Prüfungs-Gedächtnisprotokoll
Diplomprüfung praktische Informatik

Prüfer: Professor Kowalewski
Datum: 04.11.2005
geprüfte Vorlesungen: Einführung in eingebettete Systeme
Automotive Software Engineering
Formale Methoden für eingebettete Systeme
Entwurf eingebetteter Software
Software-Qualitätssicherung und Projektmanagement (Lichter)
Note: 1,0
Prüfling: Marius Wolf (marvis@gmx.de)

Kowalewski: Womit möchten Sie anfangen?

Mit der Einführung in eingebettete Systeme.

OK. Was sind denn typische Aufgaben von eingebetteten Systemen

- Abfragen von Sensoren
- Steuern von physikalischen Variablen über Aktoren
- Produkt vs. Produktionssystem

Sie haben hier schon die Unterteilung zwischen Produkt- und Produktionssystem angesprochen. Wo sind denn da die Unterschiede in Entwicklung und Anforderungen?

- verwendete Programmiersprachen
- Sicherheitsanforderungen
- andere Benutzer (trainiertes Personal bei Produktionssystemen)

Hier hat er noch ein wenig nachgebohrt und wollte auf Kosten als treibenden Faktor hinaus. Ich bin nicht direkt drauf gekommen, hab noch ein paar andere Punkte genannt, die alle richtig waren, aber letztlich hats dann doch geklingelt.

Was für Typen von eingebetteten Systemen unterscheidet man denn?

- Diskret vs. kontinuierlich, dichte Werte-/Zeitachsen vs. einzelne Werte

Wie kann man denn so ein kontinuierliches Regelsystem darstellen?

- Diagrammdarstellung aufgemalt (Sollwert, Controller, Strecke, Störgrößen)
- Unterschied zwischen Feed Forward und Feedback Control am Diagramm erklärt
- Vor- und Nachteile aufgezählt (Reagieren auf Störgrößen vs. potentielle Instabilität, die sich aufschaukeln kann)
- mögliche Realisierung durch PID-Controller erwähnt

Wie funktioniert denn eine speicherprogrammierbare Steuerung (SPS)?

Hier wollte er auf das Kreisdiagramm (Eingabe Lesen, Programm verarbeiten, Ausgabe) hinaus. Wir haben dann noch über die längstmögliche Antwortzeit (2*Zykluslänge) gesprochen und er wollte noch wissen, was denn der Vorteil von diesem Modell ist. Ich bin zuerst nicht drauf gekommen, aber kam dann doch auf die Probleme mit den

Echtzeitanforderungen, die wir etwa beim Einsatz von Interrupts bekämen. In dem bestehenden SPS-Modell ist die Antwortzeit ja ziemlich klar.

(Übergang zu Automotive Software Engineering)

Was sind denn die wesentlichen Merkmale bei der Entwicklung von Software für Autos? Wo liegen die Unterschiede zu anderen Systemen?

- gesetzliche Anforderungen, die auch noch in verschiedenen Ländern unterschiedlich sind (Produktlinienentwicklung erwähnt)
- Implementierung des V-Modells als meistbenutztes Prozessmodell (sehr viele Aufspaltungs- und Integrationsphasen)
- Beziehung zwischen Autoherstellern und Zulieferern (Autohersteller in sehr starker Position)

Es gibt da ja ein Betriebssystem, das speziell für Autos konzipiert ist. Was wissen Sie darüber?

- OSEK (hier war er erfreut, dass ich wusste, dass das für “Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen” steht)
- Komponenten (Programme bestehen aus C-Code und OSEK-OIL)
- Echtzeitbetriebssystem

Was bedeutet es denn, dass das ein Echtzeitsystem ist?

- Echtzeitanforderung erklärt (garantierte Antwortzeiten)

Hier hat er noch gefragt, ob ich wüsste, was eine weiche Echtzeitanforderung ist, aber da das in der Vorlesung nicht explizit behandelt worden war war es auch nicht schlimm, dass ich das nicht wusste.
Anschliessend ging es noch um die statische Eigenschaft von OSEK, also dass Tasks beim Systemstart schon angelegt sein müssen.

Was kann denn passieren, wenn sich mehrere Tasks begrenzte Resourcen teilen müssen?

- Starvation
- Deadlocks (hier bin ich zuerst nicht drauf gekommen, da es mir zu offensichtlich erschien; das war ja auch in der Vorlesung nicht nochmal explizit behandelt, da es in Systemprogrammierung im Grundstudium schon behandelt wurde)
- Priority Inversion (sollte ich dann erklären)

(Übergang zu formalen Methoden)

Wozu brauchen wir denn überhaupt formale Methoden? Wozu modelliere ich Systeme und wie kann ich das machen?

Hier ging es zunächst mal um Modelchecking (grundsätzliches Prinzip erläutert), wobei er noch erwähnt hat, dass man die Korrektheit von Systemen auch beweisen kann, was aber in der Vorlesung nicht wirklich behandelt wurde. Ich habe dann verschiedene Modelle erwähnt, die man zur Modellierung von Systemen verwenden kann (LTL/CTL, Büchi-Automaten, hybride- und Echtzeitautomaten).

Wie funktioniert denn CTL-Modelchecking so grob?
Hier wollte er eigentlich nur wissen, dass man mit den atomaren Formeln anfängt und dann die Kripke-Struktur sukzessiv von innen nach außen (auf die Formel bezogen) markiert. Ich hab dann noch erwähnt, wie das bei einer “Until”-Aussage geht (Rückwärtssuche), aber das wäre schon nicht mehr nötig gewesen.

Das klingt ja eigentlich nach 'ner tollen Sache...ich spezifiziere mein System und eine Anforderung und kann dann alles mögliche darüber aussagen. Warum macht man das nicht immer?

Hier wollte er darauf hinaus, dass der Aufwand für die Modellierung von realen Anwendungen einfach zu groß ist, um das für alles zu machen. Ich hab das dann am Beispiel einer Textverarbeitung erklärt, die den vertretbaren Aufwand schon total sprengen würde.

Das funkioniert ja alles nur für diskrete Systeme. Wie ist das mit hybriden Systemen?

- hybride Automaten Syntax und Semantik erläutert

Was kann man denn da bezüglich Erreichbarkeit aussagen?

Hier ging es zuerst darum, was überhaupt entscheidbar ist (Entscheidbar für Echtzeitautomaten, Unentscheidbar für allgemeine HA, partiell entscheidbar für RA) und dann darum, was es in der Praxis für Probleme gibt; hier dürften theoretische Erreichbarkeitsüberlegungen eher in den Hintergrund treten vor der Tatsache, dass die Rechner vor der Komplexität der entsprechenden Verfahren in die Knie gehen.

(Übergang zu Entwurf eingebetteter Software)

Welche Möglichkeiten zur Modellierung gibt es denn hier?

- synchronisierte Automaten
- asynchron kommunizierende Automaten
- Petri-Netze

Es wurde noch mehr gefragt, aber hier lässt mich mein Gedächtnis leider im Stich. Das war aber auch schon später in der Prüfung und Prof. Kowalewski schien so langsam zum Ende kommen zu wollen.

(Übergang zu Software-Qualitätssicherung)

Hier ging es noch kurz um Metriken (Definition von Metriken, zyklomatische Zahl, LCOM, Tiefe von Vererbungsbäumen), die Lichter-Definition von Qualität (Prof. Kowalewski hatte das in seinen Vorlesungen eher als Nicht-Funktionale Anforderung definiert, Prof. Lichter blieb eher an der ISO-Definition) und um Black-Box und White-Box-Testen. Bei letzterem wollte er wissen, was für das Testen allgemein so nötig ist (ich bin erst relativ spät auf “Testfälle” gekommen, obwohl das eigentlich klar ist). Dieser Teil wurde aber relativ kurz abgehandelt, da wir schon bei 45 Minuten waren.

Insgesamt habe ich Prof. Kowalewski als sehr angenehmen Prüfer erlebt. Er stellt präzise Fragen, hilft aber auch, wenn man nicht direkt auf die Antwort kommt. Er erwartet allerdings schon ausführliche Antworten und ich hatte den Eindruck, dass es bei ihm

schon ein klarer Pluspunkt ist, wenn man zeigt, dass man noch mehr weiss als das, wonach er explizit fragt (etwa indem man an einigen Stellen noch verwandte Themen erwähnt).

Als Hinweis sei noch genannt, dass er diese Fächerkombination wohl nicht mehr als Prüfung in praktischer Informatik machen will, da ihm das thematisch zu nah beeinander liegt und so quasi eine zweite Vertiefungsprüfung ist. Insofern möchte ich hier nochmal darauf hinweisen, dass man seine Prüfungsthemen immer frühzeitig mit dem entsprechenden Prüfer absprechen sollte, aber das sollte ja eigentlich eh selbstverständlich sein. :-)