

Langlebige Softwarearchitekturen

Dr. Carola Lilienthal
Carola.Lilienthal@wps.de

www.wps.de

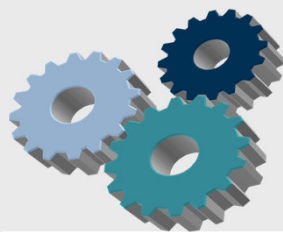
WPS - Workplace Solutions GmbH Hans-Henry-Jahn-Weg 29 22085 HAMBURG

Die zwei Architekturziele für diesen Vortrag



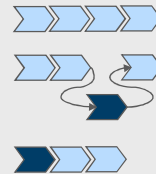
Architekturziel 1: Wartbarkeit

- Stabilität und Verständlichkeit
- Reduktion von Komplexität
- schnelle Fehleranalyse
- schnelle Changes

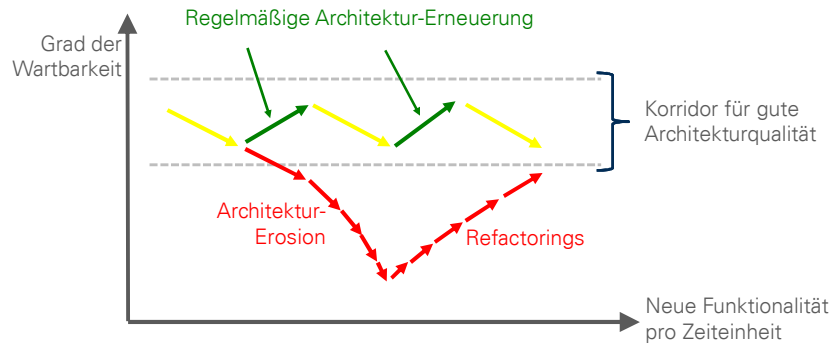


Architekturziel 2: Fachliche Flexibilität

- Geschäftsprozesse verschiedener Kunden unterstützen
- Anpassbarkeit an geänderte Anforderungen
- Baukastenprinzip



Technischen Schulden = Architektur-Erosion

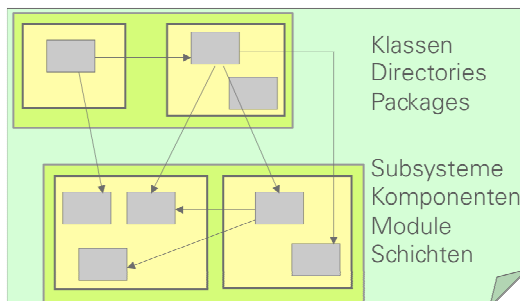


Architekturanalyse: Was ist das?



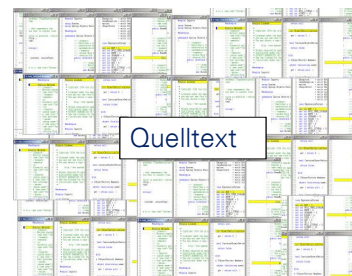
Findet sich die geplante Architektur (Soll-Architektur) in der Strukturen der implementierten Software (Ist-Architektur) wieder?

Soll-Architektur



≠

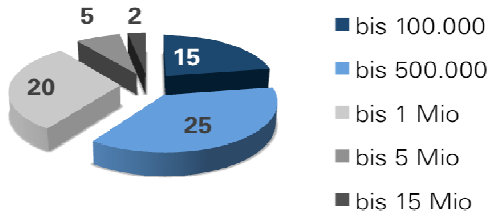
Ist-Architektur



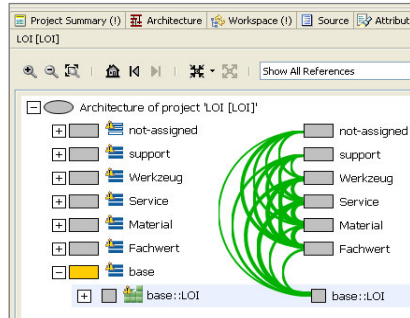
Erfahrungshintergrund



Anzahl pro Größe (LOC)



Architekturanalysen in Java, C#, C++, ABAP etc.



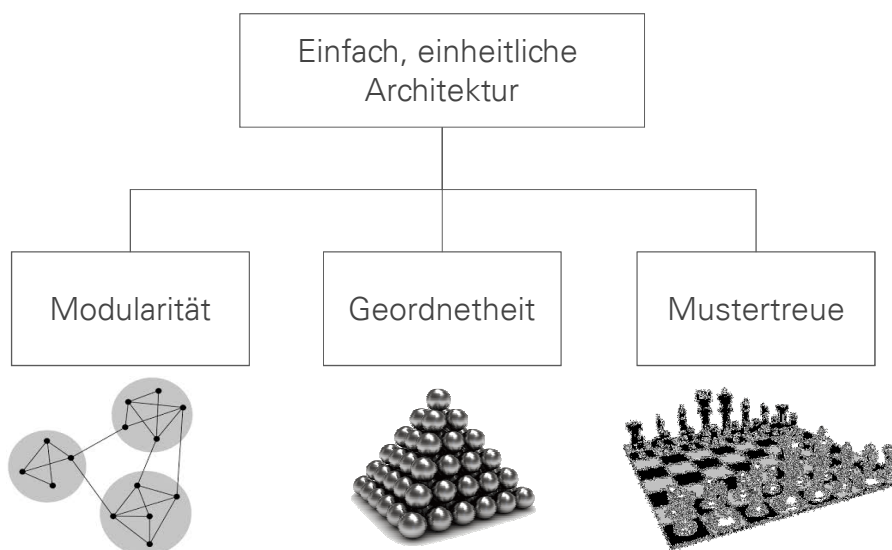
Erkenntnisse

- Typische Eigenschaften je nach Größe
- Strukturelle Einfachheit und Einheitlichkeit
- Ohne regelmäßige Architektur-Erneuerung degenerieren Systeme

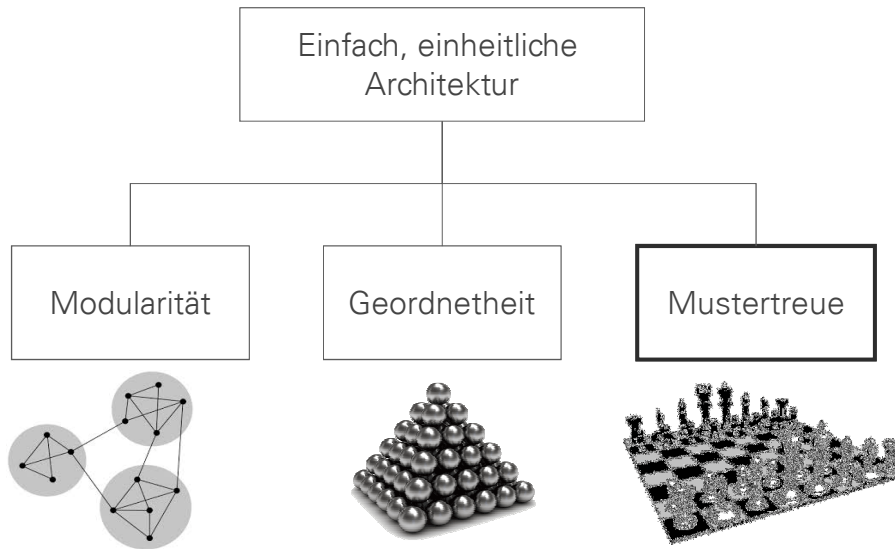
Analysewerkzeuge

- SotoArc + Sotograph
- SonarJ → Sonargraph
- Lattix

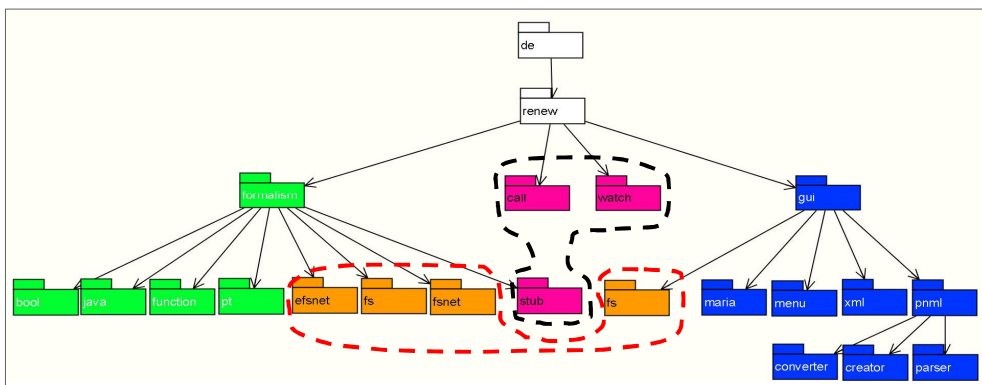
Strukturelle Einfachheit der Architektur = Zeitgewinn!



Strukturelle Einfachheit der Architektur = Zeitgewinn!



Mustertreue: Was finden wir?



- Ist die Abbildung der Architektur in der Struktur des Codes zu erkennen?

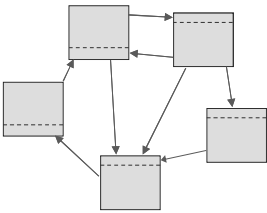
Architekturstile: Was ist das?



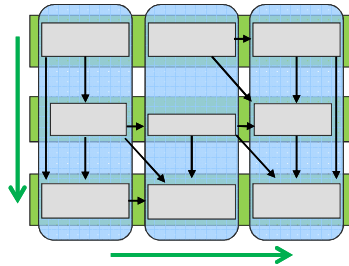
„Ein Architekturstil ist eine **prinzipielle Lösungsstruktur**, die für ein Softwaresystem **durchgängig** und unter weitgehendem **Verzicht** auf **Ausnahmen** angewandt werden sollte.“

[Reussner et al. 2006]

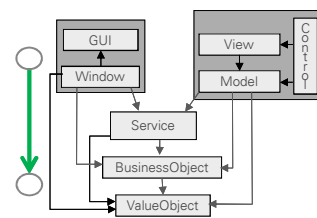
Komponentenarchitektur



Schichtenarchitektur



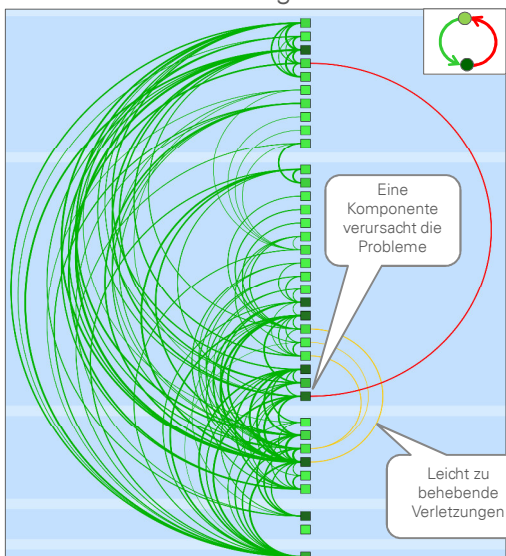
Mustersprache



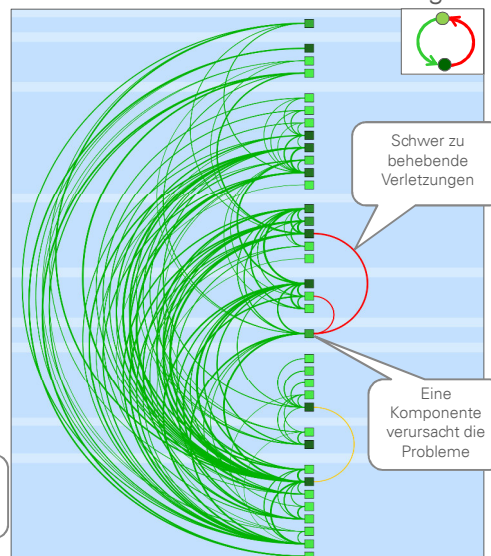
Zwei Dimensionen einer Architektur



Technische Schichtung



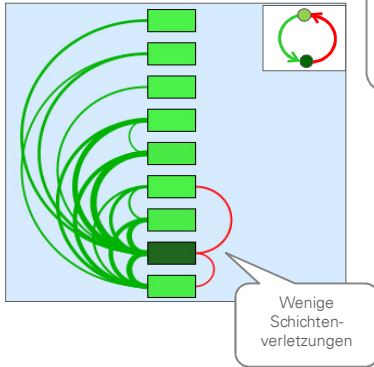
Fachliche Schichtung



Fachliche Schichtung misslungen

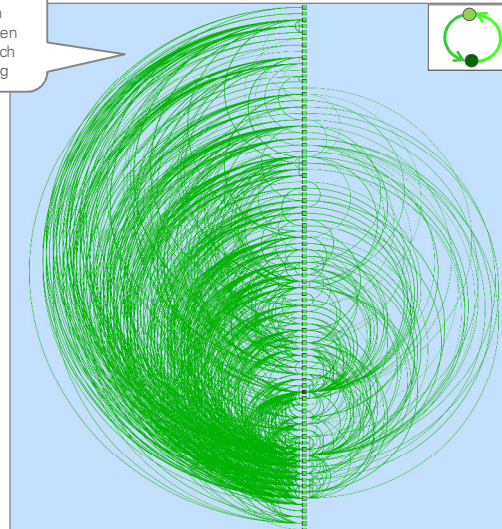


Technische Schichtung

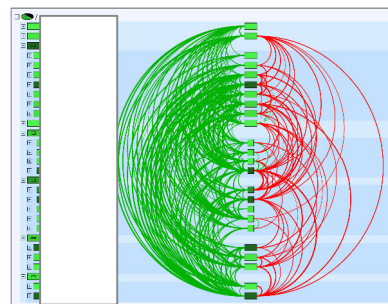
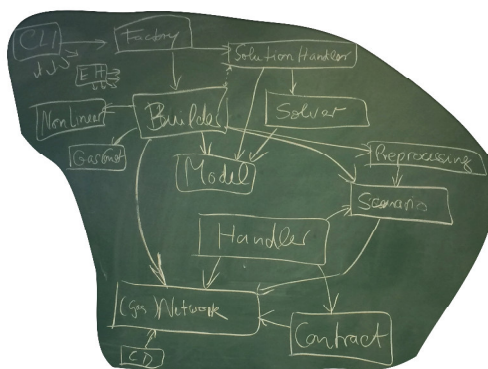


Fast alle 90 fachlichen Komponenten brauchen sich gegenseitig

Keine fachliche Schichtung

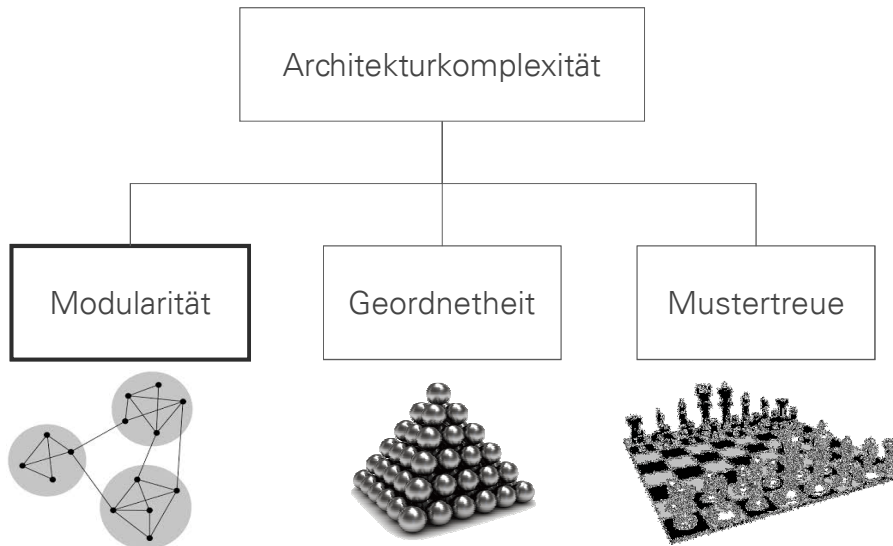


Mustersprache



😊 80% des Sourcecodes lässt sich den 23 Mustern zuordnen

😊 4% Verletzungen in den Mustern



- **Entwurf nach Zuständigkeiten** (engl.: Responsibility-Driven Design) ist eine Entwurfsphilosophie, die von Rebecca Wirfs-Brock et al. Ende der 80er Jahre formuliert wurde:

„Objects are not just simple bundles of logic and data. They are responsible members of an object community.“

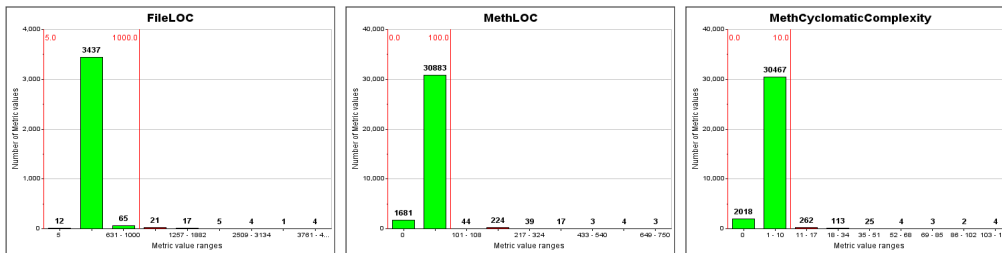
- Dazu passende Ansätze:
 - Separation of Concerns (Dijkstra)
 - Modularität (Parnas)
 - Kohäsion (Myers, Coad&Yourdon)
 - Single Responsibility Principle (SRP) (Robert C. Martin)

→ Jede Klasse, jedes Paket, jedes Subsystem, jedes Modul, jede Schicht sollte für eine klar definierte Aufgabe zuständig sein.

Modularität: Ausgewogene Größenverhältnisse



- Ist das System auf den verschiedenen Ebenen ausgewogen?
- Welche Code-Abschnitte fallen durch ihre Größe besonders auf?



Typische Metriken:

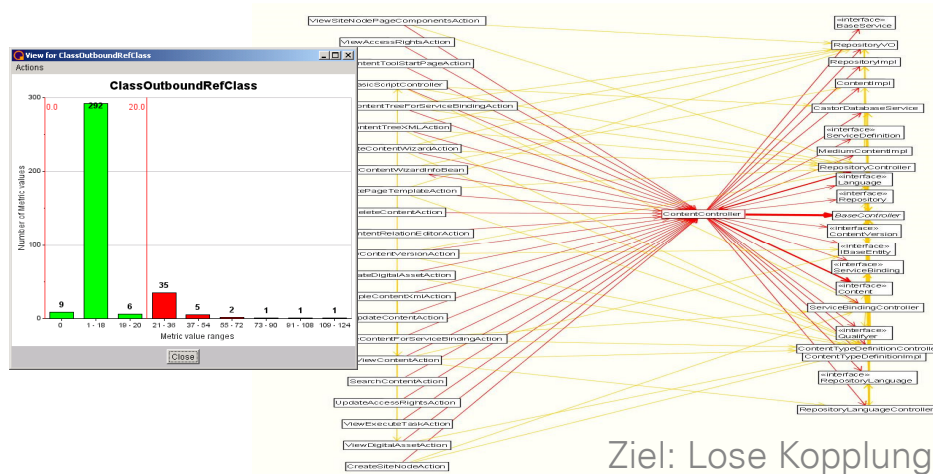
- LOC pro Methode, Klasse, Package, Komponente
- Duplizierter Code
- Zyklomatische Komplexität



Kopplungsgrad



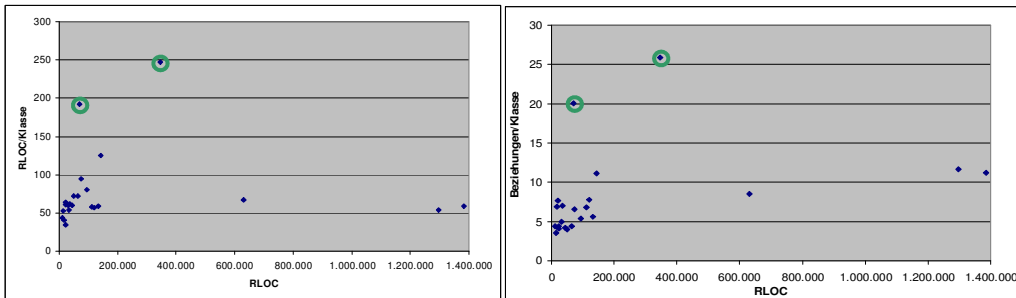
- Ist das System auf den verschiedenen Ebenen lose gekoppelt?
- Welche Code-Abschnitte fallen durch besonders viele Beziehungen auf?



Beispiel: Größenverhältnis und Kopplungsgrad

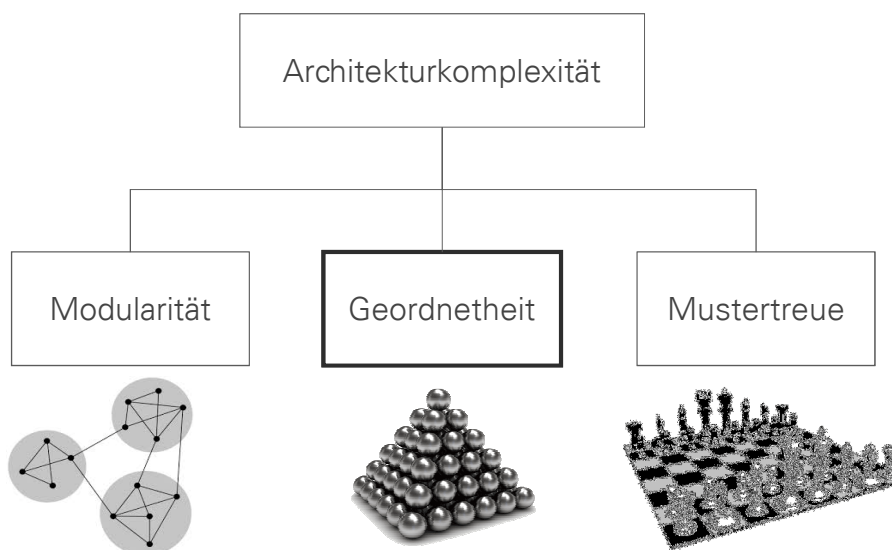


- Große Steuerungsklassen benutzen bis zu 100 – 500 andere Klassen



→ Ausgewogene Größenverhältnisse auf allen Ebenen führen zu geringerer Kopplung und besserem objektorientiertem Entwurfs

Strukturelle Einfachheit der Architektur = Zeitgewinn!

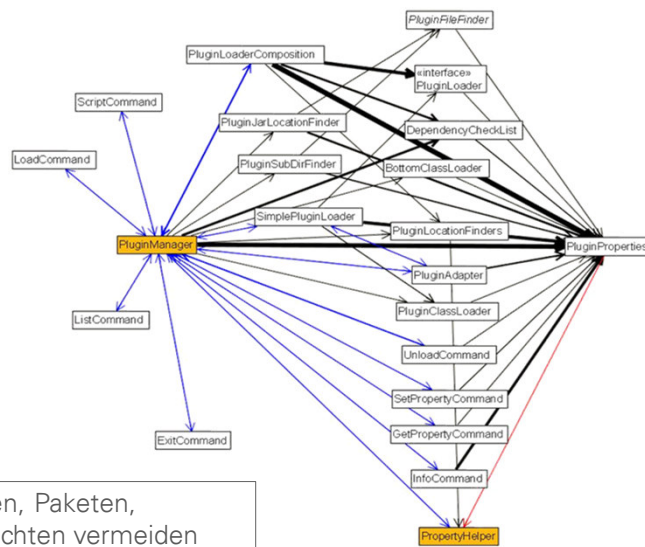


Zyklusfreiheit – Hierarchische Strukturen Acyclic Dependencies Principle (ADP)



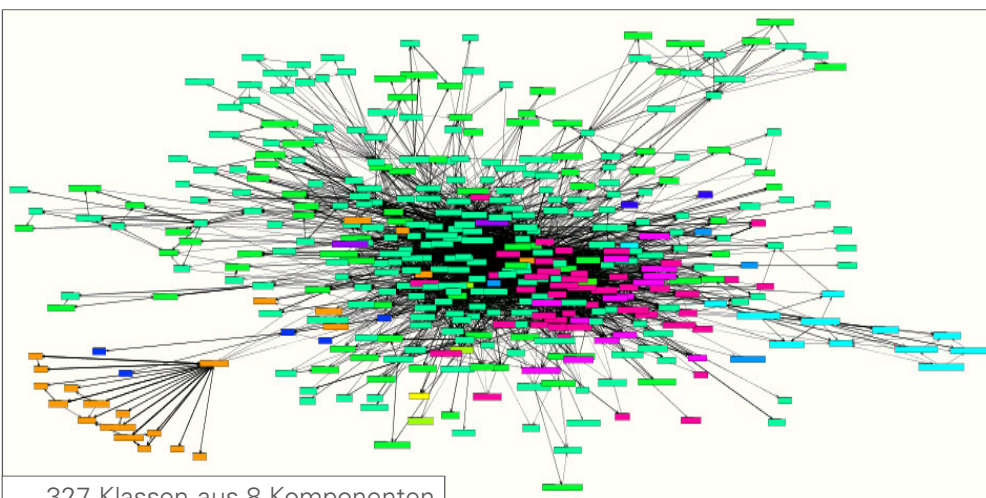
Auswirkung auf:

- Wartbarkeit
- Austauschbarkeit
- Testbarkeit
- Einstiegspunkt beim Analysieren



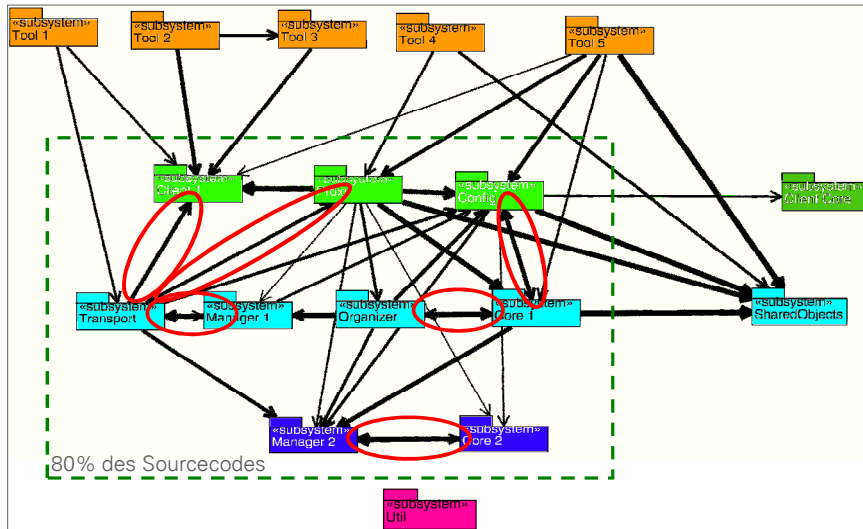
→ Zyklen zwischen Klassen, Paketen,
Komponenten und Schichten vermeiden

Große Zyklen sichtbar machen



327 Klassen aus 8 Komponenten
brauchen sich gegenseitig

Der Zwang zur Zyklentreiheit



9 Komponenten = 17 Subsysteme

Grundregeln struktureller Einfachheit für Architektur

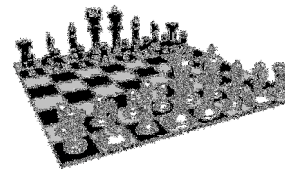
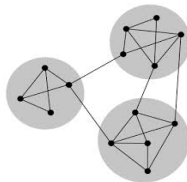


Architekturkomplexität

Modularität

Geordnetheit

Mustertreue

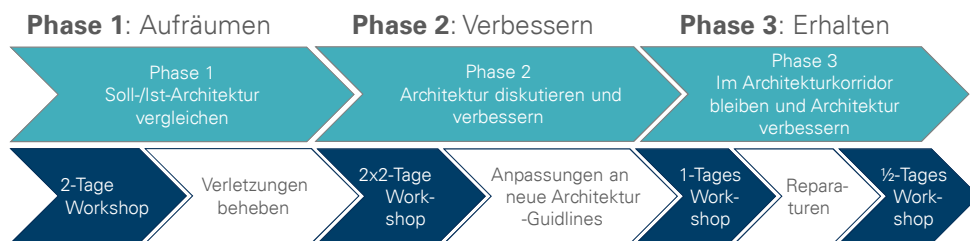


- Zuständigkeit
- Kopplung
- Größenverhältnisse
- Schnittstellen

- Zyklentreiheit auf allen Ebenen

- Architekturstil(e)
- Einheitlich und durchgängig

Schrittweise Weiterentwicklung der Architektur



Vielen Dank für Ihre Aufmerksamkeit.



WPS WORKPLACE SOLUTIONS

Diplom-Informatikerin
Dr. Carola Lilienthal
Mitglied der Geschäftsleitung

WPS – Workplace Solutions GmbH
Hans-Henny-Jahnn-Weg 29
22085 Hamburg
Telefon +49 40 51 32 26 - 82
Telefax +49 40 51 32 26 - 83
Mobil +49 170 184 77 11
c.l@workplace-solutions.de
www.workplace-solutions.de

