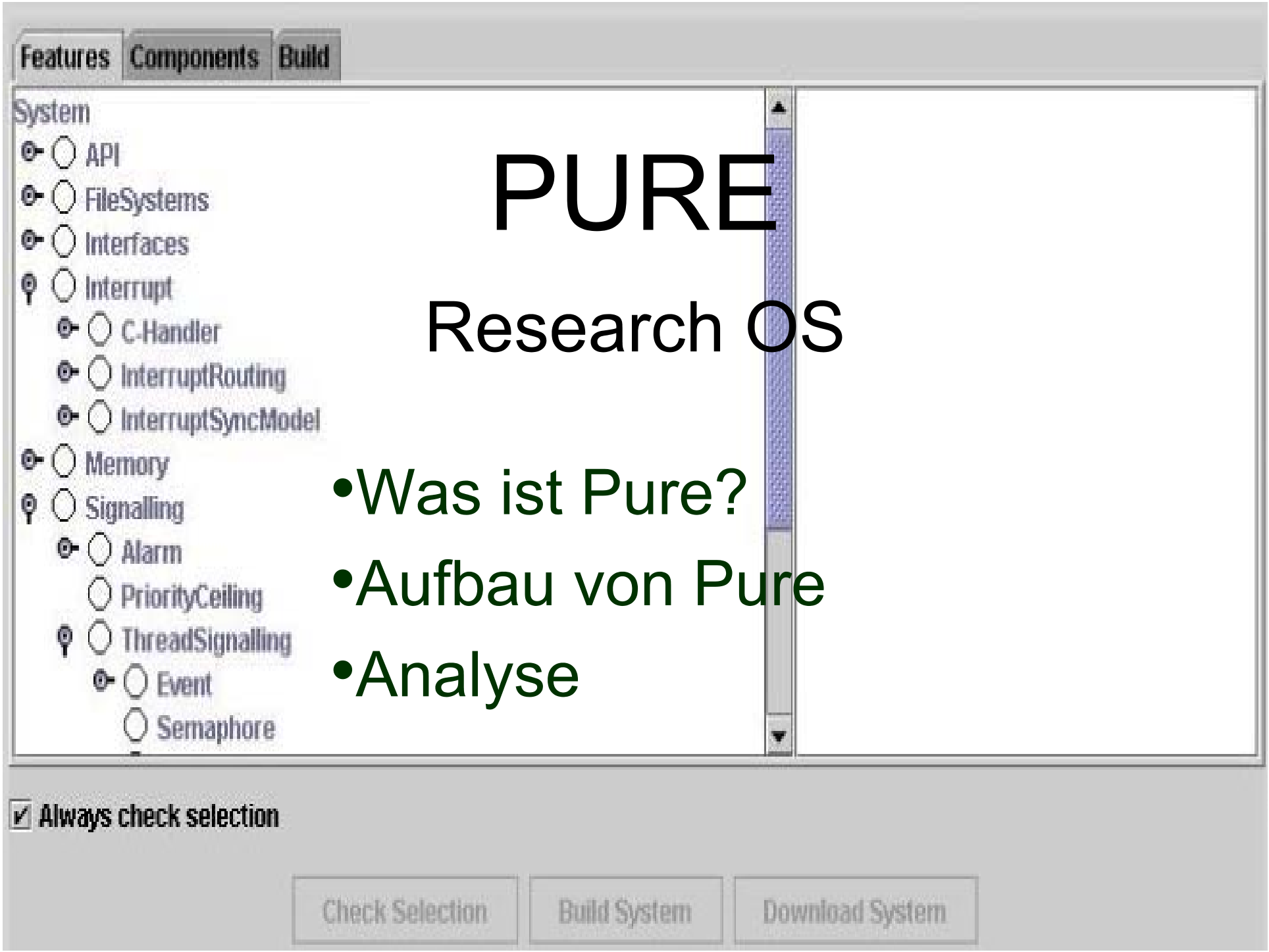
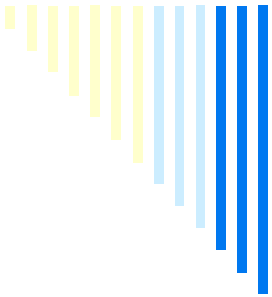


PURE

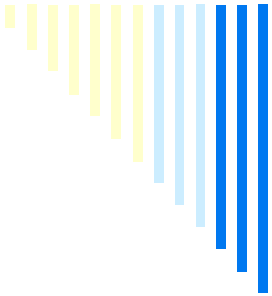
Research OS

- Was ist Pure?
- Aufbau von Pure
- Analyse



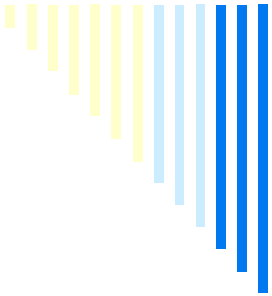
Wieso wurde Pure entwickelt?

- Normale Betriebssysteme haben feste Schnittstellen
- Problem: Fähigkeiten können fehlen oder brach liegen



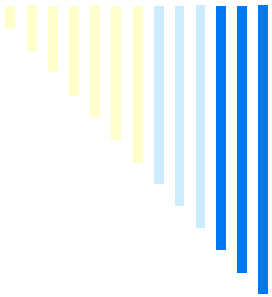
Was ist Pure?

- Einsatz in eingebetteten Systemen
- „Betriebssystembaukasten“
- Betriebssystem als objektorientiert implementierte Programmfamilie



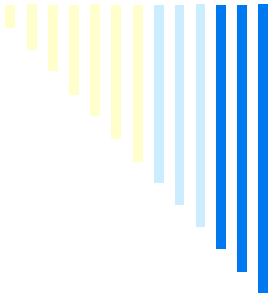
Daten zu Pure?

- Implementierung in C++
- Lauffähig auf: i80x86, sparc, alpha, m68k, ppc60x, C167, AVR, ARM
- Nucleus besteht aus über 100 Klassen und 600 Methoden



Die Programmfamilie

- Idee Mitte der Siebziger Jahre
- Minimale Menge von System-funktionen bilden Grundgerüst
- Ausbau durch minimale System-erweiterungen

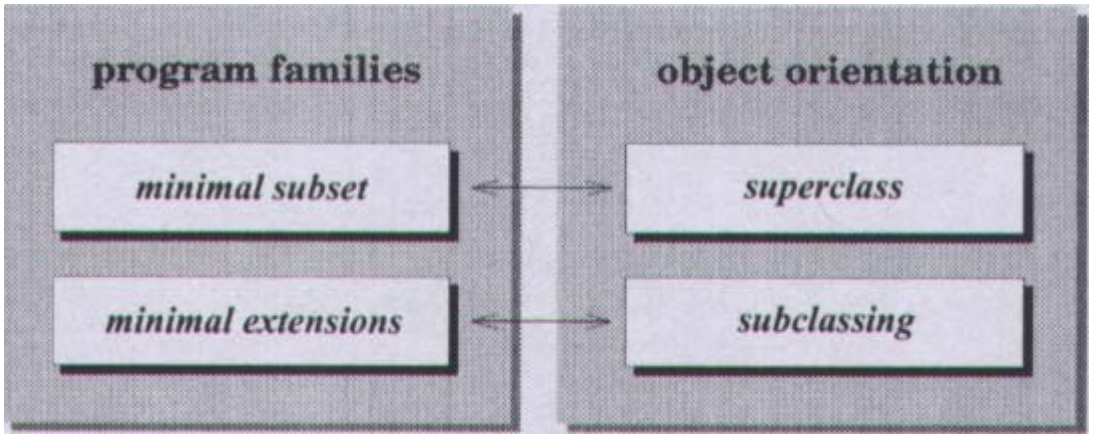


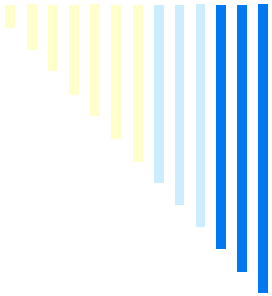
Implementierung(1)

- Aufbau als Bibliothek von Modulen
- Vorraussetzung:
Hoch modulare Systemarchitektur

Implementierung(2)

- Analogie zwischen Programmfamilie und Objektorientierung

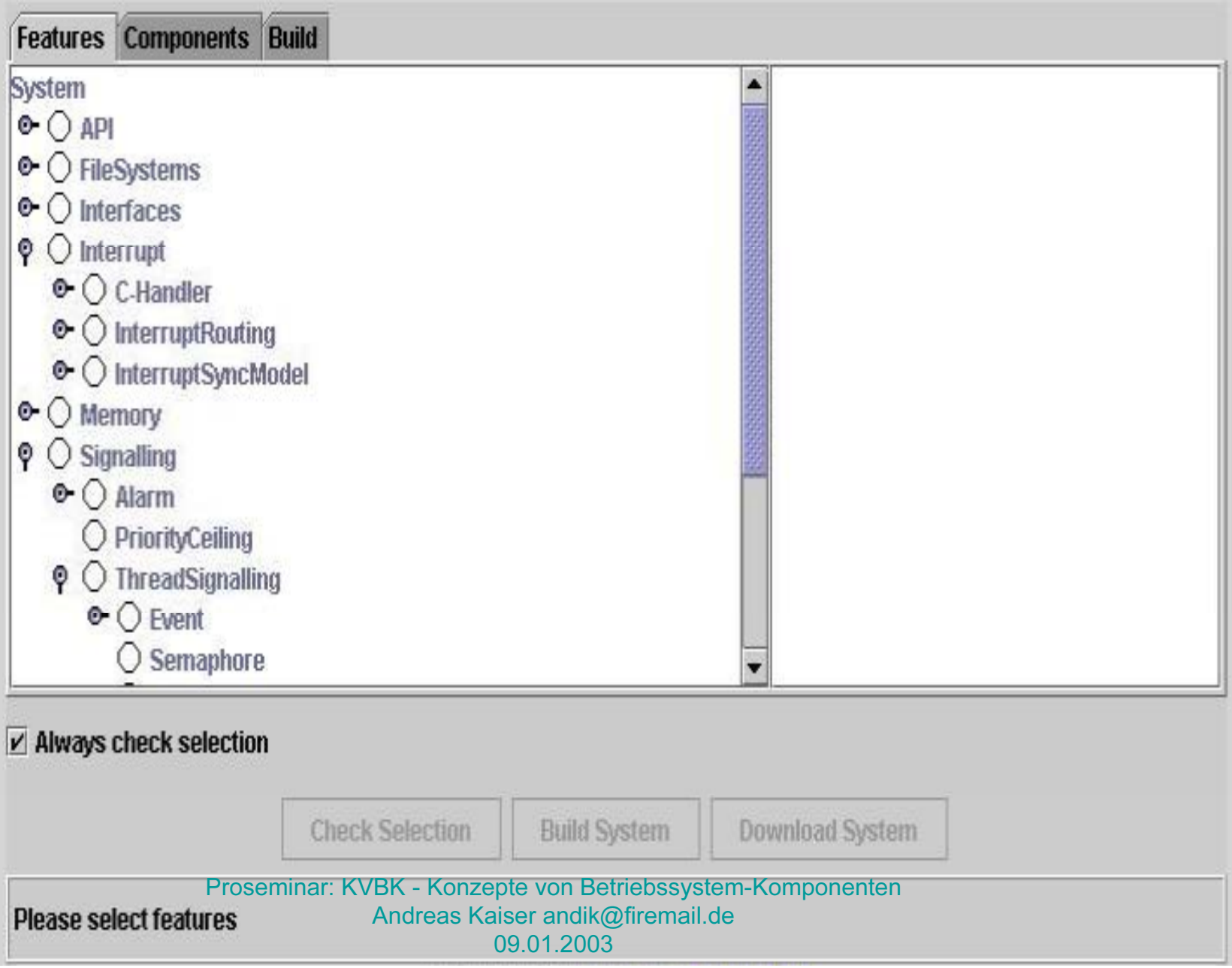




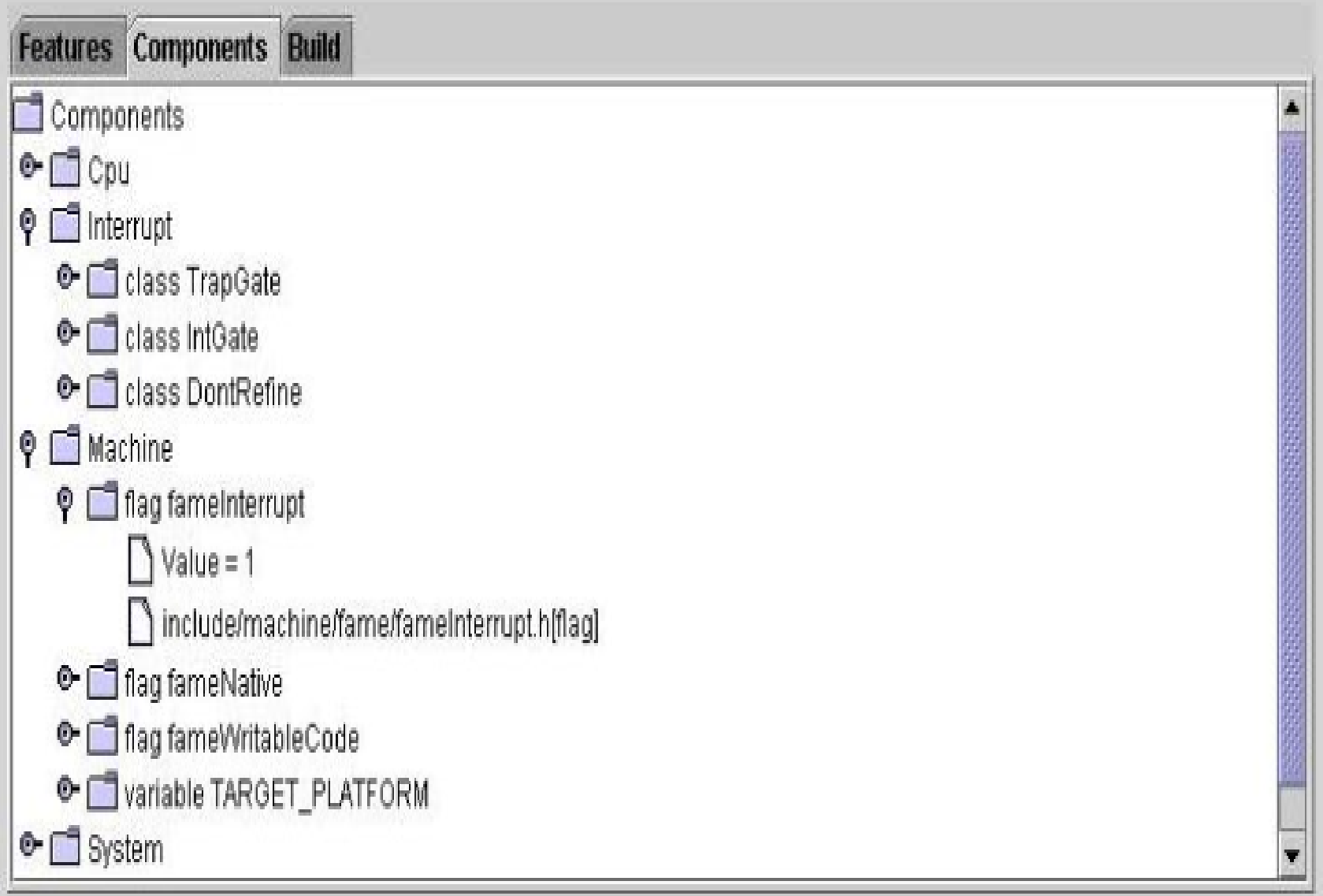
Implementierung(3)

- Inkrementeller Systementwurf
- Systemkontruktion: „bottom-up“
- Kontrolle: „top-down“

Build your own PURE embedded operating system with Pure::Consul

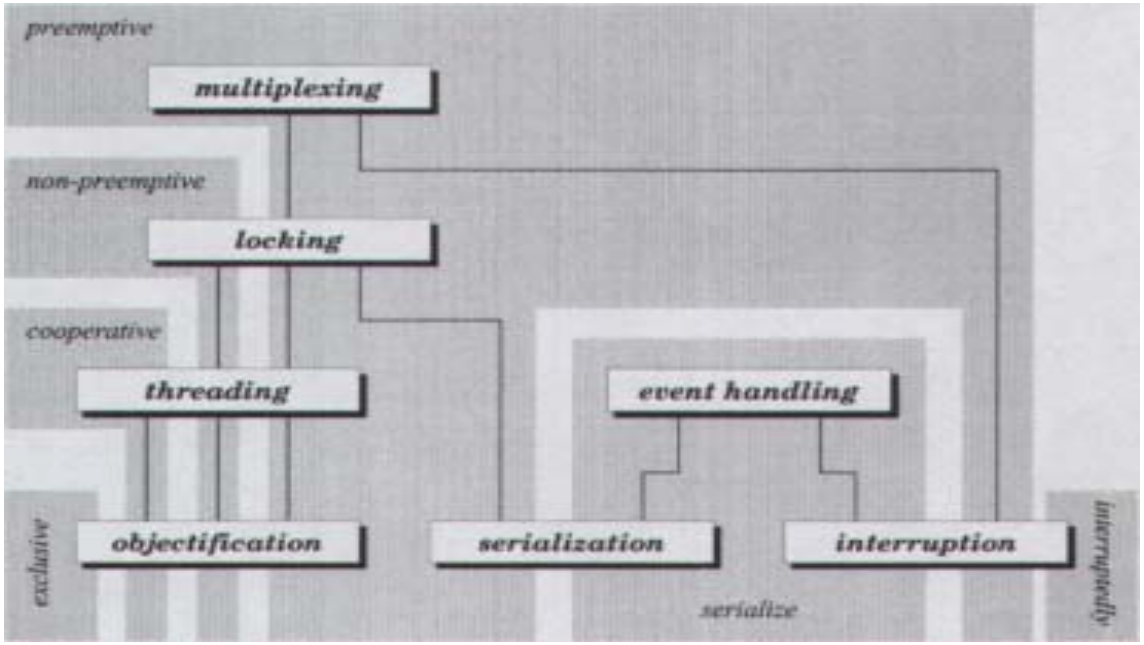


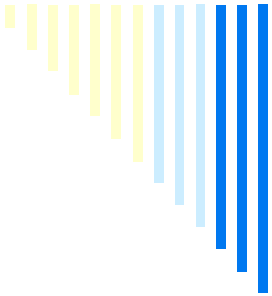
Build your own PURE embedded operating system with Pure::Consul



Der Nucleus

- Enthält mehrere Einheiten



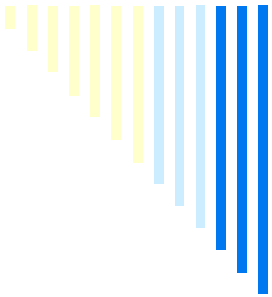


Familienmitglieder(1)

- Interruptedly:
  - -Verarbeitet nur Unterbrechungen
- Serialize:
  - -Koordinierung von Unterbrechung und Programm

Familienmitglieder(2)

- Exclusive:
  - -Nur ein einziger Thread
- Cooperative:
  - -Thread als Objekte
  - -Beliebig viele Threads



Familienmitglieder(3)

- Non-preemptive:
  - -Thread Scheduling erweitert um serialisierte Ausführung
- Preemptive:
  - -Zeitgesteuertes Thread Scheduling

Familienmitglieder(4)

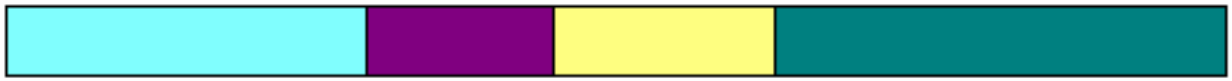
- Non-preemptive:
  - -Rechnender Prozess wird nicht verdrängt
- Preemptive:
  - -Rechnender Prozess kann verdrängt werden

Familienmitglieder(5)

Beispiel: Shortest Job First

Prozess	Ankunftszeit	Dauer
P1	0	8
P2	1	4
P3	2	9
P4	3	5

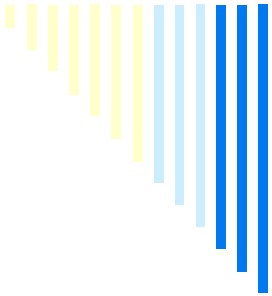
nicht-präemptiv Wartezeit (0+7+15+9)/4 = 7.75



präemptiv Wartezeit (9+0+15+2)/4 = 6.5

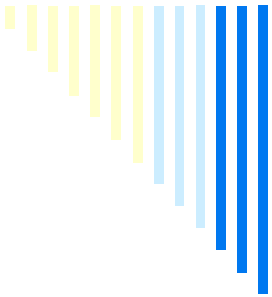
Speicherverbrauch

family member	**size** (in byte)			
	text	*data*	*bss*	*total*
interrupedly	812	64	392	1268
serialize	1882	8	416	2306
exclusive	434	0	0	434
cooperative	1620	0	28	1648
non-preemptive	1671	0	28	1699
preemptive	3642	8	428	4062



Leistungsverhalten

- Interruptbehandlung innerhalb von 200 – 300 clock cycles
- Scheduling innerhalb von ca. 50 – 350 clock cycles



Probleme

- Komponentenauswahl:
  - -Welches sind die optimalen Module für eine Anwendung
  - -nicht formal beweisbar

Beispiel: Kosinusfunktion(1)

- Applikation1: genau, keine Echtzeit
- Applikation2: genau, Echtzeit
- Applikation3: ungenauer, harte Echtzeit

Beispiel: Kosinusfunktion(2)

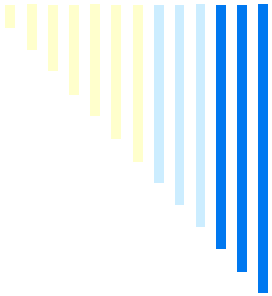
- Applikation1: iterativer Kosinusalgorithmus
- Applikation2: Tabelle von Kosinuswerten, danach Interpolation
- Applikation3: Tabelle von Kosinuswerten

Feature Based Modelling(1)

- Bibliothek der Implementierungen erstellen
- Richtige Implementierung für das jeweilige Problem des Benutzers finden

Feature Based Modelling(2)

- Beschreibung für jedes Modul
- Beziehungen zwischen den Features
- Überprüfung der Auswahl der Features



Vision für die Zukunft

- Betriebssystem wird automatisch anhand der Anforderungen generiert