7 Spezifikation der zeitlichen Ausführungsbedingungen

In Kapitel 5 und 6 wurde die Implementierung der Embedded-Interaktion behandelt. Zur Implementierung der Embedded-Interaktion gehört auch das Festlegen der zeitlichen Ausführung der Embedded Komponenten. Der schwierigste Teil ist die dynamisch zeitliche Planung (Scheduling) der Verarbeitung anstehender Ereignismeldungen, wenn harte *Deadline-Intervalle* einzuhalten sind. Diese Scheduling-Parameter wurden als bekannt vorausgesetzt. In diesem Kapitel geht es darum, die Deadline-Intervalle aus den zeitlichen Anforderungen an das Gesamtsystem herzuleiten. Dabei sind auch die Laufzeiten der externen Interaktion berücksichtigen, wie z. B. die Latenzzeiten von Peripheriegeräten oder von der Feldbuskommunikation. Weiter war man beim Unit-Scheduling von einer Aufteilung des Systems in *Embedded Units* mit festgelegten Zyklusperioden ausgegegangen. Auch diese Zyklusperioden müssen bestimmt werden, nämlich unter Berücksichtigung der spezifizierten Deadline-Intervalle und den Ausführungszeiten der reaktiven Maschine.

Zeitliche Anforderungen. Zeitliche Anforderungen werden immer in Bezug auf die externen Prozesse gestellt. Wir werden uns auf zwei Typen von harten Echtzeitanforderungen konzentrieren: Erstens die maximale Zeitspanne, während der vom System für ein bestimmtes Ereignis die entsprechende Ereignismeldung verarbeitet werden muss (maximale Reaktionszeit), und zweitens die Zeitspanne, während der eine durch ein Ereignis bewirkte Aktion ausgeführt werden muss (maximale Antwortzeit).

Qualitatives und quantitatives Zeitverhalten. Mit den Reaktiven Maschinen werden die funktionalen Anforderungen an das System erfüllt. Diese Komponenten sorgen dafür, dass das Embedded System ‘richtig’ auf die externen Ereignisse reagiert, indem geschichtsabhängig (Systemzustand) für jedes gültig auftretende Ereignis die korrekten Aktionen initiiert werden. ‘Richtig’ bedeutet hier, dass jede Ereignissequenz über den Rechner die verlangten Aktionssequenz bewirkt, d. h. es handelt hier um rein *qualitatives* Zeitverhalten. Auf das *quantitative* Zeitverhaltens hat die reaktive Maschine keinen direkten Einfluss (sequentielle Maschinen). Einzig die durch die Implementation bestimmten *Ausführungszeiten* der Ereignisverarbeitungen wirken sich auf die Grösse der Antwortzeiten eines Systems aus.

Interaktionszeiten. Ob die zeitlichen Anforderungen eingehalten werden können, ist neben den Ausführungszeiten der reaktiven Maschine vollständig durch die Implementierung der Verbindung der externen Prozesse mit den Steuerkomponenten bestimmt. Eine solche Verbindung umfasst sowohl die externen Interaktionskomponenten (Peripheriegeräte) als auch die Komponenten der Interaktionssoftware auf dem Rechner. Den stärksten Einfluss auf die zeitliche Ausführung der Ereignisverarbeitung haben die Ausführungssteuerungen und das globale Scheduling der Embedded Units. Die externe Interaktion wirkt sich durch ihre “Laufzeiten” von den externen Prozessen zum Steuerrechner, bzw. vom Steuerrechner zu den Prozessen aus. Diese durch die Installation der Peripheriegeräte (lokal oder verteilt über Feldbusse) bestimmten Zeiten sind gegeben; sie müssen aber bei der Spezifikation der Schedulingparameter berücksichtigt werden.

Bestimmung der Zyklusperioden. Eine zentrale Aufgabe bei der Spezifikation der Interaktionssoftware einer Embedded Unit ist das Festlegen der Zyklusperiode. Hier müssen die zeitlichen Anforderungen, die zeitlichen Eigenschaften der Umgebung (Prozesse und Peripheral Devices) und die Ausführungszeiten der Embedded Komponenten berücksichtigt werden. Beim Festlegen der Zyklusperiode sind bestimmte Bedingungen einzuhalten, damit das zyklusbasierte non-preemptive Deadline-Scheduling funktionieren kann. Hier muss unter Umständen damit gerechnet werden, dass das System anders in *Embedded Units* aufgeteilt werden muss. Möglicherweise existiert keine Zyklusperiode, für die alle erwähnten Bedingungen erfüllt sind. Das ist zum Beispiel der Fall, wenn das Deadline-Intervall einer Steuerfunktion kleiner ist als die Ausführungszeit einer anderen Steuerfunktion.

Schedulability Verification. Die Erfüllung der erwähnten Bedingungen für die Zyklusperioden bedeutet jedoch nicht, dass die harten Zeitbedingen immer eingehalten werden können. Diese Garantie müsste offline erfolgen, durch eine Verifikation der zeitlichen Planbarkeit, der sog. *Schedulability*. Solche Überprüfungen basieren auf pessimistisch Worst-Case-Annahmen. Damit ist es möglich, dass in einem laufenden System die Zeitbedingungen immer eingehalten werden, obwohl bewiesen wurde, dass Deadline-Überschreitungen möglich sind. Viel problematischer ist jedoch, ob die Verifikation überhaupt vollständig durchführt werden kann. Pragmatische Ansätze erachten es deshalb viel wichtiger, dass potenzieller Overload zur Laufzeit mit geringem Aufwand möglichst frühzeitig festgestellt werden kann. Damit dies möglich ist, muss das Scheduling einfach und transparent sein.

7.1 Systemreaktionen (System Reactions)

Zeitliche Anforderungen an das Gesamtsystem beziehen sich immer auf Phänomene der externen Prozesse. Wir gehen hier davon aus, dass diese Anforderungen in Bezug auf die spezifizierten Ereignisse und Aktionen formuliert sind. Eine verlangte Antwortzeit zum Beispiel ist eine Beschränkung der Zeit zwischen dem Auftreten eines bestimmten Ereignisses und der Ausführung einer zu bewirkenden Aktion. Diese Zeitspanne beinhaltet sowohl die externen Interaktionszeiten wie auch die Interaktionszeiten auf dem Rechner sowie die Ausführungszeit der eigentlichen Ereignisverarbeitung. Diese Interaktionszeiten müssen berücksichtigt werden, wenn aus den zeitlichen Systemanforderungen die Deadline-Intervalle für die Ereignisverarbeitung bestimmt werden.

Um den Bezug zwischen den zeitlichen Systemanforderungen und den zeitlichen Ausführungsbedingungen auf dem Rechner diskutieren zu können wird der Begriff *Systemreaktion* eingeführt:

> ***Systemreaktion***
> Das gesamte zeitliche Szenario, das durch ein auftretendes Ereignis bewirkt wird, bezeichnen wir als *Systemreaktion*.

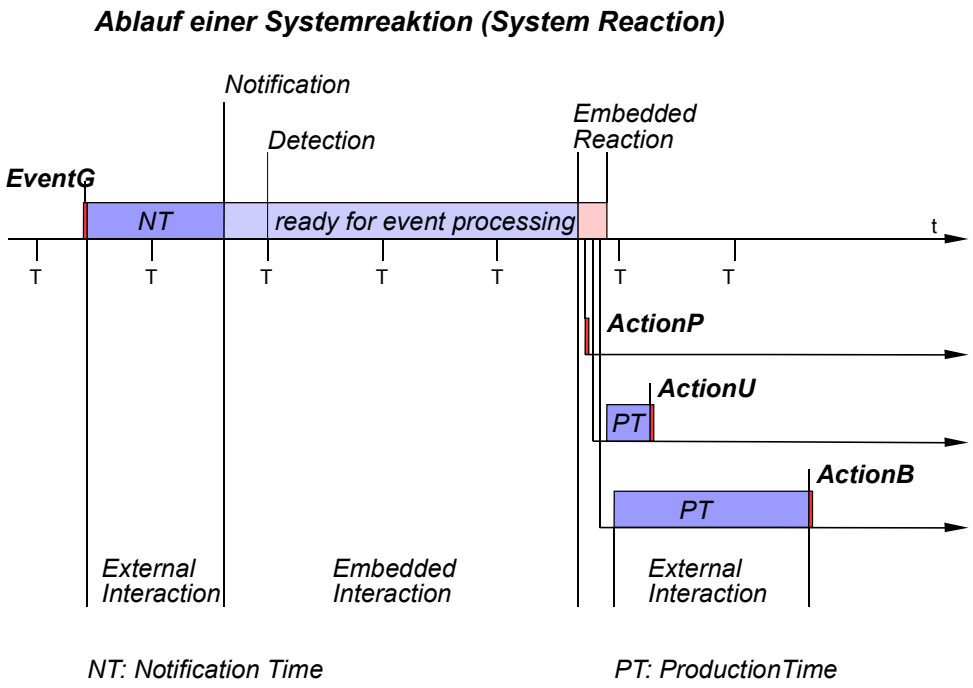

Abb. 78: Ablauf einer Systemreaktion

Abbildung 78 zeigt den zeitlichen Ablauf einer *Systemreaktion*. Das Ereignis *EventG* tritt zu einem bestimmten Zeitpunkt in der externen Prozessumgebung auf. In der anschliessenden Zeitspanne *Notification Time* (*NT*) wird ein oder mehrere Sensoren aktiviert und die erzeugte elektronische Information an die Rechnerschnittstelle übertragen. Anschliessend folgt die Erkennung (Detection)

des aufgetretenen Ereignisses auf dem Rechner und die Erzeugung der entsprechenden Ereignismeldung.

Der Zeitpunkt für die Ausführung der Ereignismeldung ist durch das non-preemptive Scheduling des *UnitController* bestimmt. Der Ablauf der aktivierten Steuerfunktion wird als *Embedded Reaction* bezeichnet. Hier wird die eigentliche Reaktionsentscheidung gefällt. Dabei wird der Zustand der reaktiven Maschine verändert und im allgemeinen mehrere Aktionen initiiert.

Die mit den Aktionsinitiierungen verursachten Ausgaben an den Rechnerschnittstellen können zu unterschiedlichen Zeitpunkten erfolgen. Jede initiierte Aktion hat ihre eigene *Production Time* (*PT*). Diese Zeitspanne umfasst das Übertragen der elektronischen Ausgabeinformation an die entsprechenden Aktoren und das Erzeugen der Aktion beim entsprechenden externen Prozess.

Eine Systemreaktion ist ein rechnerbasierter Interaktionsschritt des Gesamtsystems. Die Systemreaktionen einer *Embedded Unit* sind nebenläufig, nur die Informationsverarbeitung auf dem Rechner wird sequentiell ausgeführt. Die auszuführende Reaktionsentscheidung ist dabei immer eindeutig durch den aktuellen Zustand der Steuerkomponente bestimmt (deterministische geschichtsabhängige Steuerung).

7.2 Spezifikation der Interaktionszeiten (Interaction Time Specification - ITS)

In diesem Abschnitt wird ein Framework von Begriffen und Grössen definiert, die notwendig sind, um die Schedulingparameter für Embedded Units herzuleiten. Das *ITS*-Framework (*Interaction Time Specification*) ist eine minimale Spezifikationssprache, mit der zeitliche Anforderungen und zeitliche Systemeigenschaften festgehalten und daraus Deadline-Intervalle und Zyklusperioden bestimmt werden. Für *Embedded Units, Event Channels, Events, Action Channels* und *Actions* werden *Spezifikationsobjekte* definiert. Die Komponenten eines Objektes werden als Attribute bezeichnet. Gewisse Attribute spezifizieren Listen von Objekten, (z. B. eine Event List), für die wieder entsprechende Objekte definiert sind.

Die Erstellung der Objekte kann zum Beispiel anhand von Excel Worksheets erfolgen. Diese können so konfiguriert werden, dass das Excel-Tool die herzuleitenden Schedulingparameter automatisch berechnet. Ein entsprechendes Spezifikationswerkzeug, realisiert als Eclipse-Plugin (PSD Tool), wird zur Zeit entwickelt.

Für die Referenzierung eines Attributs wird in der folgenden Beschreibung die übliche Qualifizierung mit einem Punkt verwendet.

> *MotClose.mxRT* bezeichnet zum Beispiel die *Maximal Response Time* der Aktion *MotClose*.
>
> *MotClose* ist der Name einer Aktion und *mxRT* der Name des Attributes, das die maximal tolerierte Antwortzeit für diese Aktion spezifiziert. *Maximal Response Time* ist der sog. *Long Name* des Attributs.

Alle Attribute ausser den *Deadline-Intervallen,* der *Cycle Periods* und den *Processor Utilization Factors* werden zum Teil bereits in anderen Entwicklungsphasen beschrieben. In den *ITS-Spezifikationsobjekte* werden diese Attribute systematisch zusammengefasst und vervollständigt, damit ein klarer Zusammenhang zu den gesuchten Deadline-Intervallen hergestellt werden kann. Das wird vor allem dann wichtig, wenn es aufgrund knapper Ressourcen schwierig ist, eine zulässige Zyklusperiode zu finden.

Die meisten der gegebenen Attribute stammen aus informalen System-Beschreibungen (ad hoc type), und es besteht oft ein beträchtlicher Spielraum bei der Quantifizierung der entsprechenden Grössen. Das kann gezielt ausgenützt werden um ein machbares Scheduling zu finden, indem die Werte dieser Attribute innerhalb zulässiger Grenzen variiert werden.

Die ITS-Spezifikationsobjekte

Die Objekte werden im folgenden in Tabellenform beschrieben.

Bedeutung der 4 Kolonnen: Attributbezeichnung, lange Bezeichnung, Typ, Bedeutung.
(Typ *crd* (cardinal) bezeichnet die nicht-negativen ganzen Zahlen, *symbols* sind Zeichenketten)

Das erste Attribut eines Objektes ist der Name des spezifizierten Objektes, bei einem *Event Objekt* zum Beispiel der Name eines Ereignisses. Der Name dient als Identifier. Zusätzlich wird ein Aufzählungswert definiert, der in Tabellen als Index verwendet wird.

Die Tabellen-Felder für die herzuleitenden Attribute sind in der folgenden Darstellung grau schattiert.

In den jeweils anschliessenden Beschreibungen der Attribute eines Objekttyps wird bei jedem Attribut am Anfang unter *Quelle:* angegeben, aus welcher Systembeschreibung die entsprechenden Werte stammen. Attribute die als Quelle *ITS* haben müssen hergeleitet werden.

Unit Objekt

name	object name	symbol	Identifier der Unit
stu	Specification Time Unit	crd (us)	Zeiteinheit für Ausführungszeiten
EvtChnLst	Event Channel List	pointer	Liste der Ereigniskanäle, zeigt auf *Event Channel* Objekte
ActChnLst	Action Channel List	pointer	Liste der Aktionenkanäle, zeigt auf *Action Channel* Objekte
MaxET	Maximal Event wcET	crd (stu)	Maximum aller WCET's für Ereignisverarbeitungen
MinPDI	Minimal Processing Deadline Interval	crd (stu)	Minimum aller Processing Deadline-Intervalle,
T	Cycle Period	crd (stu)	Zyklusperiode
U	Processor Utilization Factor	crd (%)	Zu garantierender Anteil an der Prozessorzeit, zu garantieren pro Zyklusperiode

Beschreibungen der Attribute eines Unit Objekte

EvtChnLst - Event Channel List (Ereigniskanal-Liste)

Quelle: Spezifikation der virtuellen Verbindung (*Events&Actions*)

Liste der Ereigniskanäle. Die Ereignisse verschiedener Ereigniskanäle können gleichzeitig auftreten. Alle Ereignisse dieser Kanäle werden nur von dieser Unit verarbeitet.

ActChnLst - Action Channel List (Aktionenkanal-Liste)

Quelle: Spezifikation der virtuellen Verbindung (*Events&Actions*)

Liste der Aktionenkanäle. Die Aktionen verschiedener Aktionenkanäle können gleichzeitig ausgeführt werden. Die Aktionen dieser Kanäle werden nur durch diese Unit bewirkt.

MaxET - Maximal Event wcET)

Quelle: ITS - Maximum der WCET's aller Events.
Das Maximum wird benützt für die Bestimmung möglicher Zyklusperiode

MinPDI - Minimal Processing Deadline Interval

Quelle: ITS - Minimum der Processing Deadline-Intervalle aller Events.

Das Minimum wird benützt für die Bestimmung möglicher Zyklusperiode

T - Unit Cycle (Bedingungen für T siehe Kapitel 7.4)
Quelle: ITS
Periode des Unit Zyklus.

U - Processor Utilization Factor
Quelle: ITS
Zu garantierender prozentualer Anteil an der Prozessorzeit.

.

Event Channel Objekt

name	object name	symbol	Identifier des Ereigniskanals
RTC	Real-Time Conditions	{hard, soft}	Harte oder weiche Echtzeitbedingungen
cat	Event Category	evtCatType	evtCatType = {periodic, sporadic, bursty, locMsg, glob-Msg, tmUp}
EvtLst	Event List	pointer	Ereignisse des Ereigniskanals, zeigt auf *Event* Objekte
EST	Event Separation Times	Table	EvtLst x EvtLst -> crd, Separationszeiten der Ereignisse des Ereigniskanals

Beschreibungen der Attribute eines Event Channel Objektes

RTC - Real-Time Conditions
Quelle: Zeitliche Anforderungen
hard. Die hergeleiteten Deadline-Intervalle dieser Ereignisse müssen unbedingt eingehalten werden. Das Überschreiten einer Deadline ist ein Fehler und muss vom System behandelt werden.
soft. Die Zeitbedingung für das Ereignis sollen im Mittel eingehalten werden (Performance).

evtCat - Event Category
Quelle: Externe Prozessbeschreibung
Wertebereich: evtCatType = {periodic, sporadic, bursty, locMsg, globMsg, tmUp}
periodic. Ein periodisches Ereignis ist ein Zeitereignis, das repetitiv in gleichen zeitlichen Abständen auftritt. Der zeitliche Abstand wird als Periode P bezeichnet. P wird als Attribut des Ereignisses spezifiziert.
sporadic. Ein sporadisches Ereignis ist ein Ereignis, das unregelmässig in nicht zum voraus bestimmtem zeitlichen Abständen auftritt. Der zeitliche Abstand bis um nächsten Auftreten des Ereignisses ist aber gegen unten durch ein positive Zeitschranke begrenzt. Die Zeitschranke ist in erster Linie durch die physikalischen Eigenschaften des erzeugenden Prozesses bestimmt, kann aber auch von den technischen Eigenschaften der Sensorik abhängen.
bursty. Ein Burst-Ereignis ist ein Ereignis, bei dem verschiedene Instanzen beliebig schnell hintereinander auftreten können. Burst-Ereignisse können innerhalb des Systems entstehen, z. B. durch Puffereffekte bei Messreihen oder Kommunikationssystemen (Bursts). Derartige Ereignisse können keine harten Deadlines haben (RTC = soft). Sie müssen u. U. gesondert behandelt werden.
locMsg. Ein Ereignis der Kategorie *locMsg* wird immer durch eine entsprechende Aktion (Kategorie *locMsg*) einer Unit erzeugt, die auf demselben Prozessor ausgeführt wird. Solche Ereignisse werden als lokale Steuernachrichten (*local Control Messages*) bezeichnet. Mittels lokalen Steuernachrichten kann zwischen den Units auf demselben Prozessor asynchron kommuniziert werden.
globMsg. Ein Ereignis der Kategorie *globMsg* wird immer durch ein Aktion (Kategorie *globMsg*) einer Unit eines anderen Prozessors erzeugt. Solche Ereignisse werden als globale Steuernachrichten

(*global Control Messages*) bezeichnet. Mittels globaler Steuernachrichten kann zwischen den Units verschiedener Prozessor asynchron kommuniziert werden.

tmUp. Ein *tmUp*-Ereignis (TimeIsUp) ist ein Zeitereignis. Es tritt auf, wenn ein bestimmter Timer abläuft. Der Timer wurde durch eine Aktion der Kategorie *setTm* (Set Timer) gesetzt.

EvtLst - Event List (Ereignisliste)

Quelle: Spezifikation der virtuellen Verbindung (*Events&Actions*)

Die Ereignisse eines Ereigniskanals können sequentiell abhängig sein. Solche Ereignisse müssen in derselben Reihenfolge wie sie aufgetreten sind verarbeitet werden.

EST - Event Separation Times (Tabelle mit Ereignis-Separationszeiten)

Quelle: Prozessbeschreibung

Die *EST*-Tabelle ordnet gewissen Ereignispaaren Separationszeiten zu und definiert damit eine partielle Funktion *est*.

EST: EvtLst x EvtLst -> crd (enum-Werte als Indizes der 2-dimensionalen Tabelle)

Die Tabelle spezifiziert für sequentiell abhängige Ereignispaare, die einander direkt folgen können, den minmalen zeitlichen Abstand zwischen dem ersten und dem zweiten Ereignis des Paares. Für sequentiell abhängige Ereignisse, die einander nicht direkt folgen können, enthält die Tabelle *nil*, dargestellt mit dem Zeichen / (nicht im Definitionsbereich der partiellen Funktion). Parallele Ereignisse eines Ereigniskanals können gleichzeitig auftreten und haben Separationszeit 0.

Beispiele:

EST[e, f] = 90 f kann direkt nach e auftreten; der minimale zeitliche Abstand ist 90 stu.

EST[f, g] = / g kann nicht direkt nach f auftreten, es muss mindestens ein sequentiell abhängiges Ereignis dazwischen kommen.

EST[e, g] = 0 e kann gleichzeitig mit g auftreten, damit muss auch gelten EST[g, f] = 0.

EST[h, h] > 0 *sporadisches* oder *periodisches* Ereignis (siehe *Event.Category*)

EST[h, h] = 0 *Burst*-Ereignisse können sich zeitlich häufen (siehe *Event Category*)

Die Separationszeiten von Ereignissen, die nicht zum selben Ereigniskanals gehören, sind per definitionem Null, weil die Ereignisse verschiedener eines Ereigniskanäle gleichzeitig auftreten können.

Event Objekt

name	object name	symbol	Identifier des Ereignisses
mxRD	maximal Reaction Delay	crd (stu)	Maximal tolerierter zeitlicher Abstand zwischen dem Auftreten des Ereignisses und dem Ende der Verarbeitung auf dem Rechner
wcNT	worst case Notification Time	crd (stu)	Grösstmöglicher zeitlicher Abstand zwischen dem Auftreten des Ereignisses und dem Beginn der Detektierbarkeit auf dem Steuerrechner
csAct	causable Actions	pointer	Menge der Aktionen, die bei der Verarbeitung des Ereignisses bewirkt werden können
wcET	worst case Execution Time	crd (us)	Grösstmögliche Ausführungszeit der aktivierbaren Steuerfunktion
SG P	Successor Gap Period	crd (stu)	Minimaler zeitlicher Abstand zu den sequentiell abhängigen Ereignissen Für periodische Ereignisse ist SG die Periode P.
PDI_RD	PDI_ReactionDelay	crd (stu)	Einschränkung der Informationsverarbeitungszeit, die von der Beschränkung der Reaktionszeit herrührt

Event Objekt

PDI_RT	PDI_ResponseTimes	crd (stu)	Einschränkung der Informationsverarbeitungszeit, die aus den Antwortzeitbeschränkungen aller Aktionen resultiert, die vom Ereignis bewirkt werden können
PDI_SG	PDI_SuccessorGap	crd (stu)	Beschränkung der Informationsverarbeitungszeit für ein Ereignis um die Konsistenz des EDF-Scheduling zu sichern
PDI	Processing Deadline Interval	crd (stu)	PDI = min{PDI_RD, PDI_RT, PDI_SG} Minimum der spezifischen PDI's
SDI	Scheduling Deadline Interval	crd (T)	SDI = (PDI - T) / T, Zyklusbasiertes Deadline-Intervall, das für das Scheduling der Ereignismeldungen benutzt wird

Beschreibungen der Attribute eines Event Objektes

mxRD - maximal Reaction Delay (maximale Reaktionszeit)

Quelle: Zeitliche Anforderungen

mxRD spezifiziert für harte Ereignisse eine obere Schranke für die Verarbeitungszeit. Die Steuerkomponente, die das Ereignis verarbeitet, muss innerhalb der Zeit *mxRD* auf das Ereignis reagieren und die entsprechende Steuerfunktion ausführen (Aktualisierung des Systemzustandes). Diese Zeitbedingung ist unabhängig von den Zeitbedingungen für Aktionen, die bei der Verarbeitung des Ereignisses initiiert werden können.

wcNT - worst case Notification Time (grösstmögliche Ereignis-Benachrichtigungszeit)

Quelle: Sensorbeschreibung (lokal oder verteilt installiert)

wcNT spezifiziert die grösstmögliche Zeitdauer (Sensor-Latenzzeit) zwischen dem Auftreten des realen Ereignisses beim entsprechenden externen Prozess und dem Beginn der Detektierbarkeit auf dem Steuerrechner.

csAct - causable Actions (Bewirkbare Aktionen)

Quelle: Entwurfsmodell der Steuerkomponente

csAct spezifiziert Menge der Aktionen, die bei der Verarbeitung des Ereignisses bewirkt werden können (durch die Steuerfunktion des Ereignisses initiierbare Aktionen).

Beispiel: Alarm.csAct = {MotorOff, Drain, AlarmLmpOn, AlarmInfo}

wcET - worst case Execution Time (grösstmögliche Ausführungszeit)

Quelle: Code der Steuerkomponente

wcNT spezifiziert die grösstmögliche Ausführungszeit der Ereignisverarbeitung (*Worst Case Execution Time* der Steuerfunktion).

SG - Successor Gap (Nachfolger-Zeitlücke)

Quelle: Externe Prozessbeschreibung

SG spezifiziert die minimale zeitliche Separation (*Successor Separation*) zu den sequentiell abhängigen Folge-Ereignissen:

$$e.SG := \min\{x \text{ in } EvtChn \text{ und } EST[e, x] > 0 \mid EST[e, x]\}$$

Bei periodischen Ereignissen ist *SG* die *Ereignisperiode*, die mit *P* bezeichnet wird.

Bemerkungen. Jedes sporadische Ereignis ist zu sich selbst sequentiell abhängig, d. h. EST[e, e] > 0, oder EST[e, e] = / wenn sich das Ereignis nicht selber folgen kann (z. B. *On* und *Off* bei einem Schalter).

PDI_RD - PDI_ReactionDelay (siehe Unterkapitel 4.3)

Quelle: ITS

PDI_RD = mxRD - wcNT

Einschränkung der Informationsverarbeitungszeit, die von der Beschränkung der Reaktionszeit herrührt

PDI_RT - PDI_ResponseTimes (siehe Unterkapitel 4.3)

Quelle: ITS

PDI_RT = min{a in csAct | (a.mxRT - wcNT - a.wcPT)}

Einschränkung der Informationsverarbeitungszeit, die aus den Antwortzeitbeschränkungen aller Aktionen resultiert, die vom Ereignis bewirkt werden können.

PDI_SG - PDI_SuccessorGap (siehe Unterkapitel 4.3)

Quelle: ITS

PDI_SG = SG - wcNT

Beschränkung der Informationsverarbeitungszeit für ein Ereignis um die Konsistenz des Scheduling zu sichern. Für unser EDF-Scheduling von Ereignisverarbeitungen garantiert diese Beschränkung, dass der Scheduler nicht die Verarbeitung sequentiell abhängiger Ereignisse vertauscht.

PDI - Processing Deadline Interval (siehe Unterkapitel 4.3)

Quelle: ITS

PDI = min{PDI_RD, PDI_RT, PDI_SG}

Minimum der spezifischen Processing Deadline-Intervalle.

SDI - Scheduling Deadline Interval (siehe Unterkapitel 4.3)

Quelle: ITS

SDI = (PDI - T) / T,

Zyklusbasiertes Deadline-Intervall, das für das Scheduling der Ereignismeldungen benutzt wird

Action Channel Objekt

name	object name	symbol	Identifier des Aktionenkanals
RTC	Real-Time Conditions	{hard, soft}	
cat	Action Category	actCatType	actCatType = {periodic, sporadic, locMsg, globMsg, setTm}
ActLst	Action List	pointer	Aktionen des Aktionenkanäls, zeigt auf *Action* Objekte

Beschreibungen der Attribute eines Action Channel Objektes

RTC - Real-Time Conditions

hard. Ereignisverarbeitungen, die harte Aktionen initiieren, sollten unbedingt die entsprechenden Deadlines einhalten. Das Überschreiten einer Deadline ist ein Fehler und muss vom System behandelt werden.

soft. Die *mxRT* Zeitbedingungen (maximal Response Time) dieser Aktionen soll im Mittel eingehalten werden (Performance).

actCat - Action Category

Quelle: Externe Prozessbeschreibung

actCatType = {periodic, sporadic, locMsg, globMsg, setTm}

periodic. Periodische Aktionen werden nur von periodischen Ereignissen bewirkt (Regler).

sporadic. Eine sporadische Aktion kann durch jedes Ereignis bewirkt werden.

locMsg. Eine Aktion der Kategorie *locMsg* erzeugt immer ein entsprechendes Ereignis (Kategorie *locMsg*) für eine Unit desselben Prozessors. Solche Ereignisse werden als lokale Steuernachrichten (local *Control Messages*) bezeichnet. Mittels lokalen Steuernachrichten kann zwischen den Units, die auf demselben Prozessor asynchron kommuniziert werden.

globMsg. Eine Aktion der Kategorie *globMsg* erzeugt immer ein entsprechendes Ereignis (Kategorie *globMsg*) für eine Unit eines anderen Prozessors. Solche Ereignisse werden als globale Steuernachrichten (global *Control Messages*) bezeichnet. Mittels globalen Steuernachrichten kann zwischen den Units auf verschiedenen Prozessoren asynchron kommuniziert werden.

setTm. Eine *setTm*-Aktion (set Timer) setzt einen Timer für eine bestimmte Zeitspanne (timer delay). Das Ablaufen des Timers ist ein Ereignis (*TimeIsUp*) der Kategorie *tmUp*, das von derselben Unit verarbeitet wird.

ActLst - Action List (Aktionenliste)

Quelle: Entwurfsmodell der Steuerkomponente (reaktive Maschine)

Die Aktionen eines Aktionenkanals können sequentiell abhängig sein. Solche Aktionen müssen in derselben Reihenfolge erzeugt werden, wie sie auf dem Rechner initiiert werden.

Action Objekt

name	object name	symbol	Identifier der Aktion
mxRT	maximal Response Time	crd (stu)	Maximal tolerierter zeitlicher Abstand zwischen dem Auftreten des bewirkenden Ereignisses und der Ausführung der Aktion
wcPT	worst case Production Time	crd (stu)	Grösstmöglicher zeitlicher Abstand zwischen der Initiierung der Aktion und der realen Ausführung.

Beschreibungen der Attribute eines Action Objekte

mxRT - maximal Response Time (maximale Antwortzeit)

Quelle: Zeitliche Anforderungen

mxRT ist für harte Aktionen eine obere Schranke für die Antwortzeit. Wenn ein Ereignis auftritt, das über die Steuerkomponente diese Aktion bewirkt, muss die Aktion innerhalb der Zeit *mxRT* ausgeführt worden sein.

wcPT - worst case Production Time (grösstmögliche Aktions -Übertragungszeit)

Quelle: Aktorbeschreibung (lokal oder verteilt installiert)

wcPT spezifiziert die grösstmöglicher Zeitdauer (Aktor-Latenzzeit) zwischen der Initiierung einer Aktion an der Rechnerschnittstelle und der Ausführeng der realen Aktion beim entsprechenden externen Prozess.

7.3 Processing Deadline-Intervalle und Scheduling Deadline-Intervalle

Der Begriff *Deadline-Intervall*, oft als relative Deadline bezeichnet, ist im Kapitel 5 eingeführt worden. Ein *Deadline-Intervall* definiert die Zeitspanne, innerhalb welcher ein detektiertes Ereignis verarbeitet werden muss.

Die Spezifikationsobjekte für Ereignisse und Aktionen enthalten vorerst Information, die aus andern Phasen des Entwicklungsprozesses stammen (Anforderungen, Spezifikation der virtuellen Interaktion, Installation der Sensoren und Aktoren). Das Ziel diese Unterkapitels ist ein systematisches Vorgehen festzulegen, mit dem aus diesen Informationen die Deadline-Intervalle für die Ereignisverarbeitung und die Zyklusperioden der Embedded Units spezifiziert werden können. Deadline-Intervalle werden nur für diejenigen Ereignisse spezifiziert, für die das RTC-Attribut auf *hard* gesetzt ist. Das ist oft nur ein kleiner Teil aller zu verarbeitenden Ereignisse.

Die Wahl Zyklusperiode hängt von den spezifizierten harten Zeitanforderungen und den Ausführungszeiten der reaktiven Maschine ab. Es wird sich zeigen, dass es unter Umständen gar nicht möglich ist, eine zulässige Zyklusperiode zu finden. In solchen Fällen muss die Embedded Unit in zwei oder sogar drei Units aufgeteilt werden (*Unit-Refraction*).

Processing Deadline-Intervalle (PDI's) und Scheduling Deadline-Intervalle (SDI's)

Bei der Bestimmung der zyklus-basierten Deadline-Intervalle entsteht ein Problem. Einerseits müssen die Zyklusperioden bekannt sein, um die Deadline-Intervalle in Einheiten der diskreten Zykluszeit ausdrücken (Anzahl Zyklen), andererseits schränkt die Grösse der Deadline-Intervalle die Wahl der Zyklusperiode ein.

Diese zyklische Abhängigkeit kann aufgelöst werden, indem man zuerst Deadline-Intervalle bestimmt, die wie konventionelle Deadlines in Einheiten der Prozessorzeit ausgedrückt werden. Diese Deadline-Intervalle werden als *Processing Deadline-Intervalle* (*PDI's*) bezeichnet. Aufgrund dieser Grössen kann dann die Zyklusperiode bestimmt werden. Erst anschliessend werden die zyklusbasieren Deadline-Intervalle berechnet.

PDI - Processing Deadline-Intervall. Um einfache Bedingungen für die Existenz zulässiger Zyklusperioden formulieren zu können, bestimmen wir zuerst von jedem Ereignis das *Processing Deadline-Intervall* (*PDI*). Dieses Intervall spezifiziert die Zeitdauer, während der für das aufgetretene Ereignis auf dem Rechner die auszuführende Informationsverarbeitung (Detektion, Scheduling, Ereignisverarbeitung) stattfinden muss. Das Intervall beginnt, sobald am Prozessor-Interface die Information für das aufgetretene Ereignis verfügbar ist (hardware notification).

SDI - Scheduling Deadline-Intervall. Das *Scheduling Deadline-Intervall* (*SDI*) eines Ereignisses bestimmt, innerhalb welcher Zeit relativ zur Ereignis-Detektion die reaktive Maschine die entsprechende Ereignismeldung verarbeitet haben muss. Diese Intervalle definieren damit Zeitbedingungen für die Implementation des zyklusbasierten Deadline-Scheduling der Ereignismeldungen. Diese Deadline-Intervalle werden wie die diskrete Zykluszeit als ganzzahlige Vielfache der Zyklusperiode ausgedrückt.

7.3.1 PDI - Processing Deadline-Intervalle für sporadische Ereignisse

Das effektive *Processing Deadline-Intervall* für ein sporadisches Ereignis kommt aufgrund verschiedenartiger Zeitbedingungen zustande. Jede dieser Bedingung führt selbst zu einer spezifischen Beschränkung der Informationsverarbeitungszeit, ausgedrückt durch ein entsprechend *spezifisches* Deadline-Intervall.

Das effektive Deadline-Intervall für die Informationsverarbeitung, die zum Ereignis *e* gehört, ist dann durch das Minimum der spezifischen Deadline-Intervalle bestimmt:

e.PDI = min{e.PDI_ReactionDelay, e.PDI_ResponseTimes, e.PDI_SuccessorGap}

wobei die drei spezifischen Deadline-Intervalle folgendermassen definiert sind:

PDI_ReactionDelay
spezifiziert die Einschränkung der Informationsverarbeitungszeit, die von der Beschränkung der Reaktionszeit herrührt.

PDI_ResponseTimes
spezifiziert die Einschränkung der Informationsverarbeitungszeit, die aus den Antwortzeitbeschränkungen aller Aktionen resultiert, die vom Ereignis bewirkt werden können.

PDI_SuccessorGap
spezifiziert eine technische Beschränkung der Informationsverarbeitungszeit für ein Ereignis, um die Konsistenz des EDF-Scheduling zu sichern. Diese Beschränkung stellt sicher, dass bei der Verarbeitung sequentiell abhängiger Ereignisse die zeitliche Reihenfolge eingehalten wird.

Die Definitionen dieser drei spezifischen Processing Deadline-Intervalle werden im folgenden näher erläutert.

PDI_ReactionDelay (PDI_RD)
Beschränkung der Informationsverarbeitungszeit infolge der beschränkten Reaktionszeit

Die in den Anforderungen spezifizierte obere Zeitschranke *mxRD* (*maximal Reaction Delay*) beschränkt die Reaktionszeit unabhängig von den auszuführenden Aktionen. Diese Anforderung führt damit zu einer entsprechenden Beschränkung der Informationsverarbeitungszeit (Information Processing) auf dem Steuerrechner. Das entsprechende spezifische Deadline-Intervall wird als *PDI_ReactionDelay* bezeichnet. Dieses Zeitintervall spezifiziert, innerhalb welcher Zeit auf das am Prozessor-Interface detektierbares Ereignis reagiert werden muss.

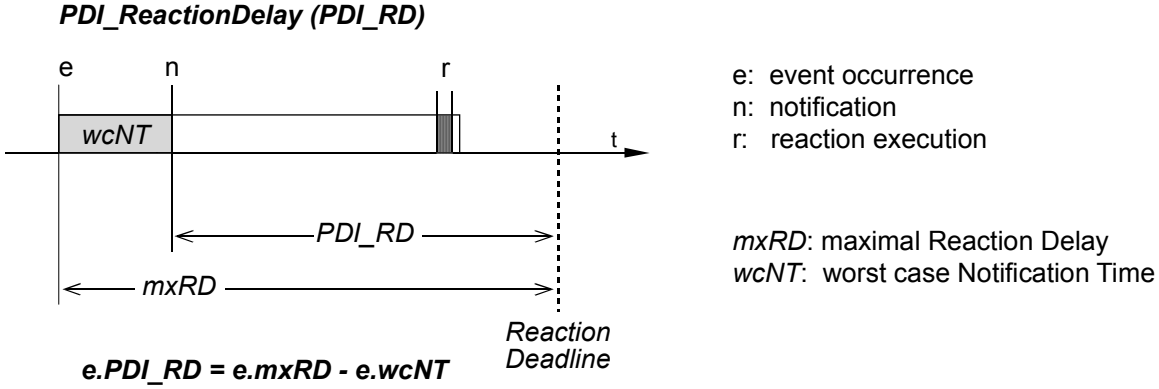

Abb. 79: Reaktionszeitbeschränkung und entsprechendes Processing Deadline-Intervall

In *Abbildung 79* ist der zeitliche Ablauf vom Auftreten des Ereignisses *e* bis zur Ausführung der ausgelösten Reaktion auf dem Steuerrechner dargestellt (ohne Aktionserzeugung). Es handelt sich um einen Fall, bei dem die Benachrichtigungszeit (*Notification Time*) den schlimmsten Wert hat, nämlich *wcNT*. Die Zeitbeschränkung für die Informationsverarbeitung ist damit gegeben durch die maximale Antwortzeit reduziert um die schlimmste Benachrichtigungszeit:

PDI_ReactionDelay: *e.PDI_RD = e.mxRD - e.wcNT*

Diese Grösse ist unabhängig von den Zeitbedingungen, die an potenzielle durch die Reaktion bewirkbare Aktionen gestellt werden (*maximal Response Times (mxRT)*).

PDI_ResponseTimes (PDI_RT)

Zeitbeschränkung infolge der beschränkten Antwortzeiten erzeugbarer Aktionen

Neben der Beschränkung der Reaktionszeiten auf Ereignisse wird auch das Einhalten maximaler Antwortzeiten (*maximal Response Times*) für die erzeugten Aktionen verlangt. Auch diese Zeitbedingungen führen bei den Ereignissen, die Aktionen bewirken können, zu einer entsprechenden Beschränkung der Informationsverarbeitungszeit. Das entsprechende Deadline-Intervall wird mit *PDI_ResponseTimes* bezeichnet.

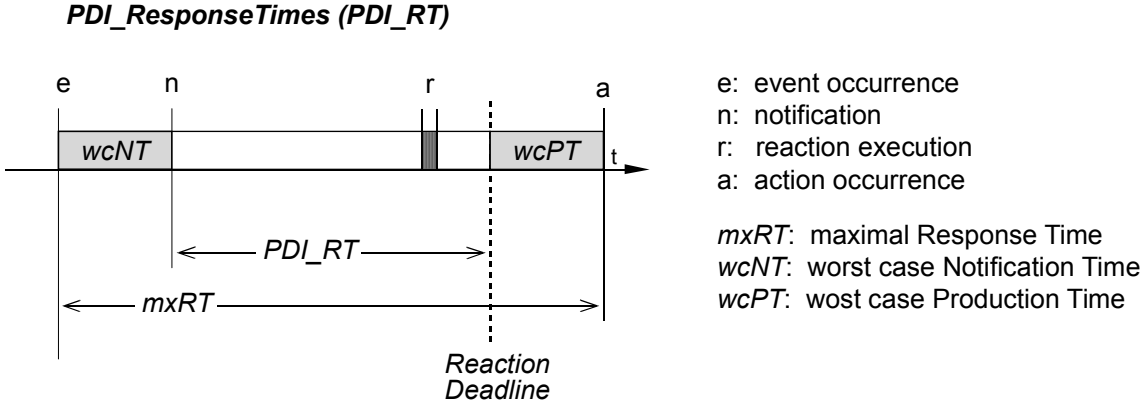

Abb. 80: Antwortzeitbeschränkung und entsprechendes Processing Deadline-Intervall

In *Abbildung 80* ist der zeitliche Ablauf vom Auftreten des Ereignisses *e* bis zur Ausführung der ausgelösten Reaktion und der Initiierung der Aktion *a* auf dem Steuerrechner dargestellt. Es handelt sich beim abgebildeten Beispiel wieder um einen extremen Fall. Sowohl die Benachrichtigungszeit (*Notification Time*) als auch die Produktionszeit für die Aktion (*Production Time*) nehmen die schlimmsten Werte an, nämlich *wcNT* und *wcPT*. Die resultierende Zeitbeschränkung für die Informationsverarbeitung ist dann durch die spezifizierte maximale Antwortzeit *mxRT* für die Aktion *a* bestimmt:

$$e.PDI\_ResponseTime = a.mxRT - e.wcNT - a.wcPT$$

Im allgemeinen Fall können bei der Verarbeitung eines bestimmten Ereignisses unterschiedliche Aktionen mit verschiedenen Antwortzeitbeschränkungen initiiert werden. Um die entsprechende Zeitbeschränkung zu bestimmen müssen deshalb alle durch die Reaktion erzeugbaren Aktionen berücksichtigt und das kleinste resultierende Deadline-Intervall genommen werden:

$$\textbf{PDI\_ResponseTimes:} \quad e.PDI\_RT = min\{a \text{ in } e.csAct \mid (a.mxRT - e.wcNT - a.wcPT)\}$$

Das Ereignis-Attribut *e.csAct (causable Actions)* spezifiziert die Menge der durch das Ereignis *e* bewirkbaren Aktionen.

PDI_SuccessorGap (PDI_SG)

Schedulingspezifische Beschränkung der Informationsverarbeitungszeit

Die Verarbeitung eines Ereignisses kann verzögert werden, wenn ein dringenderes Ereignis verarbeitet werden muss (Scheduling). Dabei darf die Verarbeitungsverzögerung aber nie so gross werden, dass ein späteres, sequentiell abhängiges Ereignis (sequential successor) vorher verarbeitet wird. Ob solche unerlaubten Vertauschungen auftreten können, hängt vom gewählten Schedulingverfahren ab.

Bei *FirstComeFirstServed*-Scheduling zum Beispiel tritt das Problem trivialerweise nicht auf; dafür kann mit diesem Verfahren das Einhalten von Zeitbedingungen nur in völlig unkritischen Fällen garantiert werden. Bei deadline-basiertem Scheduling sind solche unerlaubten Vertauschungen möglich. Sie können aber verhindert werden, indem für jedes Ereignis mit sequentiellen Nachfolgern die

Informationsverarbeitungszeit durch ein entsprechendes spezifisches Deadline-Intervall eingeschränkt wird. Diese spezifische Deadline-Intervalle wird mit *PDI_SuccessorGap* bezeichnet.

Grundlage für die Bestimmung des Scheduling-Bereichs eines Ereignisse ist die *EST*-Tabelle (*Event Separation Times)* des *Event Channel Objekts*. Für alle geordneten Ereignispaare, die nur direkt hintereinander auftreten können, spezifiziert die Tabelle den minimalen zeitlichen Abstand der beiden Ereignisse. Da ein Ereignis mehrere direkte Nachfolger haben kann, wird für jedes Ereignis das Minimum der Separationszeiten seiner Nachfolger ermittelt. Diese untere Zeitschranke für Nachfolger wird als Wert des Ereignis-Attributes *SG* (*Successor Gap*) im entsprechenden Ereignis-Objekt festgehalten. *Successor Gap* bedeutet minimale Nachfolger-Zeitlücke.

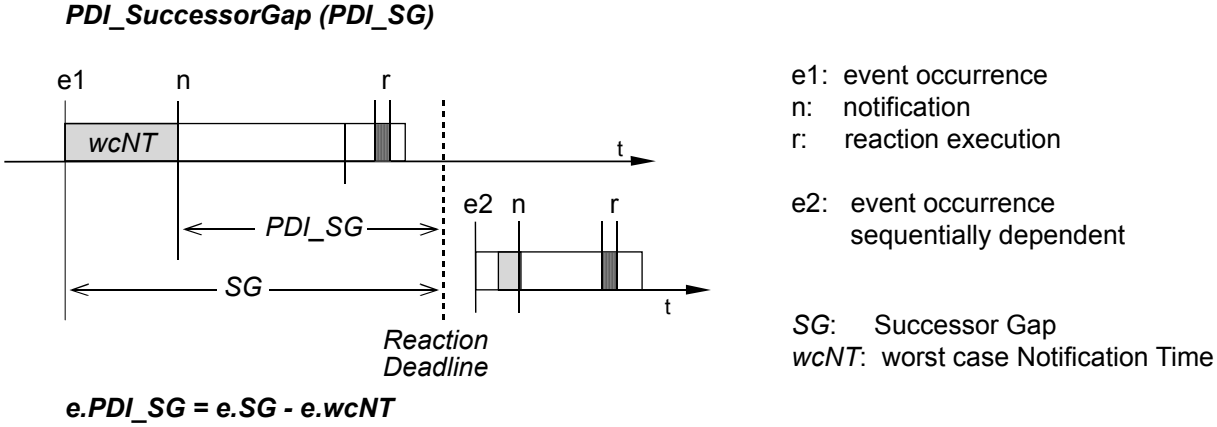

Abb. 81: Successor Gap und entsprechendes Processing Deadline-Intervall

In *Abbildung 81* ist der zeitliche Ablauf vom Auftreten des Ereignisses *e1* bis zur Ausführung der ausgelösten Reaktion auf dem Steuerrechner dargestellt (ohne Aktionserzeugung). Es wurde der extreme Fall gewählt, in dem die *Notification Time* von *e1* den schlimmsten Werte annimmt, nämlich *wcNT*. Das Ereignis *e2* ist ein von *e1* sequentiell abhängiges Ereignis, das direkt nach *e1* auftreten kann. Der zeitliche Abstand zu *e1* ist aber mindestens *e1.SG* (*Successor Gap*). Der garantierte zeitliche Vorsprung von *e1* zu *e2* ergibt damit einen Scheduling-Bereich, in welchem trivialerweise keine Vertauschung mit *e2* möglich ist. Das gilt für alle sequentiell abhängigen Nachfolger von *e1*. Wir können deshalb mit einem entsprechenden Deadline-Intervall sicherstellen, dass die Verarbeitung sequentiell abhängiger Ereignisse durch den Scheduler nie vertauscht werden:

PDI_SuccessorGap: *e.PDI_SG = e.SG - e.wcNT*

Die Bestimmung von *PDI_SuccessorGap* könnte noch optimiert werden, indem auch die wcNT und die Deadline-Intervalle von *e2* berücksichtigt würden. Meistens ist das aber nicht nötig, weil bereits der auf die einfache Art ermittelte *PDI_SuccessorGap*-Wert wesentlich grösser ist als die anderen spezifischen Deadline-Intervalle des Ereignisses.

7.3.2 SDI - Scheduling Deadline-Intervall für sporadische Ereignisse

Mit dem *PDI*-Attribut (*Processing Deadline-Intervall*) eines Ereignisses wird eine Zeitschranke für die Informationsverarbeitung auf dem Steuerrechner spezifiziert. Das Deadline-Intervall bestimmt, wieviel Zeit für diese Arbeit zur Verfügung steht, gerechnet vom Zeitpunkt an, wo die Information über das aufgetretene Ereignis am Prozessorinterface verfügbar ist (Hardware Notification).

Das *Scheduling Deadline-Intervall (SDI)* hingegegen spezifiziert Zeit, die für die zyklusbasierte Verarbeitung eines detektierten Ereignisses (event processing) zur Verfügung steht. Diese Zeitspanne wird in Vielfachen der Zyklusperiode ausgedrückt. Die *SDI's* sind die Grössen, die beim EDF-Scheduling der Ereignisverarbeitung die dynamischen Prioritäten für die Ausführung der Ereignismeldungen bestimmen.

Um das *Scheduling Deadline-Intervall* eines Ereignisses zu bestimmen, müssen das entsprechende *Processing Deadline-Intervall* und die Zyklusperiode der Embedded Unit bekannt sein.

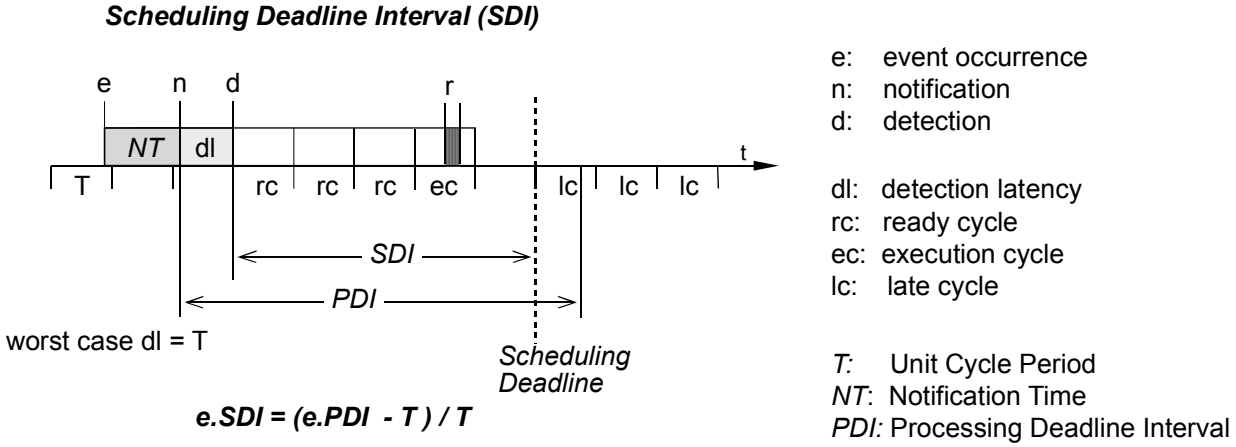

Abb. 82: Scheduling Deadline-Intervall

In *Abbildung 82* ist der Ablauf einer Ereignisverarbeitung in Bezug auf die diskrete Zykluszeit dargestellt. Die Detektion eines Ereignisses erfolgt immer am Anfang eines Zyklus; in der Abbildung ist das der Zyklus, der zum Zeitpunkt d beginnt. Das Scheduling Deadline-Intervall *SDI* ist, wie aus der Figur ersichtlich, kleiner als das bereits spezifizierte Processing Deadline-Intervall *PDI*:

1. Vor der zyklusbasierten Detektion eines Ereignisses geht für die Ereignisverarbeitung die Zeitspanne *dl* bis zum Zyklusbeginn verloren (Detektionslatenzzeit *dl*). Die Detektionslatenzzeit ist im schlimmsten Fall so gross wie die Zyklusperiode T.

2. Die Deadline-Intervalle für die Ausführung der Ereignismeldungen werden in der diskreten Zykluszeit z ausgedrückt. Bei der Umrechnung der PDI's in die SDI's wird dementsprechend auf eine ganze Anzahl Zyklen abgerundet (Ganzzahl-Division).

Damit ergibt sich für das Scheduling Deadline-Intervall *SDI*

$$e.SDI = (e.PDI - T) / T \qquad (\ /\ \text{bedeutet Ganzzahl-Division ohne Rest})$$

Wir betrachten hier die Zyklusperiode T vorläufig als freien Parameter. Für welche Werte von T das zyklische Unit-Scheduling funktionieren kann, wird im nächsten Unterkapitel diskutiert.

7.3.3 SDI - Scheduling Deadline-Intervall für periodische Ereignisse

Wir setzen voraus, des die Perioden der periodischen Ereignisse Vielfache der Zyklusperiode sind. Damit wird die Bestimmung der Scheduling Deadline-Intervalle sehr einfach.

Erstens ist die *Notification Time* kein Thema ($wcNT = 0$), weil periodische Ereignisses zu bestimmten bekannten Zeitpunkten auftreten. Es gibt keine Übertragungszeiten, weil ein Hardware-Timer als gemeinsame Clock von Sensoren und Rechner verwendet werden kann.

Zweitens ist auch die Detektion trivial. Ein Ereignis tritt immer am Anfang des entsprechenden Zyklus auf (Detektionslatenz = 0).

Drittens können wir bei einem Regler annehmen, dass die *Production Time PT* der Regler-Aktion vernachlässigt werden kann und die verlangte Reaktionszeit gleich wie verlangte Antwortzeit ist.

Für das Scheduling Deadline-Intervall erhält man damit

$$SDI = mxRD \ / \ T$$

Ein Problem entsteht, wenn $mxRD < T$ gefordert ist, weil damit $SDI = 0$ wird. Solche Fälle müssen gesondert behandelt werden.

Etwas anderes sind Bedingungen an den Input-Jitter, d. h. an die zeitliche Schwankung des periodischen Zugriffs auf die Inputdaten. Solche Bedingungen sind problemlos erfüllbar, weil Inputzugriffe von den streng-periodischen Input-Task der entsprechenden Embedded Unit ausgeführt werden.

7.4 Bedingungen für die Zyklusperiode T einer Unit

Die Grösse der Zyklusperiode bestimmt das Verhältnis zwischen dem Aufwand für das Input-Handling und der Ereignisverarbeitung. Wenn die Zyklusperiode zu kurz gewählt wird, müssen die immer gleichen Inputfunktionen viel zu oft ausgeführt werden (input handling overhead). Ein viel ernsthafteres Problem tritt jedoch auf, wenn gewisse Steuerfunktionen Ausführungszeiten haben, die grösser sind als die Zyklusperiode. In diesem Fall müsste damit gerechnet werden, dass aufgrund der non-preemptiven Ausführung der reaktiven Maschine das System aus dem Takt gerät. Ist die Zyklusperiode jedoch zu lang, kann die Einhaltung sehr kurzer Deadlines problematisch werden (grosse Detektionslatenzzeit). Zudem besteht die Gefahr, dass bei schnellen Prozessen einzelne Sensoraktivierungen durch die “Maschen” fallen.

Damit das zyklische Unit-Scheduling funktionieren kann, müssen die spezifizierten Ereignis- und Aktionsattribute gewisse Bedingungen erfüllen. Das Einhalten der in diesem Abschnitt erörterten technischen Bedingungen ist notwendig für das Funktionieren des zyklischen Unit-Scheduling. Ob die Deadlines aber immer eingehalten werden können, ist ein anderes Problem, das sog. *Scheduling Feasability Problem*. Bei dieser Problemstellung will man aufgrund der bekannten Auftrittshäufigkeiten von Tasks, bzw. Ereignissen und den gemessenen oder berechneten *Worst Case Execution Times* der Tasks, bzw. Steuerfunktionen beweisen, dass auch im schlimmsten Fall die Deadlines immer eingehalten werden können. Das kann zu mathematisch äusserst anspruchsvollen Problemstellungen führen. *Schedulability Verification* ist denn auch ein bleibendes Thema in der Forschung der Computerwissenschaften, Wir erwarten, dass das zyklusbasierte Deadline-Scheduling hier zu wesentlichen Vereinfachungen führt und *Schedulability Verification* in der Zukunft auch in der industriellen Praxis angewendet werden kann. Im Unterkapitel 4.6 wird kurz auf diesen Problemkreis eingegangen.

7.4.1 Notwendige Bedingungen für die Zyklusperiode T

Damit das zyklusbasierte non-preemptive Scheduling funktionieren kann, muss wie gesagt einerseits die Zyklusperiode T einer Embedded Unit grösser sein als die Ausführungszeiten der reaktiven Maschine. Andererseits muss die Zyklusperiode mindestens so klein gewählt werden, dass kein *Scheduling Deadline-Intervall* Null wird.

Eine *Scheduling Deadline-Intervall* mit Wert < 1 kann zum Beispiel entstehen, wenn eine angeforderte Antwortzeit kürzer als die gewählte Zyklusperiode ist. Damit wird es möglich, dass das detektierte Ereignis zum Zeitpunkt der Detektion bereits verarbeitet sein sollte. In diesem Fall ist die Auflösung der diskreten Zykluszeit zu gross, um das Einhalten der kurzen Deadline garantieren zu können (grosse Detektionslatenzzeit).

Zulässige Zyklusperioden

Bei einer Embedded Unit mit periodischen Ereignissen wird meistens die geneinsame Grundperiode dieser Ereignisse als Zyklusperiode genommen. Bei Embedded Units ohne periodische Ereignisse wird man versuchen, die Zyklusperiode nicht zu klein anzusetzen, damit der Input-Handling-Overhead nicht zu gross wird.

Um zu verifizieren, ob die gewählte Zyklusperiode T zulässig ist, muss mit allen Ereignisverarbeitungszeiten und Scheduling Deadline-Intervallen verglichen werden.

$$MaxET = \max\{e \text{ in } Evt \mid e.wcET\} \quad \text{und} \quad MinPDI = \min\{e \text{ in } Evt \mid e.PDI\}$$

MaxET bezeichnet das Maximum der Worst Case Ausführungszeiten.

MinPDI bezeichnet das kleinste aller Processing Deadline-Intervalle.

Evt ist die Menge der Ereignisse, welche die Unit verarbeitet.

Notwendige Bedingungen für T

$T > MaxET \qquad T < MinPDI$

Ist diese Bedingung nicht erfüllt, dauern die schlimmsten Ereignisverarbeitung länger als der Zyklustakt, oder es gibt bestimmte Processing Deadline-Intervalle, die kürzer als T sind.

Hinreichende Bedingungen für T

$T >= MaxET + T_{wcStatic} \qquad T_{wcStatic} = T_{wcInput} + T_{wcPeriodic} + T_{wcDetection} + T_{wcOutput}$

$T <= minPDI - T$

Die erste Bedingung stellt sicher, dass jede Ereignisverarbeitung neben den in jedem Zyklus notwendigen Arbeiten innerhalb der Zyklusperiode erledigt werden kann.

Die zweite Bedingung berücksichtigt die Detektionslatenzzeit *T,* die bei der Bestimmung der Scheduling Deadline-Intervalle in Abzug gebracht wird (siehe *Abbildung 82*).

Vereinfachte hinreichende Bedingungen für T

Meistens kommt man mit vereinfachten Bedingungen aus:

$T >= 2 * MaxET$ (Annahme: $T_{wcStatic} < MaxET$)

$T <= minPDI / 2$ (Umformulierung der oben gegebenen Bedingung)

Bei der ersten Vereinfachung nimmt man an, dass die in jedem Zyklus notwendigen Arbeiten weniger lang dauern als die längste Ereignisverarbeitung. Die zweite Vereinfachung ist nur eine Umformung der Ungleichung.

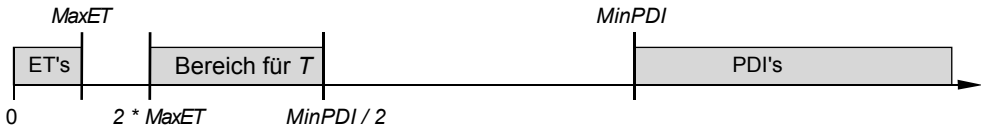

Abb. 83: Zulässiger Zeitbereich für die Zyklusperiode

In Abbildung 83 ist der zulässige Zeitbereich für die Zyklusperiode einer Embedded Unit dargestellt. Dabei wurden die *vereinfachten Bedingungen* für die Zyklusperiode angewendet.

Im folgenden wird kurz ein Fall mit zulässiger und ein Fall mit unzulässiger Zyklusperiode diskutiert. Im zweiten Fall kann es gar keine zulässige Zyklusperiode geben, welche die *vereinfachten* Bedingungen erfüllt.

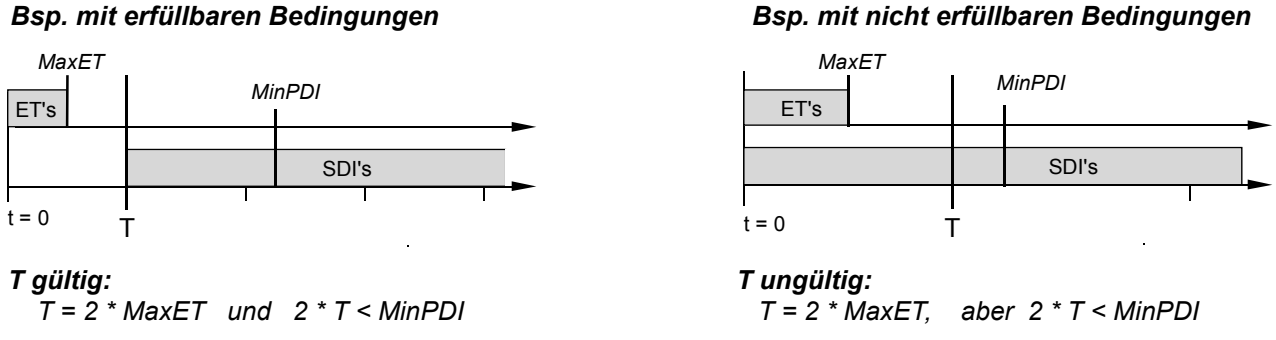

Abb. 84: Festlegung der Zyklusperiode - zwei Fälle

In *Abbildung 84* sind links die Zeitbereiche der Ausführungszeiten (ET's) und der Scheduling Deadlines (SDI's) für einen unproblemtischen Fall eingezeichnet. Die Zyklusperiode T ist doppelt so gross wie die grösste Ausführungszeit (*MaxET*) gewählt worden, und T ist etwas kleiner als *minPDI* / 2.

Im rechten Bild von *Abbildung 84* ist ein Fall mit unzulässiger Zyklusperiode dargestellt. T ist zwar doppelt so gross wie *MaxET*, aber T ist grösser als *minPDI* / 2. Bei der gewählten unzulässigen Zyklusperiode T beginnt der Bereich der SDI's bei Null, d. h. es gibt Ereignisse mit PDI = 0. Solche Ereignisse müssten verarbeitet sein, bevor sie detektiert werden!

Abbildung 84 zeigt einen Fall ohne periodische Ereignisse. Wenn auch periodische Ereignisse zu verarbeitet sind, muss die Zyklusperiode so gewählt werden, dass alle Ereignisperioden ganzzahlige Vielfache dieses Grundtakts sind, d. h. $P_i = n_i * T$ wo n_i eine natürliche Zahl ist.

Bei vielen Systemen sind die vereinfachten Zyklusbedingungen problemlos erfüllbar, weil sowohl obere Schranken für Antwortzeiten als auch die Separationszeiten sequentiell abhängiger Ereignisse von einer Grössenordnung sind, die den Beobachtungszeiten der physikalischen Prozesse entsprichen. Die Grössen für die Ausführungszeiten der reaktiven Maschine hingegen sind typischerweise um eine bis zwei Grössenordnung kleiner als diese *realen* Zeitintervalle.

Nicht erfüllbare Zyklusbedingungen

Die beiden *vereinfachten* Bedingungen

*T >= 2 * MaxET* *und* *T <= minPDI / 2*

sind nicht gleichzeitig erfüllbar wenn

2 * MaxET > minPDI / 2

Falls die Zyklusperiodenbedingungen nicht erfüllbar sind, gibt es zwei mögliche Auswege, einen *pragmatischen* und einen *konstruktiven*.

Mit dem *pragmatischen* Ansatz versucht man das Problem zu beheben, indem man zum Beispiel die Anforderungen an bestimmte Antwortzeiten abschwächt, oder indem man Steuerfunktionen mit zu grosser Ausführungszeit in kleinere aufspaltet (verkleinern der non-preemptiven Granularität). Ob das Problem behoben werden kann, indem man einen schnellerer Prozessor einsetzt, muss im jedem Fall speziell untersucht werden. Meistens ist die Leistung des Prozessors genügend, um den angeforderten Durchsatz zu garantieren. Das Problem mit den nicht erfüllbaren Zyklusbedingungen kommt fast immer daher, dass bei einzelnen Lastspitzen gewisse Deadlines nicht eingehalten werden können.

Mit dem *konstruktiven* Ansatz wird eine *kritische Embedded Unit* in kleinere Units aufgeteilt, die Funktionen unterschiedlicher Geschwindigkeitsklassen ausführen. Das Ziel ist, dass nach der Aufteilung für jede Unit die Zyklusbedingungen erfüllbar sind. Diese als *Unit-Refraktion* bezeichnete Aufteilung ist das Thema des nächsten Unterkapitels.

7.5 Unit-Refraktion - Aufteilung in verschieden schnelle Units

> *Unit-Refraktion* wird zu einer entwicklungstechnischen Herausforderung,
> wenn bei der Aufteilung der Unit funktionale Strukturen aufgebrochen werden müssen.

Wenn es nicht möglich ist, die Zyklusperiodenbedingungen einer Embedded Unit zu erfüllen, kann durch Aufteilung der kritischen Unit eine Lösung gefunden werden. Ein solche Aufteilung in quasi-parallel ausführbare Units wird als *Unit-Refraktion* bezeichnet. Die resultierenden Embedded Units, typischerweise zwei oder drei, werden normalerweise auf demselben Prozessor implementiert. Solche Units können als prioritätsgesteuerte preemptive Tasks eines RT-OS implementiert werden. Die Implementierung kann aber relativ einfach auch ohne RT-OS erfolgen. Voraussetzung ist, dass eine Funktion zur Verfügung steht, die den Kontextwechsel zeitlich unterbrochener Embedded Units implementiert (siehe Kapitel 6). Bei verteilt entworfenen Embedded Systemen wird das System von Anfang an als Multi-Unit-System konstruiert, d. h. auf jedem Prozessor wird mindestens eine Embedded Unit ausgeführt. Bei jedem Prozessor kann es dann in analoger Weise zu einer Unit-Refraktion kommen, sobald die entsprechenden Zyklusperiodenbedingungen nicht erfüllt werden können.

In diesem Unterkapitel werden generische Muster von Unit-Refraktionen erörtert. In der Praxis ist bei der Aufteilung aber von der funktionalen Struktur auszugehen, denn bei jeder Aufteilung muss mit aufgebrochenen lokalen funktionalen Abhängigkeiten gerechnet werden. Dabei gehen immer lokale Interaktionen verloren, die durch Message Passing Interaktion (*Control Messages*) zwischen den neuen Units ersetzt werden müssen. Für die Implementierung des Message Passing kommen je nach Unit-Implementation Intertask-Kommunikationsmechanismen (RT-OS) oder selbst geschriebene Kommunikationskonstrukte in Frage. Das Hauptziel bei einer Unit-Refraktion bleibt aber unabhängig von der gewählten Implementationstechmik, nämlich eine Aufteilung zu finden, bei der die funktionale Kopplung zwischen den neuen Units minimal ist.

Es ist deshalb empfehlenswert, bereits bei der Modellierung der Steuerfunktionen mehrere lose gekoppelte Steuerkomponenten (Clusters) zu entwerfen. Bei solchen Aufteilungen wird man versuchen, mögliche Unit-Refraktionen vorauszusehen (Geschwindigkeitsklassen), damit man bei der Implementierung der Embedded Units entsprechend flexibel ist. Falls dann bei der Implementierung keine Unit-Refraktion notwendig ist, können die asynchronen Steuerkomponenten immer noch mitsamt dem asynchronen Message Passing in einer einzigen Embedded Unit implementiert werden (generierte *CIP Units* für Clustergruppen bei CIP-Modellen).

Es ist klar, das mit Unit-Refraktion die Implementierung der Embedded Software aufwendiger wird, und dass auch die Wartungsfreundlichkeit des Systems Einbussen erleiden kann. Aus diesem Grund müssen bei der Systementwicklung auch strategische Überlegungen ihren Platz haben. Aufwendige Aufteilungen können zum Beispiel verhindert werden, wenn an den funktionalen Anforderungen gezielte Abstriche gemacht werden. Das ist eine wichtige Software-Engineering Strategie, die auch für andere Systemkategorien Gültigkeit hat: “Wenn das gestellte Problem schwieriger ist angenommen, muss damit gerechnet werden, dass die geplante Lösung zu teuer kommt. In diesem Fall bleibt nichts anderes übrig, als ein einfacheres System zu entwickeln.”

Wie bereits diskutiert, können unerwünschte Unit-Refraktionen unter Umständen verhindert werden, indem ein schnellerer Prozessor eingesetzt wird. Aber auch dieser Weg kann zu kritischen Kosten führen. Das gilt besonders für Systeme, die in grosser Stückzahl hergestellt werden (low cost products).

7.5.1 Notwendige Aufteilung in zwei Units

Wenn bei einer Unit ohne periodische Ereignisse die Zyklusbedingungen nicht erfüllbar sind, kann folgende in *Abbildung 85* dargestellte Aufteilung in zwei Embedded Units angestrebt werden.

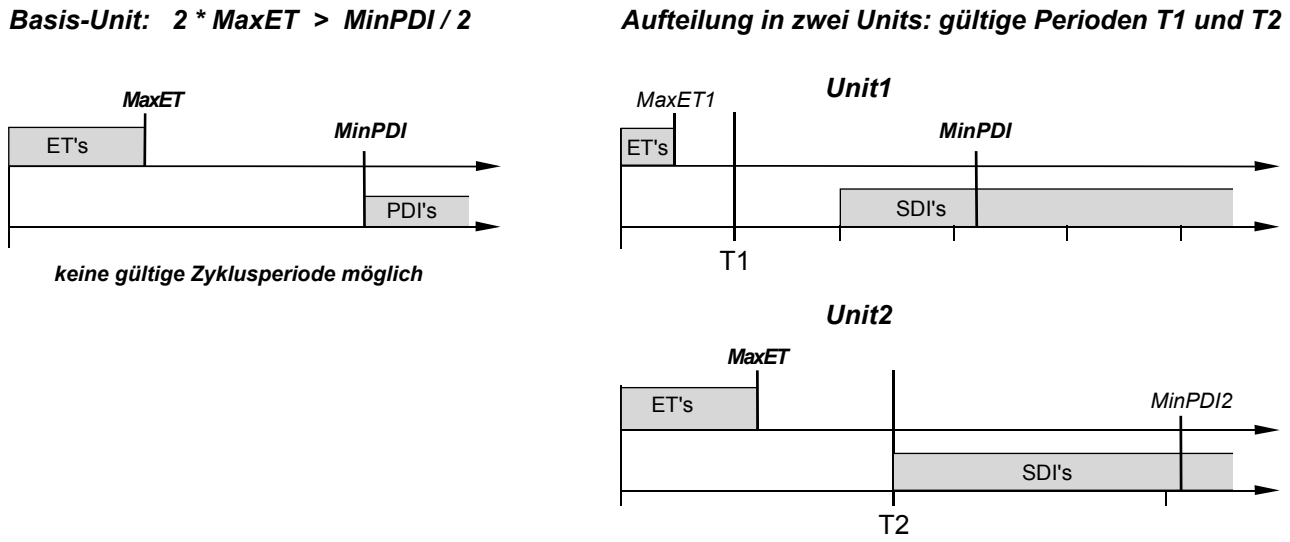

Abb. 85: Notwendige Aufteilung in zwei Units

Die vereinfachten Zyklusperiodenbedingungen der Basis-Unit in *Abbildung 85* können mit keiner Zyklusperiode erfüllt werden:

$2 * MaxET > MinPDI / 2$ verunmöglicht $T > 2 * MaxET$ und $T < MinPDI / 2$

(aus $T > 2 * MaxET$ *folgt* $T > MinPDI / 2$)

Die Aufteilung in Unit1 und Unit2 erfolgt so, dass folgende Bedingungen erfüllt sind:

Unit1: $2 * MaxET1 < MinPDI / 2$ und $MinPDI1 = MinPDI$

Unit2: $2 * MaxET < MinPDI2 / 2$ und $MaxET2 = MaxET$

Unit1 hat keine grossen Ausführungszeiten, dafür alle kleinen Deadline-Intervalle
Unit 2 hat keine kleinen Deadline-Intervalle. dafür alle grossen Ausführungszeiten.

Für *Unit1* gilt $2 * MaxET1 < MinPDI1$
und als Zykluszeit kann $T1 = 2 * MaxET1$ *gewählt werden*

Für *Unit2* gilt $2 * MaxET2 < MinPDI2 / 2$
und als Zykluszeit kann $T2 = MinPDI2 / 2$ *gewählt werden*

Die gezeigte Aufteilung ist nicht möglich, wenn z. B. für *dasselbe* Ereignis *e* gilt:

$e.wcET = MaxET$ und $e.PDI = MinPDI$

Beim diesem Fall hat das Ereignis mit der grössten Ausführungszeit das kürzeste Deadline-Intervall.

Solche seltene Fälle müssen gesondert behandelt werden. Ein möglicher Ausweg, bei dem das Problem mit der Zykluslatenz T nicht auftritt, ist mit einem *sporadischen EventDetector* zu arbeiten, d. h. das Auftreten kritischer Ereignisse wird nach *jeder* Ereignisverarbeitung überprüft (vgl. Kap. 6). Das funktioniert, wenn die Deadline-Intervalle grösser als *MaxET* sind (non-preemptive Latenz).

7.5.2 Notwendige Aufteilung in drei Units

Eine Aufteilung in drei Units kann notwendig werden, wenn Ereignisse mit der Periode P verarbeitet werden müssen und folgende Bedingungen erfüllt sind:

$$2 * MaxET > P \quad \text{und} \quad P > MinPDI / 2$$

Implizit ist in dieser Bedingung auch die Bedingung für die Aufteilung in zwei Units enthalten:

$$2 * MaxET > MinPDI / 2$$

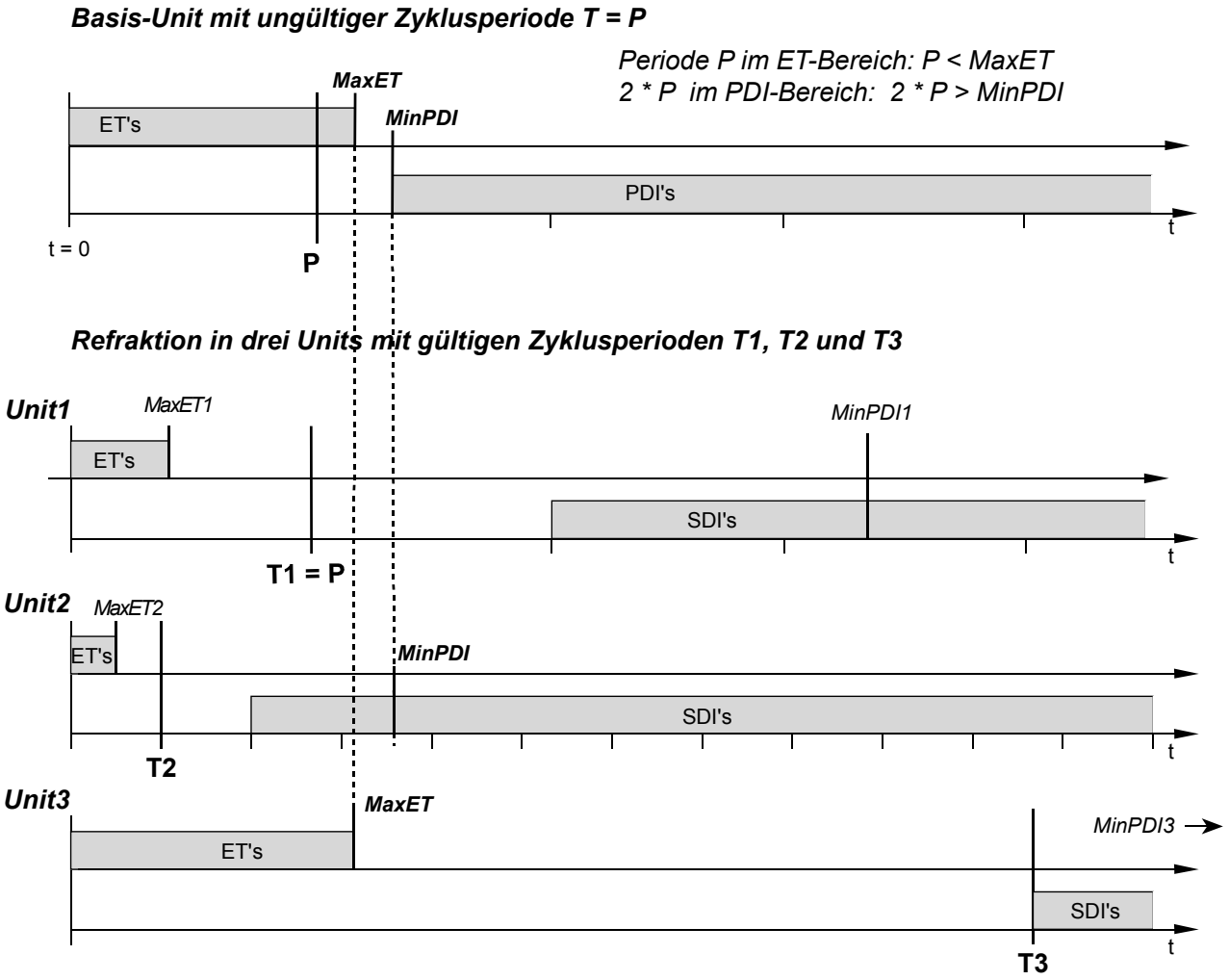

Abb. 86: Notwendige Aufteilung in drei Units

Unit 1 ist für die periodische Verarbeitung zuständig und hat Zyklusperiode T1 = P. Diese Unit kann auch sporadische Ereignisse verarbeiten, für welche die Zyklusperiode T1 zulässig ist.

Der restlichen Ereignisse werden durch *Unit2* und *Unit3* vererarbeitet. Hier erfolgt die Aufteilung nach demselben Muster wie im vorhergehenden Beispiel.

Warnung

Die vorgeführten Aufteilungen sind als Muster zu verstehen die zeigen sollen, wie Lösungen gefunden werden können. In jedem konkreten Fall muss zusätzlich sichergestellt werden, dass durch die Aufteilung nicht Antwortzeiten unzulässig vergrössert worden sind. Das kann zum Beispiel vorkommen, wenn ein Ereignis und eine von ihm zu bewirkende Aktion nicht mehr zur selben Unit gehören (Verzögerung durch asynchrones Message Passing).

Referenzen

1 Hugo Fierz, ”The CIP Method: Component- and Model-Based Construction of Embedded Systems”, European Software Engineering Conference 99, Lecture Notes in Computer Science 1687, Springer, Berlin, Germany, pages 374-391, September, 1999.

2 Hans Otto Trutmann, “Separate Connection and Functionality is the Pivot in Embedded System Design”, Computer Engineering and Networks Laboratory TIK, ETH Zurich, TIK-Schriftenreihe No. 39, Diss. ETH Zurich No. 13891, Oktober, 2000.

3 Michael Jackson, “Problem Frames: Analyzing and Structuring Software Development Problems”, Addison-Wesley, 2001.

4 K. H. Britton, R. A. Parker and D. L. Parnas, ”A Procedure for Designing Abstract Interfaces for Device Interface Modules”, Proc. of the Fifth International Conference on Software Engineering, p. 195-204, 1981.

5 H. O. Trutmann, “Well-Behaved Applications Allow for More Efficient Scheduling” 24th IFAC/IFIP Workshop on Real-Time Programming (WRTP99), Dagstuhl, Germany, pages 69-74, May, 1999.

6 J. Xu and K-Y. Lam, “Integrating Run-Time Scheduling and Pre-Run-Time Scheduling of Real-Time Processes“, 23rd IFAC/IFIP Workshop on Real-Time Programming, p. 73-80, Shantou, China, 1998.

7 Rodney R. Howell and Muralidhar K. Venkatrao, “On Non-preepmtive Scheduling of Recurring Tasks Using Inserted Idle Times“, Information and Computation 117, p. 50-62, 1995.

8 Gorgio C. Buttazzo. “Hard Real-Time Computing Systems, Predictable Scheduling Algorithms and Applications”, Springer New York, 2005.

9 Kevin Jeffay, Donald F. Stanat and Charkes U. Martel, “On Non-Preemptive Scheduling of Periodic and Sporadic Tasks“, Real-Time Systems Symposium 1991, p. 129-139, 1991.

10 Mark H. Klein et al., “A practitioner’s handbook for real-time analysis: Guide to rate monotonic analysis for real-time systems”, Kluwer Academic Publishers, Boston 1993

11 C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment”, Journal ofthe ACM 20(1), p. 40-60, 1973.

12 Giuseppe Lipari and Enrico Bini, “A methodology for designing hierarchical scheduling systems“, Journal of Embedded Computing 1, p. 257-269, IOS Press 2005.

13 Anton Cervin and Johan Eker, “Control-scheduling codesign of real-time systems: The control server approach”, Journal of Embedded Computing 1, p. 209-224, IOS Press 2005.

14 John A. Stankovic et al.,“Deadline Scheduling for Real-Time Systems”, Kluwer Academic Publishers, p. 31 (Feasability Analysis), London 1998.

15 S. K. Baruah, A. K. Mok and L. E. Rosier, “Premptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor“, Proc. 11th IEEE Real-Time Systems Symposium, p. 182-190, 1990.