

CCD Roter Dan



II/III

“Alles sollte so einfach wie möglich gemacht werden, aber nicht einfacher.”

Albert Einstein

- Prinzip
 - ~~Don't Repeat Yourself~~
 - Keep It Simple Stupid
 - Vorsicht vor Optimierungen
 - Favour Composition over Inheritance (3)
- Praktik
 - Die Pfadfinderregel beachten
 - Root Cause Analysis (3)
 - ~~Versionskontrollsystem~~
 - ~~Einfache Refaktorisierung~~
 - ~~Täglich reflektieren~~




Keep It Simple Stupid (KISS)

Warum?

*Wer mehr tut als das Einfachste,
lässt den Kunden warten und
macht die Lösung unnötig kompliziert.*

Keep It Simple Stupid (KISS)

- einfache, klare und leicht verständliche Lösung sollte immer bevorzugt werden
- Wenn man seinen eigenen Code nach kurzer Zeit nicht mehr versteht 

→ regelmäßige Reviews und Pair Programming

Vorsicht vor Optimierungen

Warum?

Optimierungen kosten immer viel Aufwand.

*Wer Vorsicht walten lässt,
spart oft wertvolle Ressourcen für das,
was dem Kunden wirklich nützt.*

Vorsicht vor Optimierungen

Rules of Optimization:

Rule 1: Don't do it.

Rule 2 (for experts only): Don't do it yet.

Vorsicht vor Optimierungen

- time/space budgeting (real-time systems)
 - give each component a budget for resources
- constant attention approach
- hot spot approach

Vorsicht vor Optimierungen

- time/space budgeting (real-time systems)
 - give each component a budget for resources
- constant attention approach
- hot spot approach

Vorsicht vor Optimierungen

- time/space budgeting (real-time systems)
- constant attention approach
 - every programmer
 - all the time
 - keep performance high
 - but: programs waste most of their time in a small fraction of the code → 90 % of the optimizations wasted
- hot spot approach

Vorsicht vor Optimierungen

- time/space budgeting (real-time systems)
- constant attention approach
- hot spot approach
 - find the 10 % hot spots
 - well-factored code
 - do it late in development
 - use a profiler to find the hot spots
 - tune them like *constant attention approach*

Vorsicht vor Optimierungen

DON'T GUESS!

Vorsicht vor Optimierungen

Chrysler Comprehensive Compensation Pay Process

5 minutes code change → doubled speed

[Refactoring, Fowler]

Vorsicht vor Optimierungen

AGAIN:
DON'T GUESS!

Die Pfadfinderregel beachten

Warum?

*Jede Beschäftigung mit einem
Gegenstand macht ihn zumindest ein
klein wenig besser.*

Ganz ohne bürokratische Planung.

Die Pfadfinderregel beachten

*„Hinterlasse einen Ort immer in einem
besseren Zustand als du ihn
vorgefunden hast.“*

Die Pfadfinderregel beachten

Broken-Windows-Theorie

"beschreibt, wie ein vergleichsweise harmloses Phänomen, [...], später zu völliger Verwahrlosung führen kann."



Die Pfadfinderregel beachten

Broken-Windows-Theorie

"beschreibt, wie ein vergleichsweise harmloses Phänomen, [...], später zu völliger Verwahrlosung führen kann."



- Rückkopplungseffekte
- abweichendes Verhalten
- Rückzug der Normalität
- mehr abweichendes Verhalten

Software Entropy (Prag.-Prog.)



<http://www.clean-code-developer.de/>