CiAO - Projektplanung bis März 2005

Daniel Lohmann

Revision: 1.0
11. November 2004

Zusammenfassung

Dieses Dokument ist als Diskussionsgrundlage und Dokumentation der kurfristigen Projektplanung für CiAO gedacht. Es beschreibt Ziele, Aufgabenpakete und (irgend wann mal) Zuständigkeiten. Zunächst mal sollen vor allem Aufgabenpakete definiert, priorisiert und zugewiesen werden.

1 Ziele

Bis etwa Ende März 2005 stehen folgende Ziele an:

1. **ACP4IS Paper (Deadline Mitte Januar)**
   Schwerpunkt des *Aspects, Components and Patterns for Infrastructure Software* AOSD-Workshops ist dieses Jahr *Produktlinien*. Da wollen wir was in Richtung *Konfigurierbare Architekturmerkmale* einreichen. Daniel hatte schon mal angefangen sich Gedanken über ein konfigurierbares IRQ-Subsystem zu machen, welches ein einheitliches Architekturmodell für z.B. Treiber bereitstellen soll. Zu diesem Thema soll das Paper geschrieben werden.

2. **AOSD-Demo (Deadline Ende Februar)**
   Die AspectC++-Demo auf der AOSD 2005 soll die wohlbekannte Wetterstation sein. Wir streben jedoch einen höheren Grad an

Aspektorientierung a,n und AspectC++ soll gegenüber dem Produktlinienansatz stärker in den Vordergrund rücken. Aus diesen Gründen soll die neue Demo auf einem minimalen CiAO für AVR beruhen (ereignisgesteuert).

3. **OSE-Veranstaltung im nächsten Semester (Deadline Ende März)**
   Olaf bietet im nächsten Sommer die OSE-Vorlesung, evtl. mit Übung/Praktikum an. Als Zielplattform für das Praktikum sind die Mindstorm-Roboter vorgesehen. Dafür wird ein minimales CiAO für H8 und ein halbwegs erprobter Tool-Chain gebraucht.

Ziel 1 ist auf der Entwurfsebene angesiedelt, Ziele 2 und 3 auf Implementierung und Tools. Ziele 2 und 3 sind außerdem in hohem Maße deckungsgleich. Insgesamt entscheiden wir uns mit diesen Zielen für einen Bottom-Up Ansatz, sowohl bei der Implementierung als auch beim Entwurf.

2 Aufgaben und Arbeitspakete

2.1 Aufgaben und Arbeitspakete für Ziel 1 (ACP4IS-Paper, IRQ-Subsystem)

2.1.1 Domänenanalyse

Eine Woche

Wosch nach XP-Sourcen fragen

Untersuchung der Domäne *Interruptverarbeitung* in bestehenden BS. Wie genau läuft die IRQ-Bearbeitung und die Ausführung des zugehörigen Treibercodes in z.B. Linux, Solaris, NT (als Beispiele für große Systeme), in L4, OS9 (als Beispiele für Mikrokernsysteme) und in Pure, ECOS, OSEK (als Beispiele für kleine Systeme). Können wir dies auf das CiAO-Modell abbilden? Hierfür ist es insbesondere erforderlich, detailierte Informationen zur Implementierung von Interrupts in den genannten Systemen zu beschaffen.

Ergebnissoll ein Featuremodell für die Domäne Interruptverarbeitung sein sowie eine Menge von Feature-Selektionen, die den Funktionalitäten der o.g. Systeme entsprechen.

2.1.2 (Genauere) Beschreibung / Spezifizierung des CiAO-Modells

Eine Woche

Das (teilweise bereits bestehende) Modell soll mit den Ergebnissen der Domänenanalyse vervollständigt werden. Wie lässt sich die Verarbeitung von Treibercode durch Interrupts verallgemeinern? Welche Gesetzmäßigkeiten können garantiert, welche Funktionalitäten angeboten werden?

Ergebnis soll ein Dokument sein, welches (informal) die Architektur (z.B. dreischichtige Verarbeitung: Handler, AST, Thread) und Spezifikation beschreibt (welche Ressourcen stehen in welcher Schicht zur Verfügung, welche können ggfs. nachgefordert werden, welche Synchronisationsbedingungen gelten etc.). Dieses beschreibt quasi die angebotene Schnittstelle (semantisch) für architekturtransparente Treiber.

2.1.3 Erste Überlegungen zur Implementierung

Anfang Januar

Eine Woche

Wie lässt sich das Modell skalierbar und konfigurierbar implementieren? Wie sähe die Transformation (z.B. der ein bis drei Verarbeitungschichten) von Treibercode aus für ein System wie PURE oder L4? Es muss untersucht werden, wo sich hier Aspekte gut einsetzen lassen und wo andere Techniken.

Ergebnis soll ein Beispiel für einen (Pseudocode-)Treiber sein, der gegen das transparente Modell entwickelt wurde und sich an das jeweilige Zielsystem anpassen lässt. Ergebnis sollte auch ein Erkenntnisgewinn bzgl. der Implementierungstechniken für die erforderliche Variabilität (Aspekte, Templates, Type-Aliasing, etc.) sein.

2.1.4 Schreiben des eigentlichen Papers

Eine Woche

Mitte Januar

Ergebnis soll ein Paper sein das aufzeigt, wie sich mit Hilfe von Aspekten und ggf. anderen Techniken eine BS-Komponente (Treiber) an kon-

figurierbare *Architektur*merkmale einer BS-Produktlinie anpassen lässt.

2.2 Aufgaben und Arbeitspakete für Ziele 2 und 3 (AOSD-Demo, OSE)

Ca. drei Wochen
bis einschließlich 2.2.4

Implementierung eines minimalen CiAO, das auf AVR/H8 bootet und als Funktionalität wenigstens Interruptverarbeitung anbietet.

2.2.1 Tool-Chain für Mini-CiAO aufsetzen

Georg hat im Rahmen seiner Diplomarbeit bereits große Teile eines geeigneten Tool-Chains aufgesetzt. Dieser muss wahrscheinlich nur noch geringfügig erweitert werden. Letztlich geht es hier aber auch um einen Wissenstransfer. Folgende Unteraufgaben stehen an:

- Subversion installieren, testen und einbinden.
- Erstellen eines Variance-Modells mit konfigurierbarer Zielplattform und konfigurierbarer Interruptunterstützung.
- Build-Chain anpassen und testen
- Aufsetzen der Hard- und Software zum Flashen der Endgeräte
  Rechner in Bello-Labs ▷ Andi fragen

Ergebnis soll eine funktionierende Tool-Umgebung sein, mit der es möglich ist ein CiAO „Hello World“ zu bauen und auf die Endgeräte zu bringen.

2.2.2 Implementierung Startup-Code

Implementierung der entsprechenden Systemstart-Funktionen / Linker-Scripts etc.

Ergebnis soll ein minimales System sein, dass in main() startet.

2.2.3 Implementierung eines Initalisierungs-Subsystems

Ergebnis soll ein Mechanismus sein, der es einfach (z.B. durch Aspekte) erlaubt, transparent Initialisierungsreihenfolgen zu definieren.

2.2.4 Implementierung eines einfachen IRQ-Subsystems

Anfang Januar (Sync. mit 2.1.3)

Ergebnis soll ein IRQ-Subsystem sein, das bereits ein bis zwei Konfigurationsmöglichkeiten des architekturtransparenten Interrupt-Modells unterstützt (z.B. nur Handler, Handler + AST)

Eine Woche

2.2.5 Implementierung der Treiber/Anwendung für die Wetterstation

Ergebnis soll eine laufende Wetterstation sein, die in verschiedenen Konfigurationen generiert werden kann.

- iolib
- interrupts
- keine Threads!

2.3 Weitere Aufgaben und Arbeitspakete für Ziel 3 (OSE)

2.3.1 Implementierung der Treiber/Geräteansteuerung für Mindstorm

Ergebnis soll eine einfache Anwendung sein, die sicherstellt, dass CiAO auf den Mindstorms funktioniert.

2.3.2 Definition von Projektthemen

Ergebnis soll eine Liste von CiAO-Subsystemen / Modulen sein, die von den Studenten im Rahmen eines Projekts entwickelt werden können.

Vorgabe könnten dabei bereits implementierte, schwach konfigurierbare Subsysteme sein (▷ alle können arbeiten), die von den einzelnen Gruppen dann erweitert werden (Domänenanlyse, konfigurierbare Implementierung).