ulm university universität
uulm

STUDIENBRIEF

ENTWURFSMETHODIK EINGEBETTETER SYSTEME

Weiterbildender Masterstudiengang „Sensorsystemtechnik“
der Fakultät für Ingenieurwissenschaften und Informatik
mit dem Abschluss „Master of Science (M. Sc.)“
an der Universität Ulm

MODULINHALT

Kürzel / Nummer:	EES
Englischer Titel:	Design Automation of Embedded Systems
Leistungspunkte:	6 ECTS
Semesterwochenstunden:	4
Sprache:	Deutsch
Turnus / Dauer:	jedes Wintersemester / 1 Semester
Modulverantwortlicher:	Prof. Dr.-Ing. Frank Slomka
Dozenten:	Prof. Dr.-Ing. Frank Slomka
Einordnung des Moduls in Studiengänge:	Sensorsystemtechnik, M.Sc., Pflichtmodul,
Voraussetzungen (inhaltlich):	Grundlagen der Rechnerarchitektur oder Architektur Eingebetteter Systeme
Lernziele:	Im Modul werden systematische Methoden für den Entwurf eingebetteter Echtzeitsysteme vermittelt. Ziel ist es, auf der Grundlage von mathematischen Modellen ingenieurmäßig eingebettete Echtzeitsysteme zu entwickeln. Dieser modellbasierte Entwurf wird in Zukunft die Ad-Hoc-Programmierung eingebetteter Software ablösen. Teilnehmer des Moduls lernen die mathematischen Grundlagen zum Nachweis der Echtzeitfähigkeit eines eingebetteten Computers kennen und anwenden. In der Praxis werden diese Methoden den Teilnehmern in Form von Entwurfswerkzeugen begegnen. Diese Tools stellen die Methoden bereit und erlauben automatische Analysen. Um die Werkzeuge zu beherrschen und deren Ergebnisse kompetent deuten zu können, ist es unerlässlich, den theoretischen Unterbau und die zugrunde liegende Mathematik zu kennen. Diese wird in dem Modul vermittelt. Nach Abschluss des Moduls sind die Teilnehmer dazu in der Lage, den modellbasierten Entwurf eingebetteter Systeme zu beschreiben und zu skizzieren. Sie können außerdem unterschiedliche Analyseverfahren zur Bewertung eingebetteter Systeme benennen und auseinanderhalten. Sie wählen aus unterschiedlichen Methoden und Algorithmen zur Analyse des Echtzeitverhaltens die richtige Methode aus, um ein gegebenes Problem zu lösen. Sie sind in der Lage, neue Methoden und Algorithmen zu konstruieren und deren Korrektheit zu beweisen. Sie bestimmen die Komplexität der Algorithmen und können Approximationen entwickeln. Die Studierenden sind in der Lage, verschiedene Entwürfe eingebetteter Systeme zu bewerten und zu vergleichen. Diese Fähigkeiten erlauben es den Teilnehmern, einfache Berechnungen per Hand durchführen und komplexere Systeme mit kommerziellen Werkzeugen zu analysieren.
Inhalt:	- Übersicht über den modellbasierten Entwurf eingebetteter Systeme - Zeit und Echtzeitsysteme - Modellierung eingebetteter Systeme: Ereignismodelle und Graphen - Intrinsische Analyse von Echtzzeitsystemen - Extrinsische Analyse von Echtzzeitsystemen - Komplexität und Approximationen der extrinsischen Analyse - Optimierung und Entwurfsraumexploration

Literatur:	- Jürgen Teich: Digitale Hardware-/Software Systeme, Springer 1996 - Peter Liggesmeyer und Dieter Rombach: Software Engineering eingebetteter Systeme, Spektrum Akademischer Verlag 2005 - Jean J. Labrosse: Embedded Systems Building Blocks, CMP 2000 - Peter Marwedel: Eingebette Systeme, Springer 2007 - Zbigniew Michalewicz und David B. Fogel: Modern Heuristics, Springer, 2000
Lehrveranstaltungen und Lehrformen:	Präsenzveranstaltungen: Einführungsveranstaltung: 8 h Vertiefende Übungen: 8 h Seminar zur Prüfungsvorbereitung: 8 h Modulprüfung: 4 h E-Learning: Webinar: 4 h Online-Gruppenarbeit: 60 h Selbststudium: 80 h Chat zur Prüfungsvorbereitung: 8 h
Abschätzung des Arbeitsaufwands:	Vermittlung des Unterrichtsstoffs: 40 h Vor- und Nachbereitung, Übungen, Anwendung: 132 h Sonstiges: 4 h Modulprüfung: 4 h Summe: 180 h
Leistungsnachweis und Prüfungen:	Zur Modulprüfung wird zugelassen, wer die Aufgaben aus den Übungen erfolgreich bearbeitet hat. Die Modulprüfung erfolgt mündlich.
Voraussetzungen (formal):	Keine
Notenbildung:	Die Modulnote ergibt sich aus der Modulprüfung.

1. Einführung

Auch wenn man sie nicht sieht – eingebettete Computersysteme sind überall. Bemerkt werden sie häufig erst dann, wenn sie ihren Dienst quittieren: Plötzlich bleibt man mit dem neuen und teuren Automobil hilflos am Straßenrand stehen, Türen im Einkaufszentrum öffnen oder schließen pünktlich zum Feierabendeinkauf nicht mehr automatisch, die Schranke im Parkhaus verharrt in geschlossenem Zustand und die Anzeigetafel im Bus zeigt statt Ankunfts- und Abfahrtszeiten nur eine Fehlermeldung. Endlich daheim befindet das Thermostat in der Wohnung 14°C für ausreichend, der Kühlschrank schließt sich dieser Meinung an. Probleme mit eingebetteten Systemen sind leidgeprüften Benutzern alltäglicher Technologien wohlbekannt: Das Smartphone erreicht kein Funknetz oder hängt komplett, der MP3-Player verweigert das Abspielen des Lieblings-Albums auf der abendlichen Jogging-Runde.

Video
Ariane 5 Explosion in 1996

Neben Fehlern, die uns lieb gewordene Komfortfunktionen rauben, können fehlerhaft entworfene Systeme gefährlich für Leib und Leben und außerdem sehr teuer werden. So ist bereits vorgekommen, dass Raketen mit teuren Satelliten an Bord gesprengt wurden, weil sie von ihrer Flugbahn abgekommen sind und damit zur Gefahr wurden. Einer der kostenintensivsten Fehler in der Geschichte eingebetteter System resultierte in der Explosion einer Ariane-Rakete im Jahr 1996. Der Schaden in Höhe von über 370.000.000 $ wurde verursacht durch einen Variablenüberlauf. Dieser wiederum kam dadurch zustande, dass die Ariane 5-Rakete schneller beschleunigte als der Vorgänger Ariane 4.

Rechnergestützter Entwurf eingebetteter Systeme

Eingebettete System sind über die technischen Entwicklungen der letzten Jahrzehnte hinweg so komplex geworden, dass Methoden des rechnergestützten Entwurfs eingesetzt werden müssen. Dieses Skript behandelt die Modellbildung und die Analyse Eingebetteter Systeme. Fm Fokus steht der Entwurf eines einheitlichen Modells für ereignisgesteuerte Echtzeitsysteme.

1.1 Historische Entwicklung

In diesem Kapitel lernen Sie wichtige historische Entwicklungen eingebetteter Systeme kennen. Sie werden anhand historischer Meilensteine die Relevanz eingebetteter Systeme darstellen können. Die Bedeutung der Erfindung des Transistors und integrierter Schaltungen wird Ihnen bewusst.

Um Flugzeuge bei nasser Landebahn in der Spur zu halten, baute der Franzose Gabriel Voisin (*15.02.1880 - †25.12.1973) in von ihm hergestellte Flugzeuge in den späten 20er Jahren des 20. Jahrhunderts hydraulische Blockierverhinderer ein. Im Jahr 1928 erhielt Karl Wessel ein Patent für einen Bremskraftregler für Automobile, die Firma Bosch konstruierte und patentierte im Jahr 1936 eine Vorrichtung zum Vermeiden des Festbremsens der Räder eines Kraftfahrzeuges. Diese Systeme funktionierten mechanisch, bestanden aus etwa tausend Bauteilen und waren unhandlich, fehleranfällig und träge in ihrer Reaktion.

Entwicklung blockierverhindernder Systeme

Fernsteuerung mittels Funk

Im Zweiten Weltkrieg wurden die ersten ferngesteuerten Großraketen entwickelt. Das mit einem Team um Wernher von Braun entwickelte „Aggregat 4" (A4) erlangte unter dem Namen „V2" traurige Berühmtheit. Die A4 war mit einem Funksender versehen, um die Rakete vom Boden aus steuern zu können. Der Sender konnte angepeilt werden, mittels Kreuzpeilung war die Positionsbestimmung möglich, über Funk konnte dann die Flugbahn angepasst werden. Eine ebenfalls gängige Methode der Fernsteuerung war, die Lage von Raketen durch ein Kreiselsystem zu regeln: Mit einem Zeitgeber (Timer) konnte der Sollwinkel für die Lageregelung verändert und so eine Flugbahn programmiert werden. Bis zum Ende des Zweiten Weltkriegs war es nicht möglich, derartige Aufgaben von einem Computer ausführen zu lassen.

Fernsteuerung von Großraketen

Entwicklung programmierbarer Binärrechner

Einer der ersten programmierbaren Rechner, die Z1 von Konrad Zuse (*22.06.1910 - †18.12.1995) aus dem Jahr 1938, war ein elektromechanisches Ungetüm mit einer Masse von etwa einer Tonne Gewicht. Die Anlage beherrschte die vier Grundrechenarten und arbeitete mit einem Takt von ca. 1 Hz. Eine Addition benötigte etwa drei Sekunden. Die aus elektromechanischen Relais aufgebaute Zuse Z3 aus dem Jahr 1941 war der erste frei programmierbare Binärrechner mit Gleitkommaarithmetik. Der amerikanische Rechner ENIAC aus dem Jahr 1946 arbeitete mit 14468 Elektronenröhren, rechnete jedoch nicht binär. Für mobile Anwendungen ließen sich diese raumfüllenden Maschinen noch nicht einsetzen – dies sollte sich erst mit der Entdeckung des Transistoreffekts ändern.

Website
Konrad Zuse
Internet Archive

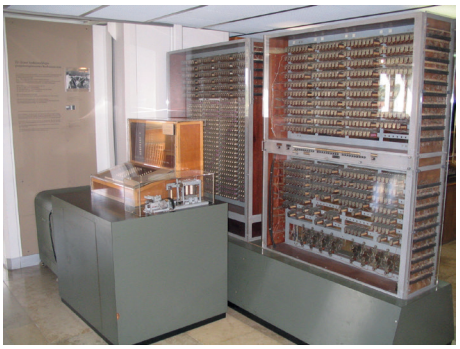

Abb. 1: Nachbau einer Zuse Z3 im Deutschen Museum (Quelle: Venusianer aus de.wikipedia.de, GFDL)

2. Modellgestützte Systemsynthese

Dieses Kapitel dient der Übersichtsbildung über die Elemente des Entwurfsablaufs und vollzieht inhaltlich die Abläufe in Abb. 22 nach. Die vorgestellten Elemente und Abläufe werden ab Kapitel 3 vertiefend behandelt.

Der systematische Entwurf eines eingebetteten Echtzeitsystems hat zum Ziel, ein möglichst kostengünstiges System zu erzeugen, das alle spezifizierten Randbedingungen einhält. Ein solcher Entwurf soll weitgehend automatisch erfolgen. Die folgende Abbildung zeigt einen typischen Entwurfsfluss für die automatische Systemsynthese.

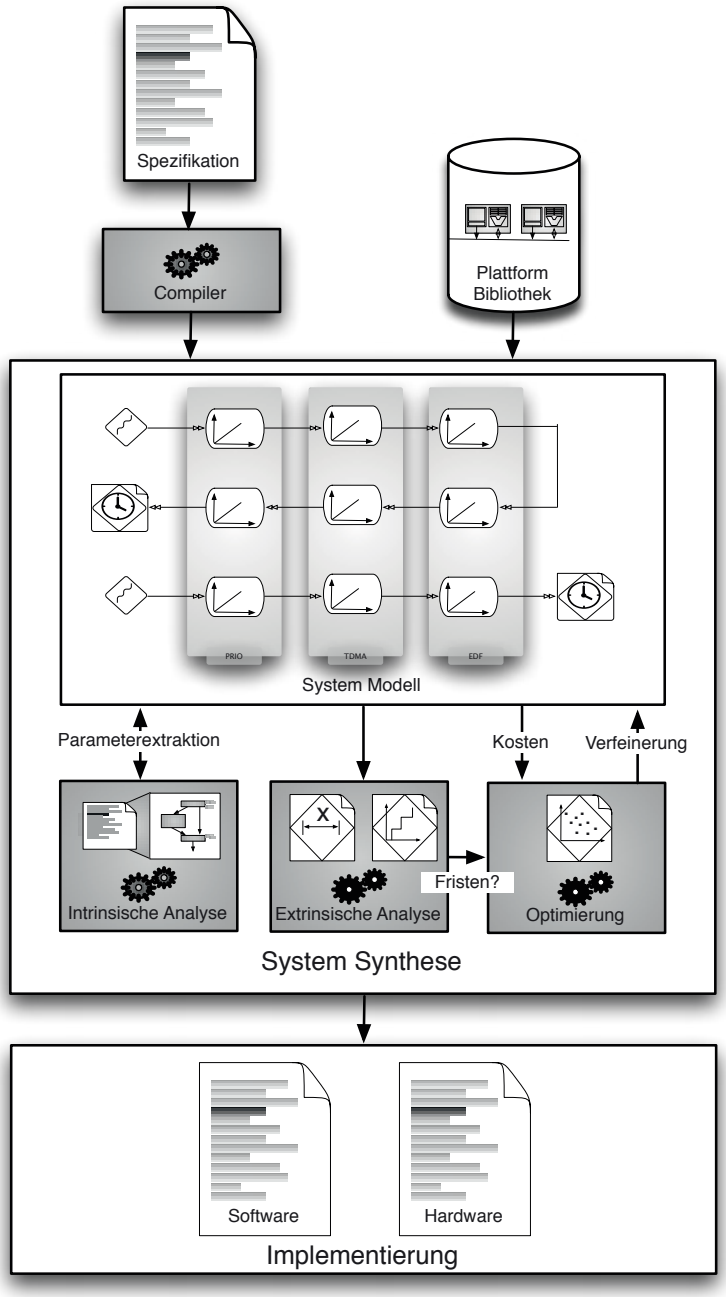

Abb. 22: Entwurfsablauf der automatischen Systemsynthese

Der folgende Text beschreibt den Entwurfsablauf der automatischen Systemsynthese, wie in Abb. 22 auf Seite 36 dargestellt.

Spezifikation

Der Entwurf eines Systems beginnt mit der Spezifikation. Diese kann unter Anwendung einer Spezifikations-, Programmier- oder Hardwarebeschreibungssprache erfolgen. Neben der reinen Funktion des Systems sind auch die Anforderungen an das System zu spezifizieren – z. B. Zeitfristen, maximal zulässiger Energieverbrauch oder maximale Systemkosten. Eine erstellte Spezifikation wird nun mit einem Übersetzungswerkzeug (Compiler) in ein Modell übersetzt. Dieses Modell enthält alle für die Systemsynthese relevanten Informationen. Aus einer Bibliothek können während der Synthese Bauelemente ausgewählt werden. Diese Bauelemente können Hardwarekomponenten wie Prozessoren und Speicher, aber auch Softwarekomponenten wie Mailboxen oder Scheduler sein. Bestimmte Modellparameter, etwa die Kosten eines Prozessors, können direkt in der Plattformbibliothek abgelegt sein. Andere Parameter, etwa die Ausführungszeit eines Programms, sind abhängig von der Programmspezifikation und dem ausgewählten Prozessor. Derartige Parameter müssen also von der Systemsynthese erzeugt werden. Dies geschieht mit der intrinsischen Analyse (intrinsisch = von innen her kommend). Diesen Schritt kann man auch als Parameterextraktion bezeichnen.

System Modell

intrinsische Analyse

Sind alle Parameter bestimmt und alle Systemkomponenten initial bestimmten Ausführungseinheiten (Prozessoren) zugeordnet, ist das resultierende System zu analysieren. Dabei ist zu ermitteln, ob die geplante Systemarchitektur die geforderten Randbedingungen auch tatsächlich einhält. Dies erfolgt mit der extrinsischen Analyse (extrinsisch = von außen her angeregt). Die extrinsische Analyse kann aus mehreren Werkzeugen bestehen: ein Werkzeug zur Analyse des Echtzeitverhaltens, ein Werkzeug zur Bestimmung des Energieverbrauchs oder aber eine Zuverlässigkeitsanalyse.

extrinsische Analyse

Optimierung

Nach der Analyse der Randbedingungen wird das Ergebnis der Analyse an ein Optimierungsverfahren weitergegeben. Dieses Optimierungsverfahren bestimmt aus dem Systemmodell die Kosten und ermittelt anhand der analysierten Randbedingungen aus der extrinsischen Analyse die Qualität des Entwurfs. Diese Entwurfsqualität wird in Relation zu anderen Entwürfen gesetzt. Existieren keine weiteren Entwürfe oder wurde keine ausreichende Qualität erreicht, kann durch Optimierung das Systemmodell verändert werden. Für ein auf diese Weise neu erstelltes Modell werden ggf. neue Modellparameter extrahiert, dann erneut mit der extrinsischen Analyse geprüft, ob die Randbedingungen eingehalten werden.

Iteration des Ablaufs

Dieser Ablauf wird solange wiederholt, bis eine optimale Systemarchitektur gefunden wurde. Danach kann aus der vorliegenden Systemarchitektur und der Spezifikation automatisch die Hardware/Software-Implementierung abgeleitet werden.

Erläutern Sie in eigenen Worten, welche Elemente eines Entwurfsablauf wie miteinander in Verbindung stehen.

Nach Bearbeitung des Abschnitts können Sie die Definitionen und die Modelle wiedergeben sowie ihre Mächtigkeit abschätzen. Sie können weiterhin anhand von Beispielen erklären, für welche Anwendungen sich ein bestimmtes Ereignismodell eignet.

5.2.1 Ereignissequenzen

Ein eingebettetes System ist mit einem technischen Prozess verknüpft. Wenn der eingebettete Rechner den Zustand des Prozesses überwacht, löst jeder Messvorgang ein Ereignis in der Rechenanlage aus. Für jeden Messpunkt wird entsprechend der Abtastrate des Systems daher im Lauf der Zeit eine Folge von Ereignissen produziert. Eine solche Folge von Ereignissen wird Ereignissequenz genannt.

Definition 26 – Ereignissequenz: Eine Ereignissequenz ist eine zeitliche Abfolge von Ereignissen und wird mit Θ bezeichnet.

Definition Ereignissequenz

In Abhängigkeit von der Zeit kann aus einer Ereignissequenz die Häufigkeit des Auftretens von Ereignissen bestimmt werden. Die Anzahl von Ereignissen pro Zeiteinheit soll Ereignisdichte heißen. Wird dabei das maximal mögliche Auftreten von Ereignissen beschrieben, spricht man von einer maximalen Ereignisdichte $\eta^{+}(t)$. Beschreibt man das minimal mögliche Auftreten von Ereignissen, spricht man von der minimalen Ereignisdichte $\eta^{-}(t)$.

Definition 27 – Begrenzte Ereignisdichte: Sei $[t_1,t_2]\ mit\ t_1,t_2 \in \mathbb{R}^{+}$ ein Zeitintervall der Länge $t = t_2 - t_1$ und Θ eine Ereignissequenz, dann bestimmt die begrenzte Ereignisdichte $\eta: \mathbb{R}^{+} \longrightarrow \mathbb{N}$ die Summe aller in diesem Zeitintervall aufgetretenen Ereignisse:

$$\eta(\Theta,t) := \begin{cases} 0 & falls\ t < 0 \\ n(t) & falls\ t \geq 0 \end{cases}$$

mit $\eta(t)$ der Anzahl der im Interval $[t_1,t_2]$ auftretenden Ereignisse. Dabei sind ausdrücklich auch Ereignisse zu den Zeitpunkten t_1 und t_2 mit eingeschlossen.

Definition der begrenzten Ereignisdichte

Was ist eine Ereignissequenz in einem eingebetteten System?

Angenommen, ein Sensor produziert eine regelmäßige Ereignissequenz. Dabei wird im festem Abstand von T ein Ereignis erzeugt. Die Anzahl der Ereignisse lässt sich mit der Funktion $f_n(t) = t/T$ bestimmen. Allerdings fordert die Definition der begrenzten Ereignisdichte, dass $n[t_1,t_2]$ ganzzahlig ist. Das bedeutet, dass die Funktion $\eta(\Theta,t)$ die Form $n(t) = \lceil f_n(t) \rceil$ oder $n(t) = \lfloor f_n(t) \rfloor$ haben muss, also auf- oder abgerundet wird.

Ganzzahligkeit durch Auf- und Abrunden

Tritt nun zum Zeitpunkt 0 bereits ein Ereignis auf, liefert $n(t) = \lceil t/T \rceil$ ein falsches Ergebnis $(n(0) = \lceil 0/T \rceil = 0)$. Dies gilt auch für $n(t) = \lfloor t/T \rfloor$. Gelöst wird das Problem mit der Funktion $n(t) = \lfloor t/T + 1 \rfloor$. Dann ergibt sich $n(0) = \lfloor 0/T + 1 \rfloor = 1$.

Begrenzende Ereignisdichten können eine Sequenz von Ereignissen nach oben oder nach unten begrenzen. Daraus ergibt sich, dass eine maximale Dichte existiert, also kein Ereignisabstand in der Ereignissequenz zu einer höheren Dichte führt. Dies gilt analog für die minimale Dichte.

Definition maximale und minimale Ereignisdichte

Definition 40 – Maximale und minimale Ereignisdichte: Die aus einer Ereignissequenz ermittelte Ereignisdichte ist stets kleiner als die maximale sowie stets größer als die minimale Ereignisdichte:

$$\eta^{+}(t) \geq \eta(t) \geq \eta^{-}(t)$$

In bestimmten Fällen kann es vorkommen, dass ein Ereignis am Ende des Intervalls t nicht mitgezählt werden soll, demnach das Intervall [t1; t2) betrachtet wird. Ein solcher Fall tritt ein, wenn etwas zum Zeitpunkt t2 beendet ist und das zu diesem Zeitpunkt auftretende Ereignis keine Rolle mehr spielt. Die nachfolgende Abbildung visualisiert diesen Aspekt:

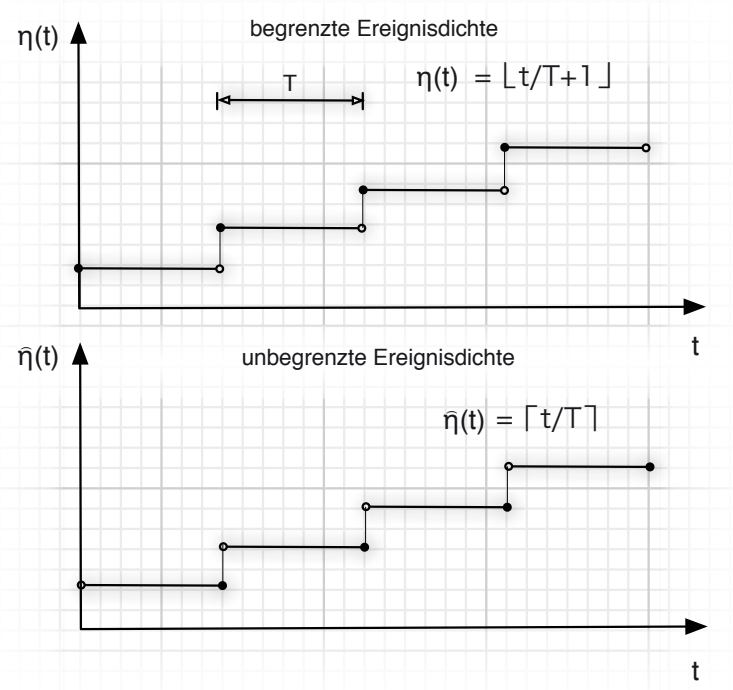

Abb. 38: Begrenzte und unbegrenzte Ereignisdichte

Definition der unbegrenzten Ereignisdichte

Definition 28 – Unbegrenzte Ereignisdichte: Sei $[t_1,t_2]$ mit $[t_1,t_2) \in \mathbb{R}^{+}$ ein Zeitintervall der Länge $t = t_2 - t_1$ und Θ eine Ereignissequenz, dann bestimmt die unbegrenzte Ereignisdichte $\eta: \mathbb{R}^{+} \rightarrow \mathbb{N}$ die Summe aller im Zeitintervall $[t_1,t_2-\varepsilon)$ aufgetretenen Ereignisse, unter der Annahme der Grenzwertbetrachtung $\varepsilon \rightarrow 0$:

$$\breve{\eta}(\Theta,t) := \lim_{\varepsilon \rightarrow 0} (\eta(\Theta,t) - \varepsilon)$$

Damit ist die unbegrenzte Ereignisdichte als linksseitiger Grenzwert der begrenzten Ereignisdichte definiert. Im Folgenden soll immer die begrenzte Ereignisdichte gemeint sein, wenn der Ausdruck Ereignisdichte verwendet wird.

Worin unterscheiden sich die begrenzte und unbegrenzte Ereignisdichte?

7.3.3 Approximation durch Superposition

Da gemäß Definition die Rechenzeitdichte in Teilfunktionen zerlegt werden kann, ist es möglich, erst die Rechenzeitdichte für jede einzelnen Task zu approximieren und die resultierende approximierte Rechenzeitdichte wieder aus den einzelnen Funktionen zusammenzusetzen (ALBERS und SLOMKA (2004)):

$$\delta^+(\Delta t,\Gamma) = \sum_{\tau\in\Gamma}\delta^+(\Delta t,\tau) \qquad (173)$$
$$\delta^+(\Delta t,\Gamma) = \left\lfloor \frac{\Delta t - d_\tau}{p_\tau} + 1 \right\rfloor c_\tau^+$$

Bei der Betrachtung einer Task kann auf die Schreibweise p_τ, c_τ^+ und d_τ verzichtet werden. Im Folgenden soll p, c^+, d äquivalent dazu sein. Die Rechenzeitdichte soll approximiert werden. Dazu wird die stufenförmige Funktion nach k Testschritten durch eine Gerade angenähert. Die Steigung der Gerade wird durch die Auslastung $u = \frac{c^+}{p}$ der Task bestimmt. Die folgende Abbildung zeigt dies.

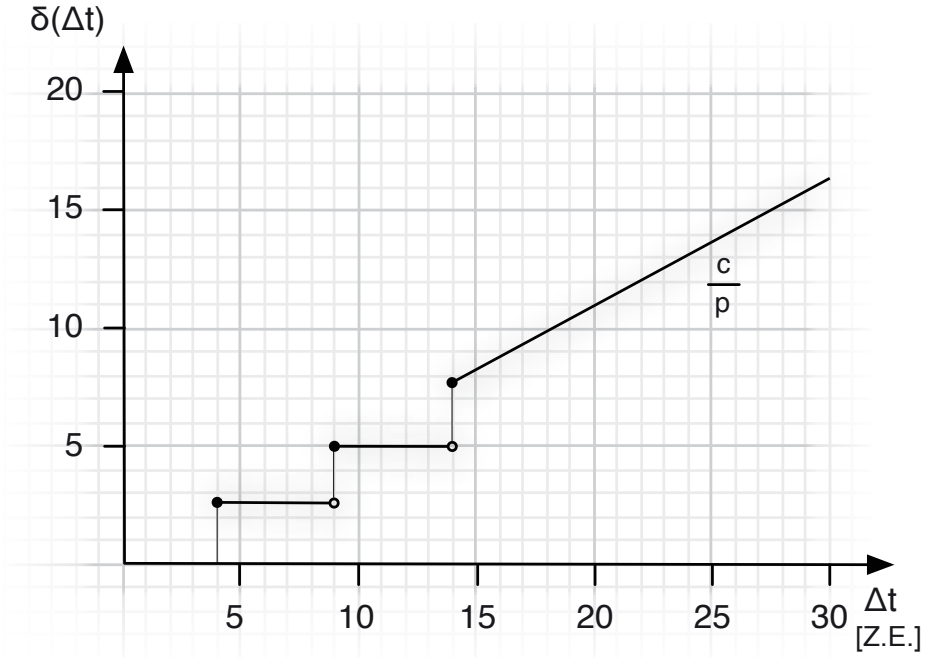

Abb. 70: Approximierte Rechenzeitdichte einer Task

Definition 47 – Approximierte Rechenzeitdichte einer Task: Die Rechenzeitdichte einer Task kann für k Testpunke exakt berechnet werden und wird dann durch die mittlere Auslastung der Task approximiert:

$$\delta'^+(k,\Delta t,\tau) = \begin{cases} \delta^+(\Delta t,\tau) & wenn\ \Delta t \le kp + d \\ \delta^+(kp+d,\tau) + \frac{c^+}{p}(\Delta t - (kp+d)) & sonst \end{cases}$$

Für die Approximation muss sichergestellt sein, dass $\delta'^{+}(k,\Delta t,\tau)$ für alle $\Delta t \geq 0$ stets größer als die ursprüngliche Funktion ist, so dass ein System, das den approximierten Test besteht, in jedem Fall auch den exakten Test bestehen würde. Im Folgenden wird $\delta'^{+}(k,\Delta t,\tau)$ durch $\delta'^{+}(\Delta t,\tau)$ abgekürzt.

Satz 18 – Gültigkeit der Approximation: Die exakte Rechenzeitdichte ist stets kleiner als die approximierte Rechenzeitdichte: Für alle Δt ist $\delta^{+}(\Delta t) \leq \delta'^{+}(\Delta t)$.

Beweis
Für $\Delta t \leq kp$ ist $\delta'^{+}(\Delta t,\tau) = \delta^{+}(\Delta t,\tau)$, daher muss nur gezeigt werden, dass für alle $\Delta t > kp + d$ gilt $\delta'^{+}(\Delta t,\tau) \geq \delta^{+}(\Delta t,\tau)$.

$$\delta'^{+}(\Delta t,\tau) = \delta^{+}(\Delta t_k,\tau) + \frac{c^{+}}{p}(\Delta t - \Delta t_k) \tag{174}$$

mit $\Delta t_k = kp + d$ dem Intervall mit dem exakt gerechnet wird. Dann gilt mit

$$\left\lfloor \frac{\Delta t - d}{p} + 1 \right\rfloor c^{+} \leq \left(\frac{\Delta t - d}{p} + 1 \right) c^{+} \tag{175}$$

folgendes:

$$\begin{aligned}
\delta^{+}(\Delta t,\tau) &\leq \left(\frac{\Delta t - d}{p} + 1 \right) c^{+} && (176)\\
\delta^{+}(\Delta t,\tau) &\leq \left(\frac{\Delta t - d + \Delta t_k - \Delta t_k}{p} + 1 \right) c^{+} \\
\delta^{+}(\Delta t,\tau) &\leq \left(\frac{\Delta t - d + \Delta t - \Delta t_k}{p} + 1 \right) c^{+} \\
\delta^{+}(\Delta t,\tau) &\leq \left(\frac{\Delta t_k - d}{p} + 1 \right) c^{+} + \frac{c^{+}}{p}(\Delta t - \Delta t_k)
\end{aligned}$$

Für $\Delta t_k = kp + d$ gilt:

$$\left\lfloor \frac{\Delta t_k - d}{p} + 1 \right\rfloor c^{+} = \left(\frac{\Delta t_k - d}{p} + 1 \right) c^{+} \tag{177}$$

daher ist:

$$\begin{aligned}
\delta^{+}(\Delta t,\tau) &\leq \left\lfloor \left(\frac{\Delta t_k - d}{p} \right) + 1 \right\rfloor c^{+} + \frac{c^{+}}{p}(\Delta t - \Delta t_k) && (178)\\
\delta^{+}(\Delta t,\tau) &\leq \delta^{+}(\Delta t,\tau) + \frac{c^{+}}{p}(\Delta t - \Delta t_k) \\
\delta^{+}(\Delta t,\tau) &\leq \delta'^{+}(\Delta t,\tau)
\end{aligned}$$

□

Definition 48 – Approximierte Rechenzeitdichte durch Superposition: Die Rechenzeitanforderungsdichte einer Taskmenge ist die Summe oder Überlagerung der approximierten Rechenzeitdichten der einzelnen Tasks der Taskmenge:

$$\delta'^{+}(\Delta t,\Gamma) = \sum_{\tau \in \Gamma} \delta'^{+}(\Delta t,\tau)$$

Ansprechpartner

Dr. Gabriele Gröger
Albert-Einstein-Allee 45
89081 Ulm

Tel 0049 731 – 5 03 24 00
Fax 0049 731 – 5 03 24 09

gabriele.groeger@uni-ulm.de
www.uni-ulm.de/saps

Postanschrift

Universität Ulm
School of Advanced Professional Studies
Albert-Einstein-Allee 45
89081 Ulm

Mod:Master
Sensorsystemtechnik

Das Studienangebot „Sensorsystemtechnik“ wurde entwickelt im Projekt Mod:Master, das aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert und aus dem Europäischen Sozialfonds der Europäischen Union kofinanziert wird (Förderkennzeichen: 16OH11027, Projektnummer WOH11012). Dabei handelt es sich um ein Vorhaben im Programm „Aufstieg durch Bildung: offene Hochschulen“.

AUFSTIEG DURCH BILDUNG OFFENE HOCHSCHULEN

EUROPÄISCHE UNION

ESF Europäischer Sozialfonds für Deutschland

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung