Prüfungsprotokoll Eingebettete Systeme
Professor Kowalewski

Einführung in eingebettete Systeme (V2, WS 03/04)
Embedded Software Design (V4, SS 04)
Automotive SWE (V2, WS 04/05)
Formale Methoden für eingebettete Systeme (V2, SS 05)
Dynamische Systeme für Informatiker (V2, WS 05/06)

30.03.2006

Matthias Stiefelhagen

Einführung in eingebettete Systeme:

Kow: Was ist ein eingebettetes System?

ich: Computersystem, integriert in einbettendes System. Anforderungen müssen vom einbettenden System abgeleitet werden. Keine oder nur sehr geringe Schnittstelle zum Benutzer, Produkt vs. Produktionssystem (mit Beispielen).

Kow: Was sind denn typische Aufgaben von eingebetteten Systemen?

ich: Sensoren abfragen und damit physikalische Variablen beobachten, überwachen und ggfs. über Aktoren beeinflussen (mit Beispiel: Temperatur im Tank regeln).

Kow: Wir hatten ja zwei Bereiche, kontinuierliche Regelung und diskrete Steuerung, was ist der Unterschied?

ich: Beide haben dieselbe Aufgabe, nämlich das System so zu beeinflussen, dass es sich wie gewünscht verhält. Bei der diskreten Steuerung bekommen wir aber diskrete Werte vom Sensor und beeinflussen die Strecke über diskrete Werte für die Aktoren (mit Beispiel).

Kow: Was hat man dann im Gegensatz dazu bei kontinuierlichen Systemen?

ich: kontinuierliche Variablen, zur Beschreibung von physikalischen Größen oder zur Annäherung von diskreten Signalen(Beispiel Verkehrsflüsse).

Kow: Das hatten wir auch bei diskreten Variablen, was war das?

ich: da approximieren wir kontinuierliche Signale durch diskrete. Beispiel Temperatur, eigentlich kontinuierlich, aber manchmal ist es nur interessant ob sie über oder unter einem bestimmen Grenzwert liegt.

Kow: Ok, kommen wir zu kontinuierlichen Systemen. Was macht man da wenn man diese beeinflussen möchte?

ich: feedback control oder feedforward control. An Zeichnung mit Beispiel Cruise Control erklärt (Sollwert: gewünschte Geschw., Ist-Wert: tatsächliche Geschw, Regeldifferenz bilden => Eingangssignal für den Regler. Dieser berechnet daraus die Stellgröße für das Stellglied (z.B. Benzinmenge wenn Stellglied = Kraftstoffpumpe), Stellglied wirkt dann auf die Strecke (umfasst Motor, Getriebe, alles was Geschw ausmacht)). Vorteile feedback-control: Berücksichtigung von Störgrößen, kann instabile Systeme stabilisieren (Bsp. invertiertes Pendel). Nachteile: Rückkoppelung kann Instabilität verursachen (z.B. Strecke hat Totzeit), teurer wegen Sensor.

Dynamische Systeme für Informatiker:

Kow: Was macht man wenn man jetzt so einen Regler wirklich einsetzen will?

ich: der am häufigsten eingesetzte Regler ist der PID-Regler, der oft genügt um die Anforderungen an geschlossene Regelkreise zu erfüllen (Stabilität, stationäre Genauigkeit, Dämpfung, Schnelligkeit). Konkret dann einstellen nach Ziegler/Nichols. 2 Methoden: 1. Strecke im geschlossenen Regelkreis nur mit P-Glied an den Stabilitätsrand bringen

(stabile Schwinung), Paramter für Tab. an Schwingung ablesen. Oft nicht möglich weil zu gefährlich. 2. Sprungantwort der Strecke betrachten, Sys. höherer Ordnung (Wendepunkt) durch PT1-Glied und Totzeitglied approximieren. Erklärt wo man die Zeitkonstante für PT1-Glied abliest.

Kow: Wie werden denn Totzeitglied und PT1-Glied verwendet, werden die in Reihe oder parallel geschaltet?

ich: in Reihe.

Kow: wie beschreibt man dynamische Systeme?

ich: Zustandsraumdarstellung als eine Möglichkeit

kont. Sys.: $\dot{x}(t) = f(x(t), u(t))$
$y(t)= g(x(t), u(t))$

diskrete Sys. $x(k+1) = f(x(k), u(k))$
$y(k) = g(x(k))$ Moore-Automat
$y(k) = g(x(k), u(k))$ Mealy-Automat

Kow: So etwas hatten wir auch bei kontinuierlichen Systemen.

ich: ja, wenn wir diskrete Zeit betrachten, dann haben wir dieselbe Darstellung x(k+1)...

Kow: Wofür brauche ich denn bei dynamischen Systemen überhaupt den Zustand?

ich: Weil dyn. Systeme, Systeme mit Gedächtnis sind. Wir können aus dem aktuellen Eingangssignal nicht direkt das Ausgangssignal bestimmen, Vergangenheit ist relevant (Beispiel gezeichnet). Zustand repräsentiert die benötigte Info über Vergangenheit

Embedded Software Design:

Kow: Ok, ich will jetzt nicht das V-Model im Detail besprechen. Wir hatten da ganz am Anfang die Phase in der die Anforderungsanalyse gemacht wird. Welche Arten von Anforderungen unterscheidet man hier?

ich: funktionale (was soll das System können) und nicht-funktionale bzw qualitative(wie gut soll es das können).

Kow: Was sind die Unterschiede?

ich: funktionale Anforderungen sind technisch orientiert, qualitative business orientiert. Da hat man dann z.B. Anforderungen wie Vermarktbarkeit, Zuverlässigkeit...

Kow: Was macht man jetzt wenn man die Architektur entwerfen möchte?

ich: Optimierungsproblem, finde die beste Lösung oder eine Lösung die gut genug ist (gemessen an den architectural drivers) unter den erlaubten Lösungen (von der funktionalen Spezifikation).

Kow: Was macht man konkret?

ich: Architekturmuster anwenden für die wichtigsten Qualitäten, bei Konflikten Kompromisse finden.

Kow: welche Architekturmuster kennen sie?

ich: z.B. um Wartbarkeit zu unterstützen, Keep changes local, konkret Information Hiding. erklärt.

Kow: Was ist denn ein Architekturmuster um Verfügbarkeit zu unterstützen?

ich: Redundanz, es gibt homogene und diverse Redundanz...

Kow: Wo kommt denn Redundanz zum tragen, in der Entwicklungs- oder zur Laufzeit?

ich: zur Laufzeit. wenn Komponten ausfallen(es tritt ein Fehler auf) können die zusätzlichen Komponenten die Aufgaben übernehmen. System ist fehlertolerant, da kein Ausfall auftritt obwohl es einen Fehler gab.

Kow: Was kann man denn in der Entwicklung machen?

ich: Ansatzpunkte sind Fehlervermeidung und Fehlererkennung. Wir hatten da Quantization Based Typing.. erklärt.

Kow: nochmal zur Redundanz. Wie kann man die Redundanz berechnen wenn wir die einzelnen Redundanzen haben (divers) und 2oo3?

ich: mit einer Tabelle. erklärt.

Kow: wofür steht hier eine Zeile?

ich: für eine bestimmte Situation, z.B. ein System funktioniert und zwei nicht.

Automotive SWE

Kow: Was passiert wenn mehrere Komponenten gleichzeitig auf den Bus schreiben?

ich: recessive bit wenn alle recessive bit schreiben, dominant sobald einer dominant bit schreibt. Bus wird beobachtet und die Komponenten hören auf zu schreiben sobald nicht ihr bit auf dem Bus anliegt. Am Ende kann so genau einer seine Nachricht versenden.

Kow: Was machen die anderen?

ich: die probieren nach einer Zeit nochmal auf den Bus zu schreiben. Zeit ist nicht genau festgelegt.

Formale Methoden für eingebettete Systeme

Kow: Wir hatten ja hybride Automaten und Echtzeitautomaten. Wo liegen da die Probleme mit der Erreichbarkeit?

ich: für allgemeine HA unentscheidbar, für rektanguläre Automaten semi-entscheidbar, für

Echtzeitautomaten entscheidbar. Eine Stoppuhr führt hier allerdings schon zur Unentscheidbarkeit. Praktisch hatten wir den Hytech-Alg, der nach wenigen Iterationen überläuft, da versucht wird die exakten Grenzen der erreichbaren Regionen zu ermitteln. Bei Annährung an z.B. 33 1/3 wird nicht gerundet und die Integer laufen über (Integer jeweils für Zähler und Nenner für die Grenze). Praktisch also nicht belastbar.

Kow: Was macht man in so einem Fall?

ich: Abstraktion durch eine einfachere Dynamik, z.B. rektanguläre Automaten. An Zeichnung erklärt. Vorteil: Menge der Verhalten des abstrahierten Systems ist Obermenge der tatsächlichen Verhalten => wenn bei der Erreichbarkeitsanalyse auf dem RA ein Zustand nicht erreichbar ist, so kann man sicher sein, dass er auch im Originalsystem nicht erreichbar ist

Fazit:

Prof. Kowalewski stellt sehr faire Fragen und lässt einen auch oft ausführlich antworten, ohne zu unterbrechen. Details sind meistens nicht so wichtig, ihm kommt es eher darauf an, dass die Ideen verstanden worden sind. Ausnahme ist hier vielleicht nur die Frage zum PT1- und Totzeitglied bei der PID-Reglereinstellung.