Betriebssysteme (BS)

VL 14 – Zusammenfassung und Ausblick

Daniel Lohmann

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 12 – 8. Februar 2012

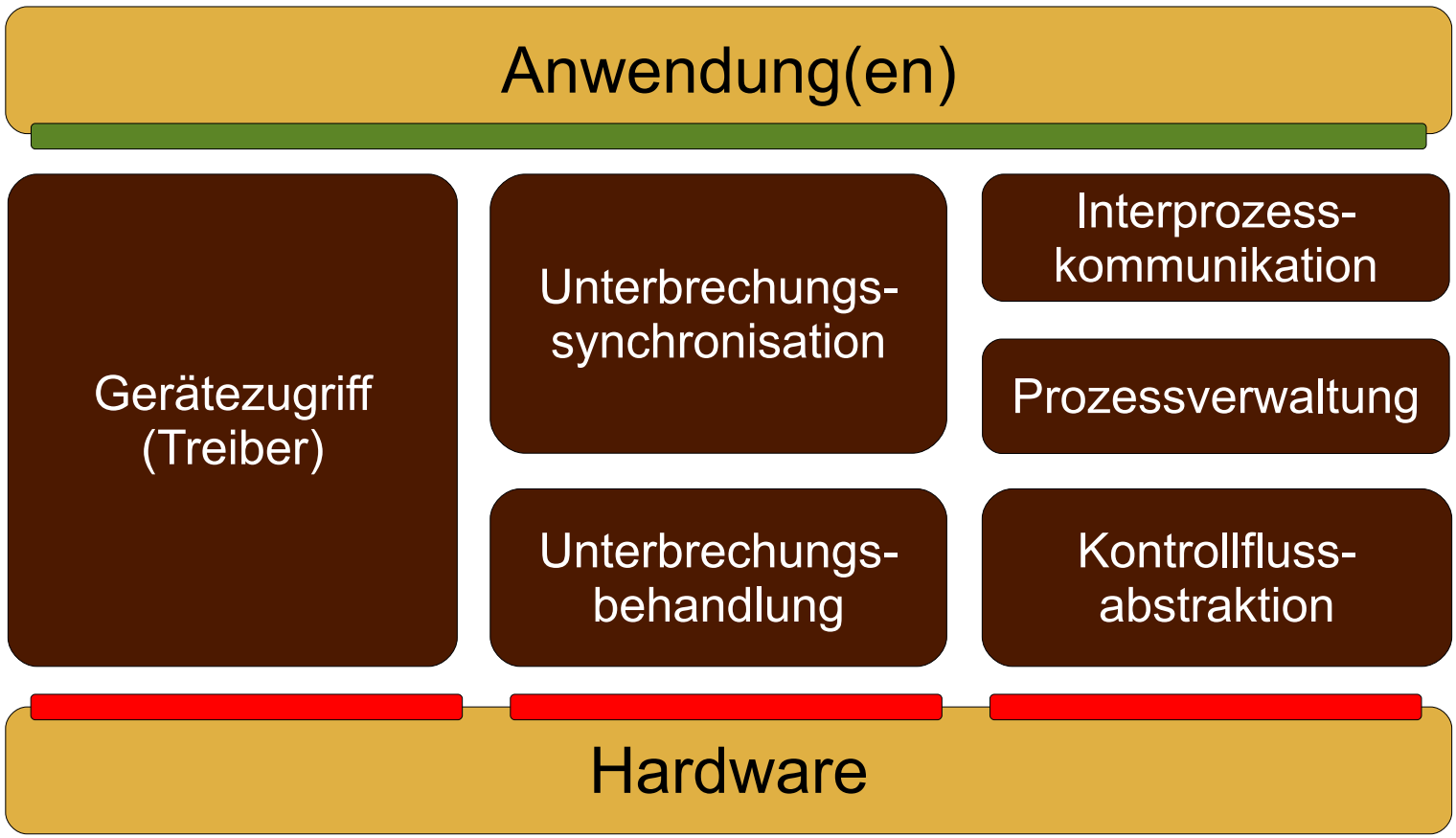
http://www4.informatik.uni-erlangen.de/Lehre/WS12/V_BS

Lernziele

⇝VL 1

- **Vertiefen** des Wissens über die interne Funktionsweise von Betriebssystemen
  - Ausgangspunkt: Systemprogrammierung
  - Schwerpunkt: Nebenläufigkeit und Synchronisation
- **Entwickeln** eines Betriebssystems *von der Pike auf*
  - OOStuBS / MPStuBS (neu!) Lehrbetriebssysteme
  - Praktische Erfahrungen im Betriebsystembau machen
- **Verstehen** der technologischen Hardware-Grundlagen
  - PC-Technologie verstehen und einschätzen können
  - Schwerpunkt: Intel x86 / IA-32

Überblick Vorlesungen



Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

- VL_1 ***Einführung***
- VL_2 ***BS-Entwicklung***
- VL_3 ***IRQs (Hardware)***
- VL_4 ***IRQs (Software)***
- VL_5 ***IRQs (Synchronisation)***
- VL_6 ***Intel IA-32***
- VL_7 ***Koroutinen und Fäden***
- VL_8 ***Scheduling***
- VL_9 ***BS-Architekturen***
- VL_{10} ***Fadensynchronisation***
- VL_{11} ***PC Bussysteme***
- VL_{12} ***Gerätetreiber***
- VL_{13} ***IPC***

Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

1. Ein Streifzug durch die PC-Architektur

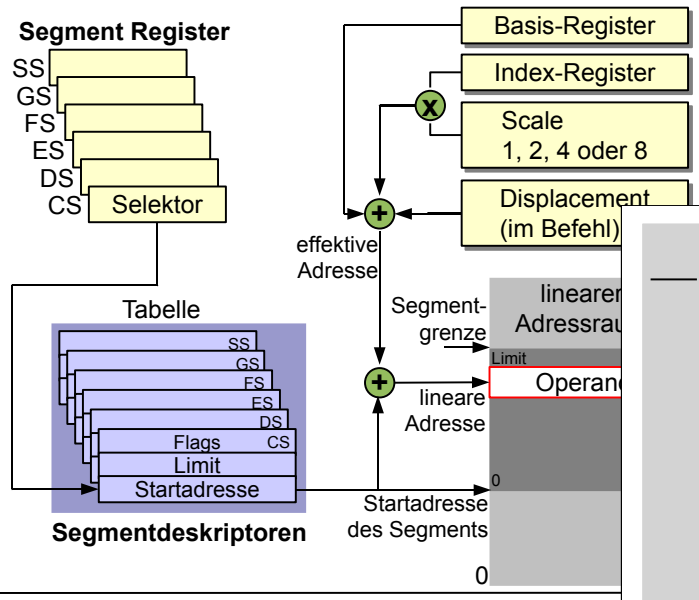
- VL_1 ***Einführung***
- VL_2 ***BS-Entwicklung***
- **VL_3** ***IRQs (Hardware)***
- VL_4 ***IRQs (Software)***
- VL_5 ***IRQs (Synchronisation)***
- **VL_6** ***Intel IA-32***
- VL_7 ***Koroutinen und Fäden***
- VL_8 ***Scheduling***
- VL_9 ***BS-Architekturen***
- VL_{10} ***Fadensynchronisation***
- **VL_{11}** ***PC Bussysteme***
- VL_{12} ***Gerätetreiber***
- VL_{13} ***IPC***

Was wir gemacht haben

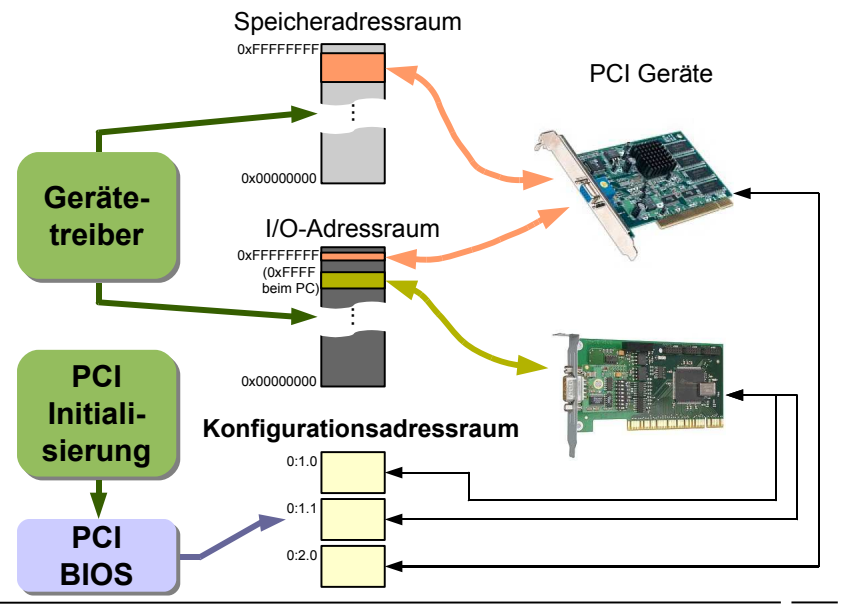
Drei inhaltliche Schwerpunkte!

1. Ein Streifzug durch die PC-Architektur

IA-32: *Protected Mode* – Segmente



Interaktion mit PCI Geräten



Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

2. Kontrollflüsse und ihre Interaktionen

- VL_1 ***Einführung***
- VL_2 ***BS-Entwicklung***
- **VL_3** ***IRQs (Hardware)***
- **VL_4** ***IRQs (Software)***
- **VL_5** ***IRQs (Synchronisation)***
- VL_6 ***Intel IA-32***
- **VL_7** ***Koroutinen und Fäden***
- **VL_8** ***Scheduling***
- VL_9 ***BS-Architekturen***
- **VL_{10}** ***Fadensynchronisation***
- VL_{11} ***PC Bussysteme***
- VL_{12} ***Gerätetreiber***
- VL_{13} ***IPC***

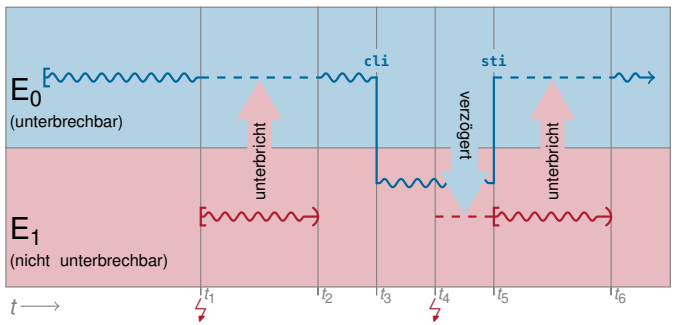
Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

2. Kontrollflüsse und ihre Interaktionen

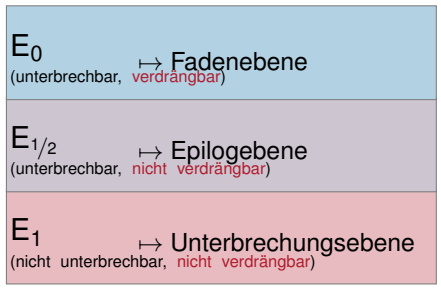
Prioritätsebenenmodell

- Kontrollflüsse können die Ebene wechseln
  - Mit `cli` wechselt ein E_0-Kontrollfluss explizit auf E_1
    - er ist ab dann nicht mehr unterbrechbar
    - andere E_1-Kontrollflüsse werden verzögert (↩ Sequentialisierung)
  - Mit `sti` wechselt ein E_1-Kontrollfluss explizit auf E_0
    - er ist ab dann (wieder) unterbrechbar
    - anhängige E_1-Kontrollflüsse „schlagen durch“ (↩ S



Erweitertes Prioritätsebenenmodell

- Kontrollflüsse auf E_l werden
  1. jederzeit unterbrochen durch Kontrollflüsse von E_m (für $m > l$)
  2. nie unterbrochen durch Kontrollflüsse von E_k (für $k \leq l$)
  3. jederzeit verdrängt durch Kontrollflüsse von E_l (für $l = 0$)



Kontrollflüsse der E_0 (Fadenebene) sind **verdrängbar**.

Für die Konsistenzssicherung auf dieser Ebene brauche wir zusätzliche **Mechanismen** zur **Fadensynchronisation**.

Was wir gemacht haben

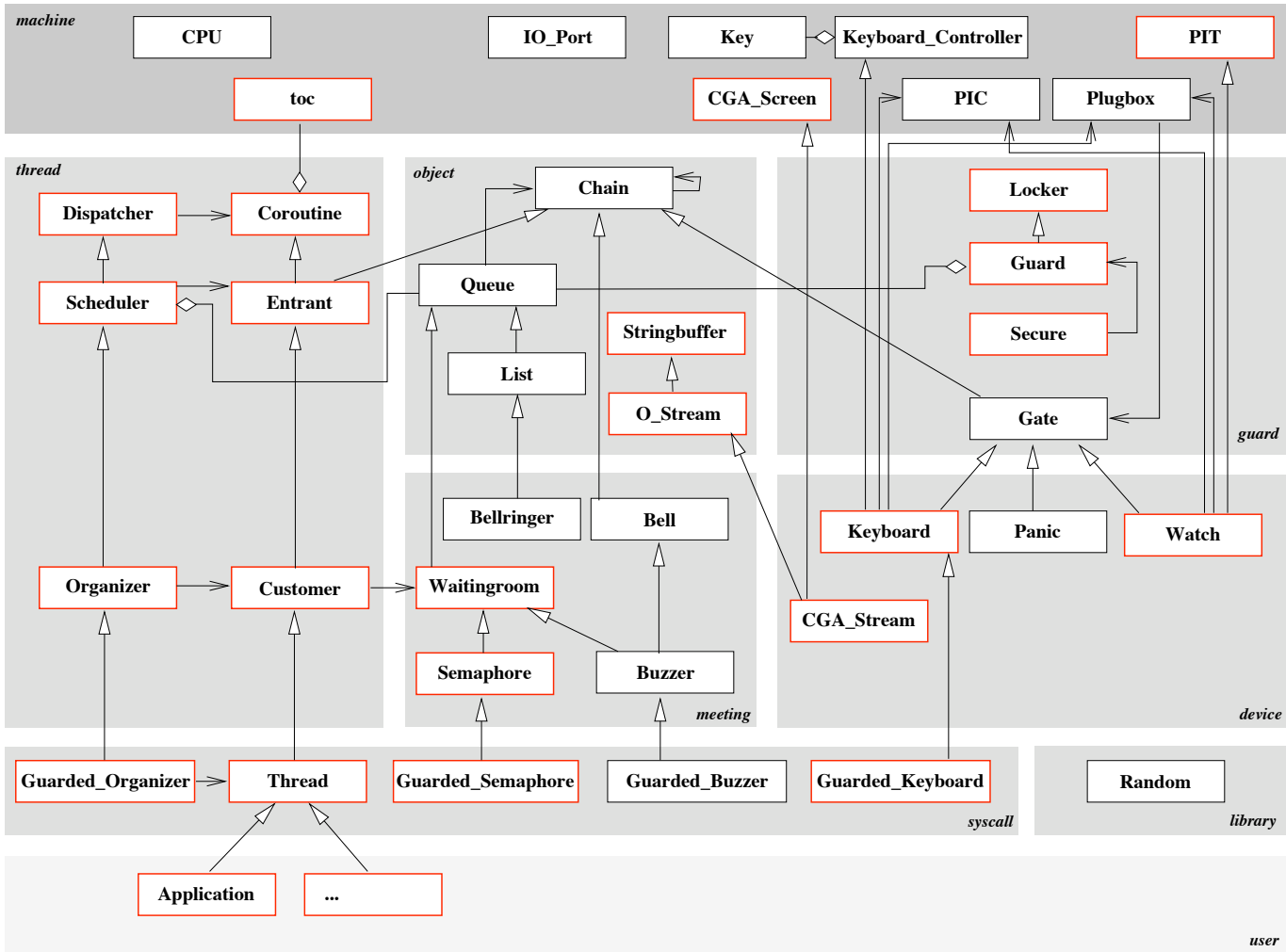
Drei inhaltliche Schwerpunkte!

2. Kontrollflüsse und ihre Interaktionen

Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

2. Kontrollflüsse und ihre Interaktionen



Was wir gemacht haben

Drei inhaltliche Schwerpunkte!

3. BS-Konzept allgemein und am Beispiel (Windows/Linux)

- VL_1 ***Einführung***
- **VL_2** ***BS-Entwicklung***
- VL_3 ***IRQs (Hardware)***
- VL_4 ***IRQs (Software)***
- VL_5 ***IRQs (Synchronisation)***
- VL_6 ***Intel IA-32***
- VL_7 ***Koroutinen und Fäden***
- **VL_8** ***Scheduling***
- **VL_9** ***BS-Architekturen***
- **VL_{10}** ***Fadensynchronisation***
- VL_{11} ***PC Bussysteme***
- **VL_{12}** ***Gerätetreiber***
- **VL_{13}** ***IPC***

Was wir gemacht haben

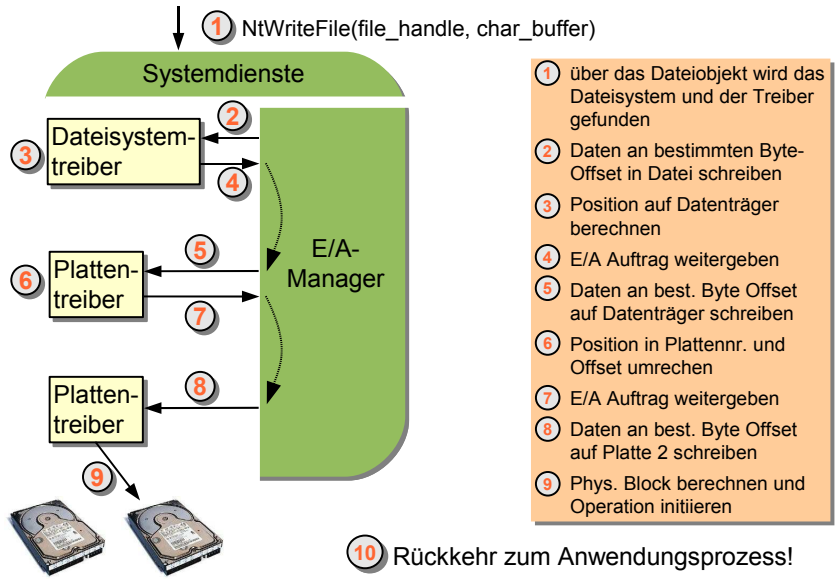
Drei inhaltliche Schwerpunkte!

3. BS-Konzept allgemein und am Beispiel (Windows/Linux)

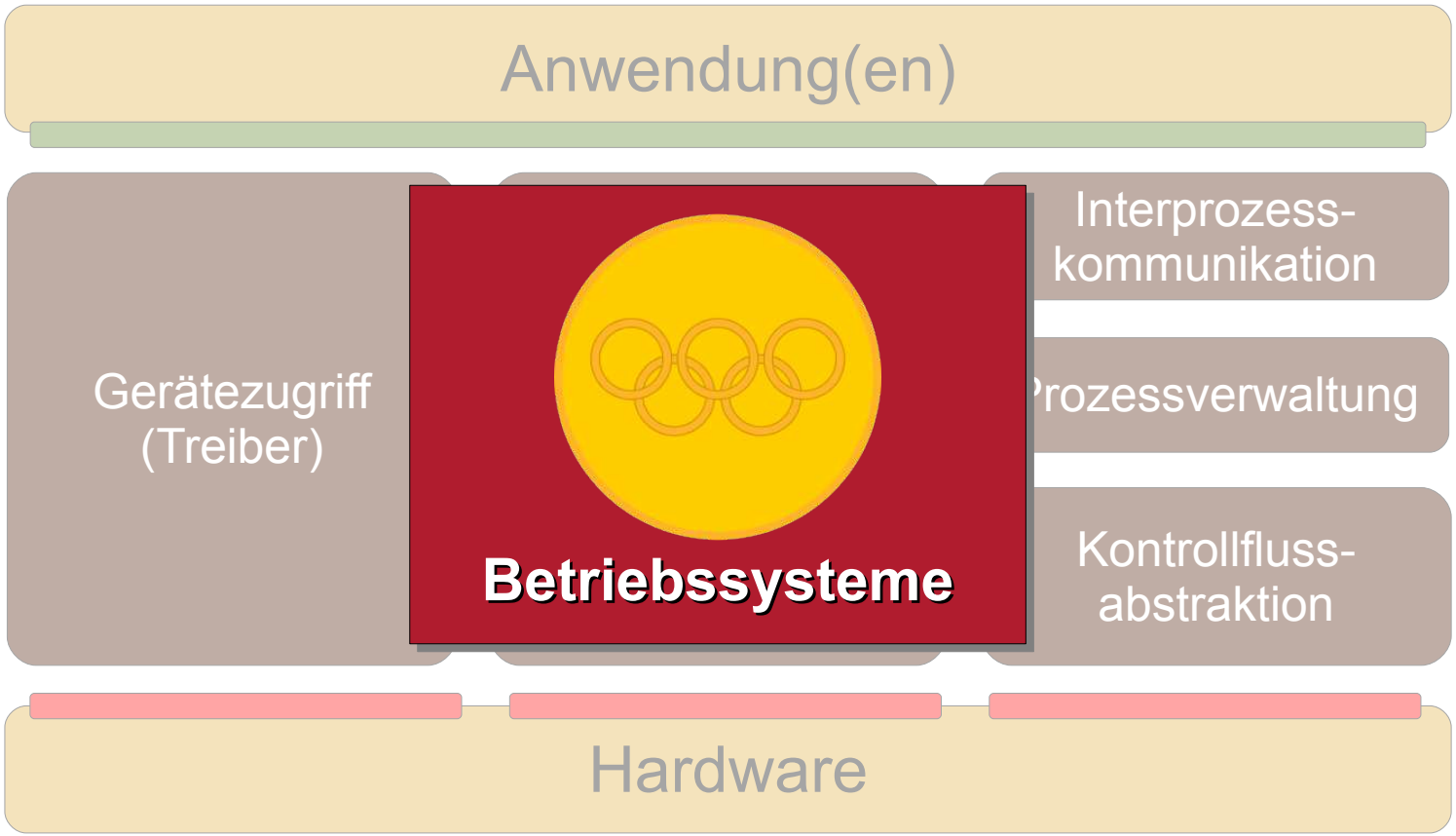
Linux *Tasks* ...

- sind die **Linux *Kernel*-Abstraktion** für ...
  - **UNIX Prozesse**: ein Kontrollfaden in einem Adressraum
  - **Linux *Threads***: spezieller Prozess, der sich seinen virtuellen Adressraum mit mindestens einem anderen *Thread* teilt
- sind die vom Scheduler betrachteten Aktivitäte
  - ein Programm mit vielen Threads bekommt unter Linux Rechenzeit (innerhalb einer *scheduling group)* als **ein** Prozess
  - gleiches gilt allerdings auch für ein Programm mit einem Prozess und vielen Kindprozessen

Windows – typischer E/A-Ablauf



Zusammen eine ganze Menge!



Realitätscheck: MPStuBS $\leftrightarrow$ “richtiges BS”

Es fehlt noch eine ganze Menge. . .

- Speicherverwaltung und Prozesskonzept
- Dateisystem und Programmlader
- Netzwerk und TCP/IP
- . . .

Realitätscheck: MPStuBS $\leftrightarrow$ “richtiges BS”

Es fehlt noch eine ganze Menge. . .

- Speicherverwaltung und Prozesskonzept
- Dateisystem und Programmlader
- **Netzwerk und TCP/IP**
- . . .

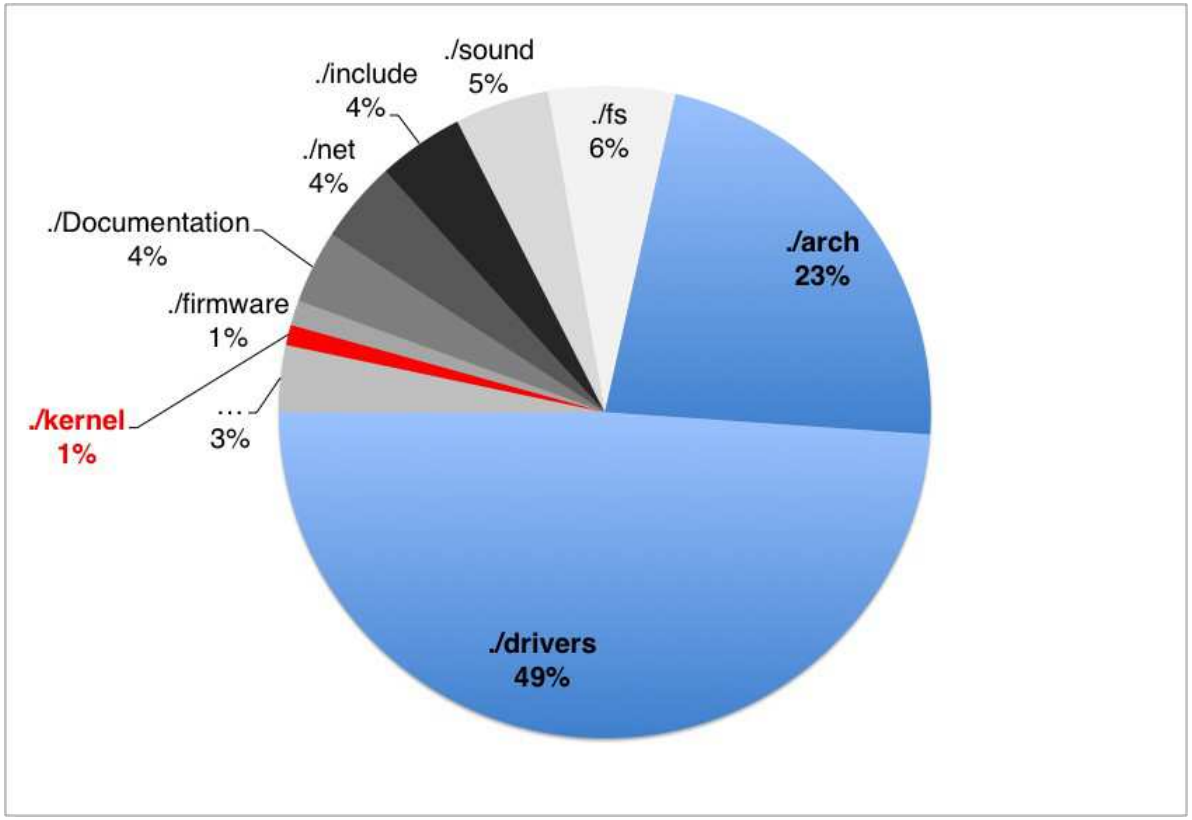
Realitätscheck: MPStuBS ↔ “richtiges BS”

Es fe

- Speic
- Datei
- **Netz**
- . . .

Bedeutung von Gerätetreibern (1)

- Anteil an Treibercode in Linux 3.2.1



Realitätscheck: MPStuBS $\leftrightarrow$ “richtiges BS”

Es fehlt noch eine ganze Menge. . .

- Speicherverwaltung und Prozesskonzept
- Dateisystem und Programmlader
- **Netzwerk und TCP/IP**
- . . .

Beispiel Linux [5]

Aug 91 Linux 0.01: bash, Dateisystem

Jan 92 Linux 0.12: Virtueller Speicher (Paging)

Mär 92 Linux 0.95: X-Windows, Unix Domain Sockets
(jetzt fehlte nur noch Netzwerk!)

Mär 94 Linux 1.00: **Netzwerk und TCP/IP**

Evaluation

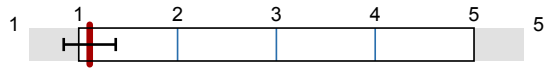
Evaluationsergebnisse

Globalindikator

Globalfragen für alle Lehrveranstaltungs-Typen (mit Gewichtung)

mw=1.23
s=0.42

Vorlesung im Allgemeinen



mw=1.11
s=0.26

Didaktische Aufbereitung

mw=1.31
s=0.51

Präsentation des Dozenten

mw=1.3
s=0.52

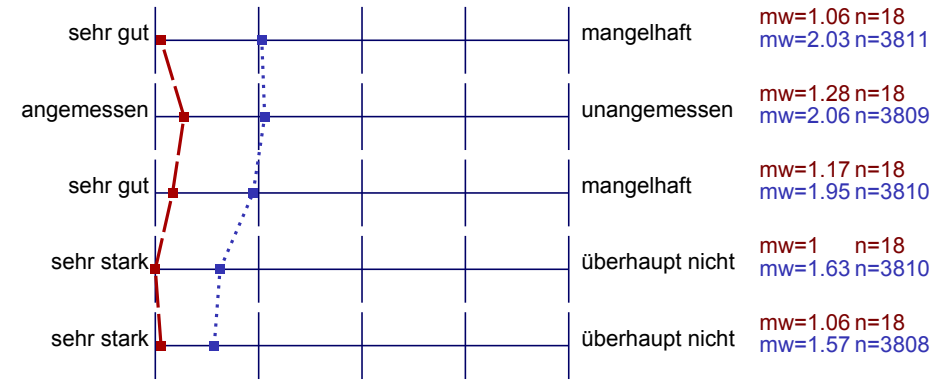
mw=1.19
s=0.4

Vergleich mit den Vorjahren

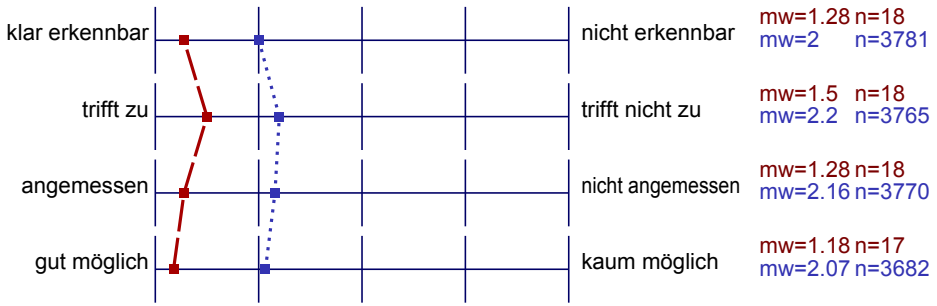
WS 12:	n=18	(51%)	mw=1.23
WS 11:	n=17	(53%)	mw=1.30
WS 10:	n=9	(29%)	mw=1.42
WS 09:	n=19	(100%)	mw=1.34
WS 08:	n=7	(27%)	mw=1.41
WS 07:	n=16	(50%)	mw=1.39

Profilline

Globalfragen für alle Lehrveranstaltungs-Typen (mit Gewichtung)



Vorlesung im Allgemeinen



Kommentare

- An der Lehrveranstaltung **gefällt mir besonders**
  - + sehr gut strukturiert + die wohl besten Vorlesungsfolien, die ich bisher in meinem Studium zur Verfügung hatte + wirkt alles sehr ausgereift, an dieser VL können sich andere LS ein Beispiel nehmen + Videos der VL werden zum Nachbereiten online zur Verfügung gestellt
  - Der Stoff ist sehr interessant. Die Folien sind sehr ansprechend gestaltet. Der Dozent weiß, wovon er redet und ist sehr engagiert. Die Umsetzung der vorgestellten Konzepte in realen Betriebssystemen ist gut und veranschaulicht der Stoff.
  - Der Vortragsstil ist super. Die Geschichten über z.B. das A20-Gate lockern den Stoff gut auf. Für mich eine der interessantesten Vorlesungen.
  - Gut strukturierter Inhalt, schöne Kombination aus Breiten- und Tiefenthemen. Motivierter, kompetenter Dozent, der den Stoff mit interessanten Anekdoten auflockert und wichtige Konzepte im Dialog mit den Teilnehmern erarbeitet.
  - Herr Lohmann :) Kompetent, motiviert, freundlich!
  - Hervorragende Vorlesung, motivierter Dozent, gute Folien. Weiter so! :-)

Kommentare (Forts.)

- Sehr interessante Übungen
- Super Veranstaltung. Mit Abstand beste Vorlesung dieses Semester. Auch die klare Terminstruktur (welche Vorlesung gehoert zur welcher Uebung, wann ist Abgabe, usw) ist sehr hilfreich.
- Tolle VL, guter Dozent
- Viele Beispiele, wo die gezeigten Konzepte tatsächlich angewendet werden, wie Betriebssysteme in der Praxis funktionieren. Top motivierter Dozent!
- sehr interessanter Stoff der gut vermittelt wird, interessante Anekdoten super Vorlesung
- sehr motivierter dozent

Kommentare (Forts.)

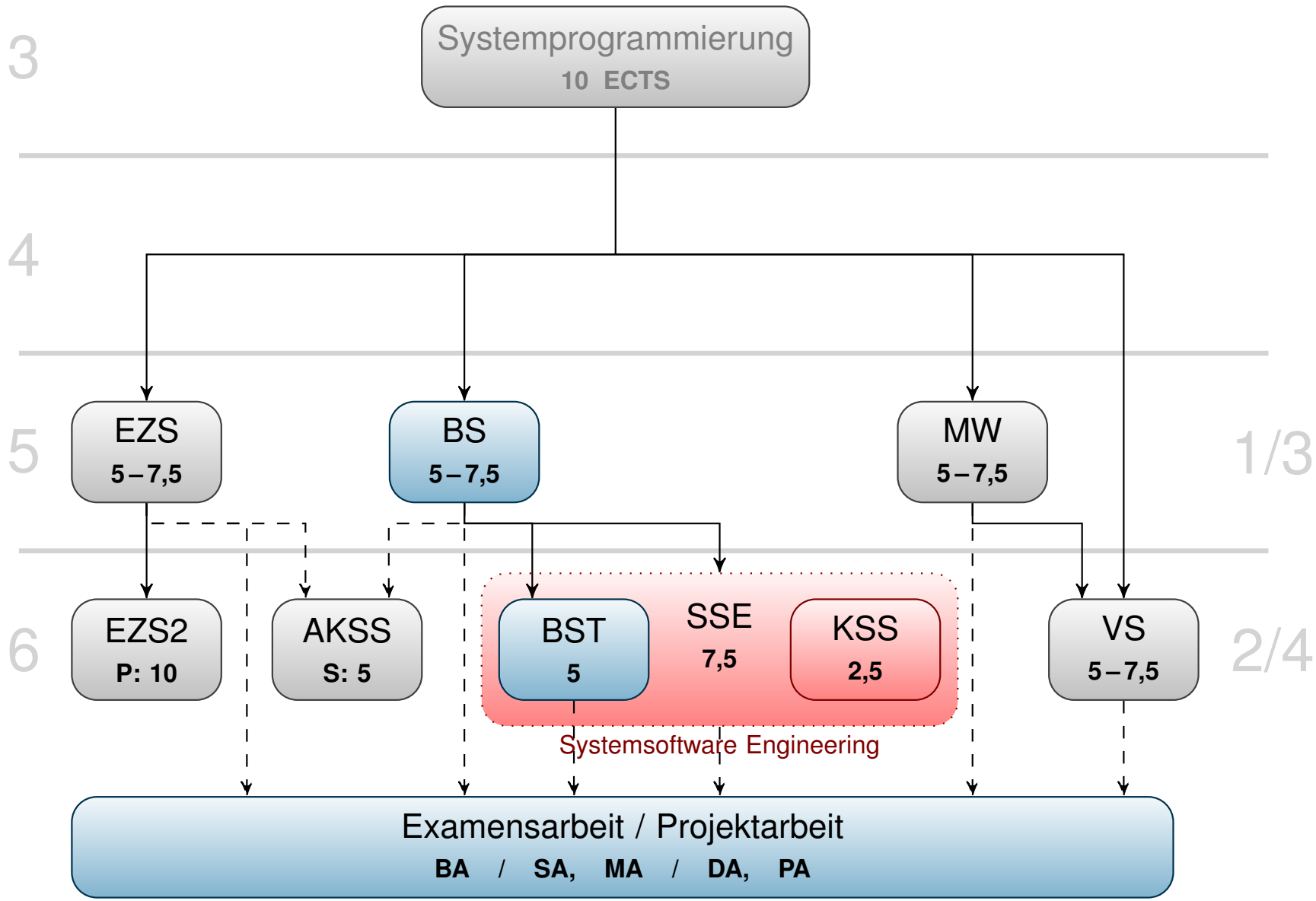
- An der Lehrveranstaltung **gefällt mir weniger**
  - Hinweise welche Inahlte jeweils besonders relevant/ kaum relevant fuer Pruefung sind waere noch hilfreich
  - Mit jeder Vorlesung steigt die Verwunderung, warum heutige Betriebssysteme überhaupt funktionieren ;-)
  - Wer, anders als wohl die meisten Teilnehmer, wenig Vorkenntnisse aus der Systemecke hat, wird ab und an abgehängt, weil manche Begriffe aus der Praxis, die man nicht aus dem Grundstudium kennt, häufig erstmal ohne Erklärung verwendet werden (als Beispiel: real mode). Bei Programmiertechnischem kommt man aber in der Regel auch ohne zusätzliches Vorwissen ganz gut mit.
  - Zwischenkommentare die zu stark vom aktuellen Thema abweichen etwas frueher unterbinden.
  - alles hat gepasst

Kommentare (Forts.)

- Weiterhin möchte ich **anmerken**
  - Bitte weiterhin so viel Spaß und Freude am Stoff vermitteln!
  - Die Videos aus dem Vorjahr waren super praktisch um versäumte Termine nachzuholen.
  - gutes Zeitmanagement des Dozenten
  - Freue mich schon auf weitere Veranstaltungen

Wie geht es weiter?

(Bachelor/Master)



Masterseminar Energiegewahre Systemsoftware

1980er **2010er**

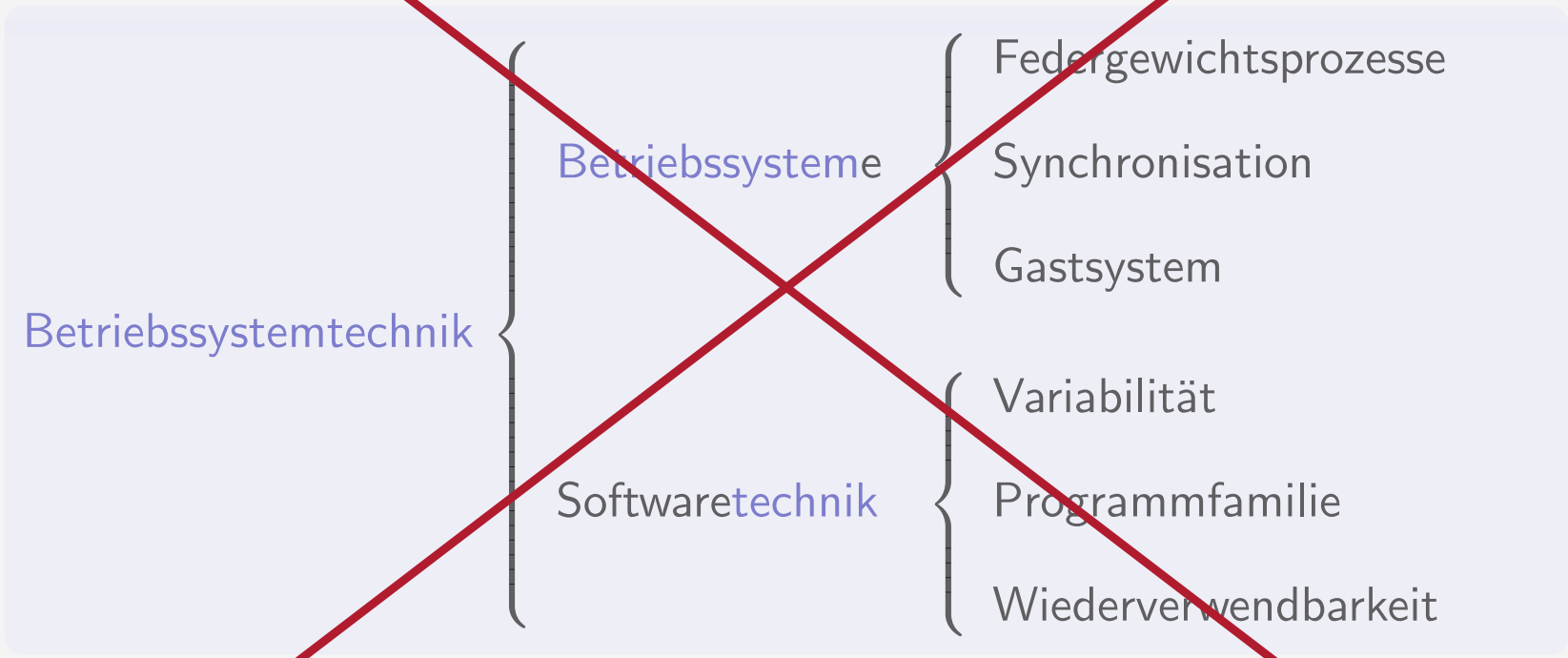
Masterseminar: Energiegewahre Systemsoftware

- Masterseminar, Ausgewählte Kapitel der Systemsoftware: Energiegewahre Systemsoftware
- 4 SWS bzw. 5 ECTS
- Das Seminar beleuchtet das breite Spektrum energiegewahrer Systemsoftware aus unterschiedlichen Richtungen, u. a.:
  - Energieeffiziente Betriebssystemkomponenten, hardwaregestützte Energiesparmechanismen
  - Energieverwaltung im Linux-Betriebssystem
  - Energieeffiziente Systemsoftware für Cloud-Computing
- Aktuelle Forschungsarbeiten aus dem Systems-Bereich
- Themenliste online:

  `http://www4.cs.fau.de/Lehre/SS13/MS_AKSS`
- Themen können bereits jetzt reserviert werden. Schickt dafür einfach eine E-Mail an Timo `<thoenig@cs.fau.de>`

Ausblick: Betriebssystemtechnik (BST)

Hinter der Kulisse: BST in aller Kürze...



Neu: Betriebssystemtechnik (BST)

Betriebssystemtechnik

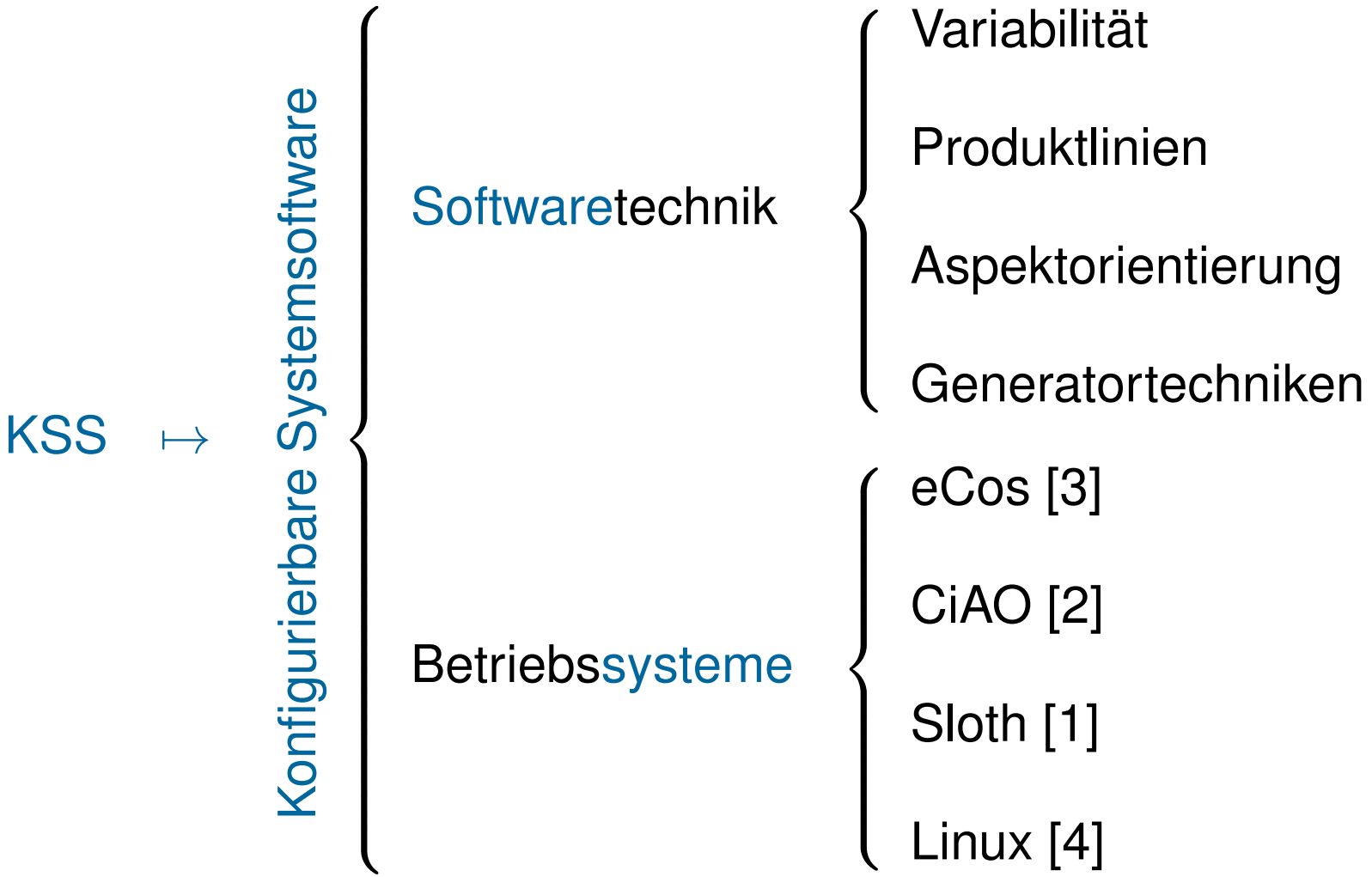
Adressräume: Trennung, Zugriff, Schutz

Einleitung

Wolfgang Schröder-Preikschat

16. April 2013

Hinter der Kulisse: KSS in aller Kürze...



Motivation: Special-Purpose Systems







“Between a Rock and a Hard Place”

functional and nonfunctional *requirements*

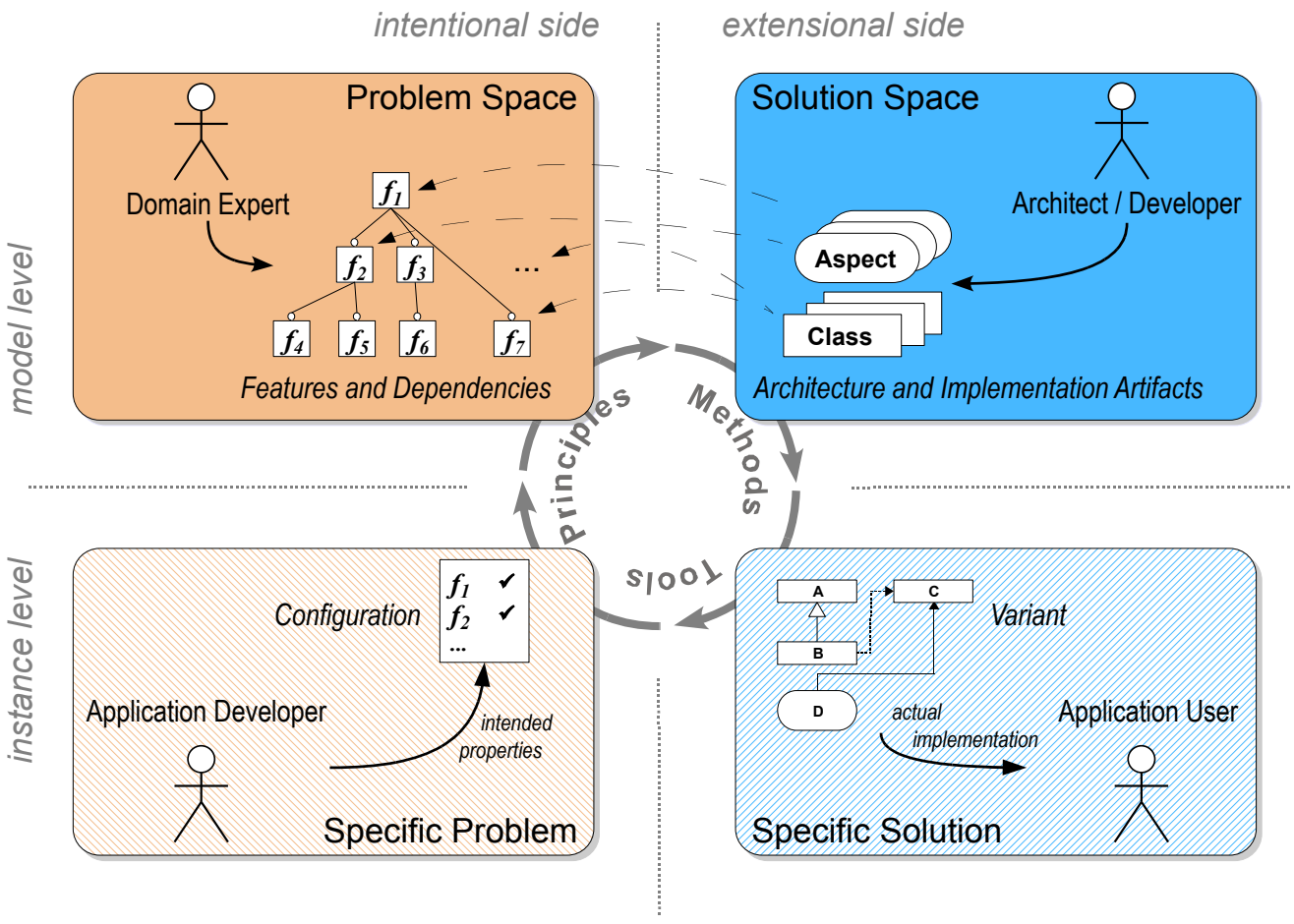
tasks
sockets
file system
...
event latency
safety
...

S y s t e m S o f t w a r e

ISA
IRQ handling
MMU / MPU
...
cache size
coherence
IRQ latency
...

functional and nonfunctional *properties*

Configurable Software ➟ Product Line



The State of the Art: eCos

The **e**mbedded **C**onfigurable **OS**

- operating system for embedded applications
- open source, maintained by eCosCentric
- broadly accepted real-world system

More than **750** configuration options

- feature-based selection
- **preprocessor-based** implementation

➥ This has a **severe impact** on the code!

eCos – Implementation of Configurability

```
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_                                  N_PROTOCOL_DEFAULT_PRIORITY;
#endif
#ifdef CYGSEM_KERNEL_SY                                  L_DEFAULT_NONE
  protocol    = NONE;
#endif
#else // not (DYNAMIC a
#ifdef CYGSEM_KERNEL_SY                                  L_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
  // Otherwise set it to zero.
  ceiling = 0;
#endif
#endif
#endif // DYNAMIC and DEFAULT defined
  CYG_REPORT_RETURN();
}
```

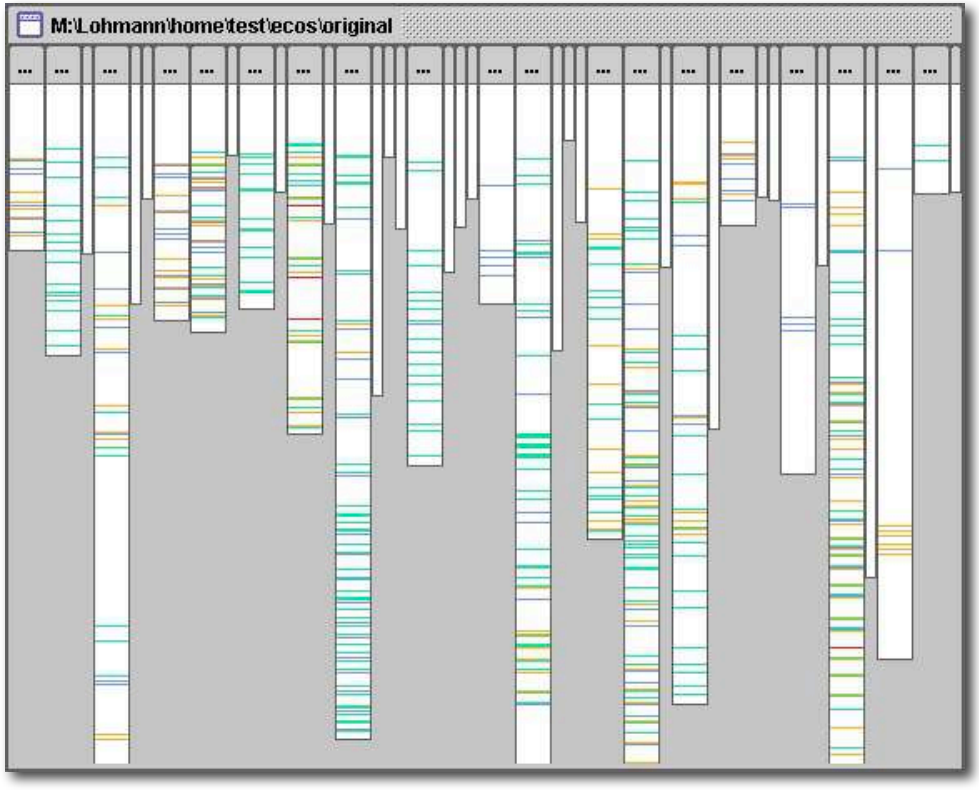
```
Cyg_Mutex::Cyg_Mutex() {
  locked      = false;
  owner       = NULL;
}
```

Mutex options:

- PROTOCOL
- CEILING
- INHERIT
- DYNAMIC

Kernel policies: Tracing | Instrumentation | Synchronization

Issue: Crosscutting Concerns



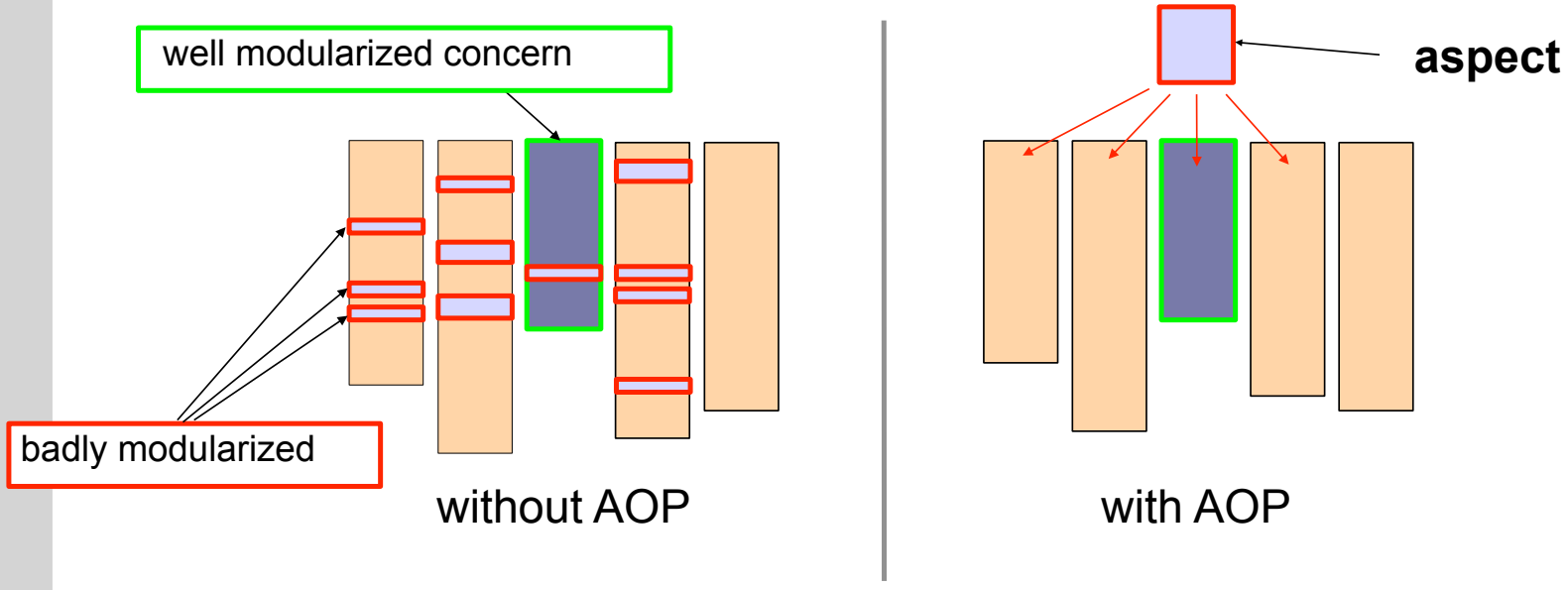
Mutex options:

- PROTOCOL
- CEILING
- INHERIT
- DYNAMIC

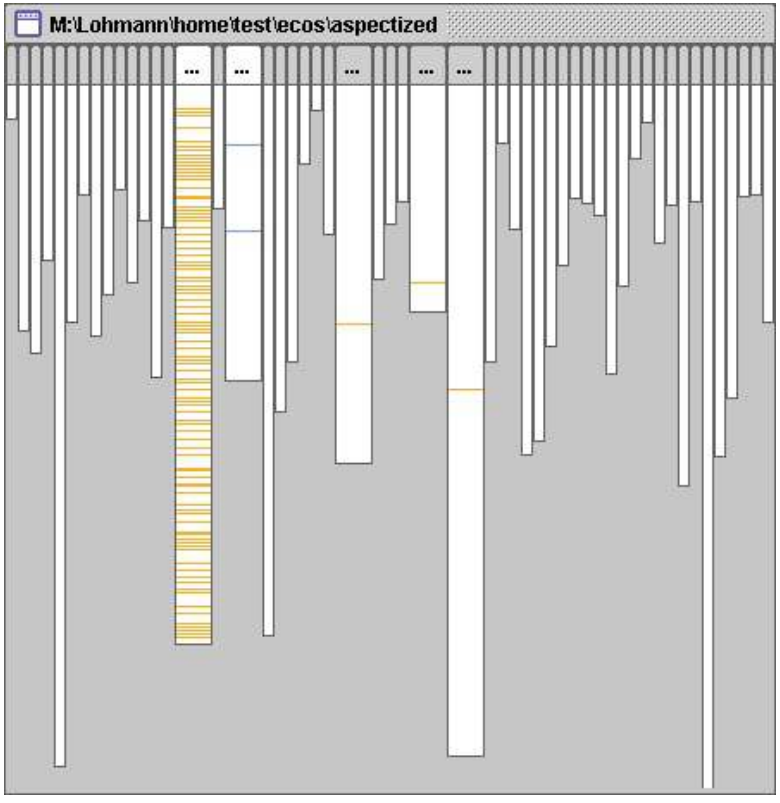
Kernel policies: Tracing | Instrumentation | Synchronization

Solution Idea: Aspect-Oriented Programming

AOP provides language means to encapsulate crosscutting and scattered concerns



Qualitative Results: eCos → AspeCos



[EuroSys '06]

Kernel policies: Tracing | Instrumentation | Synchronization

Example: Synchronization in AspeCos

```
aspect int_sync {

  pointcut sync() = execution(...) // kernel calls to sync
           || construction(...)
           || destruction(...);                                  where

  // advise kernel code to invoke lock() and unlock()
  advice sync() : before() {
    Cyg_Scheduler::lock();
  }                                                              what
  advice sync() : after() {
    Cyg_Scheduler::unlock();
  }

  // In eCos, a new thread always starts with a lock value of 0
  advice execution(
  "%Cyg_HardwareThread::thread_entry(...)") : before() {
    Cyg_Scheduler::zero_sched_lock();
  }
  ...
};
```

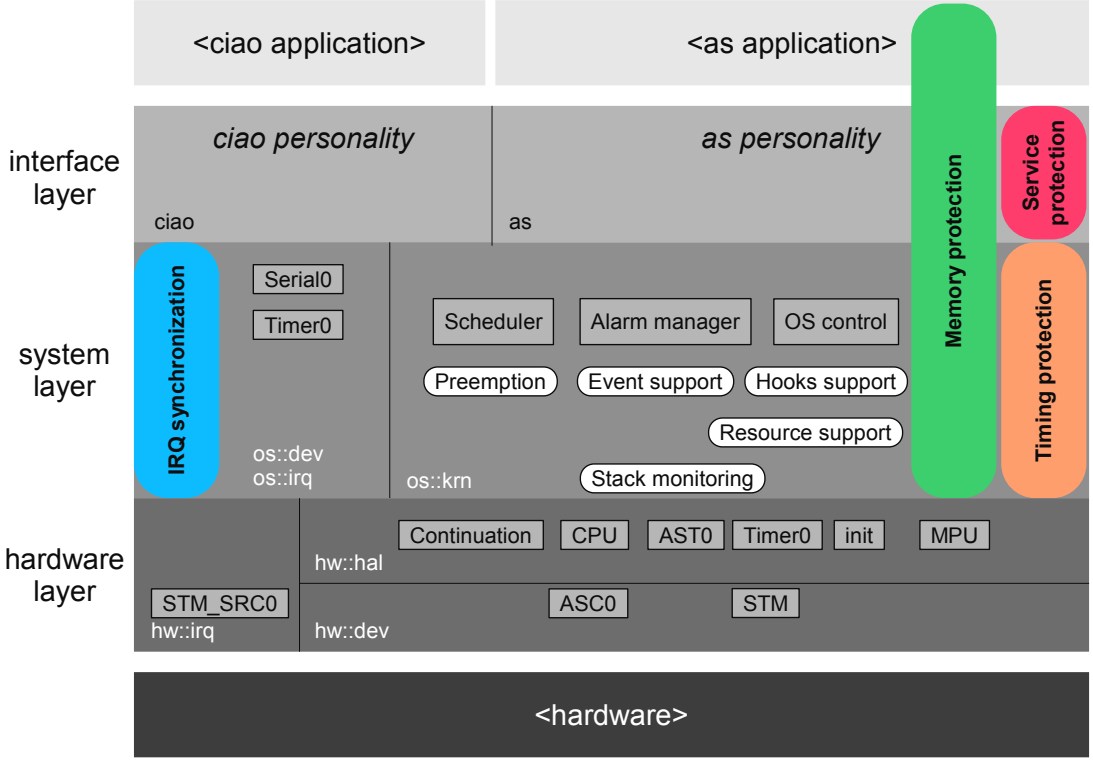
CiAO – CiAO is Aspect-Oriented

AUTOSAR

A family of aspect-oriented operating systems

[USENIX '09, AOSD '11]

- AUTOSAR-OS–like functionality
- configurability of even fundamental system policies
- achieved by aspect-aware design



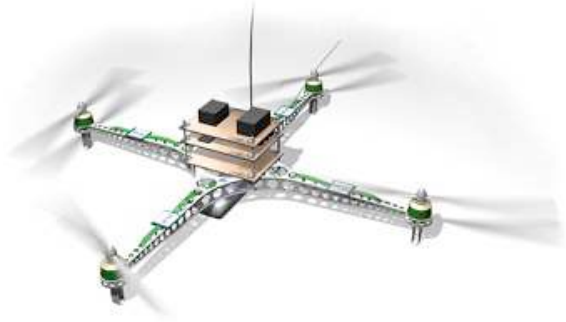
CiAO – CiAO is Aspect-Oriented

AUTOSAR

A family of aspect-oriented operating systems

[USENIX '09, AOSD '11]

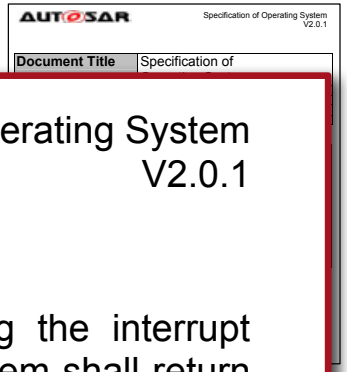
- AUTOSAR-OS–like functionality
- configurability of even fundamental system policies
- achieved by aspect-aware design



test scenario	CiAO	ProOSEK
(a) voluntary task switch	*160*	*218*
(b) forced task switch	*108*	*280*
(c) preemptive task switch	*192*	*274*
(d) system startup	*194*	*399*

Evaluation Case Study: CiAO-AS

CiAO



AUTOSAR

Specification of Operating System
V2.0.1

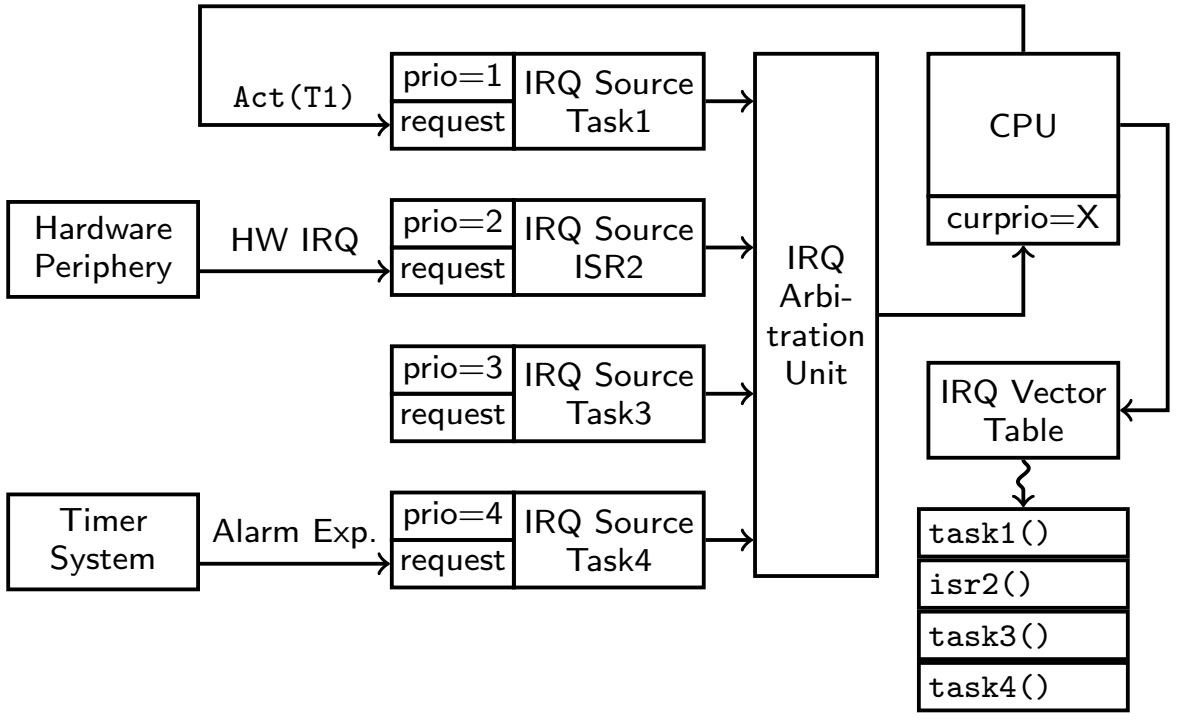
OS093: If interrupts are disabled and any OS services, excluding the interrupt services, are called outside of hook routines, then the Operating System shall return `E_OS_DISABLEDINT`

```
aspect DisabledIntCheck {
  advice call( pcOSServices() && !pcInterruptServices() )
  && !within( pcHookRoutines() ) : around() {
    if( interruptsDisabled() )
      *tjp->result() = E_OS_DISABLEDINT;
    else
      tjp->proceed();
} };
```

SLOTH: Threads as Interrupts

- **Idea: threads are interrupt handlers, synchronous thread activation is IRQ**
- Let interrupt subsystem do the scheduling and dispatching work
- Applicable to priority-based real-time systems
- Advantage: small, fast kernel with unified control-flow abstraction

SLOTH Design

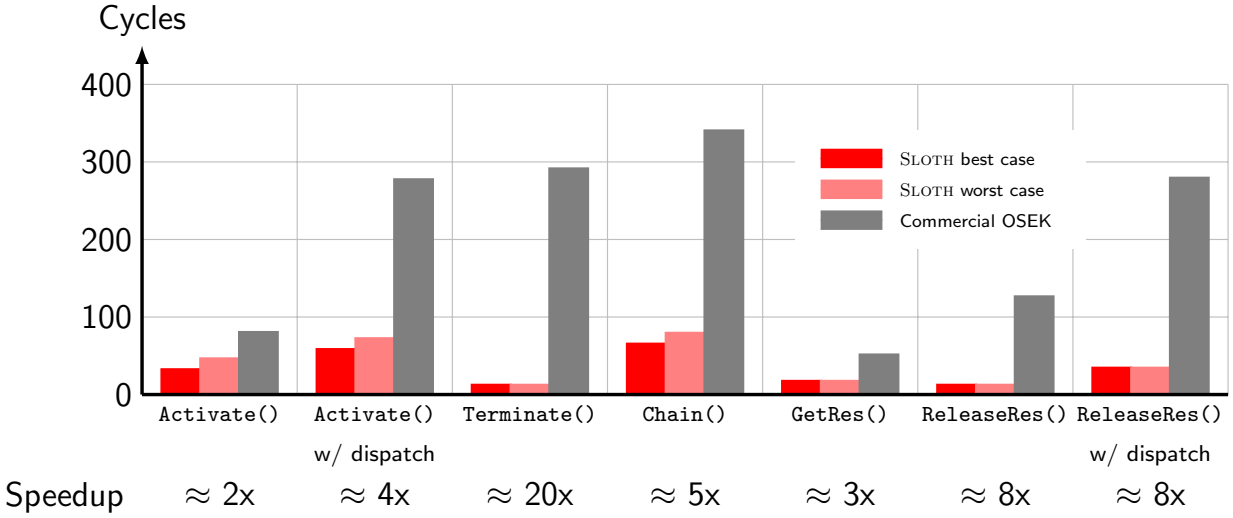


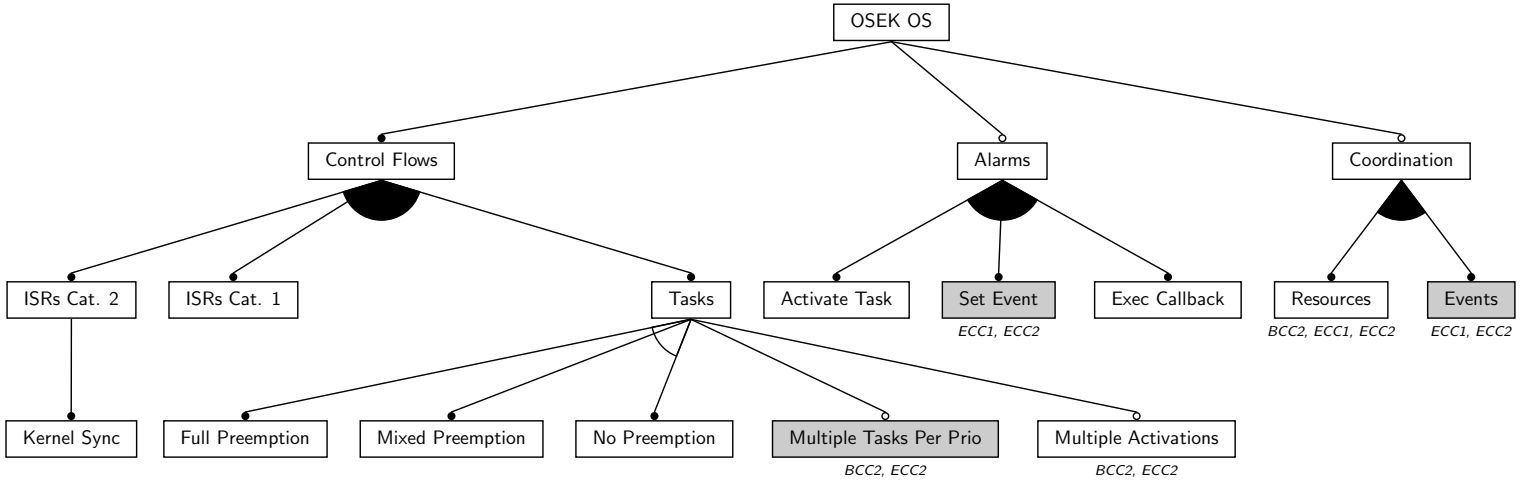
- Platform must support IR priorities and software IR triggering

Ausblick: Konfigurierbare Systemsoftware (KSS)

SLOTH

[RTSS '09]

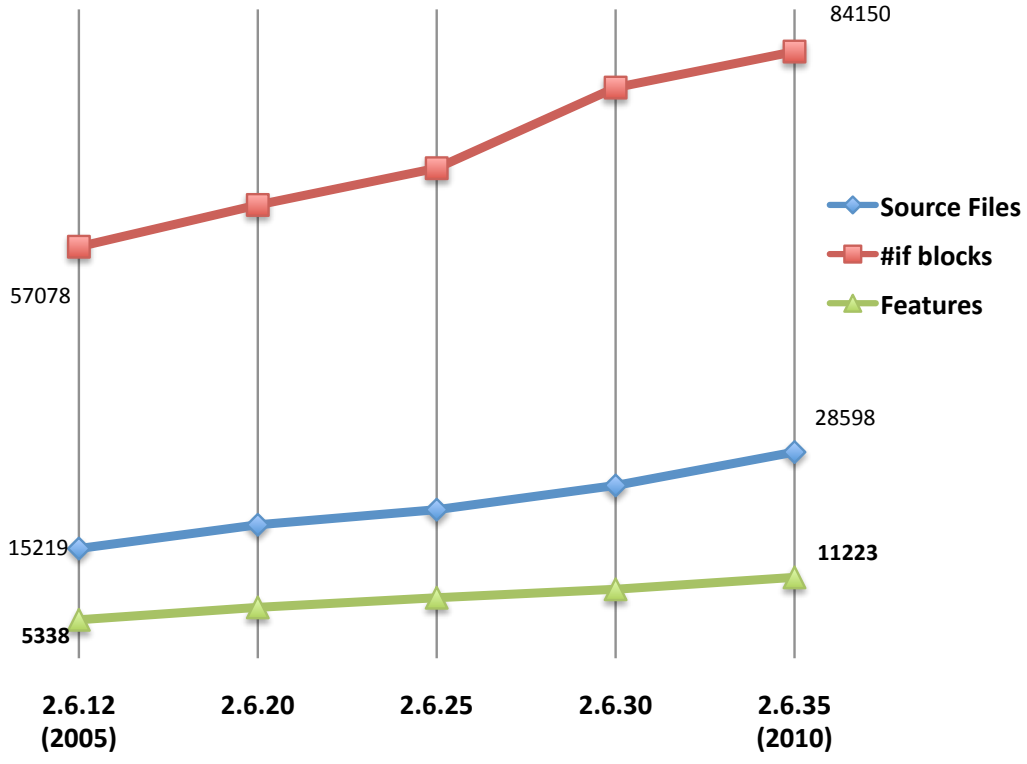




Configurability in the Large: Linux

More than **11,000** configuration options!

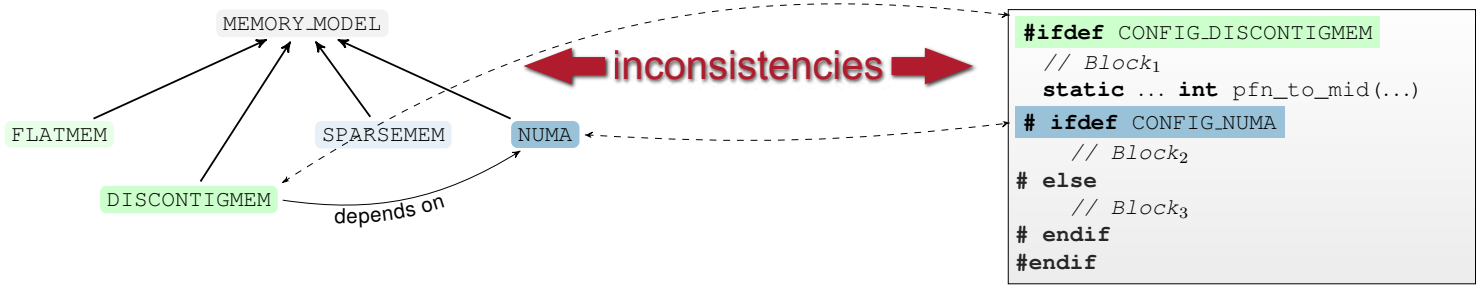
- 85,000 #ifdef blocks, sprinkled over 29,000 source files
- numbers have **doubled** within the last five years!



Configurability in the Large: Linux

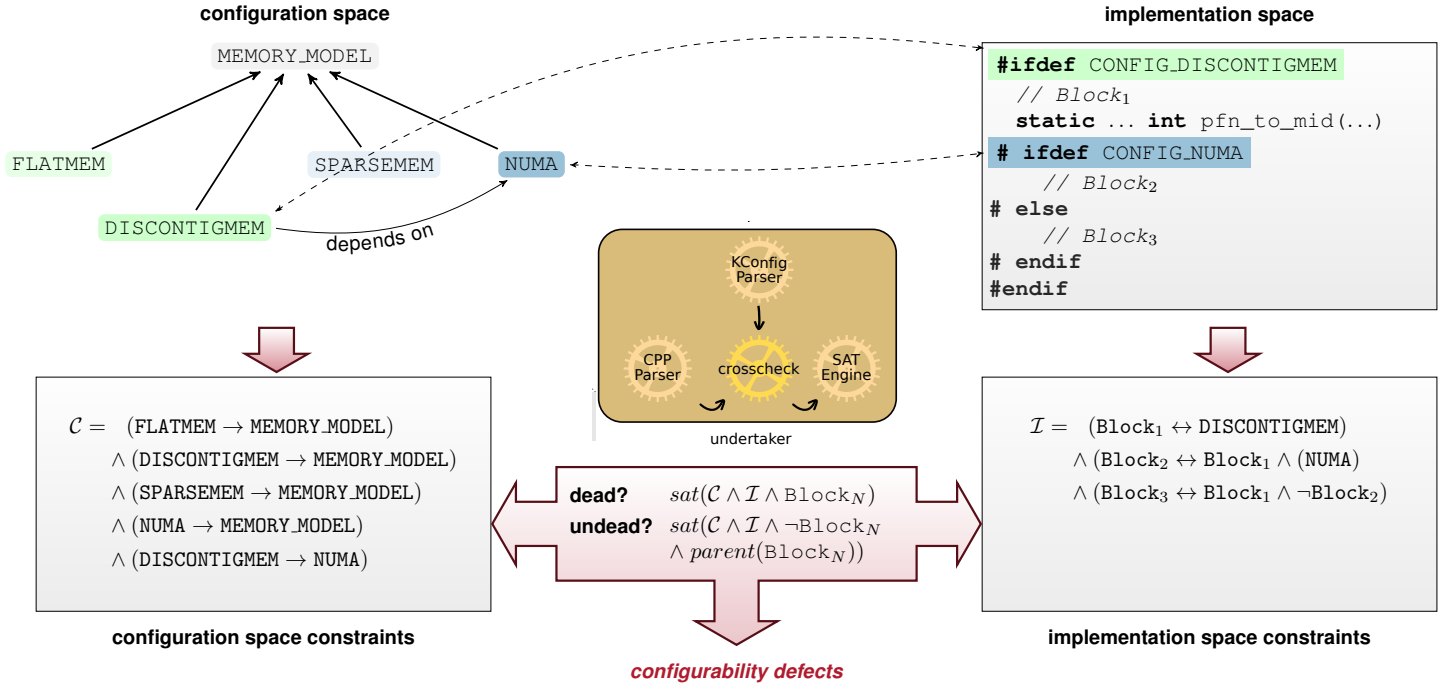
More than **11,000** configuration options!

- 85,000 #ifdef blocks, sprinkled over 29,000 source files
- numbers have **doubled** within the last five years!



The Undertaker

[EuroSys '11]



- found **1,776** defects (and that is just a lower bound!)
  - proposed fix for 364 (including 20 new bugs)
  - 123 patches submitted (49 merged into Linus-Tree)
  - removed 5,129 lines of *unnecessary* #ifdef-code
- tool suite now published as open-source project

Examensarbeiten am LS4

Zur Zeit im Angebot:

- Bachelorarbeiten
- Masterarbeiten
- Projektarbeiten

`http://www4.informatik.uni-erlangen.de/DE/Theses/`

Das war's :-)

Das Lehrstuhl 4 BS-Team wünscht erfolgreiche und erholsame "Semesterferien"




. . . und ein Wiedersehen
im Sommersemester 2013!

Referenzen

[1] Wanja Hofer, Daniel Lohmann, Fabian Scheler, et al. “Sloth: Threads as Interrupts”. In: *Proceedings of the 30th IEEE International Symposium on Real-Time Systems (RTSS ’09)*. IEEE Computer Society Press, Dec. 2009, pp. 204–213. ISBN: 978-0-7695-3875-4. DOI: 10.1109/RTSS.2009.18.

[2] Daniel Lohmann, Wanja Hofer, Wolfgang Schröder-Preikschat, et al. “CiAO: An Aspect-Oriented Operating-System Family for Resource-Constrained Embedded Systems”. In: *Proceedings of the 2009 USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, June 2009, pp. 215–228. ISBN: 978-1-931971-68-3. URL: http://www.usenix.org/event/usenix09/tech/full_papers/lohmann/lohmann.pdf.

[3] Daniel Lohmann, Fabian Scheler, Reinhard Tartler, et al. “A Quantitative Analysis of Aspects in the eCos Kernel”. In: *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2006 (EuroSys ’06)*. (Leuven, Belgium). Ed. by Yolande Berbers and Willy Zwaenepoel. New York, NY, USA: ACM Press, Apr. 2006, pp. 191–204. ISBN: 1-59593-322-0. DOI: 10.1145/1218063.1217954.

Referenzen (Forts.)

[4] Reinhard Tartler, Daniel Lohmann, Julio Sincero, et al. “Feature Consistency in Compile-Time-Configurable System Software: Facing the Linux 10,000 Feature Problem”. In: *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2011 (EuroSys ’11)*. (Salzburg, Austria). Ed. by Christoph M. Kirsch and Gernot Heiser. New York, NY, USA: ACM Press, Apr. 2011, pp. 47–60. ISBN: 978-1-4503-0634-8. DOI: 10.1145/1966445.1966451.

[5] Linus Torvalds and David Diamond. *Just for Fun: The Story of an Accidental Revolutionary*. HarperCollins, 2001. ISBN: 978-0066620725.