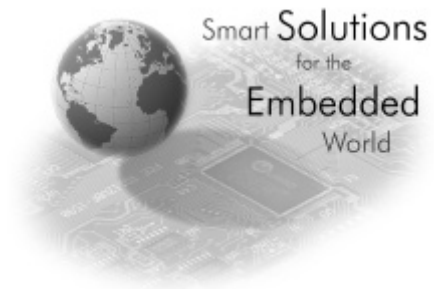


3SOFT

OSEK Deadline-Analyse



3SOFT GmbH Erlangen
Jürgen Scherg

8. Juni 2001

Ein Programmtest muß unter verschiedenen Gesichtspunkten durchgeführt werden.

⇒ verschiedene Testmethoden sind notwendig.

- Blackbox : Es wird das IO-Verhalten von Modulen und Teilsystemen getestet.
- Whitebox : Es wird getestet ob das Programm intern richtig abläuft und die richtigen Programmzustände einnimmt.
- Echtzeitverhalten : Es wird getestet ob das Programm die Zeitkriterien erfüllt.

Um eine Applikation auf ihr Echtzeitverhalten hin zu testen ist eine statische Analyse der Applikation im Zusammenhang mit dem eingesetzten Betriebssystem notwendig.

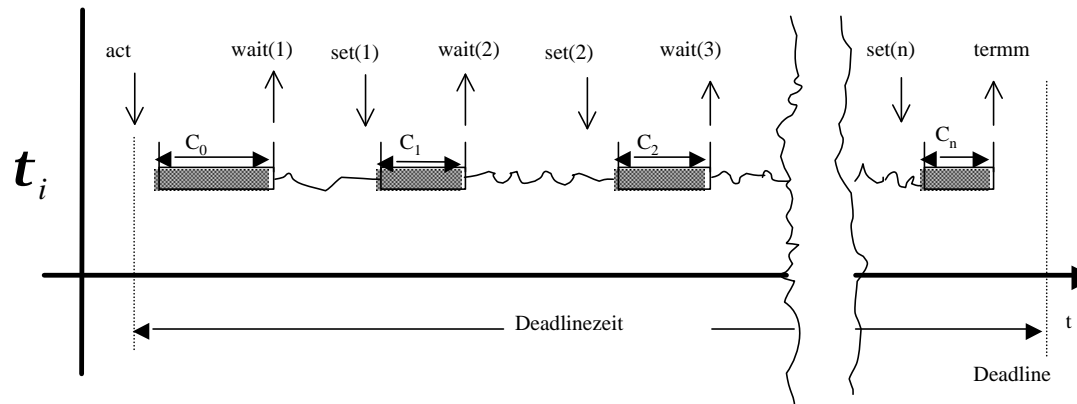
Bestehende Techniken

- RMA : (Rate Monotonic Analysis). Entstanden an der Carnegie Mellon University in Pittsburg.
- DMA : (Deadline Monotonic Analysis). Entwickelt von Ken Tindell, Mitarbeiter der Firma Northern Real-Time Applications.

RMA und DMA sind i.a. nicht auf OSEK übertragbar, da sie starke Einschränkungen an die Applikation und Betriebssystem haben

⇒ Entwicklung der OSEK Deadline-Analyse.

Taskablauf und Definition Deadline



τ_i : Task Nr i - C_i : Laufzeit von τ_i - act: Aktivierung von τ_i

term: Terminierung der Task

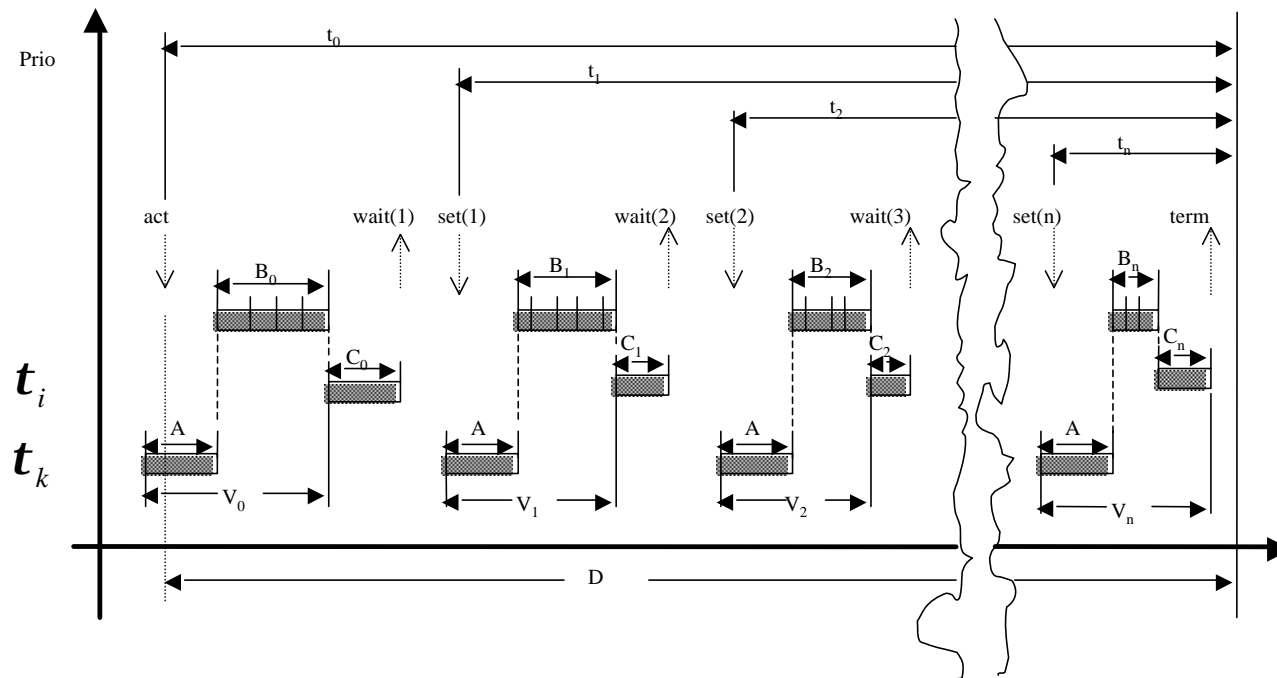
wait(k): Zeitpunkt zu dem die Task auf Events wartet.

set(k): Zeitpunkt an dem Events gesetzt werden.

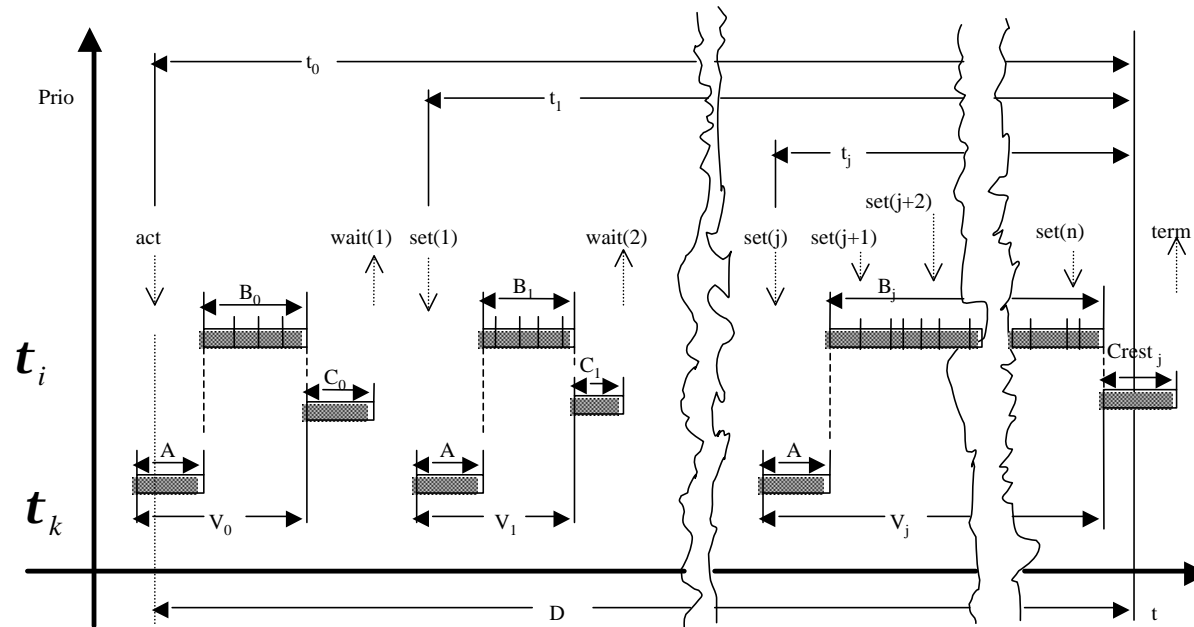
Die Ausführung von τ_i kann beeinflusst werden durch:

- 1: Eine niedriger priorisierte Task τ_k , die vor act gestartet wird.
- 2: Höher priorisierte Tasks, die während D aktiviert und vollständig abgearbeitet werden.

Bestcase: Terminierung vor der Deadline

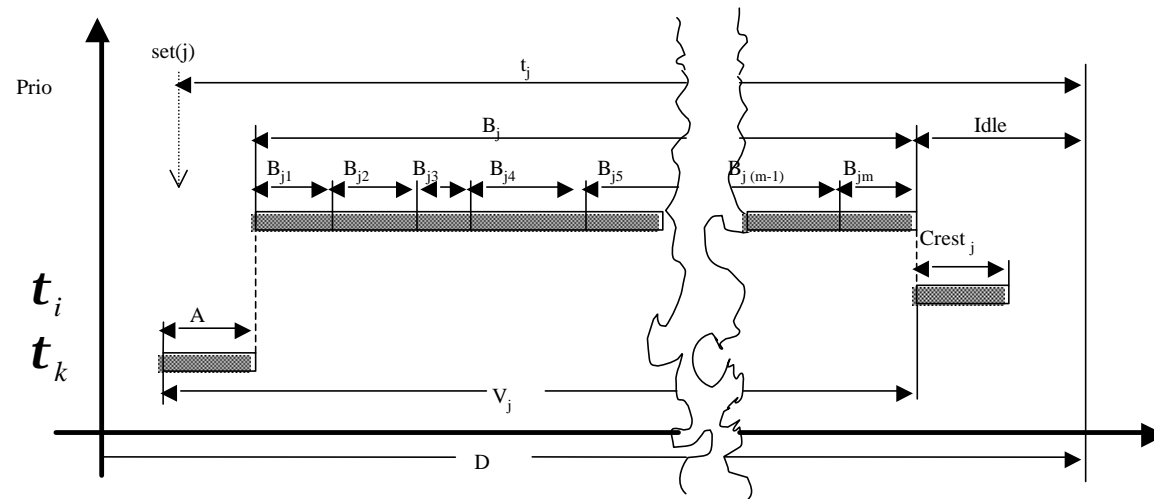


Worstcase: Terminierung nach der Deadline



Crest_j : Restlaufzeit von nach set(j).
$$Crest_j = \sum_{l=j}^n C_l$$

Berechnung der Verzögerungszeit



Bezeichnungen:

B_{jl} : Laufzeit einer einzelnen höherpriorisierten Task τ_l .

B_j : $B_j = \sum_{l=1}^m B_{jl}$ = Summe der Laufzeiten der höher priorisierten Tasks.

Idle : Verbleibende Laufzeit für τ_j bis zur Deadline.

OSEK Scheduling - Eigenschaft

Zum Zeitpunkt $set(j)$ gilt:
 $A + B_j + Crest_j < t_j$
dann kann τ_i noch innerhalb der Deadline
abgearbeitet werden.

Abschätzung von A

Es müssen folgende Fälle unterschieden werden:

$a(\tau_k)$:= Verzögerung durch Task τ_k

- $\text{prio}(\tau_k) < \text{prio}(\tau_i) \wedge \tau_k$ ist nicht preemptive
 $\Rightarrow a(\tau_k) = C(\tau_k)$
- $\text{prio}(\tau_k) < \text{prio}(\tau_i) \wedge \tau_k$ ist preemptive $\wedge \tau_k$ belegt Ressourcen
 $\Rightarrow a(\tau_k) = C(\tau_k)$.
- $\text{prio}(\tau_k) < \text{prio}(\tau_i) \wedge \tau_k$ ist preemptive $\wedge \tau_k$ belegt keine Ressourcen
 $\Rightarrow a(\tau_k) = 0$.

$$A = \max \{a(\tau_k) \mid \text{prio}(\tau_k) < \text{prio}(\tau_i)\}$$

Abschätzung von B_i

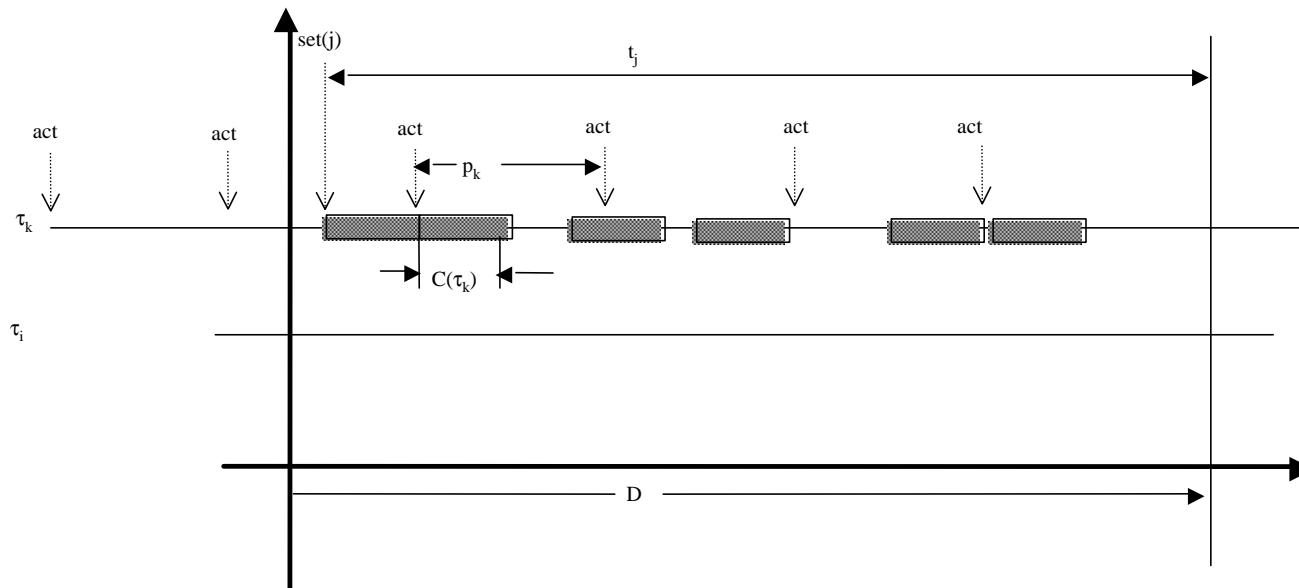
Unterscheidung in periodische und aperiodische Tasks

B_j' := Summe der Laufzeiten der periodischen Tasks

B_j'' := Summe der Laufzeiten der aperiodischen Tasks

$$\Rightarrow B_j = B_j' + B_j''$$

τ_k ist periodisch \hat{U} Basic-Task



Bezeichnungen:

p_k := Periode von τ_k (Periode der Aktivierungen)
 m_k := Höchstzahl der zulässigen Aktivierungen
(Attribut 'MultipleActivation')

τ_k wird während t_j maximal $\lceil t_j/p_k \rceil$ mal aktiviert
 $\Rightarrow \tau_k$ wird während t_j maximal $\lceil t_j/p_k \rceil + m_k$ mal abgearbeitet
 $\Rightarrow \tau_k$ benötigt während t_j maximal $(\lceil t_j/p_k \rceil + m_k) * C(\tau_k)$ Zeit.

Für alle τ_k mit $\text{prio}(\tau_k) \geq \text{prio}(\tau_i)$ und τ_k ist periodisch bestimme:

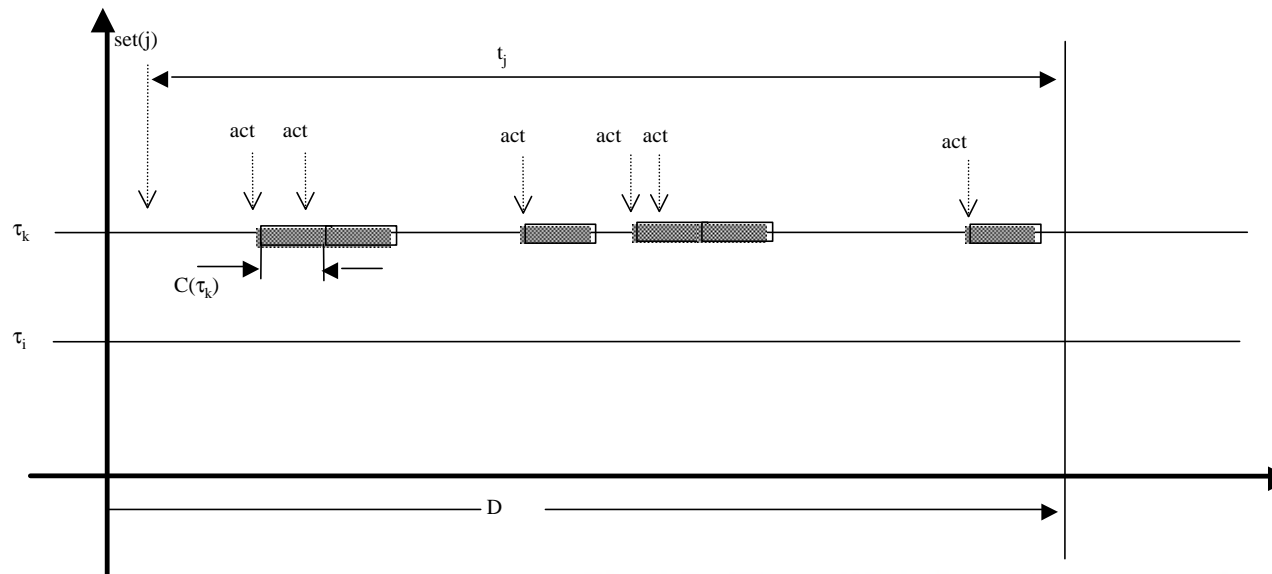
$$B_{jk} = \left(\left\lceil \frac{t_j}{p_k} \right\rceil + m_k \right) * C(t_k)$$

und berechne

$$B_j' = \sum_k B_{jk}$$

τ_k ist aperiodisch

Für eine aperiodische Task kann man nicht deterministisch bestimmen wie oft sie während t_j aktiviert und abgearbeitet wird. Theoretisch kann sie 'unendlich oft' aktiviert werden.



Einfache Lösung:

Schätze ab wie oft t_k während t_j höchstens aktiviert werden kann.
Dieser Faktor sei mit α_{jk} bezeichnet.
 \Rightarrow Verzögerung durch t_k beträgt: $\alpha_{jk} * C(t_k)$ Zeiteinheiten.

Für alle τ_k mit $\text{prio}(\tau_k) \geq \text{prio}(\tau_i)$ und τ_k ist aperiodisch bestimme:

$$B_{jk} = a_{jk} * C(t_k)$$

und berechne

$$B_j'' = \sum_k B_{jk}$$

Deadline-Analyse erfordert Wissen über Tasklaufzeiten:

- experimentelle Ermittlung
 - z.B. in OSEK über Pre/PostTaskHook
 - nur sinnvoll bei keinem oder geringem Jitter
- Abschätzung der maximalen Laufzeit
 - z.B. durch Bestimmung des maximalen Pfades
 - zweckmäßig bei komplexen Tasks
 - Vorsicht: Abschätzung gefährdet Aussagekraft!

Je ausgelasteter das System, desto genauere Werte erforderlich.