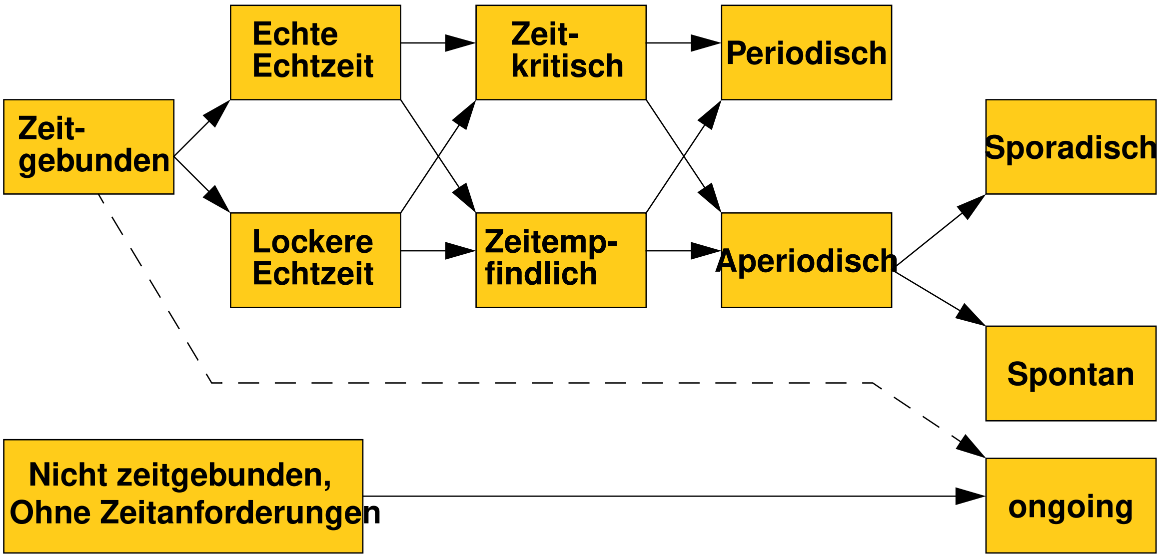
Real-Zeit Programmierung (mit public-domain Software)

Rechnerunterstützte Regelung und Steuerung wird in der Regel mit parallel laufenden Echtzeit (Real-Zeit) reaktiven Tasks implementiert. Tasks werden aktiviert als Reaktion auf Ereignisse, nach gegebenen Zeitpunkten oder können kontinuierlich laufen (ongoing). In diesem Kontext können die Echtzeit-Tasks folgendermaßen klassifiziert werden: (Sie können weitere Informationen zu diesen Themen unter **http://www.first.gmd.de/~sergio/public** finden).



Zeitgebunden / Nicht zeitgebunden Tasks

Die Korrektheit der zeitgebundenen Tasks, oder Tasks mit Zeitanforderungen, besteht nicht nur darin, einen korrekten Output-Wert zu generieren, sondern auch zum richtigen Zeitpunkt, und nicht einfach “so schnell wie möglich”. Sie haben ein Reaktionszeitfenster mit einer gewissen Toleranz. Das Ende der Zeitfenster ist die Deadline. Meistens wird im Echtzeit-Kontext nur von Deadlines gesprochen und der Anfang der Zeitfenster wird ignoriert. Dies kann zu Timing-Fehlern führen, die wiederum zu einem Anlagenausfall oder Versagen führen können.

Die Tasks ohne Zeitanforderungen (nicht zeitgebunden) brauchen nur ihre Ergebnisse zu liefern, aber der Zeitpunkt ist nicht maßgebend. Diese Tasks laufen deshalb im Hintergrund nur dann, wenn keiner der zeitgebundenen Tasks den Prozessor und andere konkurrierende Ressourcen braucht. Diese Tasks werden als ongoing Prozesse implementiert. Die ongoing Tasks können suspendiert und an derselben Stelle fortgesetzt werden, im Gegensatz zu den zeitgebundenen Tasks, die immer wieder an derselben Stelle (handle-Funktion) aktiviert werden.

Echte Echtzeit / Lockere Echtzeit

Bei den zeitgebundenen Tasks unterscheidet man je nach verfügbarer Zeit zwischen echter Echtzeit (real real-time), wo die Zeit sehr knapp gemessen ist, und lockerer

Sergio Montenegro, GMD-FIRST, Rudower Chaussee 5, D-12489 Berlin-Adlershof
Tel +49/30/6392-1878, email: sergio@first.gmd.de

Echtzeit, wo die Reaktionen mehr als genügend Zeit zur Verfügung haben.

Zeitkritisch / Zeitempfindlich

Die Echtzeit-Tasks können auch abhängig von ihrer Zeitkritikalität klassifiziert werden. Die zeitkritischen Tasks oder Tasks mit harten Echtzeitanforderungen haben feste Deadlines oder Reaktionszeitfenster, die nicht verpaßt werden dürfen. Wenn die Reaktion nicht in den vorgesehenen Reaktionszeitfenstern stattfindet, ist es ein Fehler, der zum Systemausfall führt.

Bei den zeitempfindlichen Tasks oder Tasks mit weichen Echtzeitanforderungen ist das Reaktionszeitfenster nicht so hart definiert, sondern so wie die Fuzzy-Mengen. Das Reaktionszeitfenster ist eher als eine Wunschzeit zu interpretieren.

Periodisch / Aperiodisch (sporadisch / spontan)

Die periodischen (zyklischen) Tasks führen eine Kontroll-Schleife aus, die in ein gegebenes Zeitfenster paßt. Die aperiodischen Tasks werden von asynchronen externen Interrupts (sporadische Tasks) oder von internen Ereignissen (spontane Tasks) aktiviert. Alle diese Tasks haben eine Initialisierungsfunktion, die einmal bei ihrer Erzeugung aufgerufen wird, und eine “handle”-Funktion, die bei jeder Aktivierung erneut aufgerufen wird. Die "handle"-Funktion muß ohne Verzögerungen durchlaufen und enden bevor sie wieder aufgerufen werden kann.

Scheduling (Zuteilung des Prozessors)

Der Echtzeit-Scheduler muß dafür sorgen, daß jeder Task rechtzeitig aktiviert (zugeteilt) wird, damit er sein Reaktionszeitfenster trifft. Um dies zu garantieren, ist eine Ausführungszeit-Abschätzung der Tasks nötig. Wenn bekannt ist, wie lange jeder Task braucht und wo jedes Reaktionszeitfenster liegt, kann festgestellt werden, wann jeder Task aktiviert werden soll. Im voraus kann analysiert werden, ob das System überhaupt realisierbar ist. Je mehr man über das Laufzeitverhalten des Systems weiß, um so sicherer kann der Scheduler gemacht werden und sogar die Zeit-Slots für die aperiodischen zeitkritischen Tasks können im voraus reserviert werden. Aber meistens ist eine Zeitreservierung für alle aperiodischen Tasks nicht möglich, es sei denn, der Prozessor ist sehr niedrig belastet. Darüber hinaus muß auch genügend Prozessor-Zeit frei bleiben, so daß die ongoing Tasks laufen können, besonders wenn sie zeitgebundene Aufgaben im Verborgenen durchführen. In diesem Fall muß eine Analyse gemacht werden, wieviel Ausführungszeit diese Tasks pro Zeiteinheit (z.B. Sekunde oder “**major Cycle**”) brauchen, um diese Zeit frei zu lassen. Wenn alle Laufzeitdaten bekannt sind, kann man einen Scheduler (Zuteilungsprogramm), der die Timing-Anforderungen garantiert, fest programmieren (ein statischer Scheduler). Wenn es nicht möglich ist, von Anfang an alle Tasks und ihr Verhalten zu kennen, muß ein dynamischer Scheduler angewendet werden, bei dem zur Laufzeit die Tasks angemeldet werden. Der Scheduler versucht dann, die dynamischen Timing-Anforderungen zu erfüllen, aber nur sehr wenig kann garantiert werden.

Wir haben einen experimentellen Echtzeit Java Scheduler entwickelt und bieten ihn als "Public Domain Software" unter **http://www.first.gmd.de/~sergio/public_domain** an.

Sergio Montenegro, GMD-FIRST, Rudower Chaussee 5, D-12489 Berlin-Adlershof
Tel +49/30/6392-1878, email: sergio@first.gmd.de