Grundlagen der Echtzeitplanung

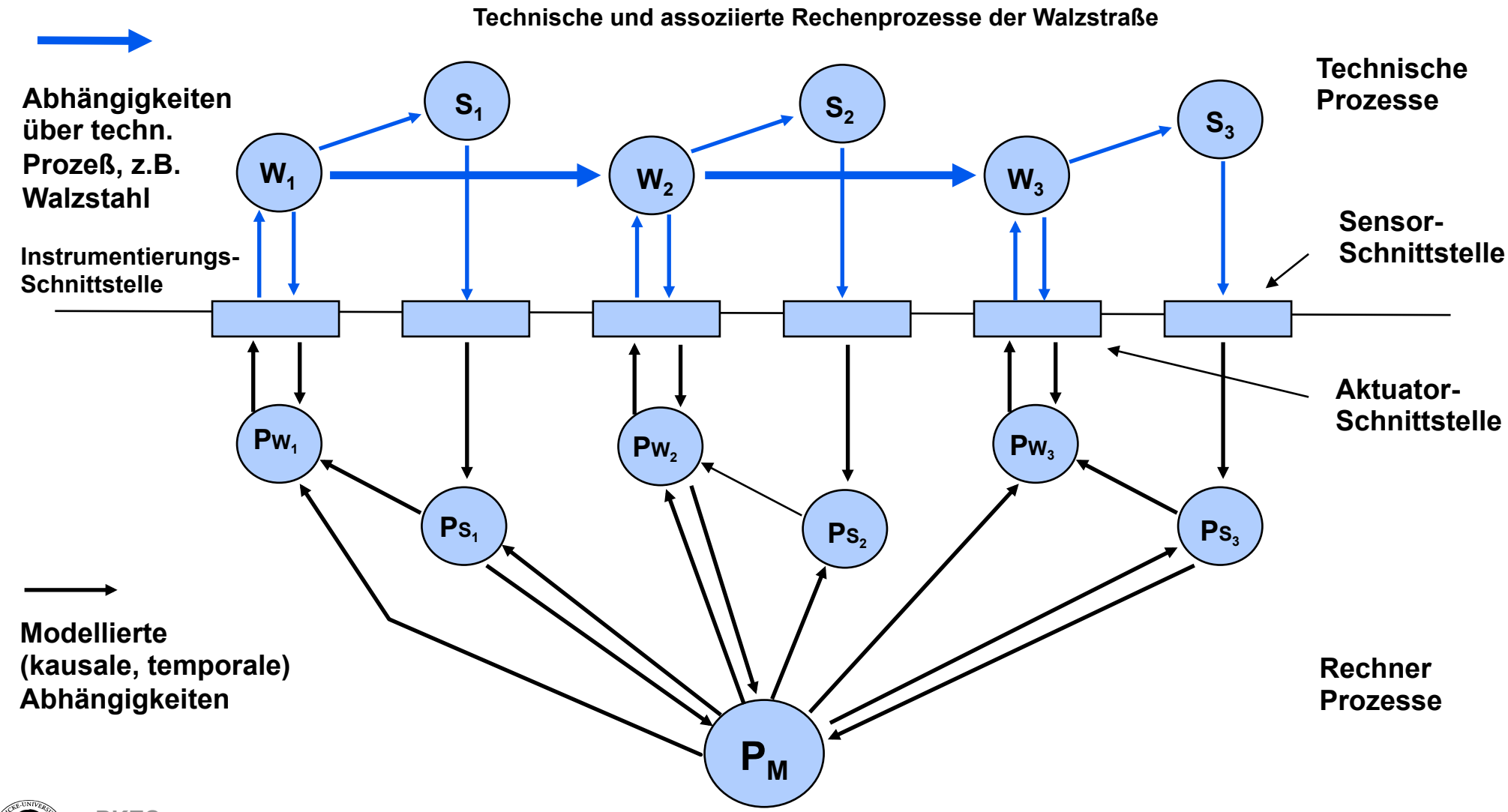
1. Grundlegende Begriffe und Konzepte

2. Planungsverfahren (Scheduling)

2.1 Planen aperiodischer Tasks
- Planen durch Suchen
- Planen nach Fristen (EDF)
- Planen abhängiger Tasks

2.2 Planen periodischer Tasks
- Planen nach monotonen Raten

Modellierung

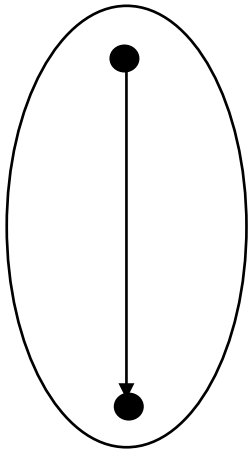


Taskmodell

Eine Task ist die Ausführung eines sequentiellen Programms auf einem Prozessor in seiner spezifischen Umgebung (Kontext).

Eine Task ist:

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

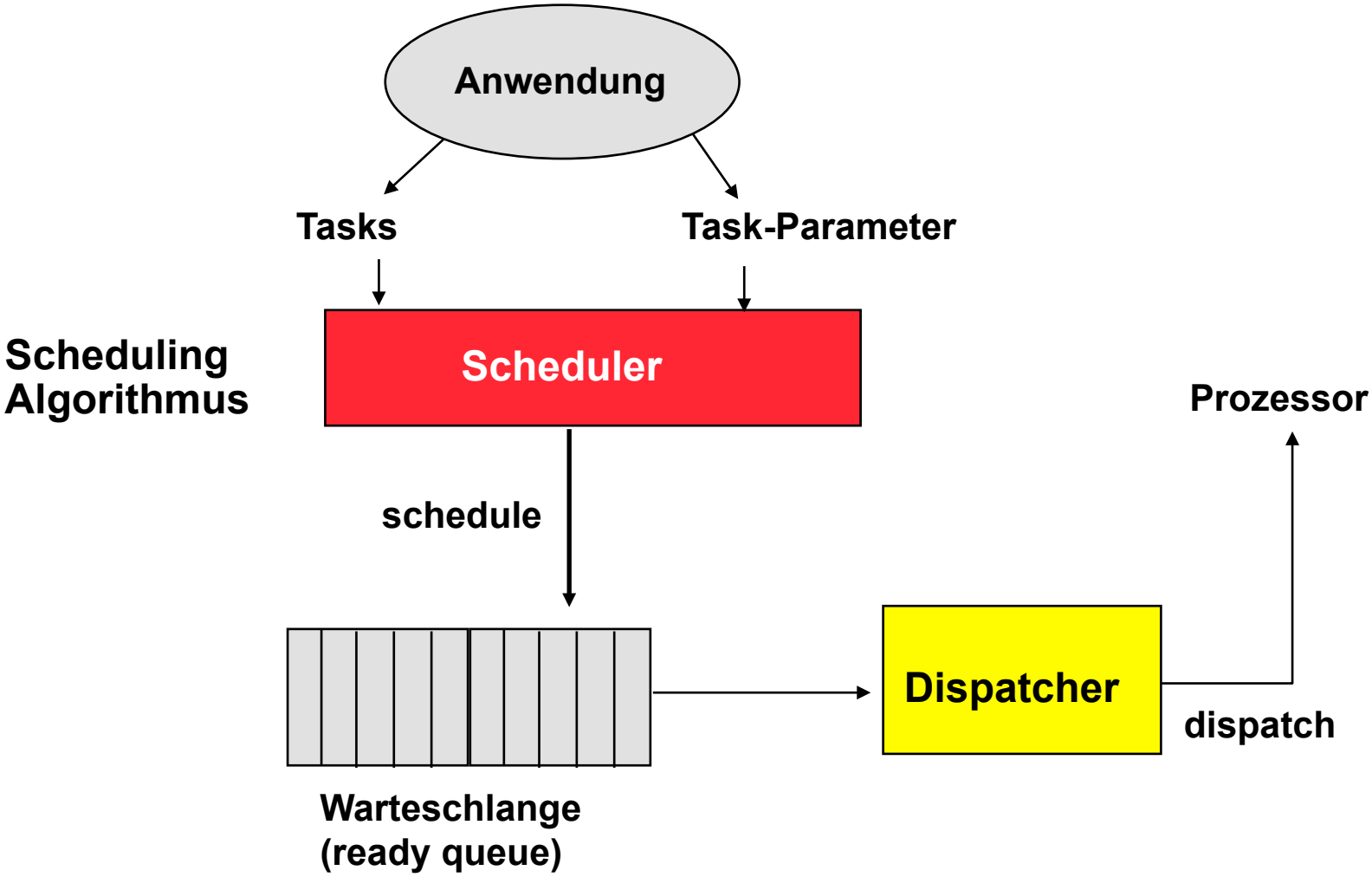


Start

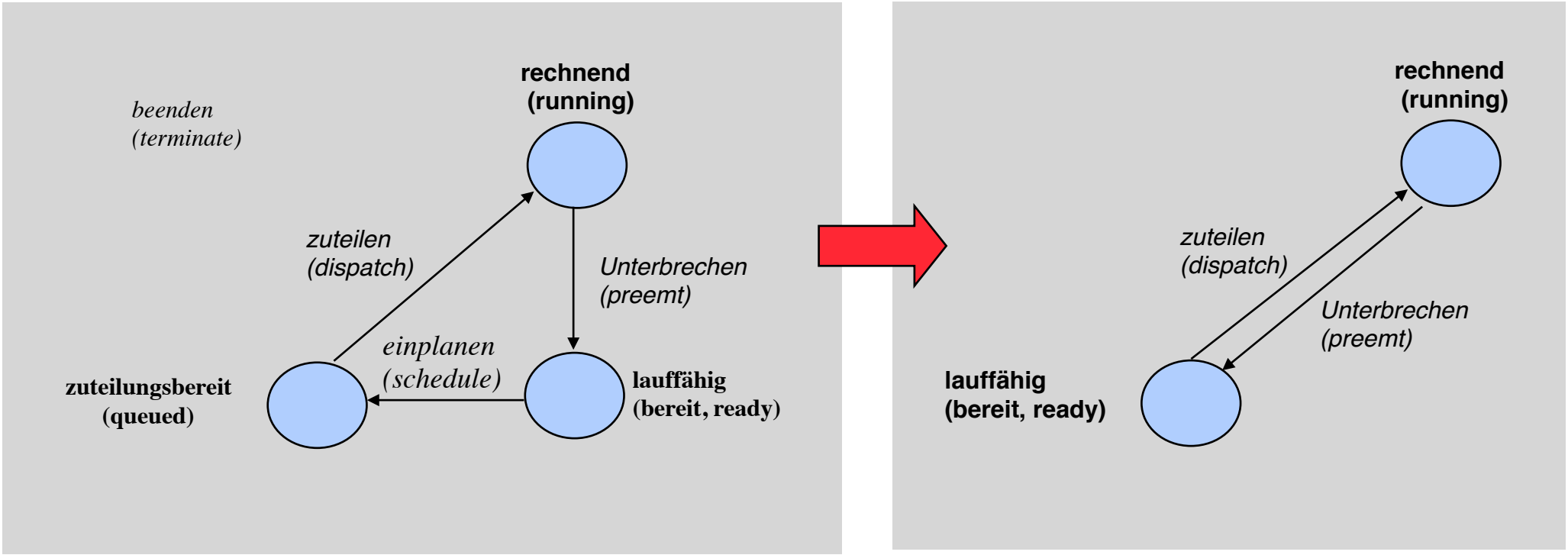
Task als sequentielle und endliche Ausführungsfolge von Instruktionen

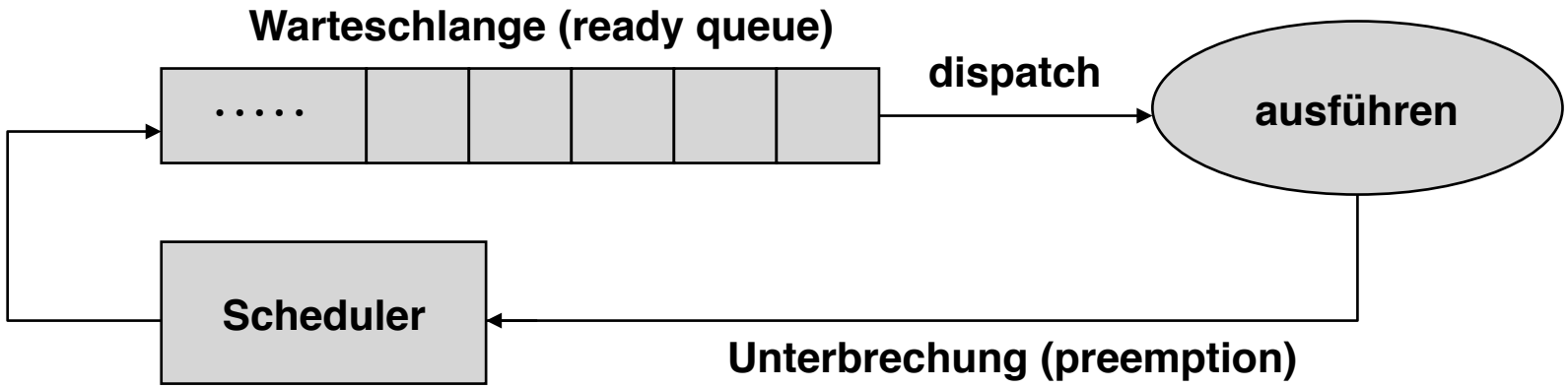
Abschluß

Scheduling und Dispatching

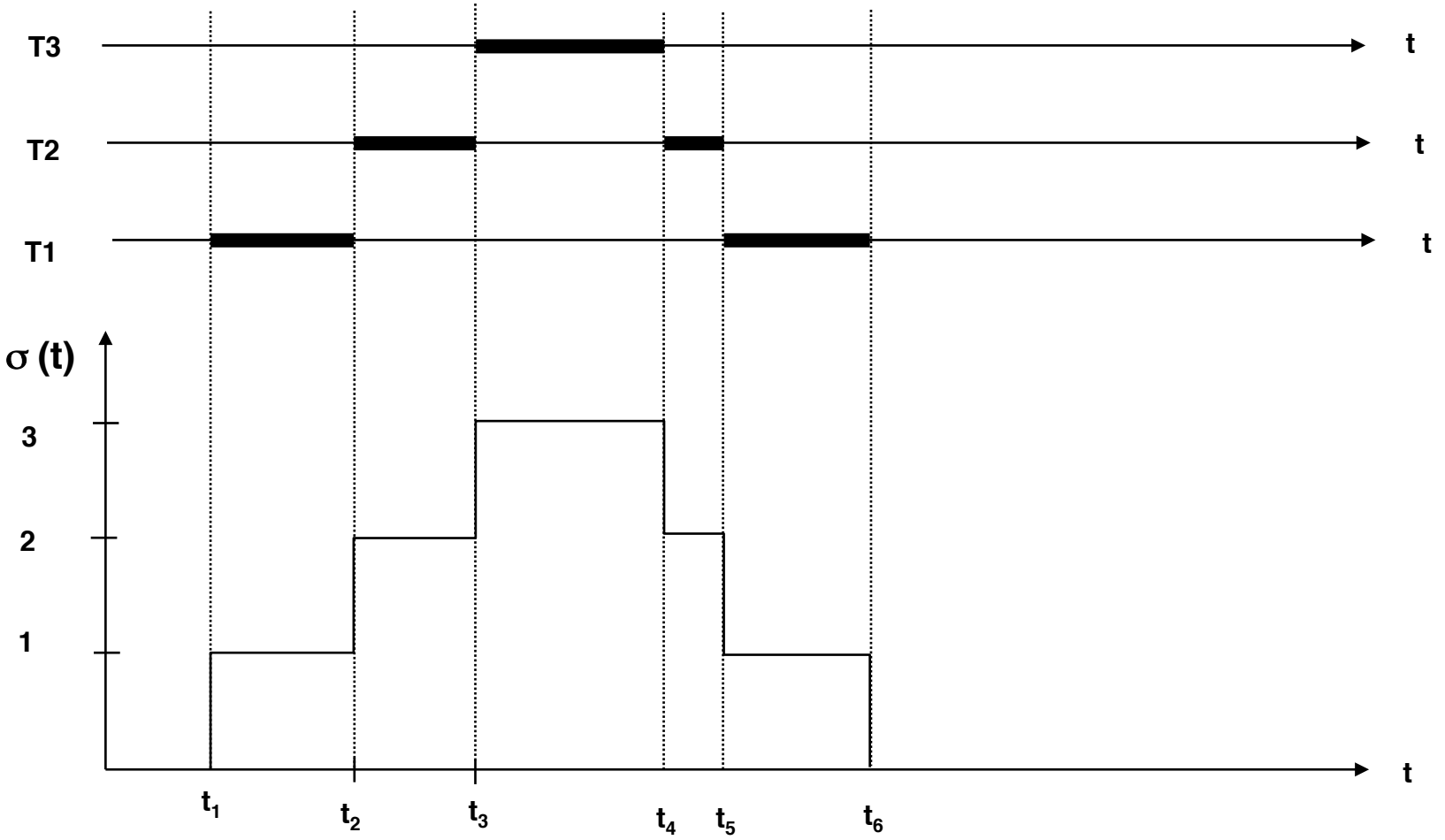


Zustandsdiagramm einer Task





Beispiel eines unterbrechbaren Schedules



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) Δ e_i :**
  Reine Rechenzeit auf dem Prozessor. Zur Planung legt man meist die "worst case execution time" (WCET) zugrunde.

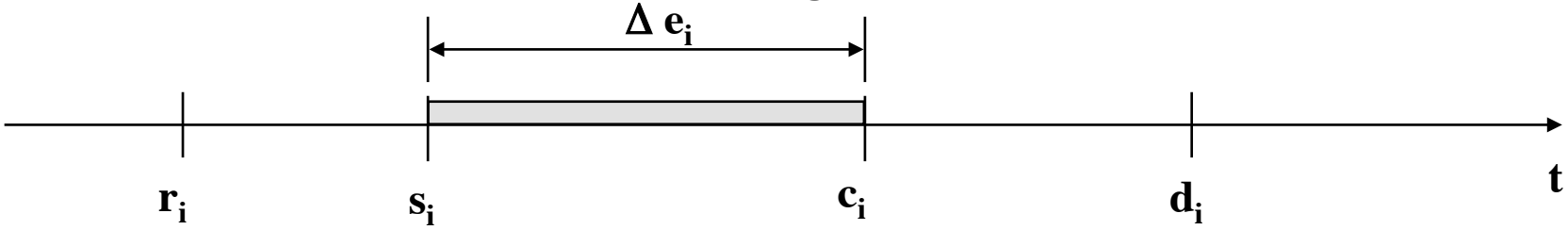
- **Startzeit (starting time) s_i :**
  Prozessor beginnt T_i auszuführen.

- **Abschlußzeit (completion time) c_i :**
  Prozessor beendet die Ausführung von T_i.

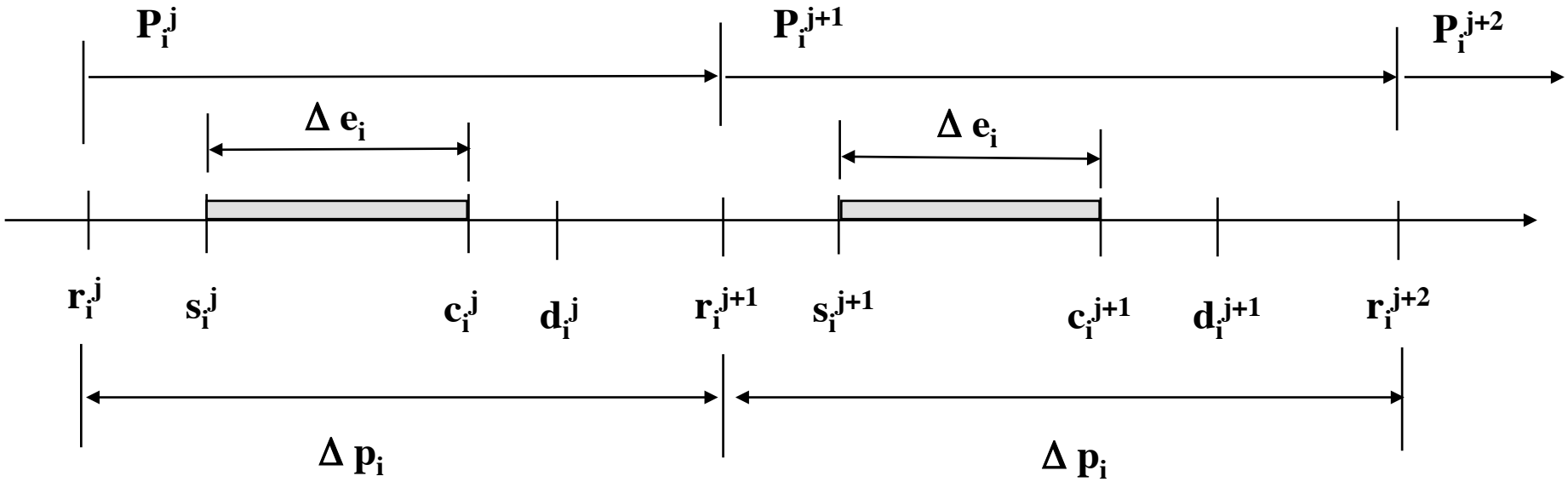
- **Frist (deadline) d_i :**
  Zu diesem Zeitpunkt muß die Ausführung von T_i abgeschlossen sein.

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 Δp_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.

Scheduling

Allgemeine Formulierung des Schedulingproblems:

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

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, ...**

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

Beispiele populärer Kostenfunktionen

- **Mittlere Antwortzeit:** $t_r = 1/n \sum_{(i=1, ..., n)} (c_i - r_i)$
  (avarage response time)

- **Zeit bis zum vollständigen Abschluß:** $t_c = \max_i (c_i) - \min_i (r_i)$
  (total completion time)

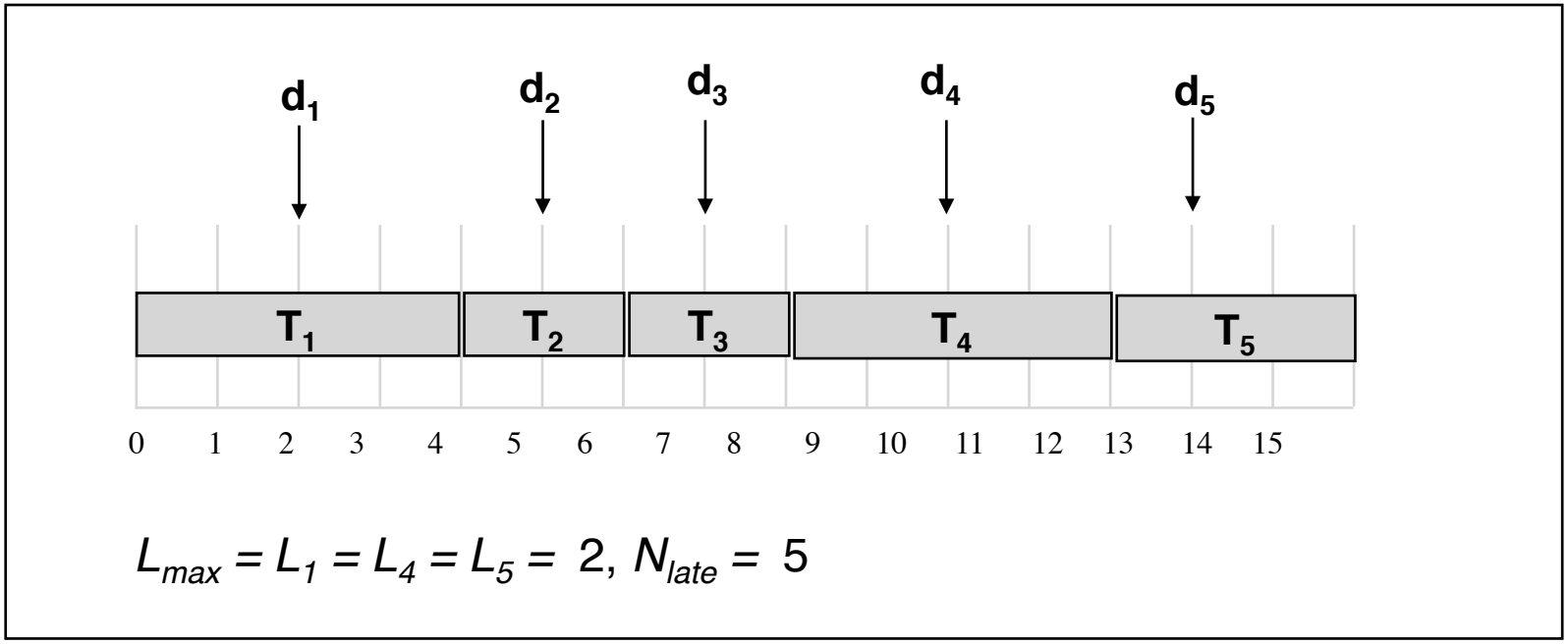
- **Gewichtete Summe der Abschlußzeiten:** $t_w = \sum_{(i=1, ..., n)} w_i c_i$

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

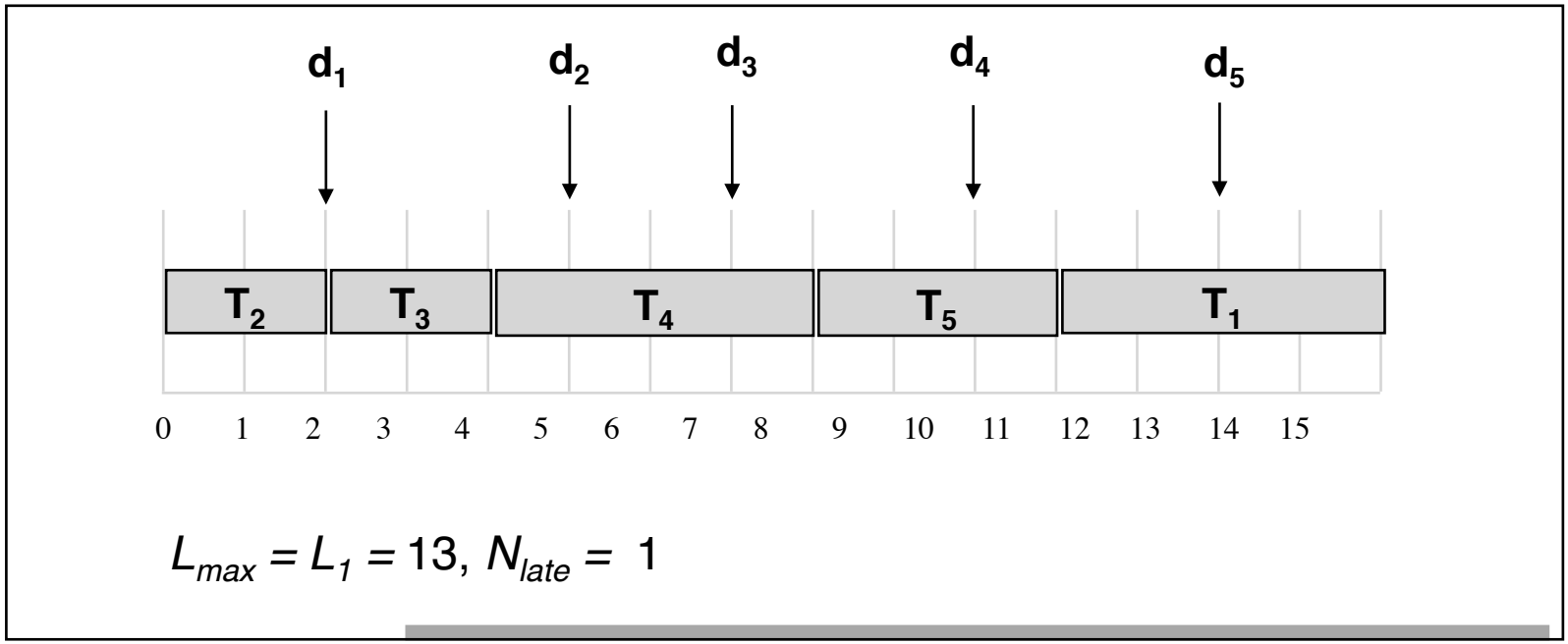
- **Maximale Zahl verspäteter Tasks:** $N_{late} = \sum_{(i=1, ..., n)} miss(c_i)$

  mit: $$miss(c_i) = \begin{cases} 0 & \text{if } c_i \le d_i \\ 1 & \text{sonst} \end{cases}$$

Beispiele für Kostenfkt.



$L_{max} = L_1 = L_4 = L_5 = 2$, $N_{late} = 5$



$L_{max} = L_1 = 13$, $N_{late} = 1$

Wesentliches Kriterium des klassischen Echtzeitschedulings:

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

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.

Planungsphasen

Einplanbarkeitsanalyse (Schedulabilty-Analysis)

Planerstellung (Schedule Construction)

Prozessorzuteilung (Dispatching)

Einplanbarkeitsanalyse (Schedulabilty-Analysis)

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

Eine Task wird charakterisiert durch : $< T_i, r_i, \Delta e_i, f_i, d_i >$

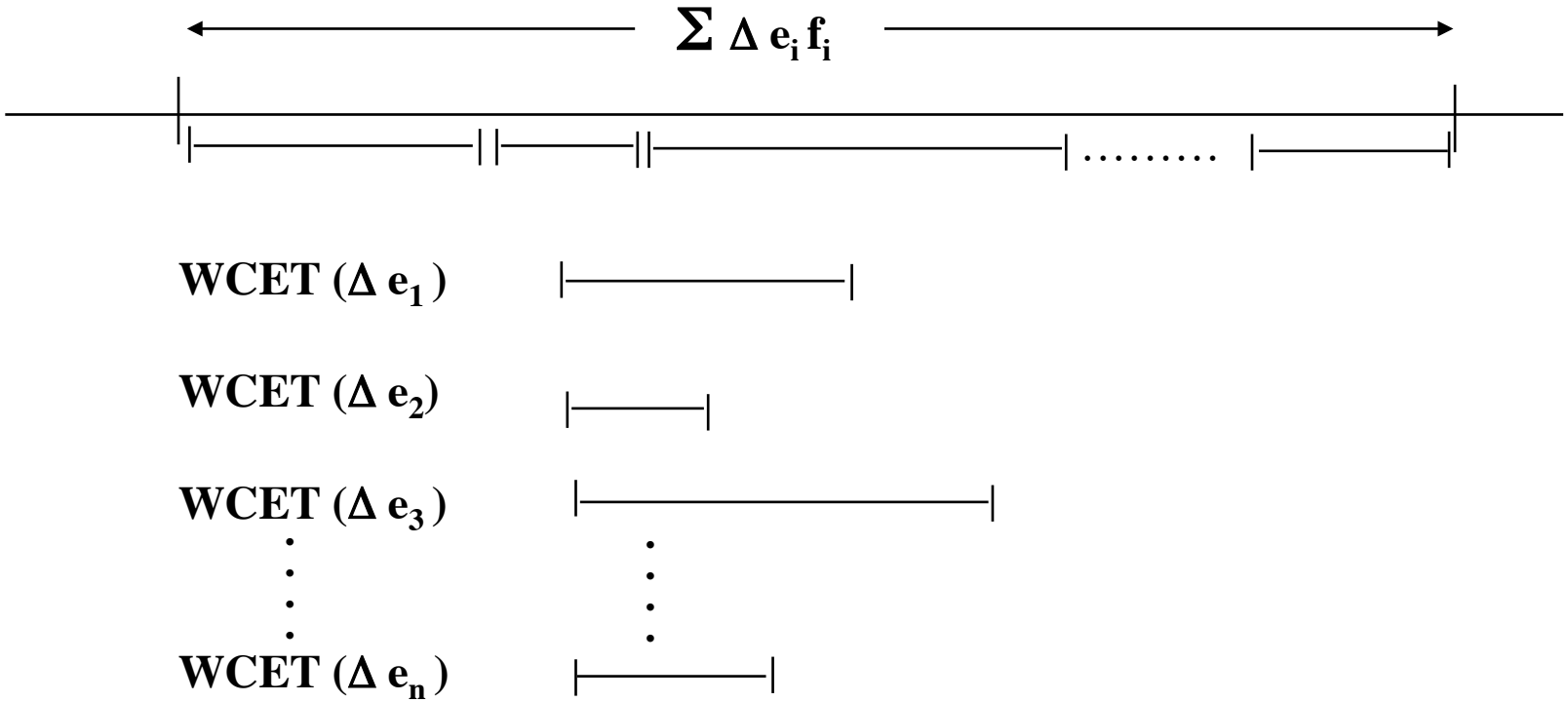
T_i : Task - Nummer,
r_i : Bereitzeit,
Δe_i : Ausführungszeit (Worst Case Execution Time (WCET),
f_i : Frequenz bei periodischen Prozessen, bei sporadischen Prozessen die obere Schranke ihrer Häufigkeit,
Δp_i : Periode bei periodischen Prozessen, bei sporadischen Prozessen die obere Schranke ihres zeitlichen Abstands : $\Delta p_i = 1/f_i$
d_i : Deadline.

Einplanbarkeitsanalyse

Notwendige Bedingungen für die Planbarkeit:

1.) $\forall i : \Delta e_i < d_i - r_i < \Delta p_i \quad (\Delta p_i = 1/f_i)$

2.) $\sum \Delta e_i f_i \leq d_{max}$ (verfügbare (Prozessor-) Zei)t

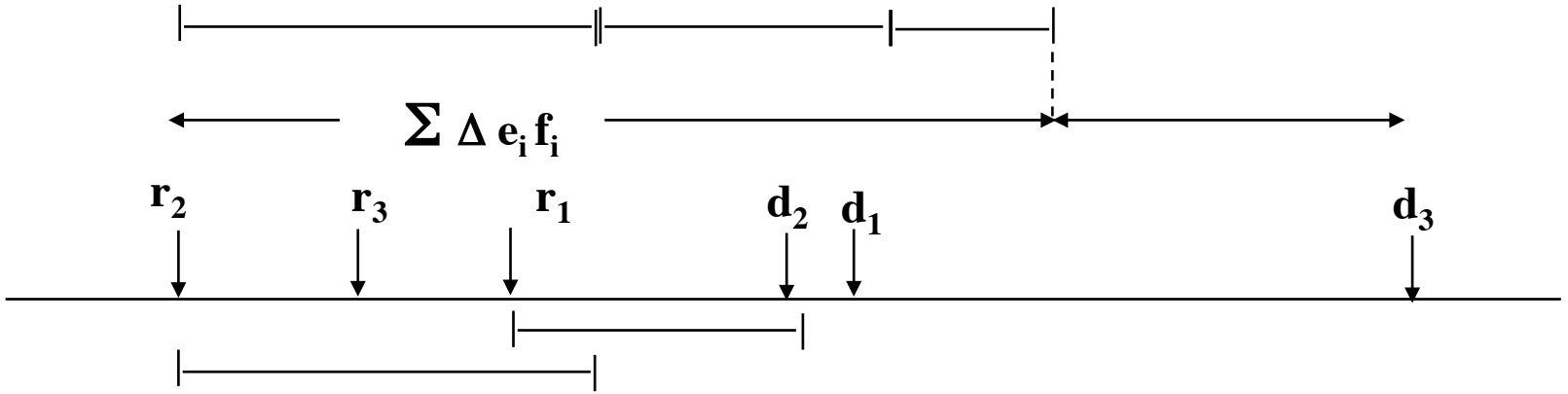


Beispiel: Notwendige aber nicht hinreichende Bedingungen

1.) $\forall i : \Delta e_i < d_i - r_i < 1/f_i = \Delta p_i$

2.) $\Sigma \Delta e_i f_i \leq d_{max}$ (verfügbare (Prozessor-) Zeit)

Beide notwendigen Eigenschaften sind erfüllt:



WCET (Δe_1)

WCET (Δe_2)

WCET (Δe_3)

Brauchbarkeit eines Plans

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

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

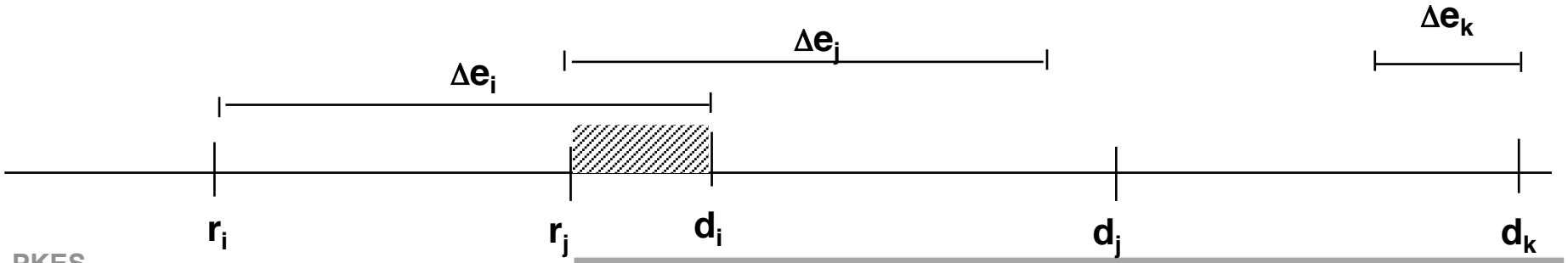
$$\forall i \; : \Delta e_i < d_i - r_i < \Delta p_i \quad \text{(notwendig)}$$

2.) **genügend Ressourcen vorhanden sind**

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

3.) **keine überlappenden Ausführungszeiten entstehen**

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



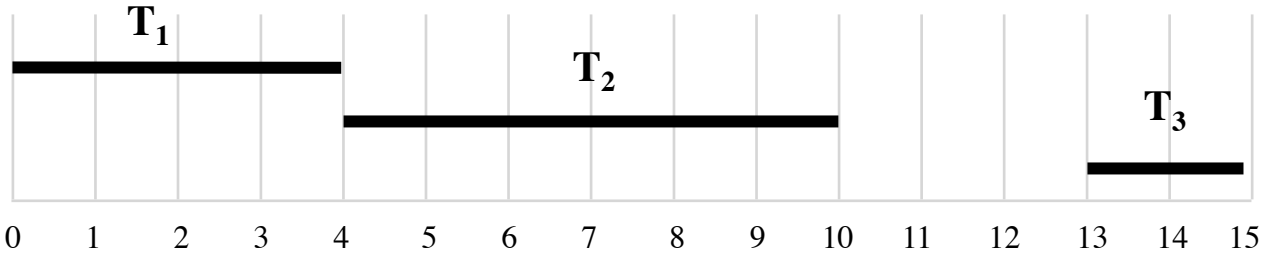
Beispiel:

Bedingung: $\forall i,j,\ d_i < d_j:\quad d_i \le d_j - \Delta e_j$

In beiden Fällen wird die Bedingung nicht erfüllt !

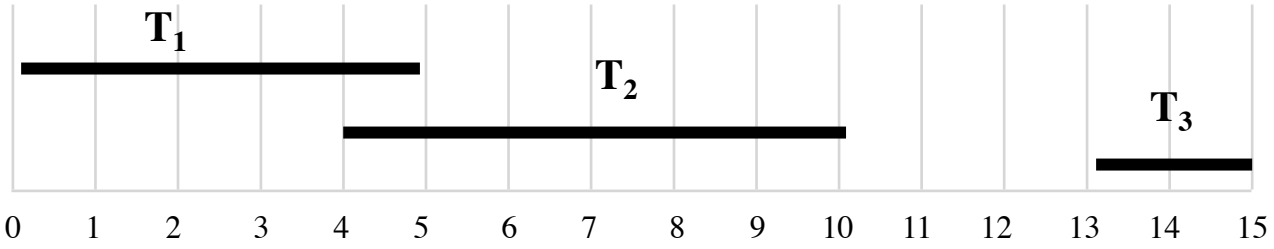
$T_1: \Delta e_1 = 4, r_1 = 0, d_1 = 6$
$T_2: \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3: \Delta e_3 = 2, r_3 = 13, d_3 = 15$

⇨ **planbar !!**

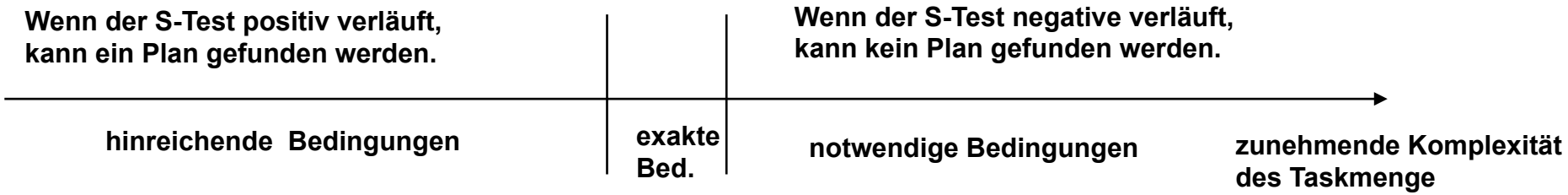


$T_1: \Delta e_1 = 5, r_1 = 0, d_1 = 6$
$T_2: \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3: \Delta e_3 = 2, r_3 = 13, d_3 = 15$

⇨ **nicht planbar !!**



Einplanbarkeitsanalyse



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.