Komposition: ProOSEK

Echtzeitsysteme 2 - Vorlesung/Übung

Fabian Scheler
Michael Stilkerich
Wolfgang Schröder-Preikschat

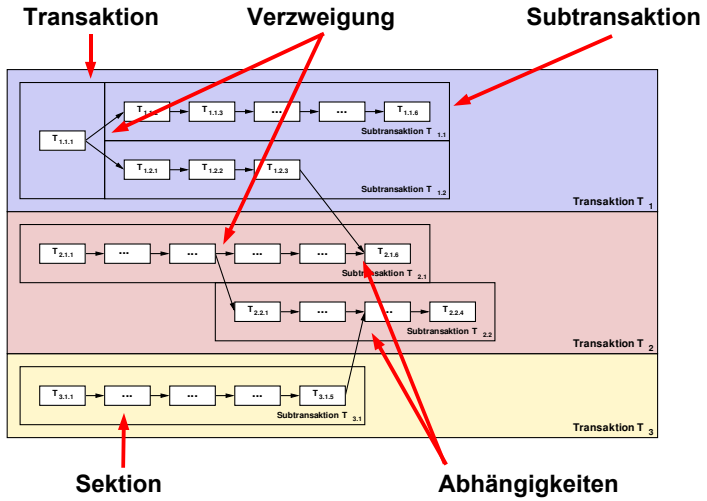
Lehrstuhl für Informatik IV
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander Universität Erlangen-Nürnberg

http://www4.cs.fau.de/~{scheler,mike,wosch}
{scheler,mike,wosch}@cs.fau.de

Übersicht

- Wiederholung
  - Ereignisbehandlungen
  - Abbildung
  - Antwortzeitanalyse
- ProOSEK
  - Abbildung
  - Antwortzeitanalyse
  - Hinweise

Wiederholung: Ereignisbehandlungen



Wiederholung: Abbildung

- Implementierung von
  - Transaktionen, Subtransaktionen und Sektionen
  - Konkatenation von Sektionen
  - Verzweigungen
  - Abhängigkeiten
- Implementierung mit den Mitteln das zu Grunde liegenden Betriebssystems

Wiederholung: Antwortzeitanalyse

- **Auslösezeitpunkt**: Zeitpunkt an dem ein Ereignis eintritt
- **Antwortzeit**: Zeitpunkt an dem die Behandlung des Ereignisses abgeschlossen ist
  - mit anderen Worten: eine Subtransaktion
- **maximale Antwortzeit**:

$$w_i(t) = e_i + \sum_{k=1}^{i-1} ceiling(\frac{t}{p_k}) e_k$$

$$w_i(t) \leq t \,;\, t = jp_k \,;\, k = 1,2,\ldots,i \,;\, j = 1,2,\ldots,floor(min(d_i, p_i)/p_k)$$

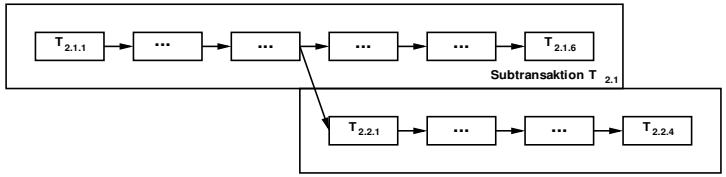
- hinreichendes und notwendiges **Kriterium für Planbarkeit**:

$$max_t w_i(t) \leq d_i \, \forall Tasks \, t_i$$

ProOSEK: Abbildung

- Verzweigungen
- Abhängigkeiten
- aperiodische Ereignisbehandlungen

ProOSEK: Verzweigungen

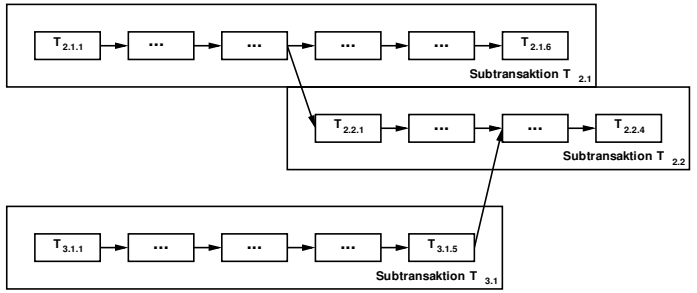


- Transaktion spaltet sich in unabhängige Kontrollflüsse
- Subtransaktionen münden nicht wieder zurück

```
TASK(A) {
  ...
  ActivateTask(B);
  ...
  ChainTask(C);
}
```

- eine Subtransaktion ensteht nur wenn
  - Task A nicht-präemptiv ist oder
  - Priorität(B) < Priorität(A)
- nieder-priore Subtransaktionen zweigen immer von hochprioren Subtransaktionen ab

ProOSEK: Abhängigkeiten



- Arten von Abhängigkeiten
  - ODER-Abhängigkeiten
  - UND-Abhängigkeiten
- Nachfolger hängt von mehreren Vorgängern ab

ProOSEK: ODER-Abhängigkeiten

- **Ansatz**: mehrfache Auslösbarkeit des Nachfolgers
  - jeder Vorgänger löst jeweils den Nachfolger aus
- **zu beachten**:
  - Prioritätenvergabe
  - Anzahl der Auslösungen

ProOSEK: UND-Abhängigkeiten

- **Ansatz**: EVENTs
  - Nachfolger wartet zyklisch bis mehrere EVENTs gesetzt sind
  - Vorgänger setzen jeweils ein EVENT

```
Task(Successor) {
  EventMaskType eventsAwaited;
  EventMaskType eventsNotSet;
  eventsAwaited = e1 | e2 | e3;
  eventsNotSet = eventsAwaited;

  do {
    EventMaskType eventsSet = 0;
    WaitEvent(eventsNotSet);
    GetEvent(Successor,&eventsSet);
    eventsNotSet ^=
      (eventsSet & eventsAwaited);
  } while(eventsNotSet != 0);
  ...
  TerminateTask();
}
```

```
Task(Predecessor1) {
  ...
  SetEvent(Successor,e1);
  TerminateTask()
}
Task(Predecessor2) {
  ...
  SetEvent(Successor,e2);
  TerminateTask()
}
Task(Predecessor3) {
  ...
  SetEvent(Successor,e3);
  TerminateTask()
}
```

ProOSEK: UND-Abhängigkeiten

- **zu beachten**:
  - Overhead durch Kontextwechsel
  - erschwert Antwortzeitanalyse
  - Prioritätenvergabe

ProOSEK: UND-Abhängigkeiten

- **Ansatz**: ALARM & USERCOUNTER
  - Nachfolger wird von einem ALARM ausgelöst
  - ALARM wird durch einen USERCOUNTER gesteuert
  - Vorgänger inkrementieren den USERCOUNTER

```
TASK(Successor) {
  ...
  SetRelAlarm(alarm,3,0);
  TerminateTask();
}

Task(Predecessor1) {
  ...
  AdvanceCounter(counter);
  TerminateTask()
}
```

```
Task(Predecessor2) {
  ...
  AdvanceCounter(counter);
  TerminateTask()
}

Task(Predecessor3) {
  ...
  AdvanceCounter(counter);
  TerminateTask()
}
```

ProOSEK: UND-Abhängigkeiten

- **zu beachten**:
  - Prioritätenvergabe
  - Wann soll der ALARM ablaufen?
    - Vorgänger können unterschiedliche Periode besitzen
  - Der ALARM muss initialisiert werden

ProOSEK: aperiodische Ereignisse

- verfügbare Methoden: *deferred handling*
- Vorgehen
  - Ereignis wird initial durch ISR behandelt
  - ISR löst einen TASK aus
  - Auslösung *speichert* das Ereignis
- nicht implementierbar:
  - *deferred server*
  - *sporadic server*
  - ...
- zu beachten:
  - minimale Zwischenankunftszeit
  - pessimistische Antwortzeitanalyse
  - Anzahl der Auslösungen

ProOSEK: Ablaufplanung

- Begriffe
- Algorithmus
- einfache Antwortzeitanalyse
- erweiterte Antwortzeitanalyse

Begriffe

- Transaktionen, Subtransaktionen, Sektionen
- **Ausführungspfad**

  Ein **Ausführunspfad** ist ein Kontrollfluss innerhalb einer Subtransaktion und besteht aus Sektionen die sequentiell hintereinander ausgeführt werden. Dieselbe Sektion kann auch mehrmals in einem Ausführungspfad auftreten.
- **Abschnitt**

  Ein Abschnitt ist eine Menge von Sektionen, die in einem Ausführungspfad direkt aufeinander folgen.
- WCET der Subtransaktion $T_{n.m}$:

$$e_{n.m} = max_{Pa_{n.m.l} \in T_{n.m}} \sum_{T_{n.m.i} \in Pa_{n.m.l}} e_{n.m.i} k_{n.m.l}(T_{n.m.i})$$

$Pa_{n.m.l}$	Pfad l der Subtransaktion $T_{n.m}$
$k_{n.m.l}(T_{n.m.i})$	Auftreten der Sektion $T_{n.m.i}$ im Pfad $Pa_{n.m.l}$

Begriffe

- Deadline, Periode, minimale Zwischenankunftszeit
- Planbarkeit
- **kritischer Zeitpunkt**

  Eine Subtransaktion erreicht ihre **maximale Antwortzeit** genau dann, wenn sie ihre maximale Ausführungszeit erreicht und an einem **kritischen Zeitpunkt** ausgelöst wird.

Algorithmus

- OSEK
  - Vorrangsteuerung
  - MLQ-Scheduler
  - statisch bestimmte Prioritäten
- → zur Auswahl stehende Algorithmen
  - RMA
  - DMA
- → Entscheidung: DMA
  - Begründung: weniger restriktiv als RMA

Einfach Ablaufplanung

- trivialer Fall
- TASKs
- kritische Abschnitte
- ALARMe
- Self-Suspension
- beliebige Deadlines

Trivialer Fall

- Ereignisse
  - strikt periodisch
  - Deadline kleiner oder gleich Periode
  - keine Abhängigkeiten
- OSEK
  - ausschließlich ISRs Cat. 1 und Cat. 2
  - ISRs verhalten sich präemptiv
  - kein ISR sperrt Interrupts
  - ausreichende Anzahl von Prioritätsebenen
  - keine
    - TASKs
    - RESOURCEn
    - ALARMe
    - EVENTs

Trivialer Fall

- allgemein: $r_{n.m} = e_{n.m} + i_{n.m} + b_{n.m}$

  $i_{n.m}$ Zeit in der $T_{n.m}$ durch höher-priore Subtransaktionen unterbrochen wird

  $b_{n.m}$ Zeit in der $T_{n.m}$ durch nieder-priore Subtransaktionen blockiert wird

- hier: $i_{n.m} = \sum_{T_{i.j} \in H_{n.m}} ceiling(r_{n.m}/p_i) e_{i.j}$

  $H_{n.m}$ Menge der höher-prioren Subtransaktionen

  $ceiling(r_{n.m}/p_i)$ Auftreten des Ereignisses T_i während der Ausführung der Subtransaktion $T_{n.m}$

- ingesamt: $r_{n.m} = e_{n.m} + \sum_{T_{i.j} \in H_{n.m}} ceiling(r_{n.m}/p_i) e_{i.j} + b_{n.m}$

TASKs

- keine nicht-präemptiven TASKs
- kein TASK sperrt die Interrupts
- ein TASK aktiviert maximal einen Nachfolger
- keine RESOURCEn, EVENTs oder `Schedule()`

- Problematik:
  - **ISRs nieder-priorer Ereignisse** unterbrechen
  - **TASKs hoch-priorer Ereignisse**

TASKS

- hier: $i_{n.m} = \underbrace{\sum_{T_{i.j} \in H_{n.m}} ceiling(r_{n.m}/p_i) e_{i.j}}_{(1)} + \underbrace{\sum_{T_{i.j} \in L_{n.m}} ceiling(r_{n.m}/p_i) e^{isr}_{i.j}}_{(2)}$

  (1) Verdrängung durch höher-priore Subtransaktionen

  (2) Verdrängung durch ISRs nieder-priorer Subtransaktionen

  $L_{n.m}$ Menge der nieder-prioren Subtransaktionen

  $e^{isr}_{i.j}$ Ausführungszeit des ISR-Anteils der Subtransaktion $T_{n.m}$

Kritische Abschnitte

- Implementierungsvarianten
  - gesperrte Interrupts
  - nicht-präemptive TASKs
  - RESOURCEn

- hier: $b_{n.m} = b^{non}_{n.m} + b^{isr}_{n.m} + b^{res}_{n.m}$

  $b^{non}_{n.m}$ Blockade durch nicht-präemptive TASKs

  $b^{isr}_{n.m}$ Blockade durch gesperrte Interrupts

  $b^{res}_{n.m}$ Blockade durch belegte RESOURCEn

Kritische Abschnitte

- Begründung
  - TASK A belegt eine RESOURCE
  - TASK A wird von einem nicht-präemptiven TASK B verdrängt
  - ISR C unterbricht TASK B und sperrt die Interrupts
- Bestimmung der einzelnen Komponenten durch
  - Sektionen
  - Ausführungspfade
  - Abschnitte

ALARMe

- ALARM werden von COUNTERn gesteuert
  - USERCOUNTER betrachten wir hier nun nicht
  - hier: TIMERCOUNTER
- TIMERCOUNTER
  - Hardware-Timer
  - löst zyklisch Interrupt aus
  - je nach Zählerstand läuft ALARM ab
- Problem
  - Timer-ISR
    - unterbricht andere Subtransaktionen
    - welche Priorität soll die ISR haben

ALARMe

- insgesamt: $w_{n.m}(t) = e_{n.m} + tc^{isr}_{n.m}(t) + i_{n.m} + b_{n.m}$

  $tc^{isr}_{n.m}(t)$ Unterbrechungen der Subtransaktion $T_{n.m}$ durch Timer-ISRs
- hier: $tc^{isr}_{n.m}(t) = \sum_{tc \in TC} tc^{tc}_{n.m}(t)$

  $tc^{tc}_{n.m}(t) = 0$ Subtransaktion $T_{n.m}$ enthält keine TASKs und alle ISRs haben eine höhere Priorität als der ISR des Hardware-Timers

  $tc^{tc}_{n.m}(t) = ceiling(t / p_{TC}) e^{isr}_{TC}$ sonst

  TC Menge aller Timer-Counter
- Priorität der Timer-ISR
  - so hoch wie nötig, so niedrig wie möglich
  - es darf **kein** Tick verloren gehen
  - iterative Annäherung

Self-Suspension

- in OSEK durch
  - Warten auf ein EVENT: `WaitEvent()`
  - Aufruf des Schedulers: `Schedule()`
- Problematik:
  - Blockade durch nieder-priore Subtransaktionen kann nach jedem Verzicht auf den Prozessor erneut auftreten
- insgesamt: $w_{n.m}(t) = e_{n.m} + tc^{isr}_{n.m}(t) + i_{n.m} + s_{n.m} + (s^k_{n.m} + 1) b_{n.m}$

  $s_{n.m}$ so lange verzichtet Subtransaktion $T_{n.m}$ auf den Prozessor

  $s^k_{n.m}$ so oft verzichtet Subtransaktion $T_{n.m}$ auf den Prozessor

Self-Suspension

- Alternative: aktives Warten

```
Task(UglyTask) {
  ...
  SetRelAlarm(alarm1,time_to_wait,0);
  while(GetAlarm(alarm1) != E_OS_NO_FUNC);
  ...
  TerminateTask();
}
```

- Problematisch
  - erhöht Prozessorauslastung unnötigerweise
  - nieder-priore Subtransaktionen werden lange verzögert

→ Solche Probleme auf Entwurfsebene vermeiden

beliebige Deadlines

- bisher: Deadline ≤ Periode
- jetzt: Deadline > Periode
  - → Ereignis kann erneut eintreten bevor es fertig behandelt wurde
  - → mehrere Inkarnationen einer Subtransaktionen können existieren
  - → TASKs **müssen mehrfach aktivierbar** sein
  - → ISRs **können nicht mehrfach** aktiviert werden
- Analyse am kritischen Zeitpunkt reicht nicht mehr aus
  - → Analyse muss sich über ein **Auslastungsintervall** erstrecken

Ein **Priorität-A-Auslastungsintervall** [a,b] ist ein Zeitintervall in dem nur Subtransaktionen mit einer Priorität ≥ A ausgeführt werden. In den Zeiträumen]a – ε,a[und]b,b + ε[werden nur Subtransaktionen mit einer Priorität < A ausgeführt.

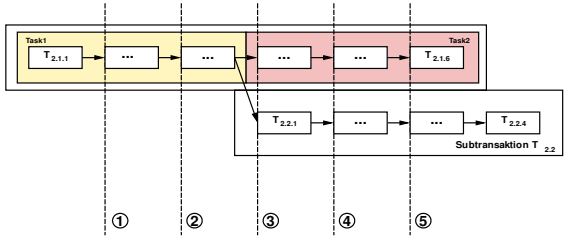
Erweiterte Ablaufplanung

- Laufzeitprioritäten
- Verzweigungen
- abhängige Ereignisse

Laufzeitprioritäten

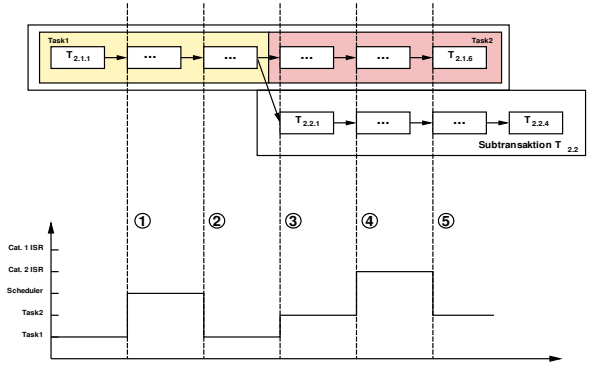
- bisherige Verwendung von Sektionen
  - Berechnung der WCET
  - Berechnung der Belegungszeit von RESOURCEn
  - Sperrung von Unterbrechungen
- jetzt: Priorität kann zwischen Sektionen variieren

Laufzeitprioritäten




```
1 GetResource(RES_SCHEDULER)
2 ReleaseResource(RES_SCHEDULER)
3 ChainTask(Task2)
4 SuspendOSInterrupts()
5 ResumeOSInterrupts()
```


Laufzeitprioritäten




```
1 GetResource(RES_SCHEDULER)
2 ReleaseResource(RES_SCHEDULER)
3 ChainTask(Task2)
4 SuspendOSInterrupts()
5 ResumeOSInterrupts()
```


Laufzeitprioritäten: Antwortzeitanalyse

- Begriffe
  - **Unterbrechungsblock** einer Sektion

    Ein **Unterbrechungsblock** einer Sektion ist ein längstmöglicher **Abschnitt eines Ausführungspfades** dessen Sektionen alle eine **größere Priorität** haben als diese Sektion.

  - **führender Unterbrechungsblock**

    Ein **führender Unterbrechungsblock** ist ein Unterbrechungs dessen erste Sektion zugleich die **erste Sektion eines Ausführungspfades** ist.

  - **Verzögerungsblock** einer Sektion

    Ein **Verzögerungungsblock** einer Sektion ist ein längstmöglicher **Abschnitt eines Ausführungspfades** dessen Sektionen alle eine **größere oder die gleich Priorität** haben als/wie diese Sektion.

Laufzeitprioritäten: Antwortzeitanalyse

- Klassen von Subtransaktionen:
  - $L_{n.m.l}$

    Alle Subtransaktionen enthalten keinen Unterbrechungsblock der Sektion $T_{n.m.l}$.

  - $U_{n.m.l}$

    Alle Subtransaktionen enthalten einen führenden Unterbrechungsblock der Sektion $T_{n.m.l}$.

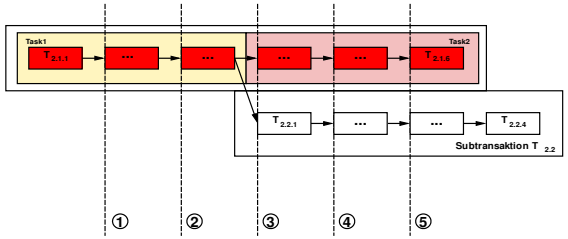
  - $V/L_{n.m.l}$

    Alle Subtransaktionen enthalten einen mindestens einen Verzögerungsblock der Sektion $T_{n.m.l}$.

- Zweck
  - führender Unterbrechungsblock: **mehrmalige Unterbrechung**
  - Unterbrechungs-/Verzögerungsblock: **einmalige Unterbrechung**

Laufzeitprioritäten: Antwortzeitanalyse

- Analyse erfolgt für einen bestimmten Ausführungspfad



- alle Ausführungspfade müssen analysiert werden
- Analyse erfolgt sukzessive
  - Berechnung der Antwortzeit der ersten Sektion
  - Berechnung der Antwortzeit der zweiten Sektion ausgehend von diesem Ergebnis

Laufzeitprioritäten: Antwortzeitanalyse

- Antwortzeit der Sektion $T_{n.m.1}$

$$w^k_{n.m.1}(t^{(l)}) = b_{n.m} + e_{n.m.1} + (k-1)\,e_{n.m} + \sum_{T_{i,j} \in U_{n.m.1}} ceiling\left(\frac{t^{(l)}}{p_i}\right) e^u_{i.j\,,\,n.m.1} + \sum_{T_{i,j} \in V / L_{n.m.1}} e^u_{i.j\,,\,n.m.1}$$

Laufzeitprioritäten: Antwortzeitanalyse

- Antwortzeit der Sektion $T_{n.m.1}$

Blockade

wiederholte Verzögerung durch führende Unterbrechungsblöcke

$$w^k_{n.m.1}(t^{(l)}) = b_{n.m} + e_{n.m.1} + (k-1)\,e_{n.m} + \sum_{T_{i,j} \in U_{n.m.1}} ceiling\left(\frac{t^{(l)}}{p_i}\right) e^u_{i.j\,,\,n.m.1} + \sum_{T_{i,j} \in V / L_{n.m.1}} e^u_{i.j\,,\,n.m.1}$$

Zeitbedarf von $T_{n.m.1}$ und der (k -1) vorherigen Auftreten von T_n

WCET der Unterbrechungs – bzw. Verzögerungsblöcke

einmalige Verzögerungen durch Verzögerungsblöcke

Laufzeitprioritäten: Antwortzeitanalyse

- Antwortzeit der Sektion $T_{n.m.r}$, r > 1

$$w^k_{n.m.r}(t^{(l)}) = \underbrace{f^k_{n.m.(r-1)} + e_{n.m.r}}_{(1)} + (2) + (3) + (4) + (5)$$

(1) Antwortzeit der vorhergehenden Sektion und WCET der gegenwärtigen Subtransaktion

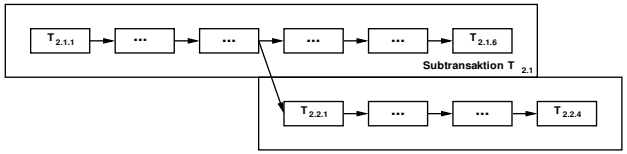
(2) Unterbrechung durch führende Unterbrechungsblöcke – unter bestimmten Umständen (Prio($T_{n.m.r}$) < Prio($T_{n.m.(r-1)}$)) können sich Aktivierungen für Unterbrechungsblöcke aufstauen – diese aufgestauten Aktivierungen werden hier nicht berücksichtigt

(3) Aufgestaute, frühere Aktivierungen führender Unterbrechungsblöcke

(4) Verzögerungsblöcke, die seit Beginn der Subtransaktion aktiv sind und nicht durch einen vorhergehenden Unterbrechungsblock aktiviert wurden.

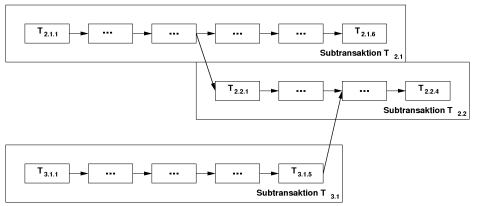
(5) Verzögerungsblöcke, die durch einen vorhergehenden Unterbrechungsblock aktiviert wurden.

Verzweigungen



- Prioritäten
  - Priorität des Vorgängers richtet sich nach den Nachfolgern
  - Prio(Vorgänger) = max Prio(Nachfolger)
- Antwortzeitanalyse
  - Vorgänger: erweiterte Antwortzeitanalyse
  - Nachfolger: erweiterte Antwortzeitanalyse für alle Nachfolger ausgehend vom Vorgänger

ODER-Abhängigkeiten

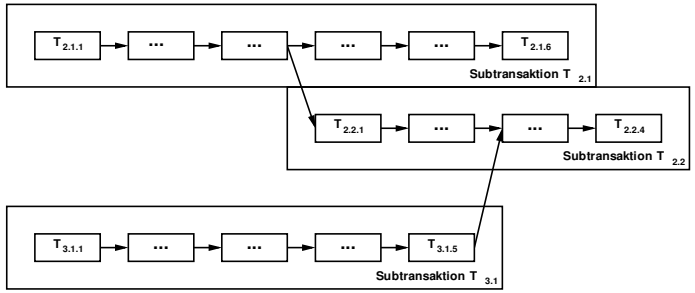


- Antwortzeitanalyse
  - für alle Vorgänger: erweiterte Antwortzeitanalyse
  - Nachfolger wird für jeden Vorgänger analysiert
- Achtung
  - Aktivierungen des Nachfolgers
  - ein Nachfolger muss mehrere Vorgänger behandeln

ODER-Abhängigkeiten

- Prioritäten
  - mehrere Subtransaktionen *teilen* sich *denselben Nachfolger*
  - diese Subtransaktionen können *verschiedene Deadlines* haben
  - → der Nachfolger müsste *verschiedene Prioritäten* besitzen
- Lösung
  - Priorität des Nachfolgers richtet sich nach den Vorgängern
  - → Prio(Nachfolger) = max Prio(Vorgänger)

UND-Abhängikeiten

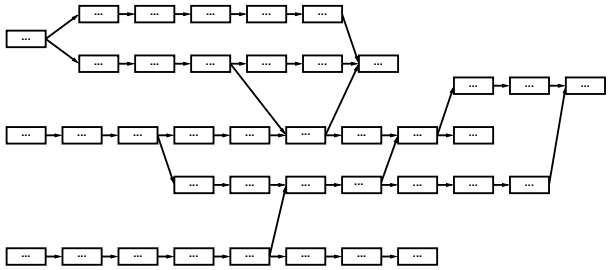


- Antwortzeitanalyse
  - für alle Vorgänger: erweiterte Antwortzeitanalyse
  - Nachfolger wird für den spätesten Vorgänger analysiert

UND-Abhängigkeiten

- Prioritäten
  - alle Vorgänger haben einen *gemeinsamen Nachfolger*
  - alle Vorgänger haben einen *gemeinsame Deadline*
  - *innerhalb* der Vorgänger kann es auch *separate Deadlines* geben
- Lösung
  - Prioritäten der Vorgänger richten sich nach
    - der Priorität des Nachfolgers: Prio(Vorgänger) = Prio(Nachfolger)
    - separaten Deadlines

Probleme - Prioritäten



- Problematik: viele Subtransaktionen, viele Abhängigkeiten
- → Entstehung hoher Prioritäten
- → evtl. Konflikte

- → nach Möglichkeit im Design vermeiden

Zusammenfassung

- ProOSEK Abbildung
  - Verzweigungen
  - ODER-Abhängigkeiten
  - UND-Abhängigkeiten
  - aperiodische Ereignisse
- ProOSEK: einfache Ablaufplanung
  - Algorithmus
  - trivialer Fall
  - TASKs
  - kritische Abschnitte
  - ALARMe
  - Self-Suspension
  - beliebige Deadlines
- ProOSEK: erweiterte Ablaufplanung
  - Prinzip
  - Verzweigungen
  - ODER/UND-Abhängigkeiten

Ergebnis

- voll ausformulierte Steuerung
  - OSEK Konfiguration
  - Implementierung der Anwendung
  - Abbildung der Komponenten auf Ablauftabellen
- Antwortzeitanalyse