# Betriebssystemtechnik

Operating System Engineering (OSE)

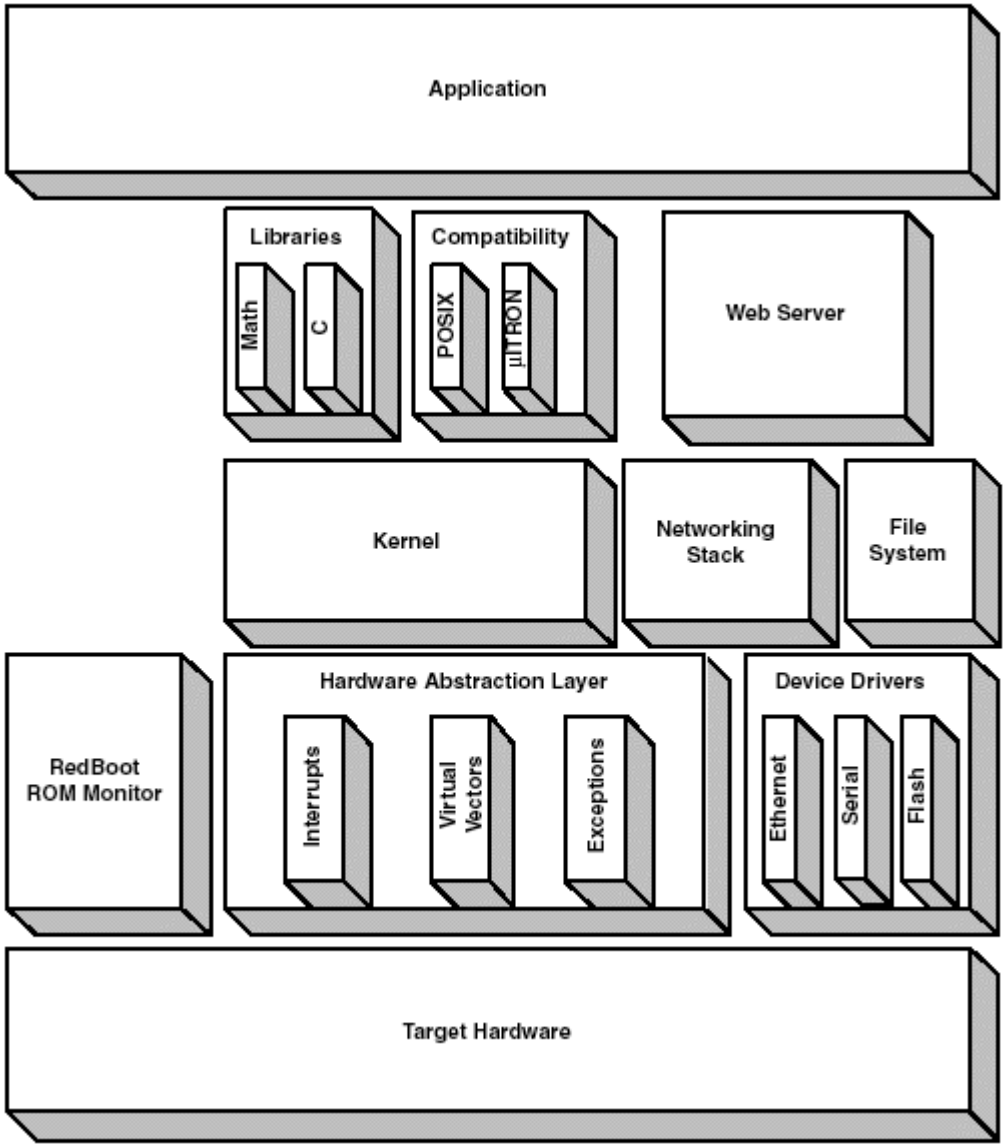
## Stand der Kunst

# eCos – eine Betriebssystemfamilie

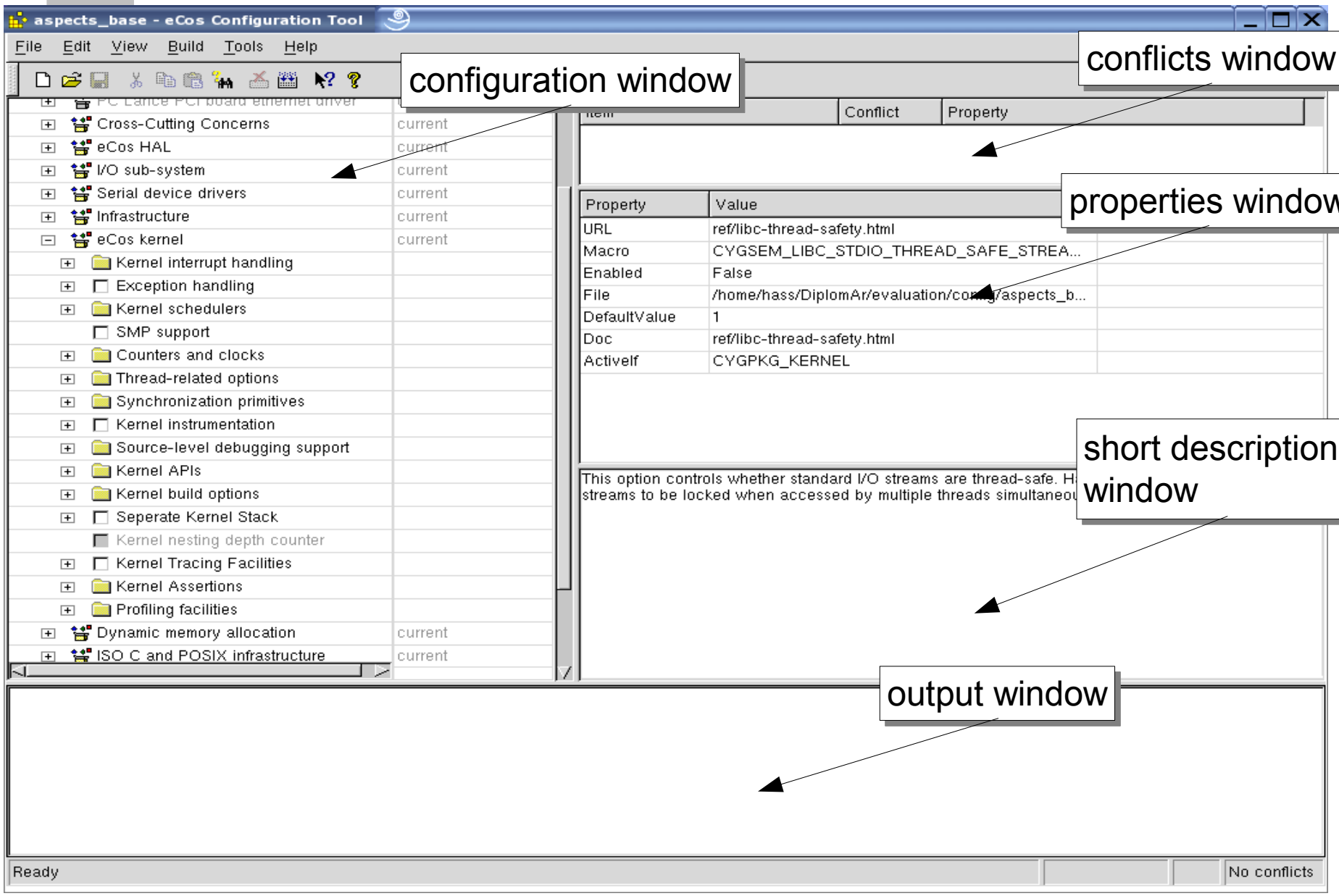
... dient hier als Beispiel für den Stand der Kunst

- Zieldomäne: eingebettete Systeme
- Ansatz: Ressourcen sparen durch statische anwendungsspezifische Konfigurierung
- Implementierungssprache: C und C++ (Kernel!)
- Lizenz: Open Source (früher Cygnus Solutions, heute RedHat)

# eCos *building blocks*

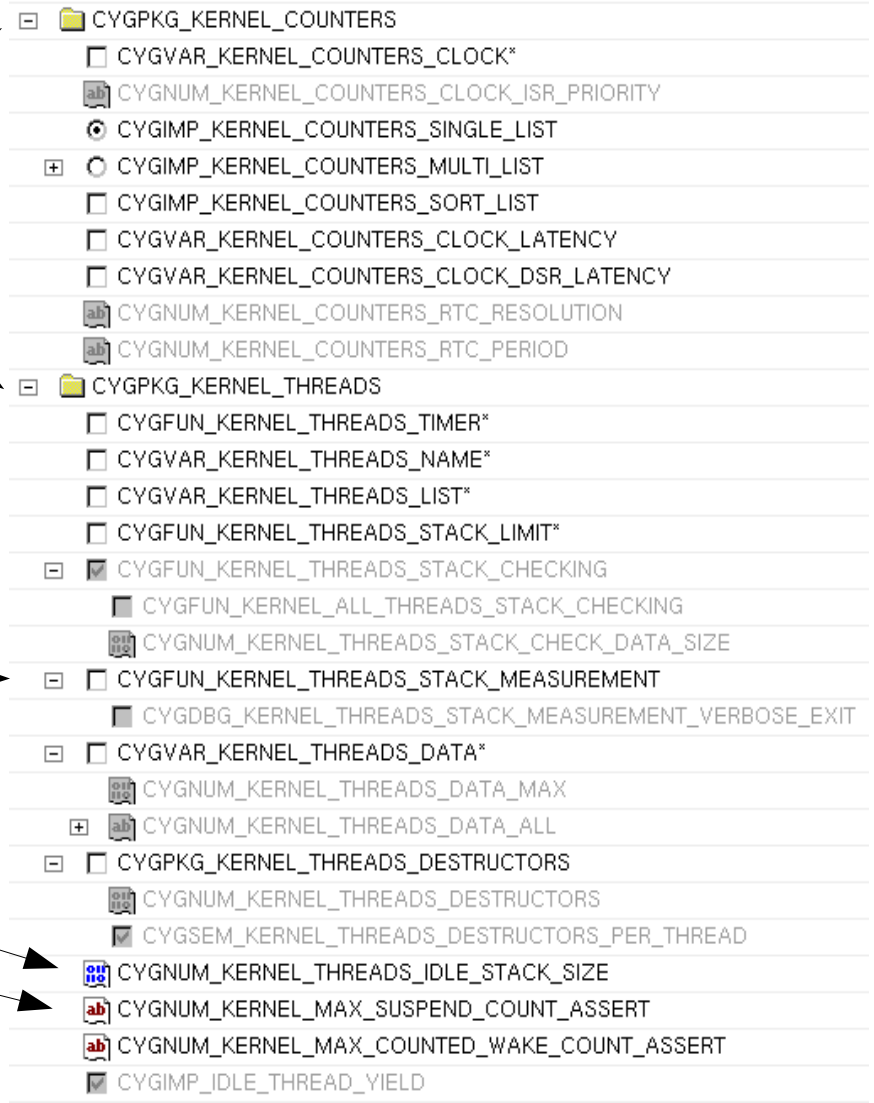



# Konfigurationswerkzeug



# Konfigurierungseinheiten

- Pakete
  - Quellcodebündel, oberste Konfigurierungsebene
- Komponenten
  - Logische Konfigurierungs-einheiten unterhalb der Packages (hierarchisch)
- Optionen
  - logisch
  - Integer Werte
  - Zeichenketten
  - Aufzählungstypen




# Komponentenbeschreibung (1)

- in der *Component Description Language* verfasste Dateien beschreiben je ein Paket, seine Komponenten und dazugehörige Optionen:

```
cdl_package CYGPKG_INFRA {
    display       "Infrastructure"
    include_dir   cyg/infra
    description   "
        Common types and useful macros.
        Tracing and assertion facilities.
        Package startup options."
    compile startup.cxx prestart.cxx pkgstart.cxx userstart.cxx      \
            dummyxxmain.cxx null.cxx simple.cxx fancy.cxx buffer.cxx \
            diag.cxx tcdiag.cxx memcpy.c memset.c delete.cxx
}
```

über die Paketauswahl werden indirekt Dateien selektiert

# Komponentenbeschreibung (2)

```
cdl_component CYGPKG_IO_SERIAL_POWERPC_COGENT_SERIAL_A {
    display       "Cogent PowerPC serial port A driver"
    flavor        bool
    default_value 0
    requires      (CYGIMP_KERNEL_INTERRUPTS_CHAIN || \
                   !CYGPKG_IO_SERIAL_POWERPC_COGENT_SERIAL_B)
    …
}
```

komplexe Abhängigkeiten können formuliert werden

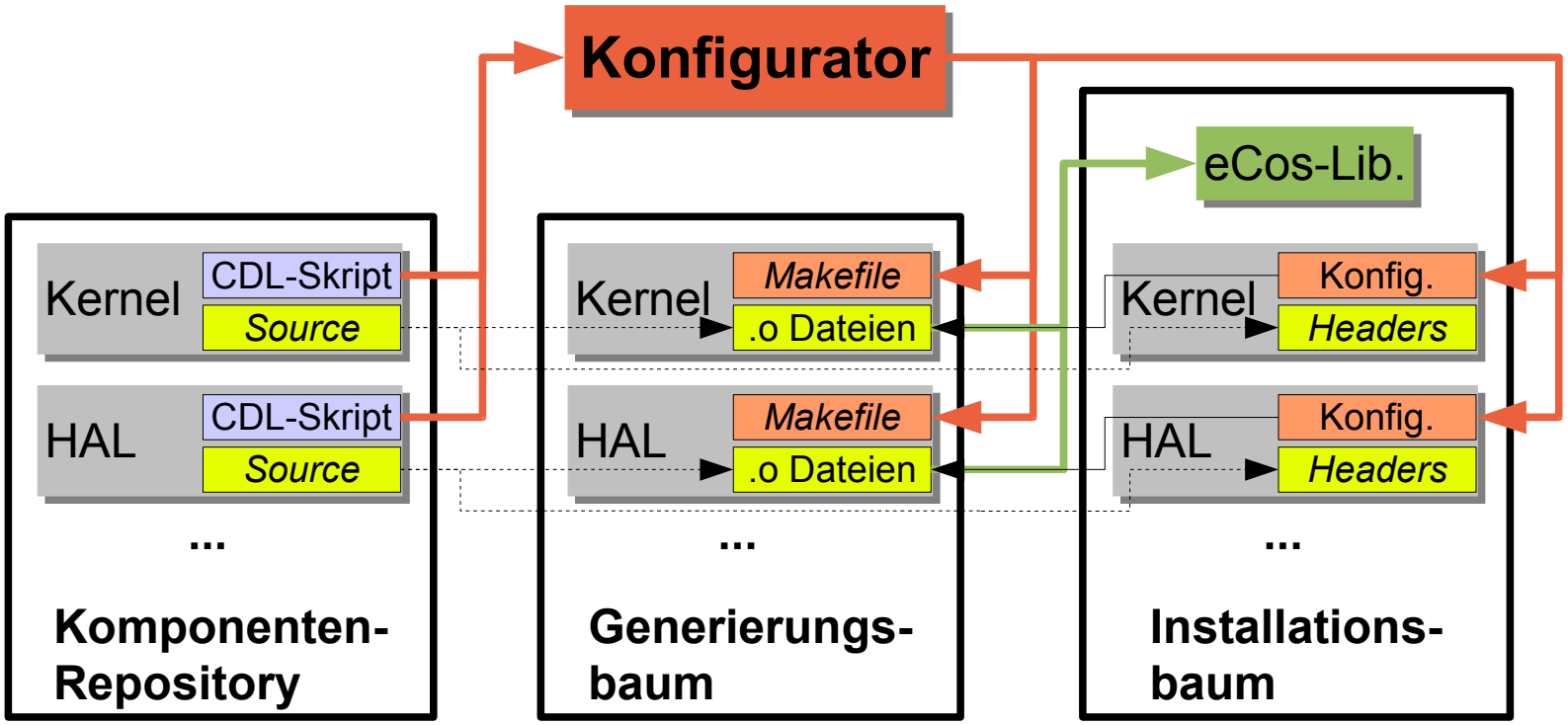
```
cdl_option CYGNUM_HAL_RTC_PERIOD {
    display       "Real-time clock period"
    flavor        data
    calculated    12500
}
```

Werte von Optionen können auch berechnet werden

```
cdl_option CYGNUM_LIBC_TIME_STD_DEFAULT_OFFSET {
    display       "Default Standard Time offset"
    flavor        data
    legal_values  -- -90000 to 90000
    default_value -- 0
    description   "
        This option controls ..."
}
```

Wertebereich und Default-Werte können festgelegt werden.

# Systemgenerierungsprozess




- Generierte Makefiles stellen sicher, dass die gewählten Dateien übersetzt werden.
- Optionen werden als C-Makros in Konfigurationsdateien geschrieben und bei der Übersetzung berücksichtigt.

# Komponentenkonfigurierung

```
#include <pkgconf/kernel.h>
#include <cyg/infra/cyg_trac.h>

void some_func() {
  CYG_REPORT_FUNCTION();
  ...
#ifdef SOME_OPTION
  ...
#endif
  ...
  CYG_REPORT_RETURN();
}
```

```
#define SOME_OPTION
// #define TRACE_KERNEL
```

```
#include <pkgconf/kernel.h>
#ifdef TRACE_KERNEL
#define CYG_REPORT_RETURN() \
  ...
#else // leer!
#define CYG_REPORT_RETURN()
#endif
```

- Berücksichtung der Konfiguration erfolgt über bedingte Übersetzung (#ifdef)
- Konfigurierbare quer schneidende Belange werden über Makros realisiert, um #ifdefs zu reduzieren

# Eine Beispielkomponente

27 Zeilen Quelltext

```
Cyg_Mutex::Cyg_Mutex() {
    CYG_REPORT_FUNCTION();
    locked   = false;
    owner    = NULL;
#if defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT) && \
    defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DYNAMIC)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_INHERIT
    protocol = INHERIT;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_CEILING
    protocol = CEILING;
    ceiling  = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRI;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_NONE
    protocol = NONE;
#endif
#else // not (DYNAMIC and DEFAULT defined)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_CEILING
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY
    // if there is a default priority ceiling defined, use that to initialize
    // the ceiling.
    ceiling = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY;
#else
    ceiling = 0; // Otherwise set it to zero.
#endif
#endif
#endif // DYNAMIC and DEFAULT defined
     CYG_REPORT_RETURN();
}
```

# Eine Beispielkomponente

```
Cyg_Mutex::Cyg_Mutex() {
    CYG_REPORT_FUNCTION();
    locked   = false;
    owner    = NULL;
#if defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT) && \
    defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DYNAMIC)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_INHERIT
    protocol = INHERIT;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_CEILING
    protocol = CEILING;
    ceiling  = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRI;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_NONE
    protocol = NONE;
#endif
#else // not (DYNAMIC and DEFAULT defined)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_CEILING
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY
    // if there is a default priority ceiling defined, use that to initialize
    // the ceiling.
    ceiling = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY;
#else
    ceiling = 0; // Otherwise set it to zero.
#endif
#endif
#endif // DYNAMIC and DEFAULT defined
    CYG_REPORT_RETURN();
}
```

2 Zeilen für die
**Kontrollflussverfolgung**

# Eine Beispielkomponente

21 (unleserliche) Zeilen für **optionale Merkmale**

```
Cyg_Mutex::Cyg_Mutex() {
    CYG_REPORT_FUNCTION();
    locked   = false;
    owner    = NULL;
#if defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT) && \
    defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DYNAMIC)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_INHERIT
    protocol = INHERIT;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_CEILING
    protocol = CEILING;
    ceiling  = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRI;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_NONE
    protocol = NONE;
#endif
#else // not (DYNAMIC and DEFAULT defined)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_CEILING
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY
    // if there is a default priority ceiling defined, use that to initialize
    // the ceiling.
    ceiling = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY;
#else
    ceiling = 0; // Otherwise set it to zero.
#endif
#endif
#endif // DYNAMIC and DEFAULT defined
    CYG_REPORT_RETURN();
}
```

# Eine Beispielkomponente

```
Cyg_Mutex::Cyg_Mutex() {
    CYG_REPORT_FUNCTION();
    locked   = false;
    owner    = NULL;
#if defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT) && \
    defined(CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DYNAMIC)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_INHERIT
    protocol = INHERIT;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_CEILING
    protocol = CEILING;
    ceiling  = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRI;
#endif
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_NONE
    protocol = NONE;
#endif
#else // not (DYNAMIC and DEFAULT defined)
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_CEILING
#ifdef CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY
    // if there is a default priority ceiling defined, use that to initialize
    // the ceiling.
    ceiling = CYGSEM_KERNEL_SYNCH_MUTEX_PRIORITY_INVERSION_PROTOCOL_DEFAULT_PRIORITY;
#else
    ceiling = 0; // Otherwise set it to zero.
#endif
#endif
#endif // DYNAMIC and DEFAULT defined
    CYG_REPORT_RETURN();
}
```

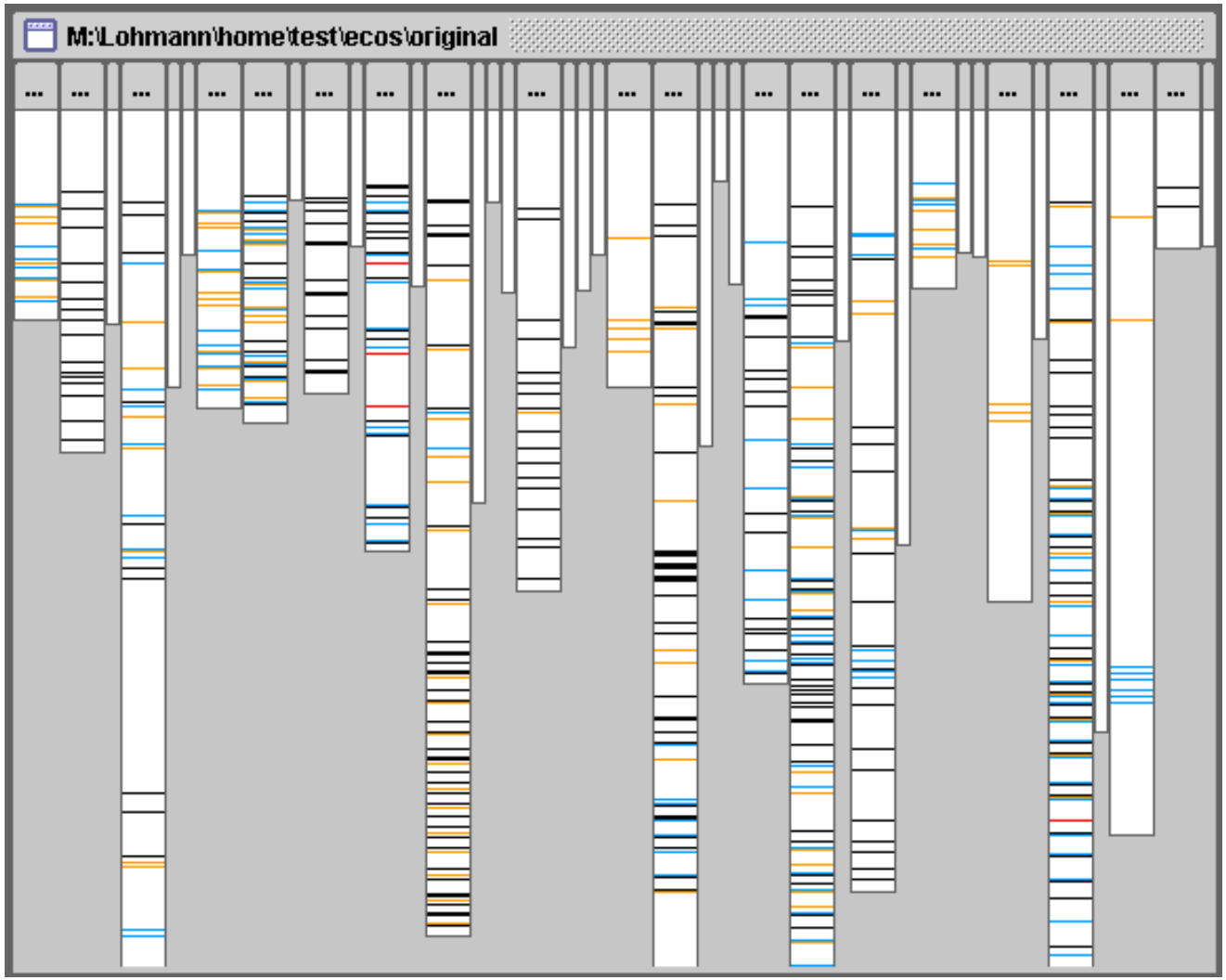
4 Zeilen für die
**eigentliche Implementierung**

# Quer schneidende Belange

synchronization

instrumentation

tracing




# Anteil quer schneidender Belange

- Untersucht wurden in C++ implementierte Pakete
  - Kernel
  - libc
  - Memory Management
  - Wallclock/Watchdog
  - POSIX/µITRON

| | Kernel | | Memory Management | | Gesamt | |
|---|---|---|---|---|---|---|
| **LOC** | 5205 | 100,00% | 2813 | 100,00% | 16535 | 100,00% |
| **Tracing** | 336 | 6,46% | 66 | 2,35% | 938 | 5,67% |
| **Assertions** | 384 | 7,38% | 151 | 5,37% | 793 | 4,80% |
| **Profiling** | 319 | 6,13% | 0 | 0,00% | 319 | 1,93% |
| **Locking** | 186 | 3,57% | 40 | 1,42% | 300 | 1,81% |
| **Gesamt** | 1225 | 23,54% | 257 | 9,14% | 2350 | 14,21% |

# Konfigurationsoptionen

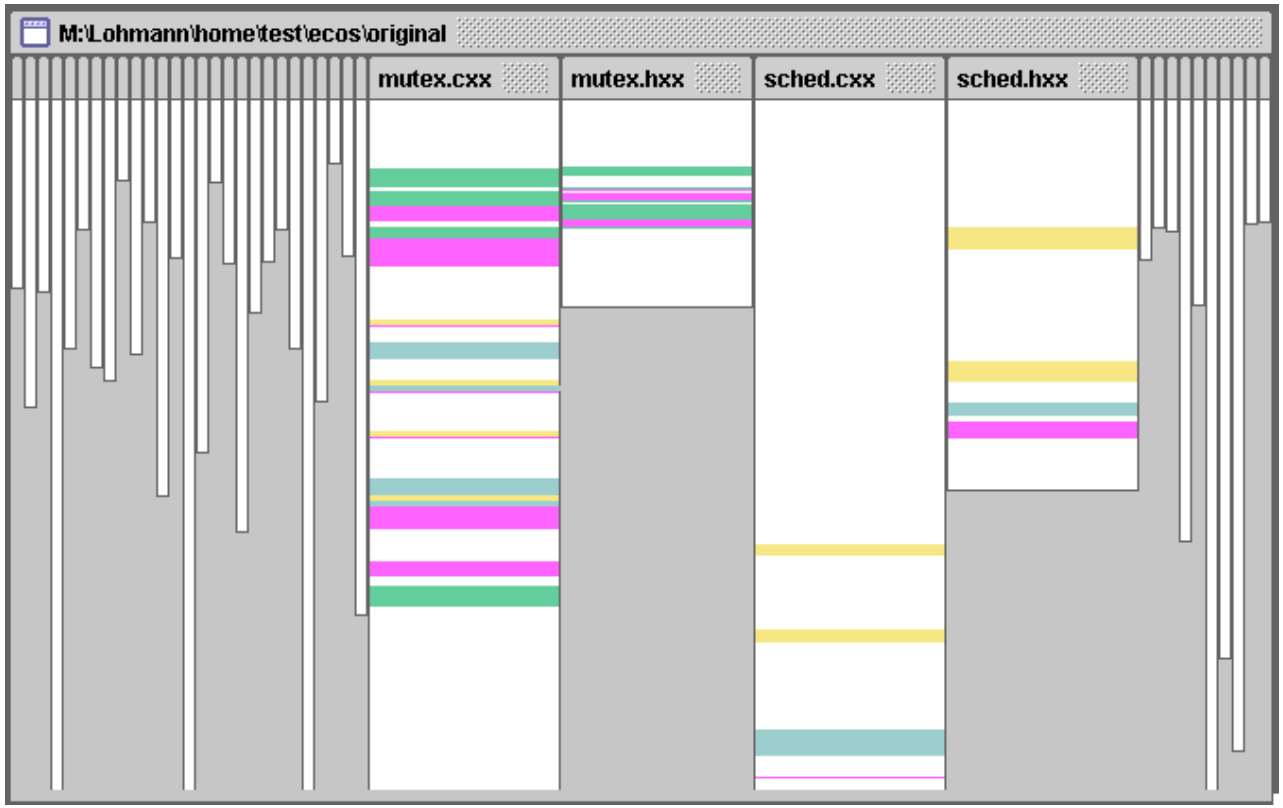
Varianten des Protokolls zur Vermeidung der Prioritätsumkehr bei *Mutex*-Verwendung

simple

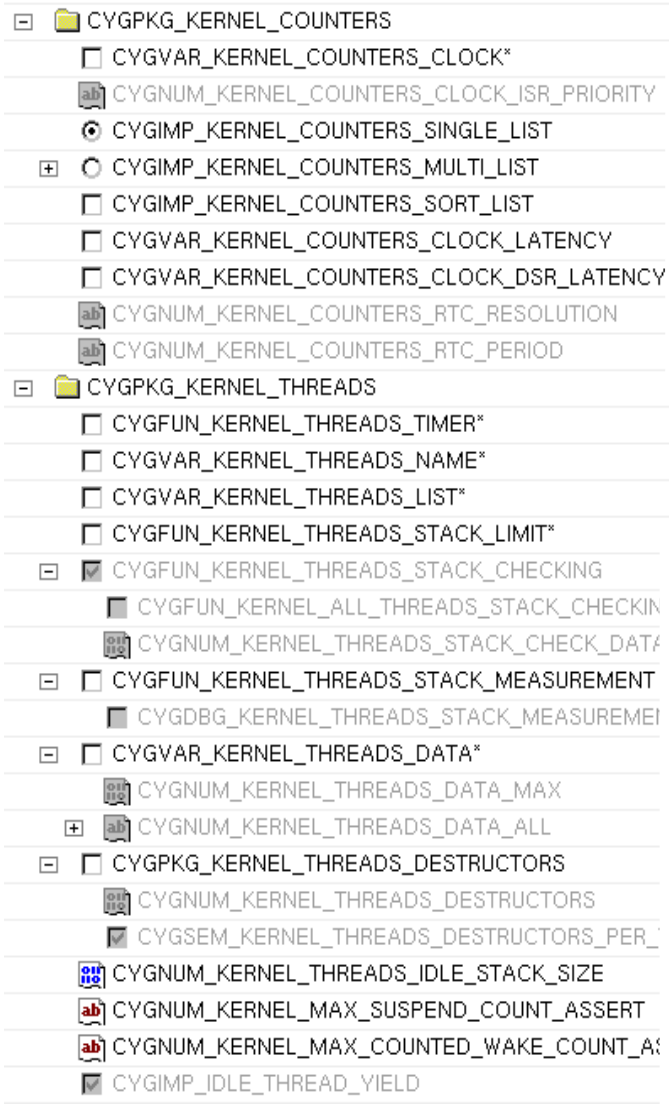
ceiling

inheritance

dynamic




# Variationspunkte pro Option




| Option | # |
|---|---|
| CYG_VAR_KERNEL_COUNTERS_CLOCK | 42 |
| CYG_VAR_KERNEL_COUNTERS_SINGLE_LIST | 7 |
| CYG_VAR_KERNEL_COUNTERS_MULTI_LIST | 7 |
| CYG_VAR_KERNEL_COUNTERS_SORT_LIST | 2 |
| CYG_VAR_KERNEL_COUNTERS_CLOCK_LATENCY | 20 |
| CYG_VAR_KERNEL_COUNTERS_CLOCK_DSR_LATENCY | 3 |

| Option | # |
|---|---|
| CYGFUN_KERNEL_THREADS_TIMER | 95 |
| CYGVAR_KERNEL_THREADS_NAME | 15 |
| CYGVAR_KERNEL_THREADS_LIST | 10 |
| CYGFUN_KERNEL_THREADS_STACK_LIMIT | 9 |
| CYGFUN_KERNEL_THREADS_STACK_CHECKING | 10 |
| CYGFUN_KERNEL_ALL_THREADS_STACK_CHECKING | 1 |
| CYGFUN_KERNEL_THREADS_STACK_MEASUREMENT | 10 |
| CYGFUN_KERNEL_THREADS_STACK_MEASUREMENT_ | 2 |
| CYGVAR_KERNEL_THREADS_DATA | 8 |
| CYGVAR_KERNEL_THREADS_DESTRUCTORS | 6 |
| CYGVAR_KERNEL_THREADS_DESTRUCTORS_PER_... | 13 |

# Zusammenfassung

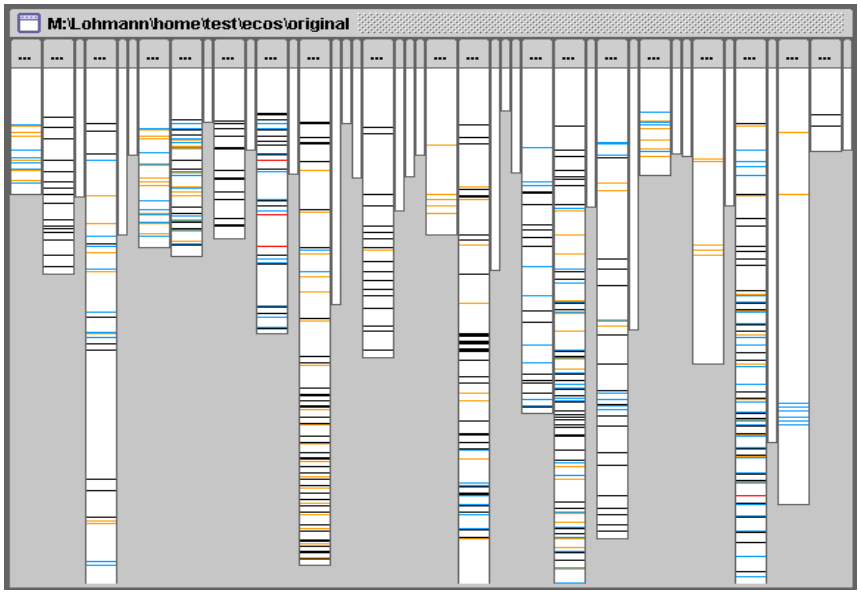
- eCos ist ein modernes konfigurierbares Betriebssystem
- einfache Konfigurierung durch GUI Unterstützung
  - die CDL ist eine mächtige Sprache
    - Festlegung von Abhängigkeiten zwischen Komponenten
    - Typisierte und berechnete Werte für Optionen
- **Mängel** (im Hinblick auf die Umsetzung einer Produktlinie)
  - Klassische Umsetzung der Konfigurierungsentscheidungen in den Komponenten mit Hilfe von #ifdef und Makros
    - Schutz vor ungewollten Ersetzungen nur durch strikte Namenskonvention
  - mangelnde Trennung der Belange
    - viel Konfigurierungswissen ist im Quellcode verankert
    - quer schneidende Belange blähen die Funktionen auf
    - bedingte Übersetzung macht den Code schwer verständlich, zu warten und wiederzuverwenden

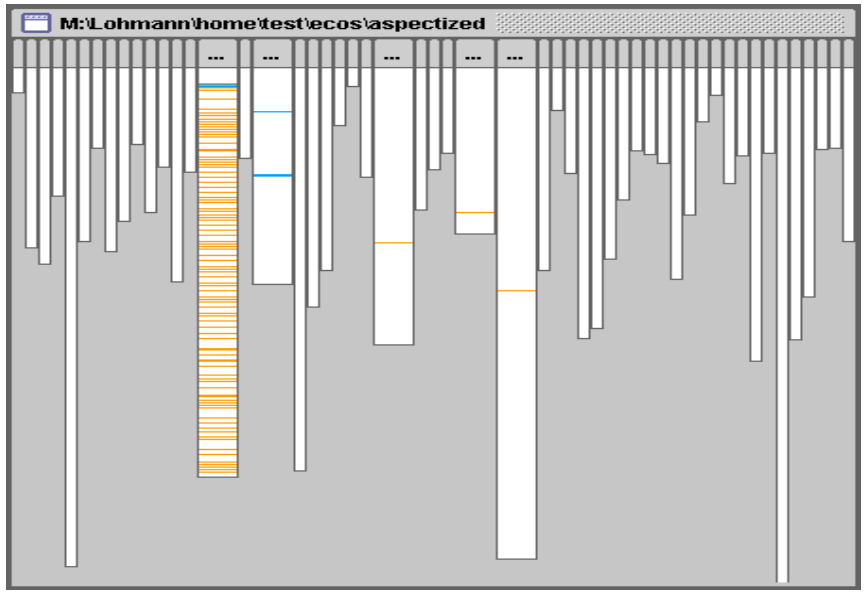
# Epilog: eine vergleichende Studie [2]

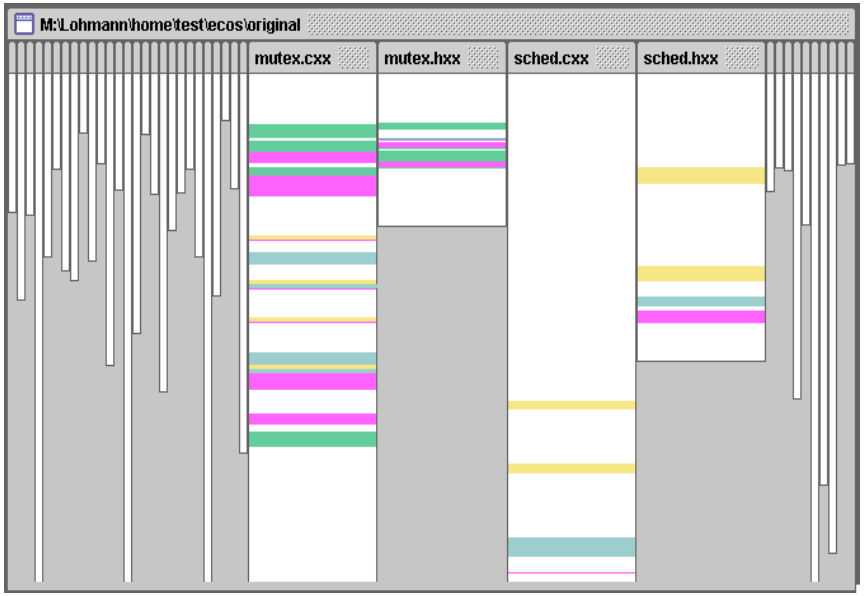
- **Vergleich: *original Kernel* → *Kernel mit Aspekten***
  - **3 quer schneidende Belange**
    - *interrupt synchronization* 187 Aufrufe → 160 Code *Joinpoints*
    - *kernel instrumentation* 162 Aufrufe → 139 Code *Joinpoints*
    - *tracing* 336 Aufrufe → 632 Code *Joinpoints*
  - **12 Konfigurationsoptionen**
    - *Mutex* Optionen
    - *Thread* Optionen
- betrachtete Eigenschaften
  - *scattering*, Performanz, Speicherplatzverbrauch

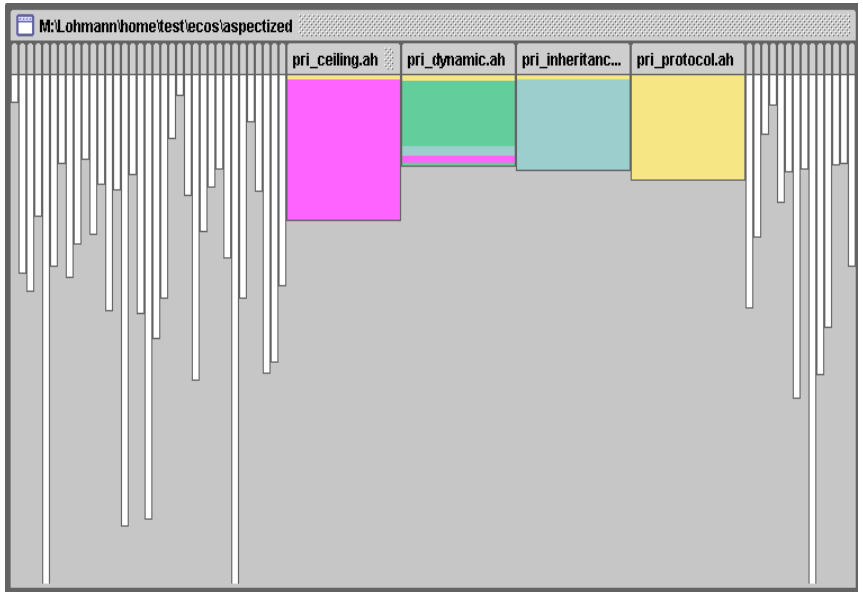
# original Kernel

# Kernel mit Aspekten



# Ausblick

- Untersuchung verschiedener Techniken zur Umsetzung von Variabilität in der Implementierung der Komponenten
  - werkzeugbasierte Lösungen
    - pure::variants als Beispiel eines Variantenmanagement Systems
    - XVCL als Beispiel für eine besser geeignete Präprozessorlösung
  - programmiersprachenbasierte Lösungen
    - Aspekte
    - Objekte
    - *Templates*
    - *Mixin Layers*

# Literatur

[1] A. J. Massa. *Embedded Software Development with eCos*. Prentice Hall, 2003, ISBN 0-13-035473-2.

[2] D. Lohmann, F. Scheler, R. Tartler, O. Spinczyk, and W. Schröder-Preikschat. *A quantitative analysis of aspects in the eCos kernel*. In *EuroSys'06*, pages 191-204. ACM Press, April 2006.