

Betriebssystemtechnik

Operating System Engineering (OSE)

Querschneidende Belange



1

Was ist ein Belang (concern)?

„A **concern** is an area of interest or focus in a system.“

Glossar bei www.aosd.net

Belange können sehr unterschiedlich sein:

- „Pi soll auf 20 Dezimalstellen genau berechnet werden“
- „Reaktion erfolgt nach spätestens 5 µs“
- „detaillierte Fehlerinformationen im Fehlerfall ausgeben“

Belang = Anforderung?

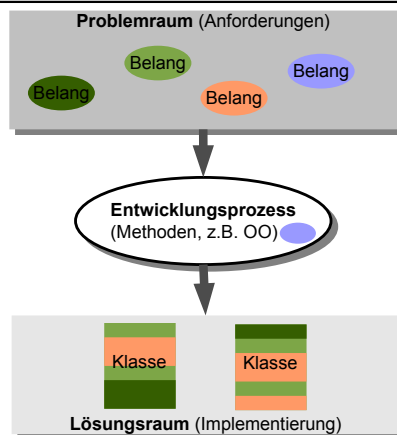
- Im Prinzip ja. Man verwendet jedoch lieber „Belang“, wenn es darum geht, was aus einer „Anforderungen“ im Zuge der Entwicklung wird.
- Anforderungen sind „dokumentierte Belange“



© 2007 Olaf Spinczyk

2

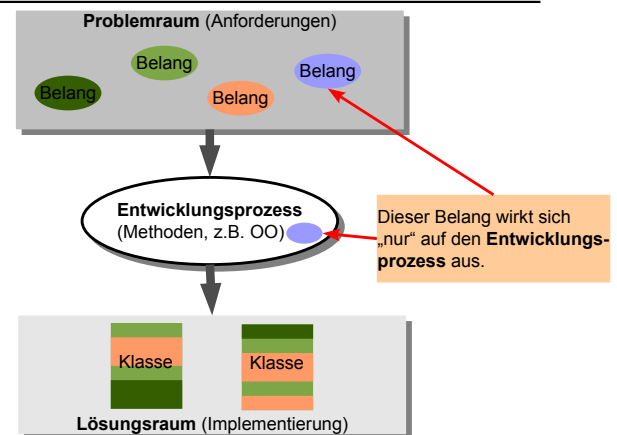
Belange auf dem Weg in die Lösung



© 2007 Olaf Spinczyk

3

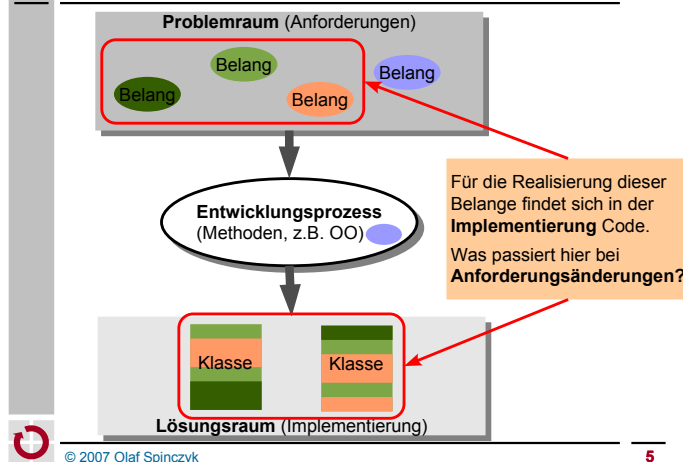
Belange auf dem Weg in die Lösung



© 2007 Olaf Spinczyk

4

Belange auf dem Weg in die Lösung



Belange auf dem Weg in die Lösung

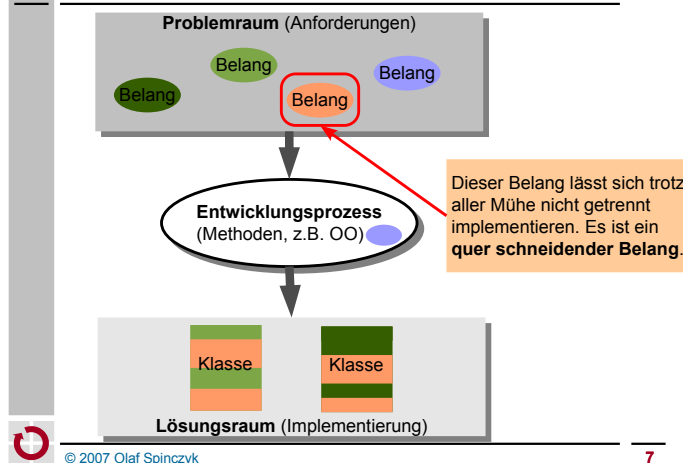
Prinzip der Trennung der Belange (separation of concerns - SoC):

„Dinge, die nichts miteinander zu tun haben, sind auch getrennt unterzubringen und zu behandeln.“

→ Ziel: Jedes Artefakt der Implementierung ist ein Modell genau **eines Belangs**



Belange auf dem Weg in die Lösung



Quer schneidende Belange - Beispiele

- „auf einen Fehlerzustand nach Aufruf einer libc Funktion soll mit dem Werfen einer *exception* reagiert werden“
 - nach **jedem libc Funktionsaufruf** muss ein Test erfolgen
 - quer schneidend in der Instruktionsabfolge (**dynamisch**)
- „garbage collection durch *reference counting*“
 - jedes Objekt **jeder Klasse** muss einen Zähler enthalten
 - quer schneidend in der Programmstruktur (**statisch**)
- „vor dem Betreten des BS-Kerns durch die Anwendung muss ein *lock* gesetzt werden.“
 - **alle system calls** werden abgesichert

Belange auf dem Weg in die Lösung

Quer schneidender Belang
(crosscutting concern - CCC):

„Ein quer schneidender Belang ist dadurch charakterisiert, dass er in der gewählten Modellierungssprache nicht als eigenständige Entität ausdrückbar ist“

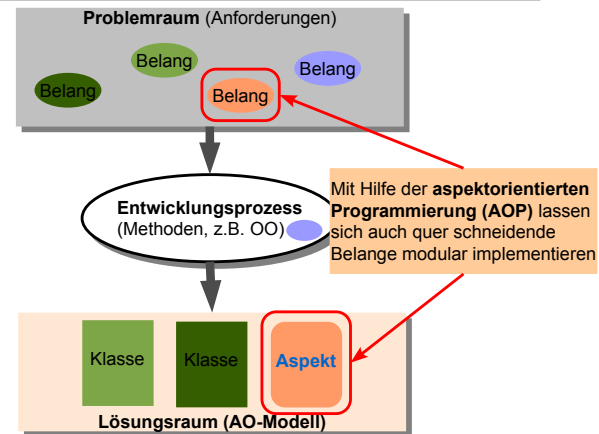
→ Ziel: Auch quer schneidende Belange modular im Sinne des SoC implementieren

Lösungsraum (Implementierung)

© 2007 Olaf Spinczyk

9

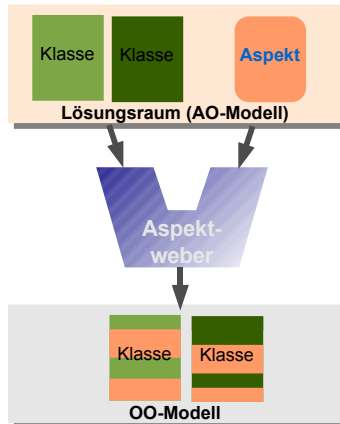
Belange auf dem Weg in die Lösung



© 2007 Olaf Spinczyk

10

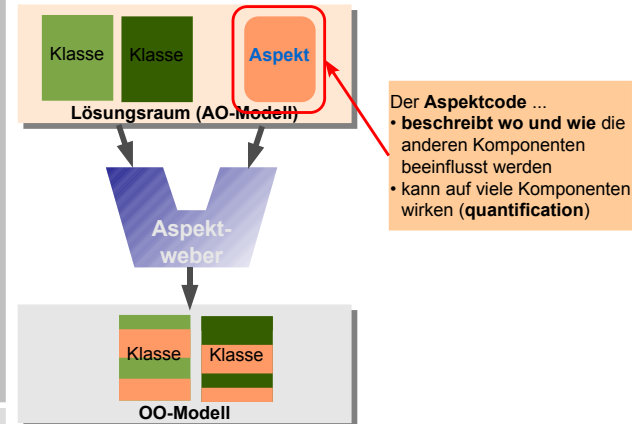
Die Rolle des Aspektwebers



© 2007 Olaf Spinczyk

11

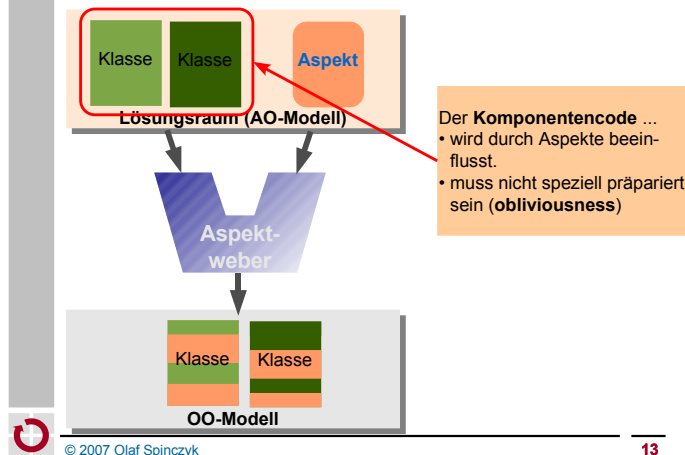
Die Rolle des Aspektwebers



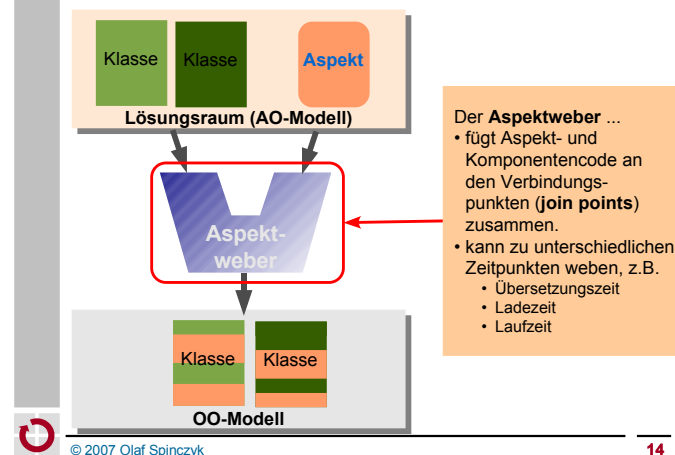
© 2007 Olaf Spinczyk

12

Die Rolle des Aspektwebers



Die Rolle des Aspektwebers



Was bringt AOP [1]?

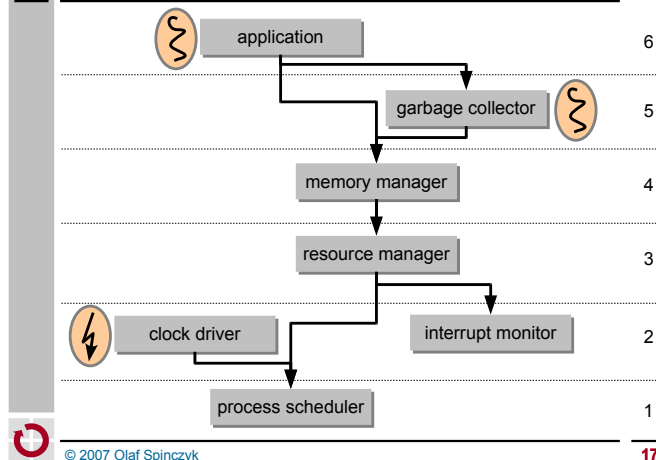
- **Wiederverwendbarkeit**
 - der Komponentencode kann u.U. auch ohne den Aspektcode (und umgekehrt) genutzt werden
- **Lesbarkeit**
 - Aspekte vermeiden oft Code-Replikation im Komponentencode
 - der Komponentencode realisiert nur die „eigentliche Funktion“
- **Erweiterbarkeit und Wartbarkeit**
 - Ein „gut formulierter“ Aspekt bezieht zukünftige Erweiterungen „automatisch“ mit ein
- **Qualität**
 - *Policies* können durch Aspekte oftmals explizit ausgedrückt und durchgesetzt werden
- **Konfigurierbarkeit**
 - bei Konfigurierung des Aspektcodes kann ein kleiner Konfigurierungseingriff große Wirkung haben

Aspekte im Domänenentwurf?

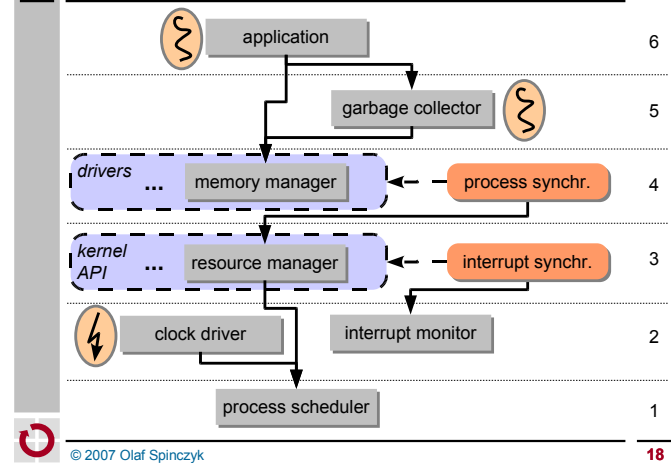
Wie könnten auch die quer schneidenden Belange beim Entwurf einer Referenzarchitektur berücksichtigt werden?

- **Experiment: Belang-Hierarchien**
 - Erweiterung der funktionalen Hierarchien
 - bestehen aus Funktionen, Aspekten und deren Beziehungen
- **Funktionen**
 - repräsentieren (später) modularisierbare Belange
 - können logische Abhängigkeiten zu anderen Funktionen haben
- **Aspekte**
 - repräsentieren quer schneidende Belange
 - können auf/in Funktionen und anderen Aspekten wirken
 - können auch logische Abhängigkeiten zu Funktionen haben

Bisher: Funktionale Hierarchie



Jetzt: Belang-Hierarchie



Zusammenfassung

- Mit Hilfe von AOP können auch quer schneidende Belange modular implementiert werden
- Gerade im Hinblick auf konfigurierbare Software verspricht AOP Vorteile
 - ein Aspekt(modul) kann leicht weggelassen werden
 - konfigurierbare Aspekte versprechen eine große Wirkung
- Beim Entwurf der Systemarchitektur sollten quer schneidende Belange berücksichtigt werden
 - Verfeinerung ergibt eine Modulstruktur **mit Aspekten**

Ausblick

- AspectC++ (jetzt)
 - und in der Übung
- Durchführung eines (Sub-)Domänenentwurfs in der Übung
 - unter Berücksichtigung von AOP
- Untersuchung verschiedener Techniken zur Umsetzung von Variabilität in der Implementierung der Komponenten
 - „Domänenimplementierung“

Literatur

- [1] R.E. Filman, T. Elrad, S. Clarke, and M. Aksit (eds).
Aspect-Oriented Software Development.
Addison-Wesley, 2005, ISBN 0-321-21976-7.

