Software-Architektur, Entwurfsmuster und Refactoring

Ingolf Krüger

Die Rolle der SW-Architektur

„»Buy, don't build« is the anthem of the software community today. But buying means less control over every aspect of a system's development. How can this loss of control be reconciled with our desire for quality? Part of the answer lies in our assertion that, for large systems, quality lives primarily in the architecture.“

[BCK98]

Übersicht

- Architekturbegriff und Architekturaspekte
- Die Rolle der Software-Architektur
- Architektur-Modellierung und -Dokumentation
- Architektur- und Entwurfsmuster
- Refactoring-Techniken

Architekturbegriff

Eine Software-Architektur beschreibt die Dekomposition eines Systems in

Einheiten / Komponenten / Teilsysteme,

sowie deren

Verbindungen / Interaktionen / Beziehungen

unter Beachtung

von Qualitätsmerkmalen / Entwurfsrichtlinien / Beschränkungen

Architekturbegriff

Eine Software-Architektur be eines Systems in

Einheiten / Kompon

sowie deren

Verbindungen / Interaktionen / Beziehungen

unter Beachtung

von Qualitätsmerkmalen / Entwurfsrichtlinien / Beschränkungen

genauer:

- Schnittstellen (angeboten/benutzt)
- Verhalten (an den Schnittstellen)

Architekturbegriff

Eine Software-Architektur beschreibt die Dekomposition eines Systems in

Einheiten / Komp

sowie deren

Verbindungen / .

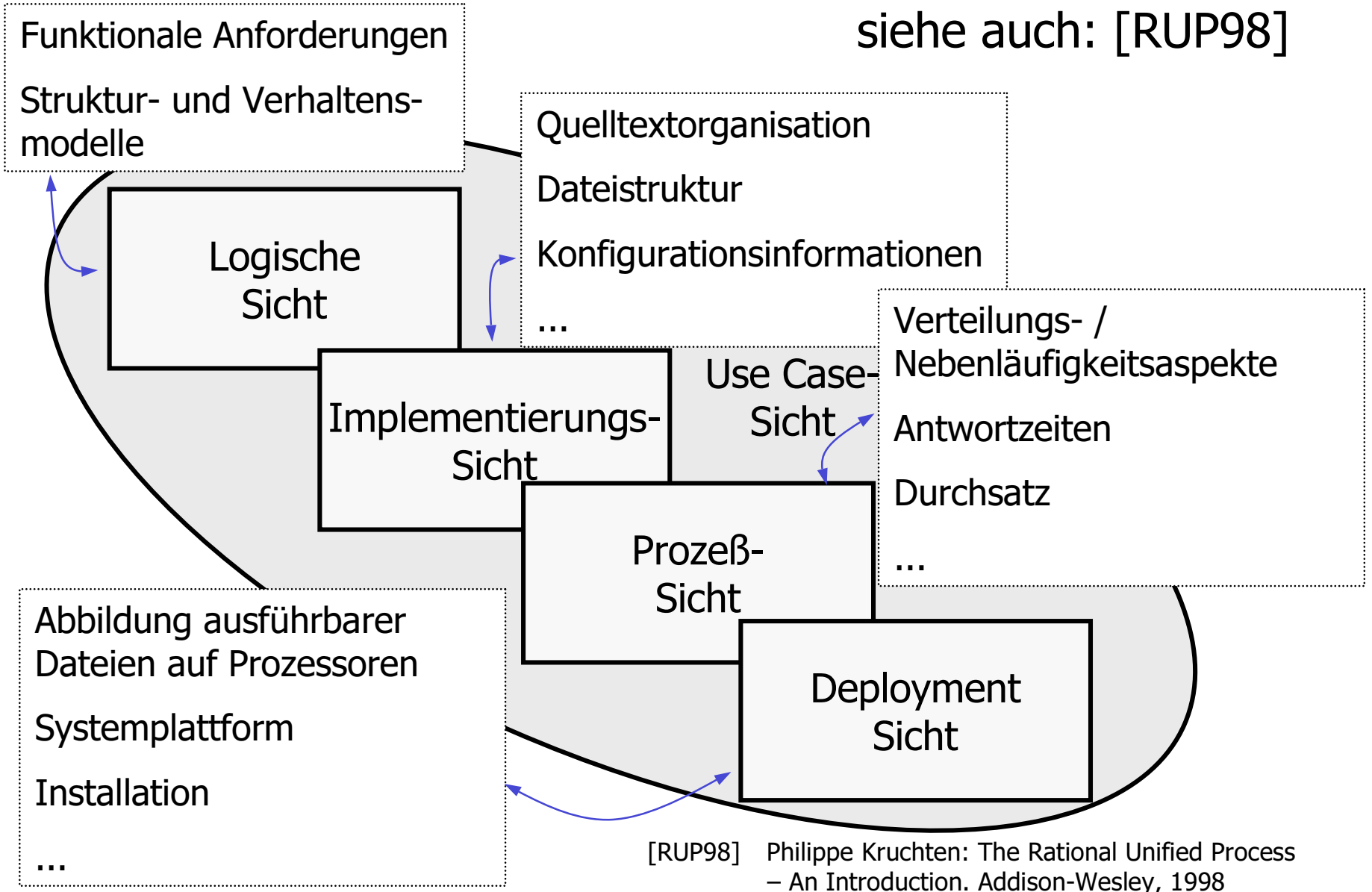
unter Beachtung

von Qualitätsmerkmalen / Entwurfsrichtlinien / Beschränkungen

Oft:
(willkürliche) Unterscheidung zwischen „funktionalen“ und „nichtfunktionalen“ Eigenschaften

Architekturaspekte

siehe auch: [RUP98]



[RUP98] Philippe Kruchten: The Rational Unified Process – An Introduction. Addison-Wesley, 1998

Architekturaspekte

- Strukturen:
  - Modul-/Komponentenstruktur (logisch/technisch)
  - Prozeßstruktur
  - physikalische Struktur
  - Aufrufstruktur
  - Abhängigkeitsstruktur
  - Klassenstruktur
  - Quellcodestruktur
  - ...
- Datenfluß
- Kontrollfluß
- ...

Architekturarten

Applikationsspezifische Architektur

Applikationsunabhängige Architektur

Architekturarten

- anwendungsspezifisch
- logische Komponentenstruktur
- Aufteilung der „business logic“
- weitgehend technologieunabhängig

Architektur

Applikationsunabhä

- anwendungsunspezifisch
- Infrastruktur
- Bibliotheken
- Werkzeuge
- ...

Übersicht

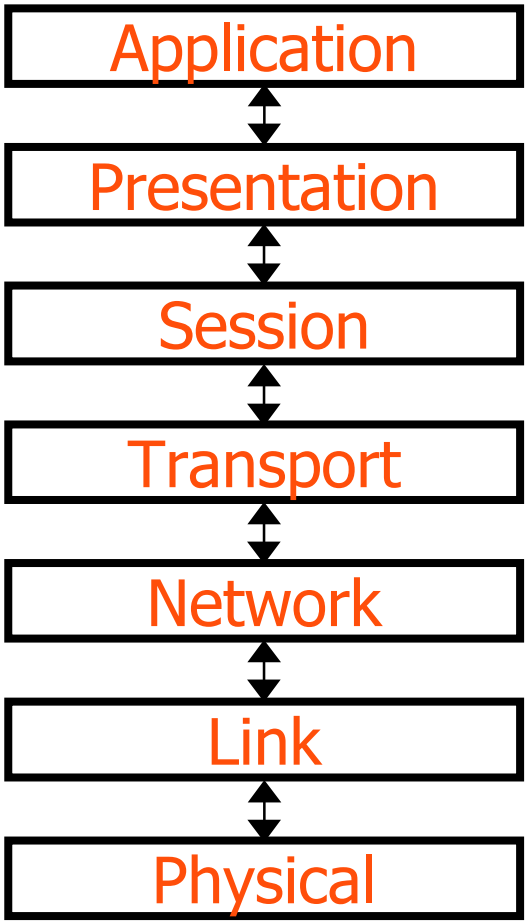
- Architekturbegriff und Architekturaspekte
- Die Rolle der Software-Architektur
- Architektur-Modellierung und -Dokumentation
- Architektur- und Entwurfsmuster
- Refactoring-Techniken

Die Rolle der SW-Architektur

- beschränkt den Spielraum für die Implementierung
- fördert (oder verhindert!) die Qualität des Systems
- unterstützt die kritischen use cases des Systems
- definiert Ansatzpunkte für
  - Änderungsmanagement
  - Variantenbildung (Produktfamilien!)
  - Arbeitsteilung

Die Rolle der SW-Architektur

Der Klassiker: ISO/OSI-Schichtenarchitektur

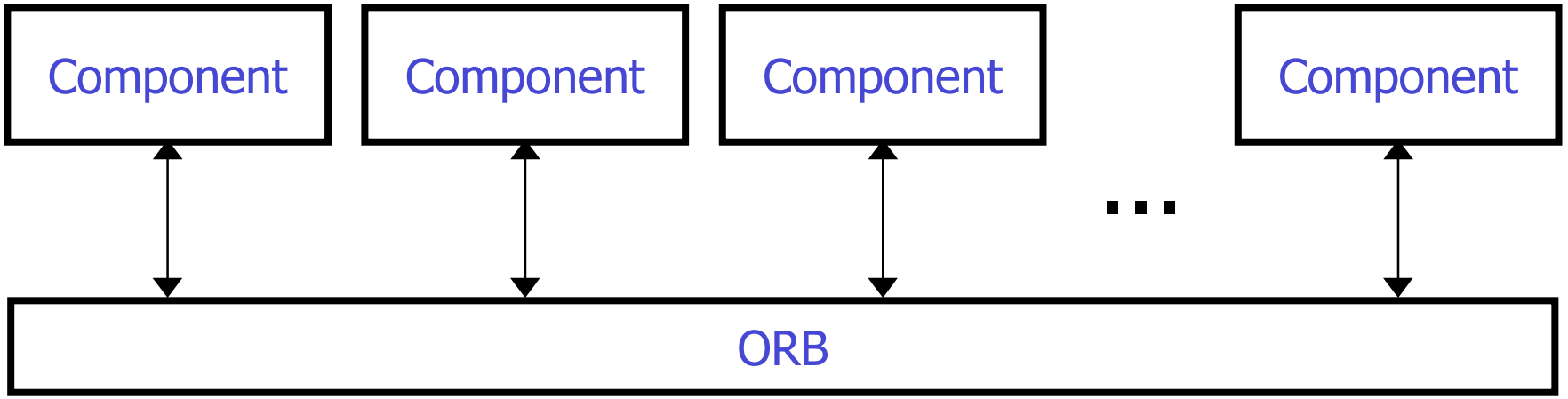


- klare Zuordnung von Aufgaben zu Schichten
  - Welche Komponente leistet was und wo ?
  - Wer entwickelt welchen Systemteil ?
  - Positionierung des Business-Modells
- klare Schnittstellen zwischen Schichten
- Aufrufbeschränkungen
  (kein „Überspringen" von Schichten)

Die Rolle der SW-Architektur

„CORBA“ (stark vereinfacht)

- Klare Schnittstellen, Verantwortlichkeiten, Entkopplung
- Infrastruktur für Kommunikation, Verteilung
- Implementierungsstrategie



Die Rolle der SW-Architektur

- Die Wahl einer geeigneten Architektur ist ein wesentlicher Erfolgsfaktor im Systementwurf

- Klar strukturierte Architektur ist Basis für:
  - Projektorganisation
  - Komplexitätsbeherrschung
  - Wiederverwendung
  - komponentenorientierte Entwicklung
  - teamübergreifendes Systemverständnis
  - beherrschbare Systemevolution

⇒ Architekturbeschreibung gehört zu jedem Software-Projekt!

Die Rolle der SW-Architektur

Eine gute SW-Architektur

- verleiht dem System Stabilität
  - ⟹ kleine Änderungen an den Anforderungen führen zu kleinen Änderungen am System
- besitzt „konzeptuelle Integrität“
  - ⟹ Ausgewogenheit
  - ⟹ Einfachheit
  - ⟹ Eleganz, Klarheit
  - ⟹ Praktikalität

Einsatz

- eines dedizierten SW-Architekten
- von Architektur-Reviews

Übersicht

- Architekturbegriff und Architekturaspekte
- Die Rolle der Software-Architektur
- Architektur-Modellierung und -Dokumentation
- Architektur- und Entwurfsmuster
- Refactoring-Techniken

Architektur-Modellierung und -Dokumentation

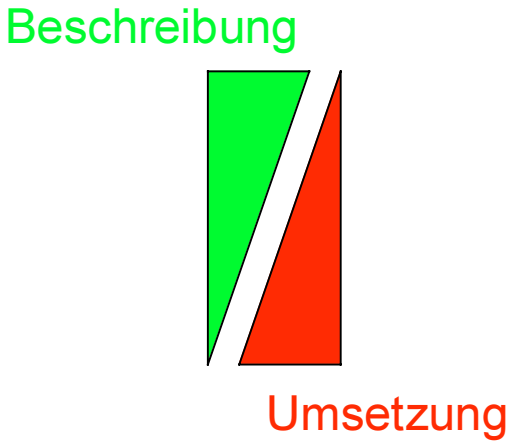
Wie lassen sich

- *Einheiten / Komponenten / Teilsysteme*,
- *Verbindungen / Interaktionen / Beziehungen,*
- *Qualitätsmerkmale / Entwurfsrichtlinien / Beschränkungen*

geeignet beschreiben und umsetzen?

⇒ Ansätze:

- *Architecture Description Languages (ADLs)*
- *Architekturstile und Muster (Patterns)*
- *Domänenspezifische Architekturen*
- *Infrastrukturen*



Architektur-Modellierung und -Dokumentation

Wie lassen sich

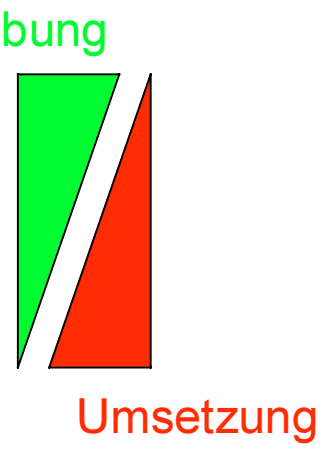
- *Einheiten / Ko*
- *Verbindungen*
- *Qualitätsmerk*

geeignet beschr

- Unicon/Wright (Garlan et al., CMU)
- Darwin (Kramer et al., Imperial College)
- Rapide (Luckham et al., Stanford)
- AutoFocus ?!
- ...

⇒ Ansätze:

- *Architecture Description Languages (ADLs)*
- *Architekturstile und Muster (Patterns)*
- *Domänenspezifische Architekturen*
- *Infrastrukturen*



Architektur-Modellierung und -Dokumentation

- Architekturstile und -muster:
  - Pipes and Filters
  - Layered Architecture
  - Model-View-Controller
  - Broker, ...
- Domänenspezifische Architekturen:
  - Telekommunikation
  - Standardarchitekturen im Business-Bereich
  - Avionik, Automotive
- Architektur-Ausprägungen:
  - Middleware-Technologien (CORBA, DCOM, Jini, ...)
  - Betriebssysteme
  - Datenbanken

Architektur-Modellierung und -Dokumentation

- Architekturstile und -muster:
  - Pipes and Filters
  - Layered Architecture
  - Model-View-Controller
  - Broker, ...

Ziel:

transparente Darstellung

- der wesentlichen Eigenschaften
- der Begründung für die „Güte"

einer Architektur

- Domänenspezifische Architekturen:
  - Telekommunikation
  - Standardarchitekturen im Bu
  - Avionik, Automotive

Ziel:

Bereitstellung von Infrastrukturen / Architekturen für spezielle Anwendungs-bereiche

- Architektur-Ausprägungen:
  - Middleware-Technologien (C
  - Betriebssysteme
  - Datenbanken

Ziel:

Bereitstellung allgemeiner Infrastruk-turen als Basis für die Systemimplemen-tierung

Architektur-Modellierung und -Dokumentation

Ziel:

- Durchführung der weiteren Entwicklungsschritte „entlang" der gewählten Architektur

  ⟹ Modellierung des Systems unter Beachtung der Architektur

  ⟹ Zuordnung von Komponenten zu ihrem „Platz" / ihren Aufgaben innerhalb der Architektur

- Beibehaltung der positiven Eigenschaften der Architektur

  ⟹ konzeptuelle Integrität

- Grundsätzlich neue Anforderungen:

  ⟹ Architektur überdenken

Übersicht

- Architekturbegriff und Architekturaspekte
- Die Rolle der Software-Architektur
- Architektur-Modellierung und -Dokumentation
- Architektur- und Entwurfsmuster
- Refactoring-Techniken

Was sind Muster/Patterns?

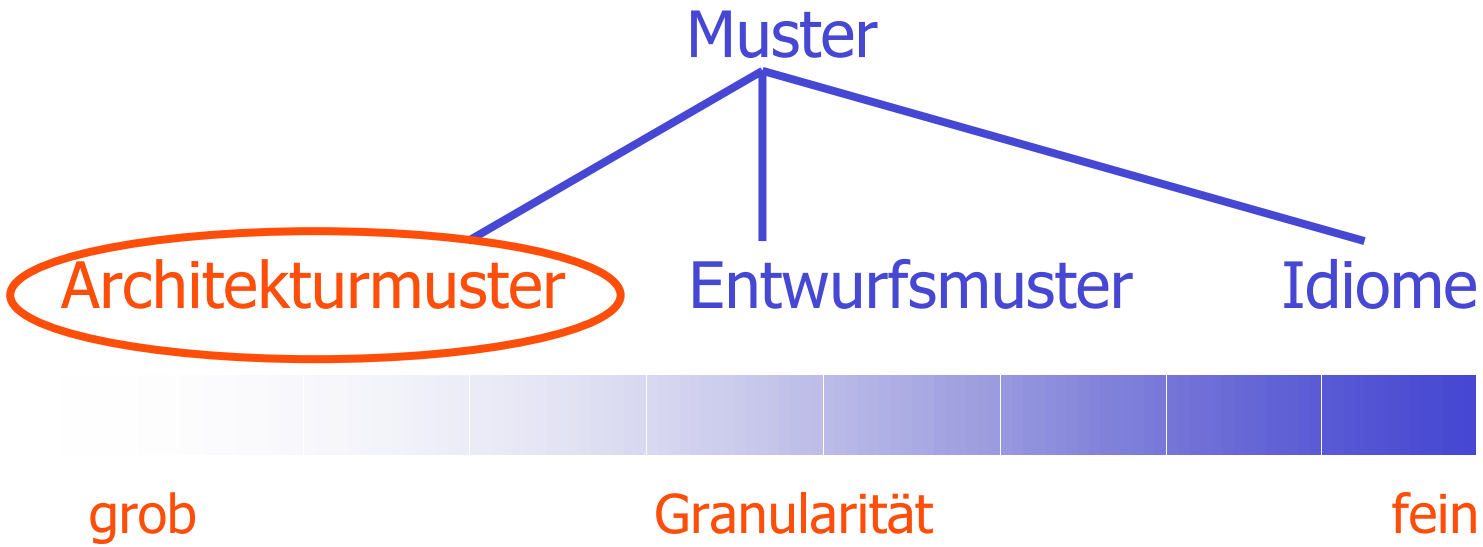
A pattern for software architecture *describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.*

[POSA96]

Was sind Muster/Patterns?

nach [POSA96]

- beschreibt *eine* bewährte Lösung für ein wiederholt auftretendes Entwurfsproblem
- definiert den Kontext für die Anwendbarkeit der Lösung

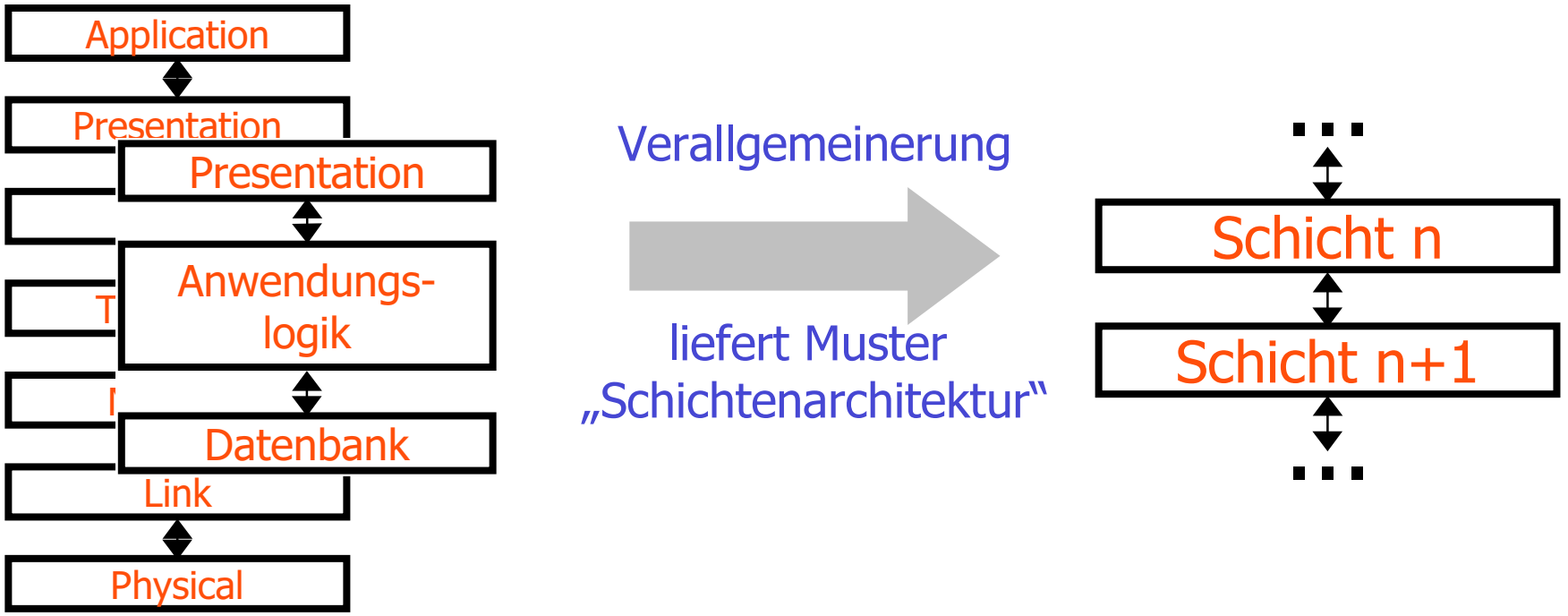


Architekturmuster

"An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them."

[POSA96]

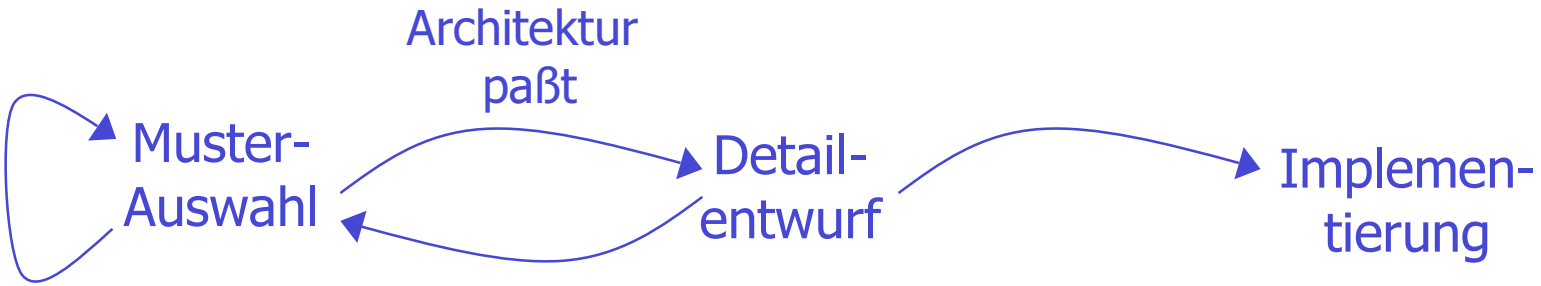
Architekturmuster



Architekturmuster – Anwendung

Architekturmuster

- unterstützen bei der Auswahl einer problemadäquaten Architektur
- helfen, die Vor- und Nachteile von Architekturen zu erkennen und gegeneinander abzuwägen
- können den Startpunkt für eine Machbarkeitsanalyse bilden:
  - unterstützt die Architektur die Echtzeitanforderungen?
  - berücksichtigt die Architektur vorhandene Platzbeschränkungen?
  - erlaubt die Architektur die vorhersehbaren Erweiterungen?
  - ...
- bereichern das „Lösungsvokabluar" des Entwicklerteams



Architekturmuster – Schema

- Name:
  - prägnante Bezeichnung für das Muster
- Kontext:
  - wann ist das Muster anwendbar?
- Problem:
  - welches Problem löst das Muster?
  - welche Randbedingungen bestehen (tradeoffs!)?
- Lösung:
  - Struktur
  - Verhalten
  - Implementierung
- Beispielanwendungen, Varianten, Konsequenzen

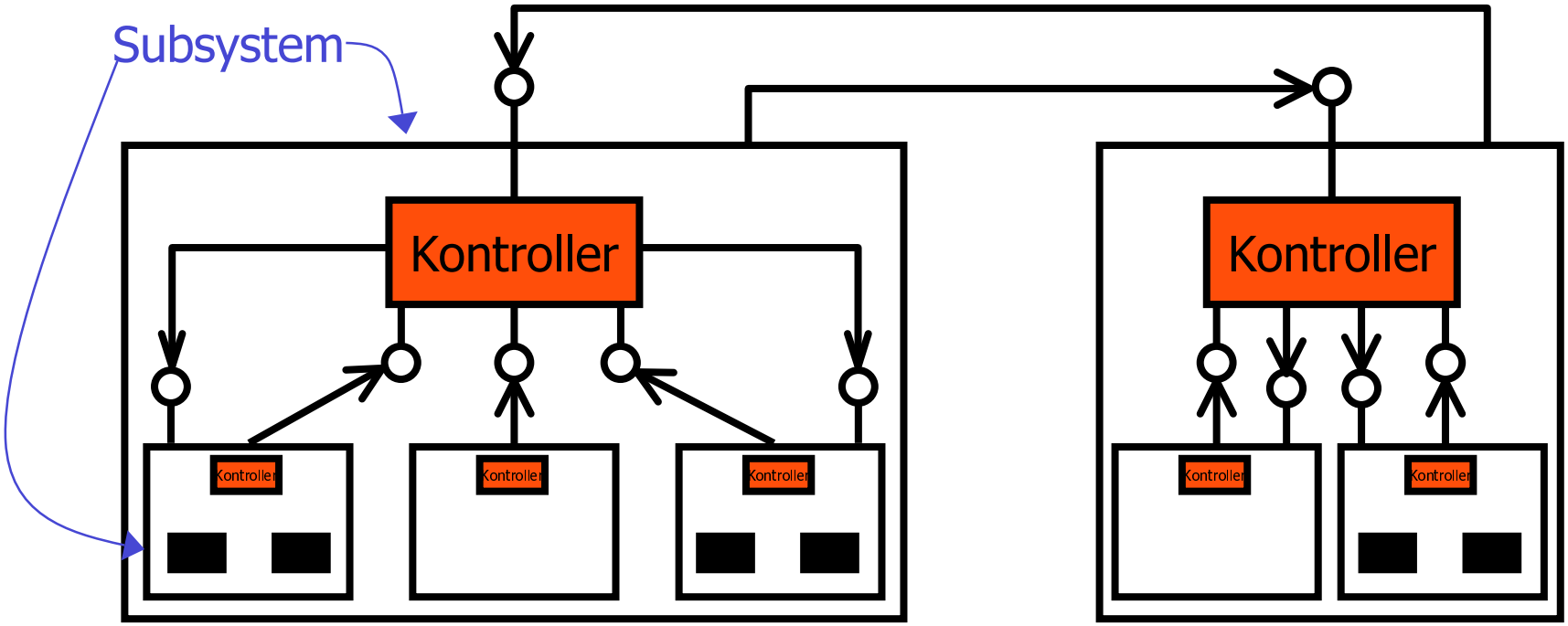
Beispiel: Subsystem Controller

- Kontext & Problem
  - Komplexes System mit hoher Vernetzung zwischen Einzelkomponenten
  - Starke Kopplung der Komponenten untereinander
  - Im System lassen sich kohärente Subsysteme identifizieren

- Lösung:
  - Hierarchische Dekomposition des Systems in Subsysteme
  - Einführung von Kontroller-Komponenten zur Entkopplung von Subsystemen
  - Jedes Subsystem besteht aus einem Kontroller und einer Menge weiterer Subsysteme
  - Kontroller steuern die Kommunikation
    - von Subsystemen auf der selben Abstraktionsebene
    - von Subsystemen auf niedrigeren Abstraktionsebenen
  - Subsysteme interagieren ausschließlich über „ihren" Kontroller

Beispiel: Subsystem Controller

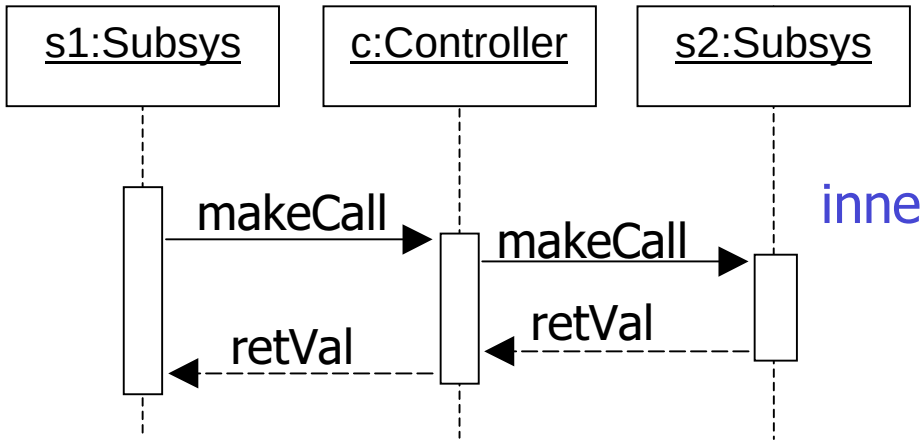
- Struktur:



Kommunikation findet ausschließlich über die Schnittstellen des Kontrollers statt

Beispiel: Subsystem Controller

- Verhalten

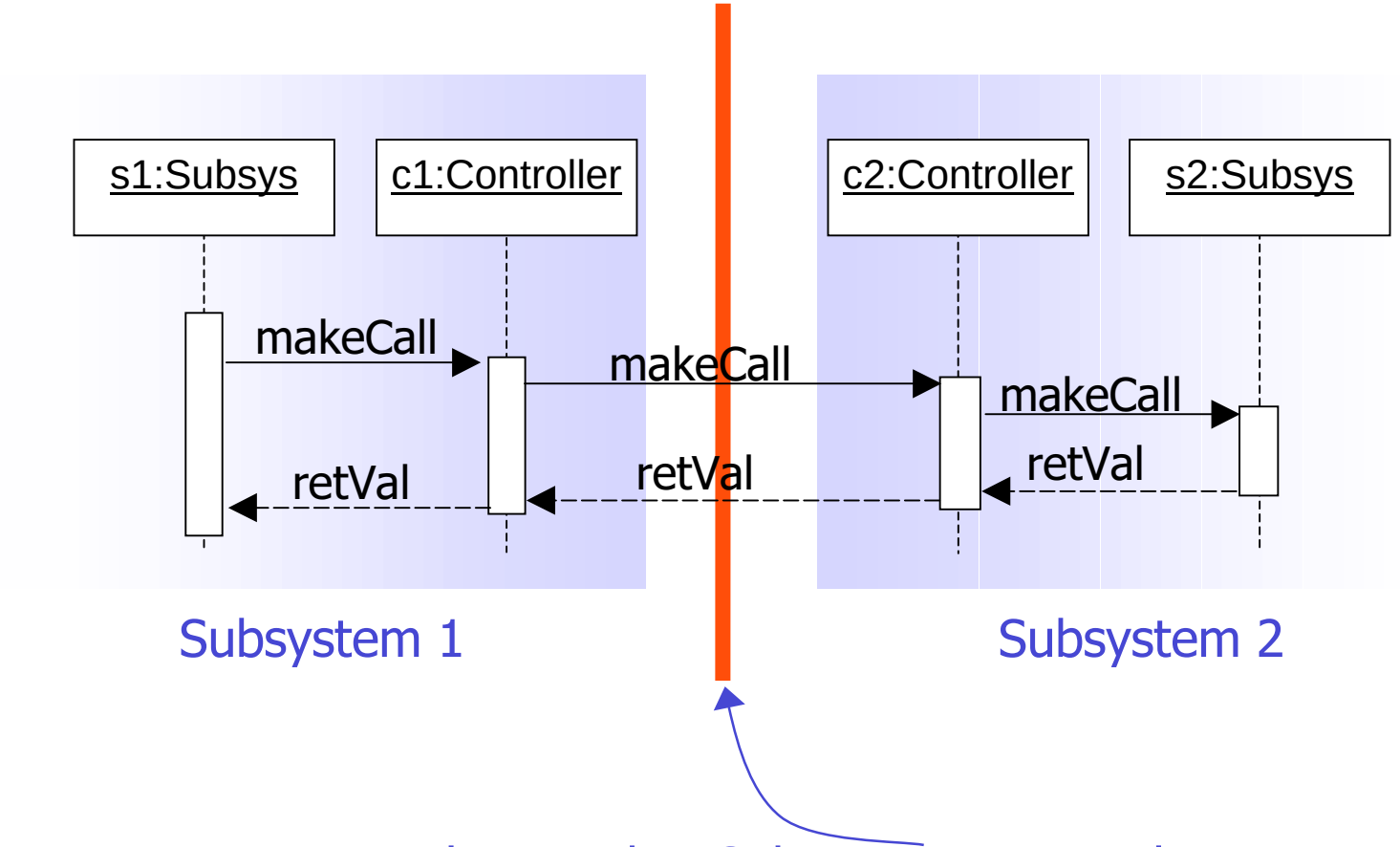


innerhalb desselben Subsystems

Einsatz effizienter Kommunikationsmechanismen!

Beispiel: Subsystem Controller

- Verhalten



Kommunikation über Subsystemgrenzen hinweg,
evtl. Einsatz weniger effizienter Kommunikationsmechanismen

Beispiel: Subsystem Controller

- Beispielanwendungen
  - Telekommunikation: Vermittlungsanlagen
- Konsequenzen:
  - klare Kommunikationsstruktur
  - hohes Maß an Entkopplung
  - evtl. Performanzverlust durch Kommunikation über mehrere Hierarchiestufen hinweg
  - evtl. Performanzverlust bei Kommunikation zwischen Kontrollern (Überschreitung der Subsystemgrenze)
  - Ansatzpunkt für Performanzanalyse der Kommunikationswege
  - Ansatzpunkt zur Lokalisierung harter Echtzeitanforderungen vorhanden:

    ⟹ innerhalb von Subsystemen: „schnelle" Kommunikation

    ⟹ zwischen Kontrollern: „langsame" Kommunikation

Übersicht

- Architekturbegriff und Architekturaspekte
- Die Rolle der Software-Architektur
- Architektur-Modellierung und -Dokumentation
- Architektur- und Entwurfsmuster
- Refactoring-Techniken

Refactoring-Techniken

- Gegeben:
  - ein Modell des Systems als Ergebnis des Entwurfs oder
  - eine Implementierung auf der Basis des Entwurfs

- Problem:
  - die Anforderungen an das System ändern sich
  - dem System fehlen notwendige/wünschenswerte Qualitäten:
    - klare Struktur
    - Wiederverwendbarkeit
    - Übersichtlichkeit
    - Änderbarkeit
    - ...
  - die Anpassung des Systems an die sich ändernden Anforderungen gestaltet sich schwierig
  - die Investition in den bisher erzielten Entwurf soll sich auszahlen

Refactoring-Techniken

- Läßt sich der vorhandene Entwurf/die Implementierung so modifizieren, daß
  - sich das beobachtbare Verhalten des Systems nicht ändert, aber
  - danach die notwendigen/wünschenswerten Eigenschaften im System vorhanden sind?

- Beispiel:
  - gegeben sei eine Aufzugssteuerung für eine Kabine
  - die Bedienstrategie für die Kabine ist eng mit der Steuerung gekoppelt
  - dies erschwert die Erweiterung des Systems auf mehrere Kabinen mit unterschiedlicher Bedienstrategie
  - welche Modifikationen sind an der gegebenen Aufzugssteuerung vorzunehmen, um die Bedienstrategie stärker von der Steuerung zu entkoppeln?
  - wie ist sicherzustellen, daß die Modifikationen keine Fehler in das System einführen?

Refactoring-Techniken

„Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs. In essence when you refactor you are improving the design of the code after it has been written."

[F99]

Refactoring-Techniken

Refactoring:

- „minimalinvasive" Eingriffe in die Struktur des Systems

  ⇒ Strategie der kleinen Schritte

- Aufstellen geeigneter Testfälle **vor** der Durchführung einer Änderung am System

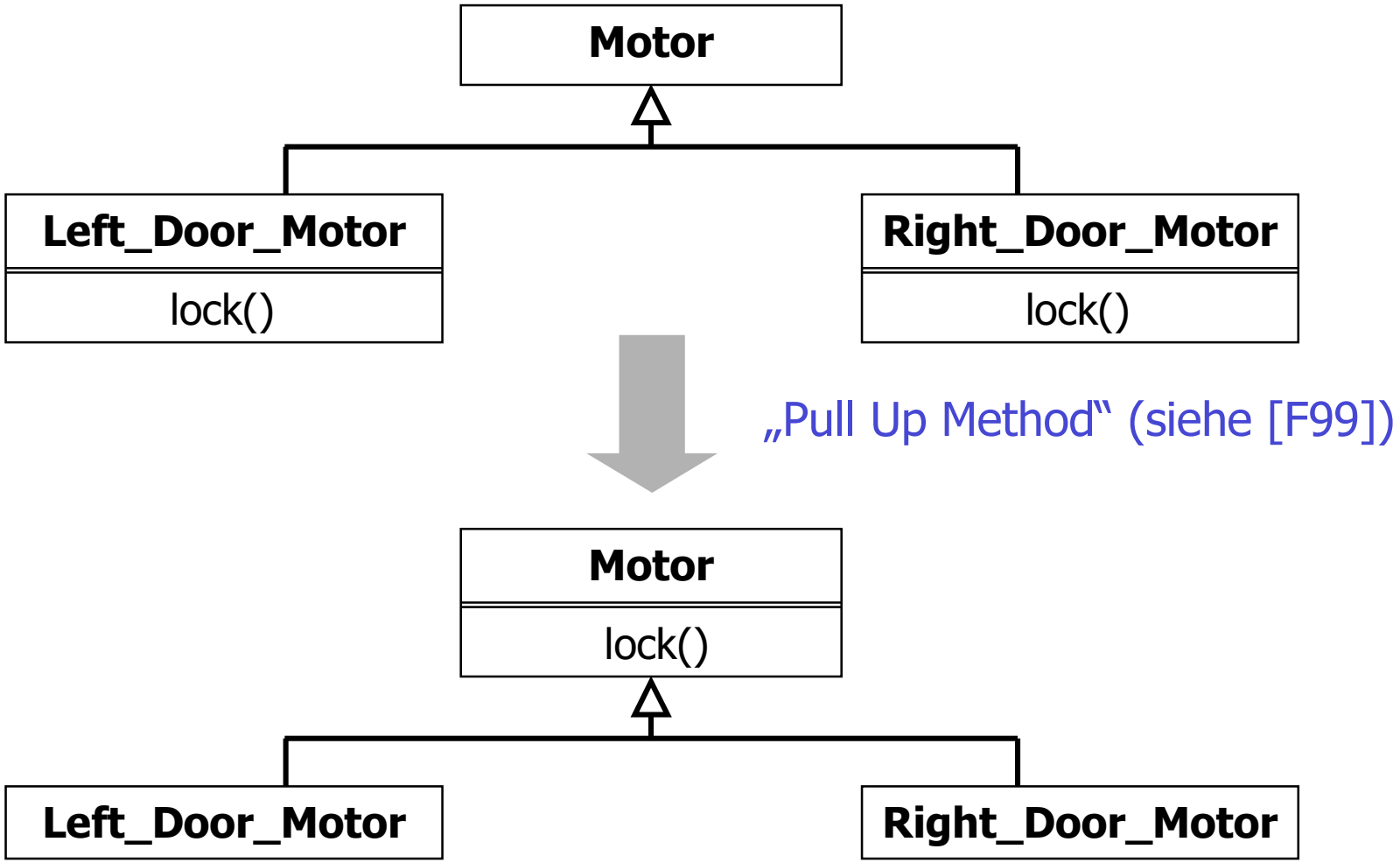
- Durchführen der Tests während und nach Durchführung der Änderung

  ⇒ erhöht das Vertrauen in die Korrektheit der Änderung;

  Ziel: keine Änderung des beobachtbaren Verhaltens

Refactoring-Techniken

Beispiel: Verschieben einer Methode zur Superklasse



Refactoring-Techniken

- Refactoring findet in extrem kleinen Schritten statt
- Dadurch bleibt die jeweilige Änderung überschau- und handhabbar
- Refactoring läßt sich im Entwurf besonders gut in Zusammenhang mit Architektur- und Entwurfsmustern einsetzen:
  1.) aktuellen Stand bewerten
  2.) Ziel-Muster auswählen
  3.) Sequenz von Refactoring-Schritten identifizieren, die zum Ziel-Muster führt
  4.) Refactoring durchführen, bis Ziel-Muster erreicht

Refactoring-Techniken

- Für effektiven Einsatz muß ein Katalog von Refactoring-Techniken und Architektur-/Entwurfs-Mustern bekannt sein (siehe [F99,GOF95,POSA96]):
  - Vereinfachung von
    - Methodenaufrufen
    - Klassenhierarchien
  - Erhöhung/Vermeidung loser Kopplung
  - Vereinfachung der Kontrollstruktur
  - ...

Refactoring-Techniken

- Bei geschicktem Einsatz von Refactoring kann zur Lösung eines Problems zunächst stets der einfachste Ansatz gewählt werden

  ⟹ Refactoring hilft, die gegebene Lösung an geänderte Anforderungen anzupassen

  ⟹ Vermeidung von „overengineering" möglich

  ⟹ Anknüpungspunkt an „eXtreme Programming"

- Fazit:
  - Refactoring ist hilfreich, um einen Entwurf/eine Implementierung in kleinen Schritten zu verbessern
  - hohes Maß an Erfahrung/Kenntnis über Refactoring-Schritte erforderlich

Zusammenfassung

- Die Wahl einer belastbaren Software-Architektur ist einer der wesentlichen Erfolgsfaktoren für die Entwicklung komplexer, hochqualitativer Systeme

- Beschreibungssprachen (ADLs) und Muster helfen, Struktur- und Verhaltensaspekte einer Software-Architektur herauszuarbeiten und darzustellen

- Refactoring-Techniken: „Politik der kleinen Schritte“, um eine Zielarchitektur mit den gewünschten Qualitätseigenschaften zu erreichen

Literatur

[BCK98] Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice, SEI Series in Software Engineering, Addison-Wesley, 1998

[DW98] Desmond F. D'Souza, Alan C. Wills: Objects, Components and Frameworks with UML – the CATALYSIS Approach, Addison-Wesley, 1998

[F99] Martin Fowler: Refactoring - Improving the Design of Existing Code, Addison-Wesley, 1999

[GOF95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[POSA96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture – A System of Patterns, Wiley, 1996

[RUP98] Philippe Kruchten: The Rational Unified Process – An Introduction, Addison-Wesley, 1999