



AOSTA

Aspects in Operating Systems: Tools and Applications

*Michael Engel, Philipps-Universität Marburg
AG Verteilte Systeme
engel@informatik.uni-marburg.de*

Aspektorientierung in 5 Minuten ;-)

- *Crosscutting concerns (CCCs)*
 - *nicht mit vorhandener Modularisierungs-Struktur vereinbar*
 - *Code ist “tangled” und “scattered”*
 - *Joinpoint = Punkt, an dem CCC in den Codefluß integriert werden soll*
 - *Pointcut = Beschreibung einer Menge von Joinpoints*
 - *Beispiele: Logging, Security, Quality of Service*

Aspektorientierung in 5 Minuten ;-)

- *Modularisierung des Codes, der CCCs implementiert, als “advice”*
- *Pointcut-Beschreibung + Advice-Code = Aspekt*
- *Einfügen (“weaving”) in den existierenden Code*
 - *statische AOP = Deployment zur compile time*
 - *dynamische AOP = Deployment zur runtime*
- *AOP für refactoring existierenden Codes...*
- *...und zur Implementierung neuer Eigenschaften*

AOSTA

- *Projekt: Infrastruktur für Aspekte in Native Code*
- *Einsatz im Umfeld von Betriebssystem-Kernen*

AOSTA: Struktur

- *Identifikation von crosscutting concerns in prozeduralem (C) Code*
- *TOSKANA: Toolkit für dynamische AOP in kernel space*
- *Tool support (compiler backend, linker) für verbesserten Support dynamischer AOP in native code*
- *Aspects in the execution layer*
- *Applikationen*
 - *Cross-layer problems*
 - *Energy conservation*

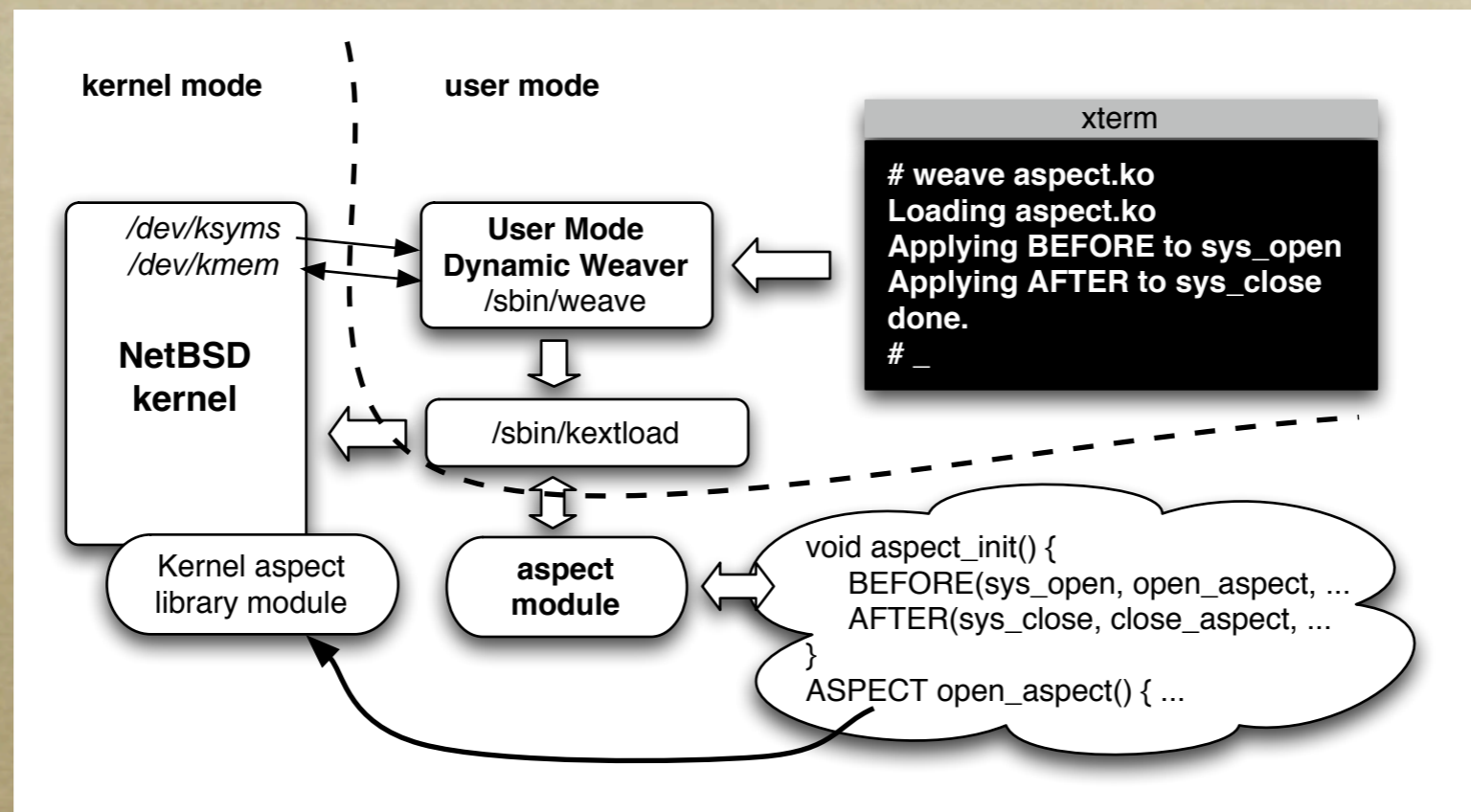
Identifikation von CCCs

- *Identifikation von tangling und scattering*
 - *In C-Code*
- *Ähnlichkeitssuche (duplizierter/ähnlicher Code)*
- *Abhängigkeitssuche (call graphs)*
- *Analyse der Evolution (cvs log dependencies)*
- *Ermittlung eines Maßes für die “Wahrscheinlichkeit” eines ccc’s*
 - *Hinweise für human experts für die Identifikation*

TOSKANA

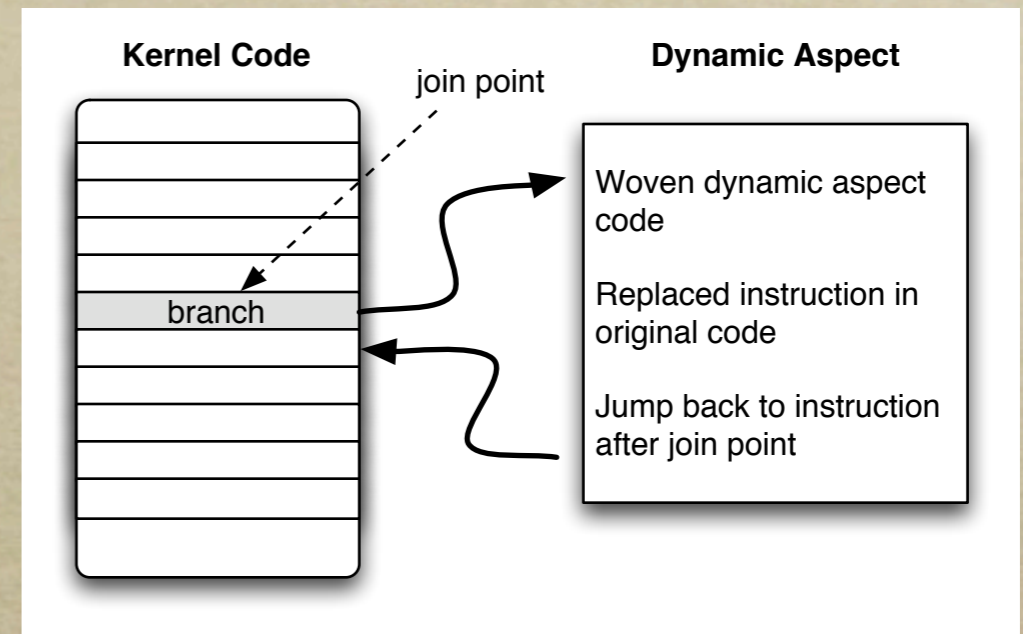
- *“Toolkit for Operating System Kernel Aspects with Nice Applications” ;-)*
- *Dynamische AOP in NetBSD*
- *before/after/around-advice für Funktionen im Kernel*
- *advice als Kernelmodul*
- *portabel - Darwin, L4 service (geplant)*

TOSKANA



- *Struktur von TOSKANA*

TOSKANA Code Splicing



- *Basis: Instrumentation*
- *Ersetzen von Instruktionen durch branch*
- *Advice Code Ausführung*
- *Rücksprung zum join point*

Toolkit Support

- *Probleme des deployment von advice in native code:*
 - *localization of inlined functions*
 - *avoiding optimization over inlined function borders*

Aspects in the execution layer

- *Problem: Joinpoints auf Funktionsaufrufe beschränkt*
- *Erweiterung des Joinpoint-Modells mit existierenden Methoden nicht machbar*
- *Lösungsansatz: low-level virtual machine im Kernel*
 - *LLVM (Univ. of Illinois), VVM (INRIA)*
- *Aufbauend auf L4 Microkernel: Servers implementiert als VM-Bytecode*

Anwendungen

- *Cross-layer problems*
- *Kernelstruktur weist verschiedene Layerstrukturen auf, z.B.:*
 - *TCP/IP Stack*
 - *Virtual File System Layer*
- *Cross-Layer-Probleme sind crosscutting:*
 - *TCP/IP: QoS, Packet Filtering*
 - *VFS Layer: Mounting/unmounting, permission control, locking*

Anwendungen

- *Energy conservation*
 - *Verschiedene Komponenten eines Systems sind softwaremäßig abschaltbar bzw. frequenz- und spannungsreduzierbar - Code ist scattered und tangled*
 - *Frequenzskalierung/Spannungsskalierung*
 - *Scheduling*
 - *Accounting, Profiling*
 - *Device-Treiber Code*
 - *Deaktivierung und Reaktivierung*
 - *Interrupt management, Lock management*
 - *Virtual Memory (swap space, buffer space)*
 - *Device Driver Code*