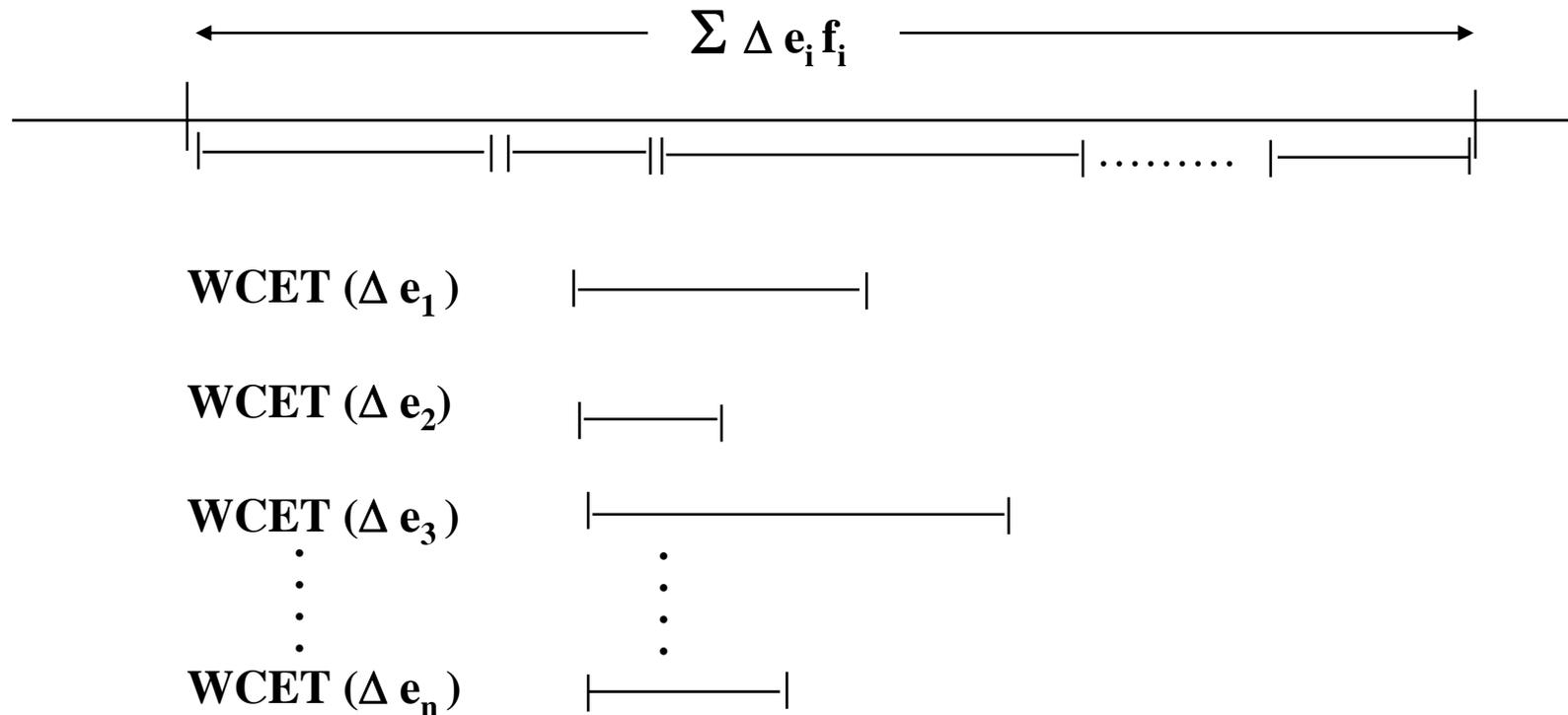
$d_i$  : Deadline.



# Einplanbarkeitsanalyse

## Notwendige Bedingungen für die Planbarkeit:

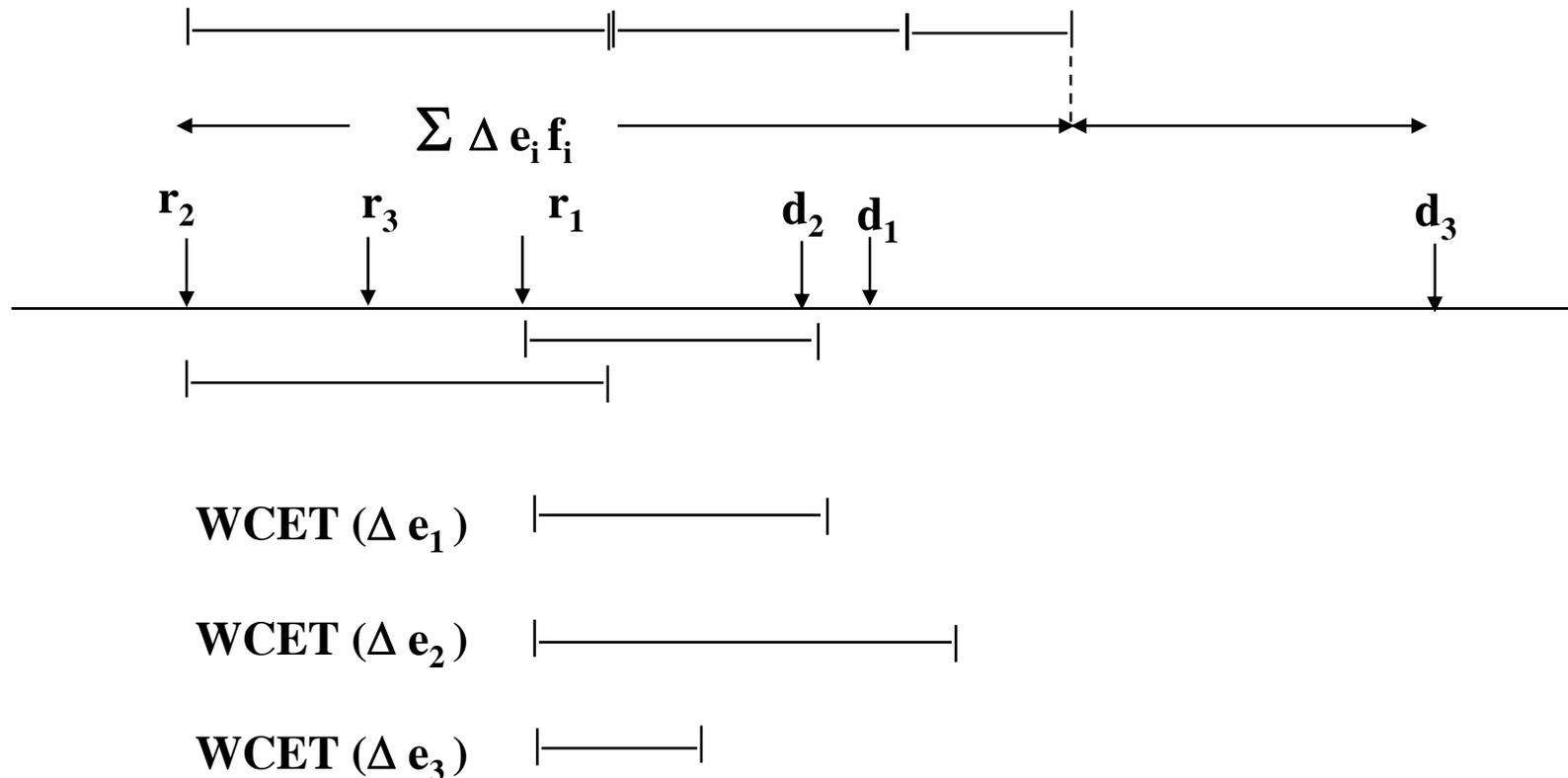
- 1.)  $\forall i : \Delta e_i < d_i - r_i < \Delta p_i$  ( $\Delta p_i = 1/f_i$ )
- 2.)  $\sum \Delta e_i f_i \leq$  verfügbare (Prozessor-) Ressourcen



## Beispiel: Notwendige aber nicht hinreichende Bedingungen

- 1.)  $\forall i : \Delta e_i < d_i - r_i < 1/f_i = p_i, f_i = 1$
- 2.)  $\sum \Delta e_i f_i \leq$  verfügbare (Prozessor-) Ressourcen

Beide notwendigen Eigenschaften sind erfüllt:



# Brauchbarkeit eines Plans

**Def.:** Ein Plan heißt brauchbar (valid, feasible) für eine Taskmenge  $\{T_1, T_2, \dots, T_n\}$ , falls bei vorgegebenem  $r_i, \Delta e_i, d_i$  und  $\Delta p_i$  die Startzeit  $s_i$  und die Abschlußzeit  $c_i$  so gewählt sind, daß:

- 1.) Alle Zeitbedingungen (für eine einzelne Task) eingehalten werden

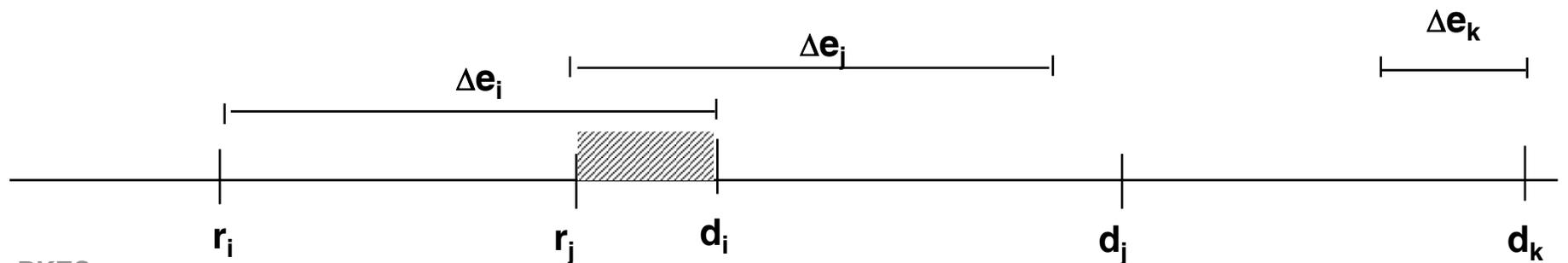
$$\forall i : \Delta e_i < d_i - r_i < \Delta p_i \quad (\text{notwendig})$$

- 2.) genügend Ressourcen vorhanden sind

$$\sum \Delta e_i f_i < \text{verfügbare Prozessorzeit} \quad (\text{notwendig})$$

- 3.) keine überlappenden Ausführungszeiten entstehen

$$\forall i, j, d_i < d_j : d_i \leq d_j - \Delta e_j \quad (\text{hinreichend})$$



**Beispiel:**

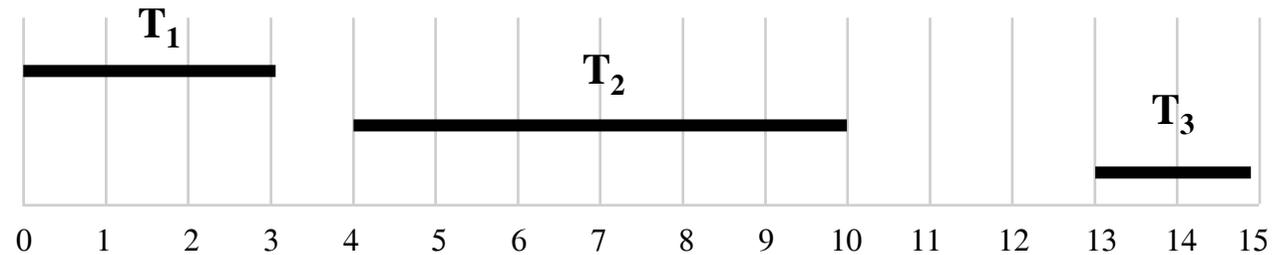
**Bedingung:**  $\forall i,j, d_i < d_j: d_i \leq d_j - \Delta e_j$

---

**In beiden Fällen wird die Bedingung nicht erfüllt !**

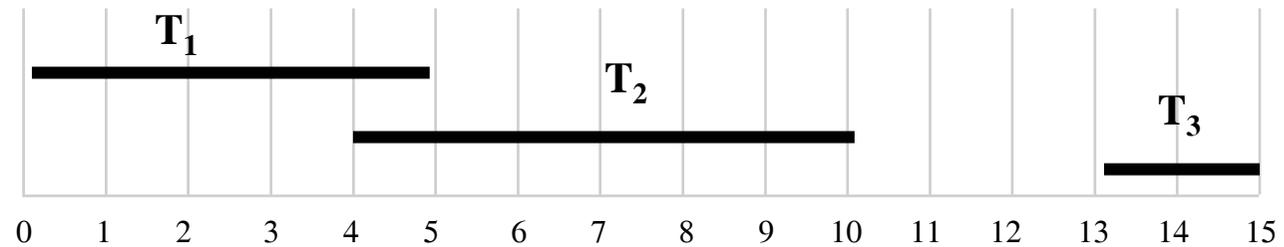
- $T_1 : \Delta e_1 = 3, r_1 = 0, d_1 = 6$
- $T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
- $T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

➡ **planbar !!**



- $T_1 : \Delta e_1 = 5, r_1 = 0, d_1 = 6$
- $T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
- $T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

➡ **nicht planbar !!**



# Einplanbarkeitsanalyse

---

Wenn der S-Test positiv verläuft,  
kann ein Plan gefunden werden.

Wenn der S-Test negative verläuft,  
kann kein Plan gefunden werden.

hinreichende Bedingungen

exakte  
Bed.

notwendige Bedingungen

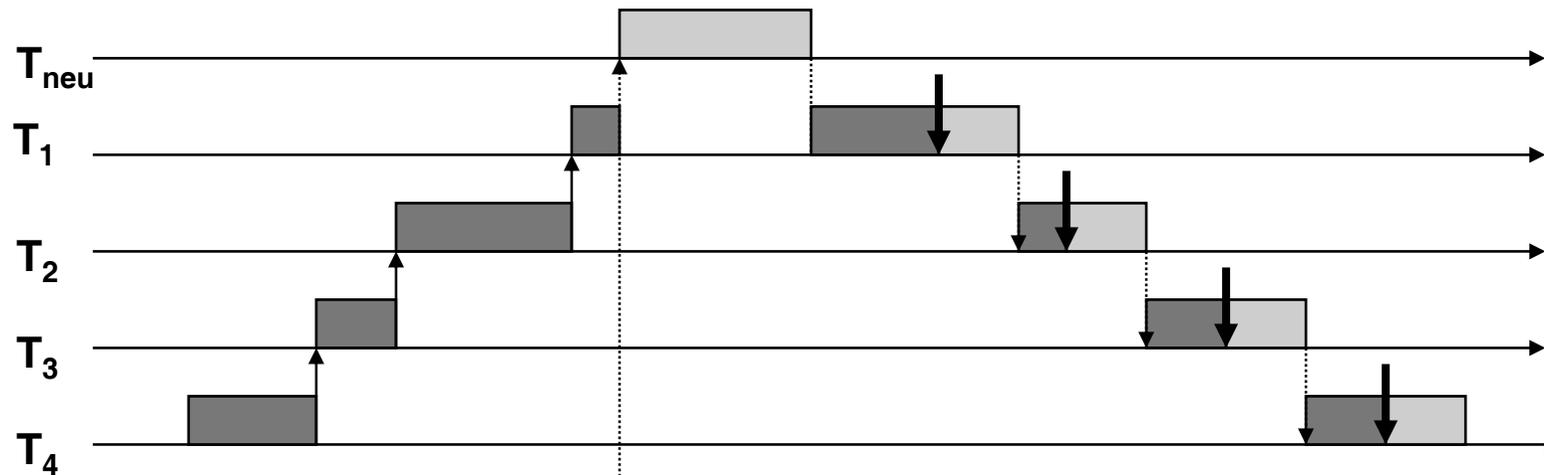
zunehmende Komplexität  
des Taskmenge

dieser Test wird einfacher, wenn man in Kauf nimmt, daß er für einige Taskmengen negative Resultate liefert, obwohl die Taskmenge planbar ist

eine einfache notwendige Bedingung ist, daß der Spielraum einer Task (laxity) größer als seine Ausführungszeit sein muß, d.h.  $d - r > \Delta e$ . Wenn das für alle Tasks gilt, bedeutet das aber noch nicht, daß ein Plan gefunden wird.



# Garantie-basierte Planungsverfahren

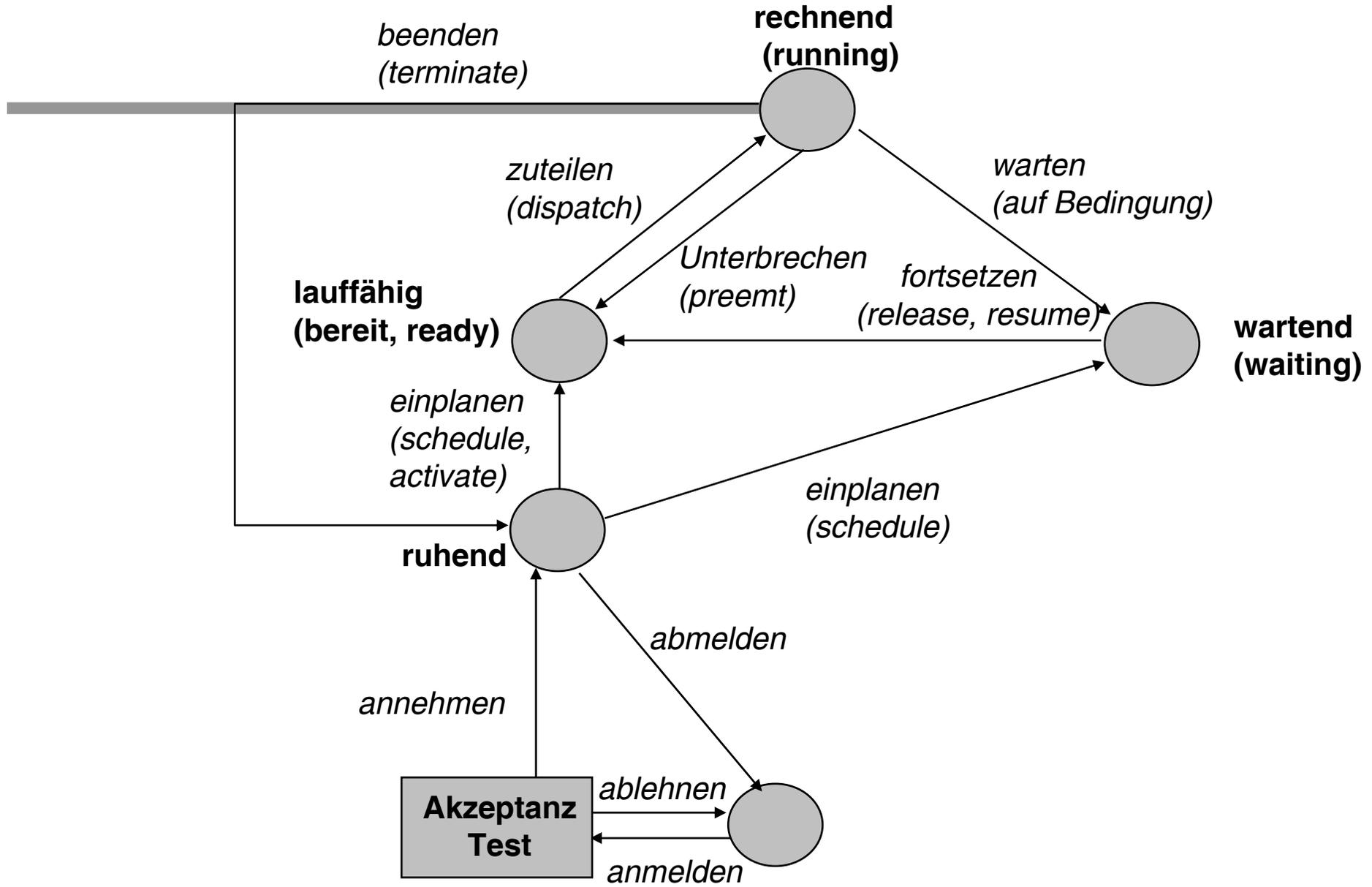


Die Ausführung der neuen Task führt dazu, dass keine der Deadlines mehr eingehalten wird.

→ Dominoeffekt

Zur Verhinderung dieses Effektes muss ein Einplanbarkeitstest dynamisch für jede neu hinzukommende Task ausgeführt werden.





---

# Annahmen und Zielkonflikte bei der Planerstellung

statischer

dynamischer



- statische, nicht unterbrechbare Tasks.
- Off-line Planbarkeitsanalyse
- Einplanbarkeit bewiesen.
- Plan z.B. in Form einer Tasktabelle mit nach Startzeiten geordneten Einträgen.
- Minimaler Overhead zur Laufzeit

+

- Mix aus periodischen und sporadischen Prozessen
- Prozesse unterbrechbar.
- harte und weiche Zeitbedingungen
- Plan liefert Wichtigkeit und Prioritäten
- On-line Planbarkeitsanalyse und Planerstellung
- Graceful degradation

---

Probleme mit Komplexität und der Bestimmung der individuellen WCET.

-

---

Planung erfordert Aufwand zur Laufzeit und muß mitgeplant werden



# Planungsphasen

---

**Einplanbarkeitsanalyse (Schedulability-Analysis)**

**Planerstellung (Schedule Construction)**

**Prozessorzuteilung (Dispatching)**



---

**Nach welchen Kriterien wird entschieden, ob ein Scheduler gut ist oder nicht ?**



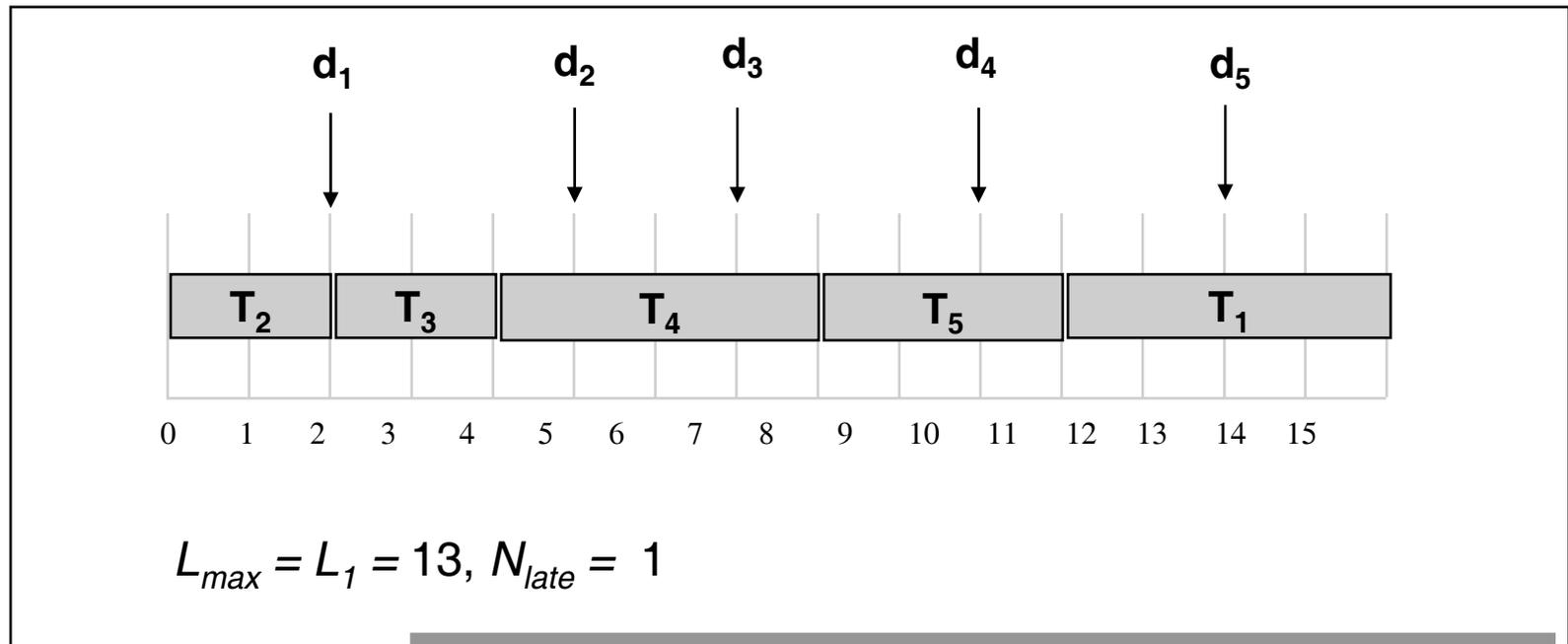
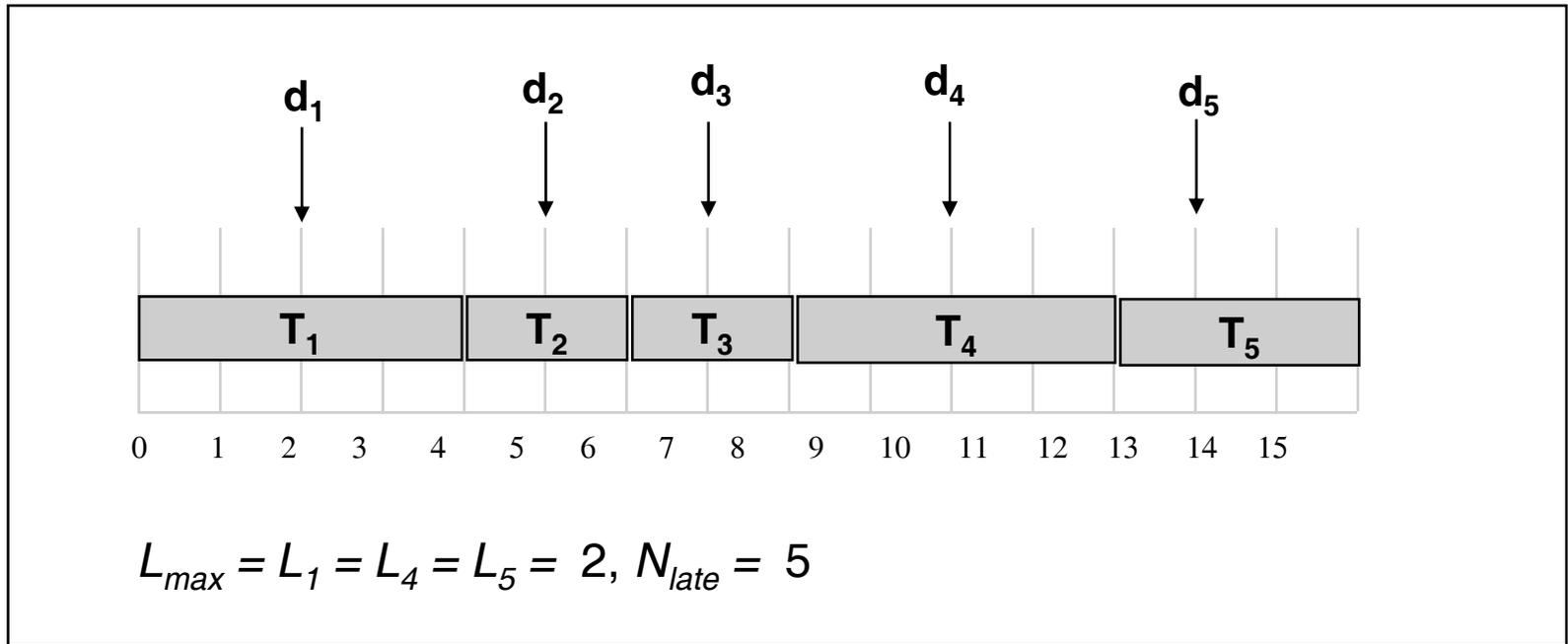
# Beispiele populärer Kostenfunktionen

---

- **Mittlere Antwortzeit:**  $t_r = 1/n \sum_{(i=1, \dots, n)} (c_i - r_i)$   
(average response time)
- **Zeit bis zum vollständigen Abschluß:**  $t_c = \max_i (c_i) - \min_i (r_i)$   
(total completion time)
- **Gewichtete Summe der Abschlußzeiten:**  $t_w = \sum_{(i=1, \dots, n)} w_i c_i$
- **Maximale Verspätung:**  $L_{max} = \max_i (c_i - d_i)$   
(maximum lateness)
- **Maximale Zahl verspäteter Tasks:**  $N_{late} = \sum_{(i=1, \dots, n)} miss(c_i)$   
mit:  $miss(c_i) = \begin{cases} 0 & \text{if } c_i \leq d_i \\ 1 & \text{sonst} \end{cases}$



Beispiele für  
Kostenfkt.



## Wesentliches Kriterium des klassischen Echtzeitschedulings:

- **Maximale Verspätung:**  $L_{max} = \max_i (c_i - d_i)$   
(maximum lateness)

Für Prozesse mit harten Zeitbedingungen gilt dabei:

$$L_{max} \leq 0$$

**Die vorgegebene Deadline muss immer eingehalten werden.**



# Allwissende Scheduler (Clairvoyant Scheduler)

---

**Ein Scheduler ist allwissend, wenn er beliebig weit in die Zukunft schauen kann.**

-Ein allwissender Scheduler findet daher immer einen Plan, wenn ein solcher existiert.

-Ein Scheduler heißt optimal, wenn er einen Plan findet, falls ein allwissender Scheduler einen solchen findet.



# Klassen von Planungsalgorithmen

---

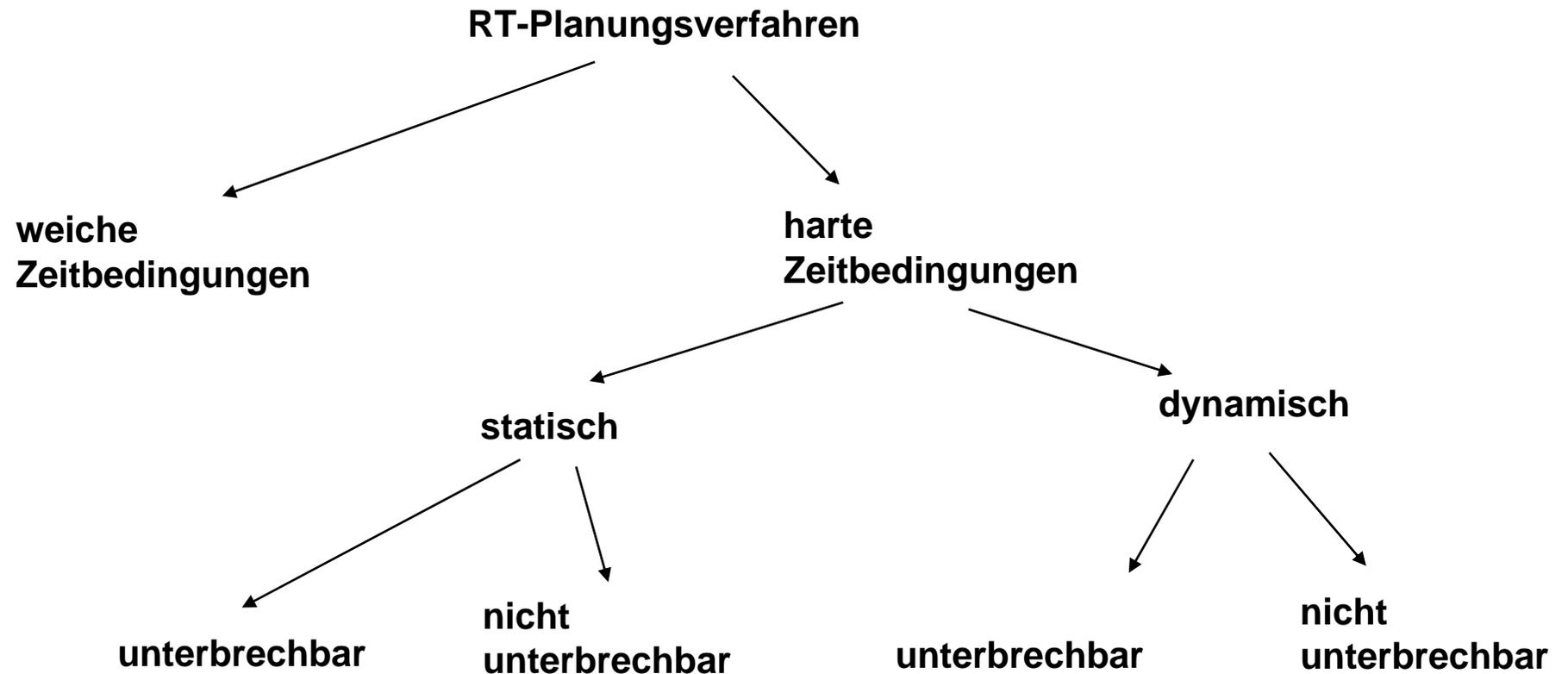
- **Off-line :** Es wird ein vollständiger Schedule für die Menge der Tasks erstellt, z.B in Form einer Tabelle, die später von einem Dispatcher ausgeführt wird.
- **On-line:** Planungsentscheidungen werden zur Laufzeit getroffen, wenn eine Task beendet wird, die Bereitzeit einer Task erreicht ist, oder eine neue Task zur Menge der Tasks hinzu kommt.
- Statisch:** Planungsentscheidungen basieren auf festen Parametern, die vor Aktivierung der Tasks festgelegt sind.
- Dynamisch:** Parameter ändern sich während der Laufzeit.
- Optimal:** Eine Algorithmus ist optimal, wenn er eine gegebene Kostenfunktion für eine Menge von Tasks minimiert. Angenommen es sei keine Kostenfunktion gegeben und das einzige Ziel ein brauchbarer Plan ist. Ein Algorithmus heißt dann optimal, wenn er nur dann keinen brauchbaren Plan liefert, wenn auch kein anderer Algorithmus derselben Klasse einen brauchbaren Plan liefert.
- Heuristisch:** Ein Algorithmus heißt heuristisch, wenn es nicht garantiert werden kann, daß er einen optimalen Plan findet (er aber diesem Ziel möglichst nahe kommt).



# Taxonomie der RT-Planungsverfahren

(Cheng, Stankovic, Ramamritham)

---



# Taxonomie der RT-Planungsverfahren (Einprozessor, harte Realzeitbedingungen)

---

