

Software Engineering

SS 2005

Prof. Dr. Barbara Paech, Jürgen Rückert



Institut für Informatik
Im Neuenheimer Feld 326
69120 Heidelberg
<http://www-swe.informatik.uni-heidelberg.de>
paech@informatik.uni-heidelberg.de



RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG



Kapitel Vorlesung

▶ 4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

1. Einführung SWE
2. Beschreibungstechniken (Dokumentation)
3. Qualitätssicherung
4. **Methode (Entwicklung)**
5. Projektmanagement
6. Evolution

4.Methode

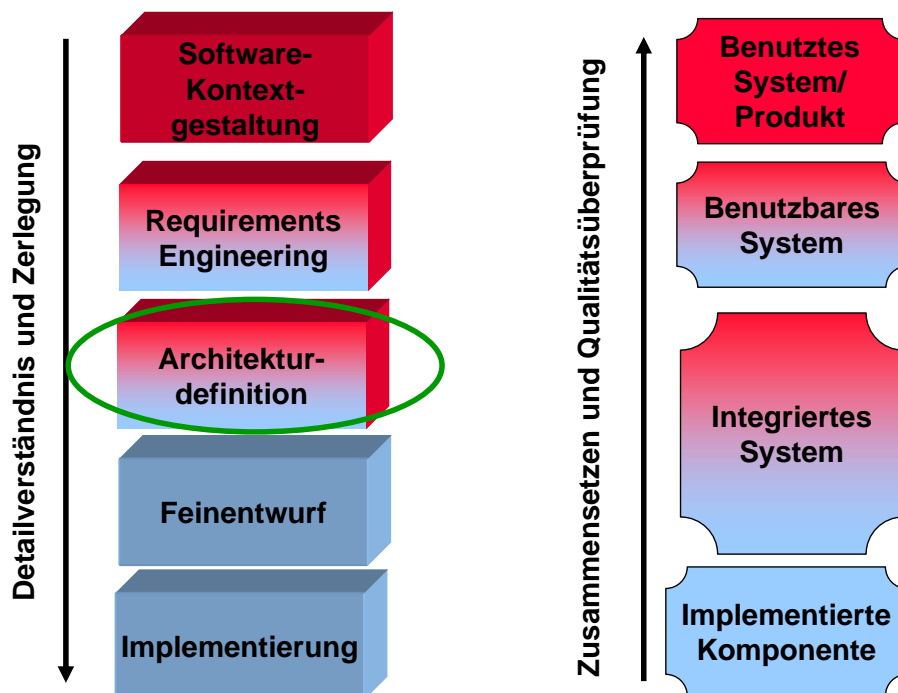
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ 4.1. Allgemeines
- ◆ 4.2. Vorgehen Requirements Engineering
- ◆ **4.3. Vorgehen Architektur**
- ◆ 4.4. Vorgehen Entwurf
- ◆ 4.5. Vorgehen Testen

1.3.2. Aktivitäten und Ergebnisse der Entwicklung

4.Methode

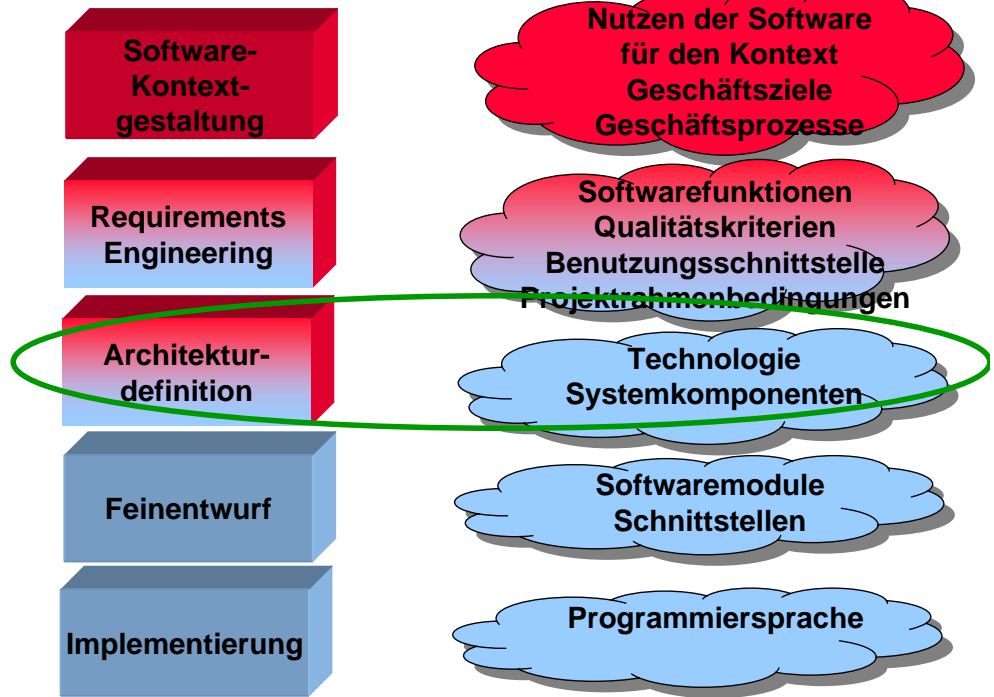
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



1.3.2. Gestaltungsentscheidungen

4.Methode

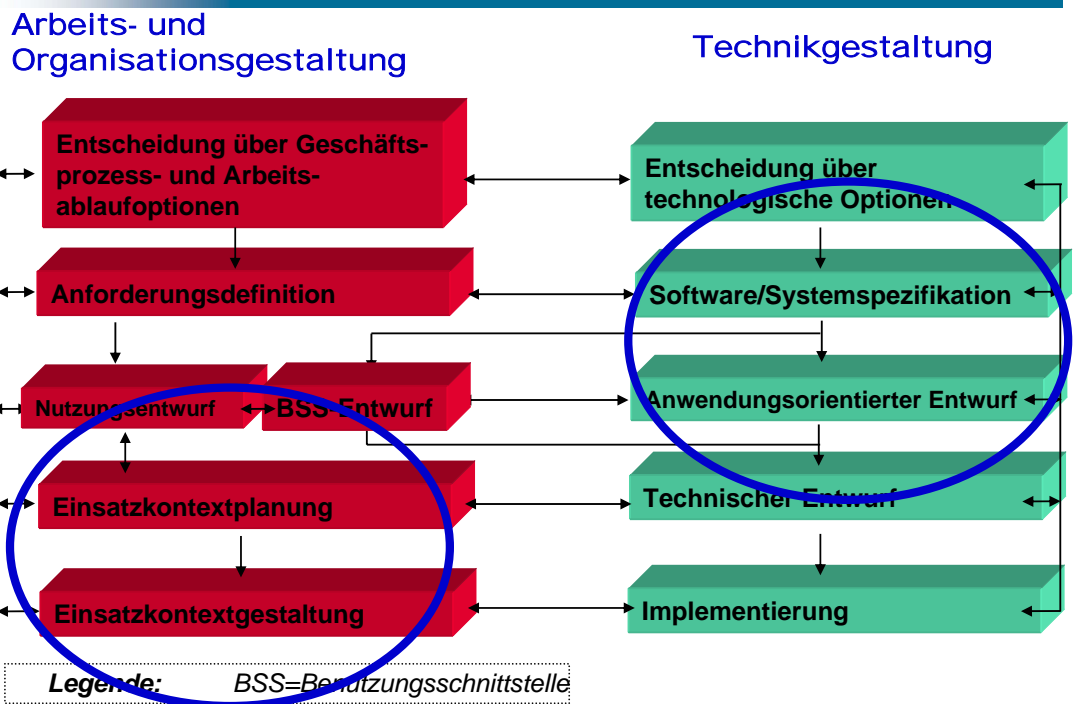
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.2.4.SW-Entwicklung als Arbeits- und Technikgestaltung

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

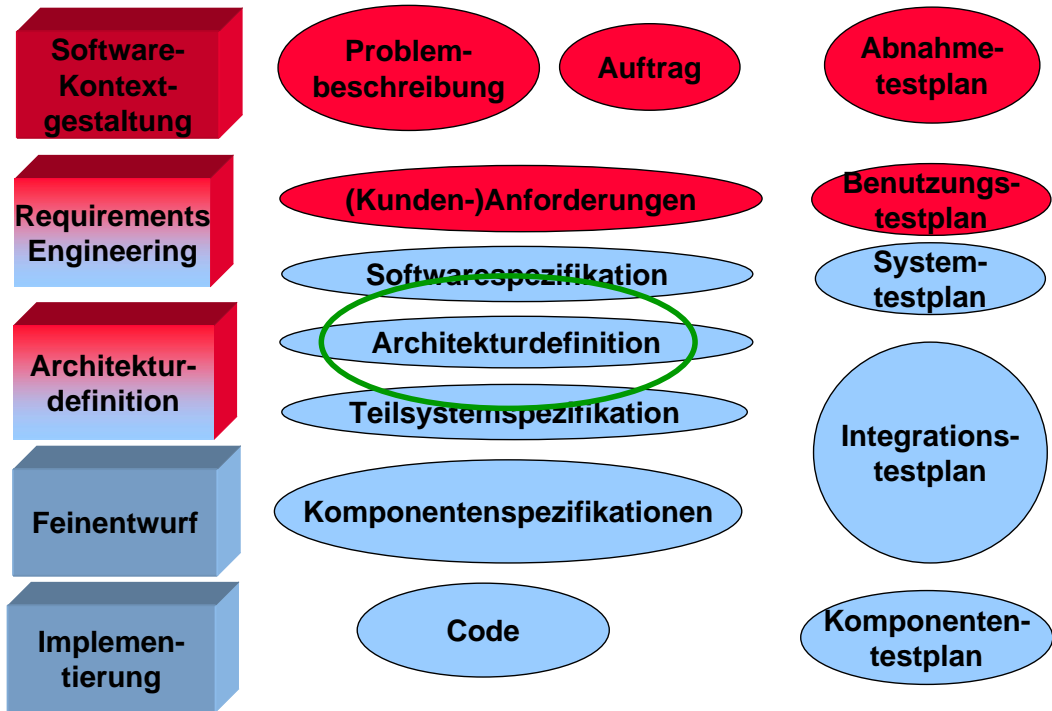


[Paech 2000]

1.3.2. Entwicklungsdokumente / Abstraktionsebenen

4.Methode

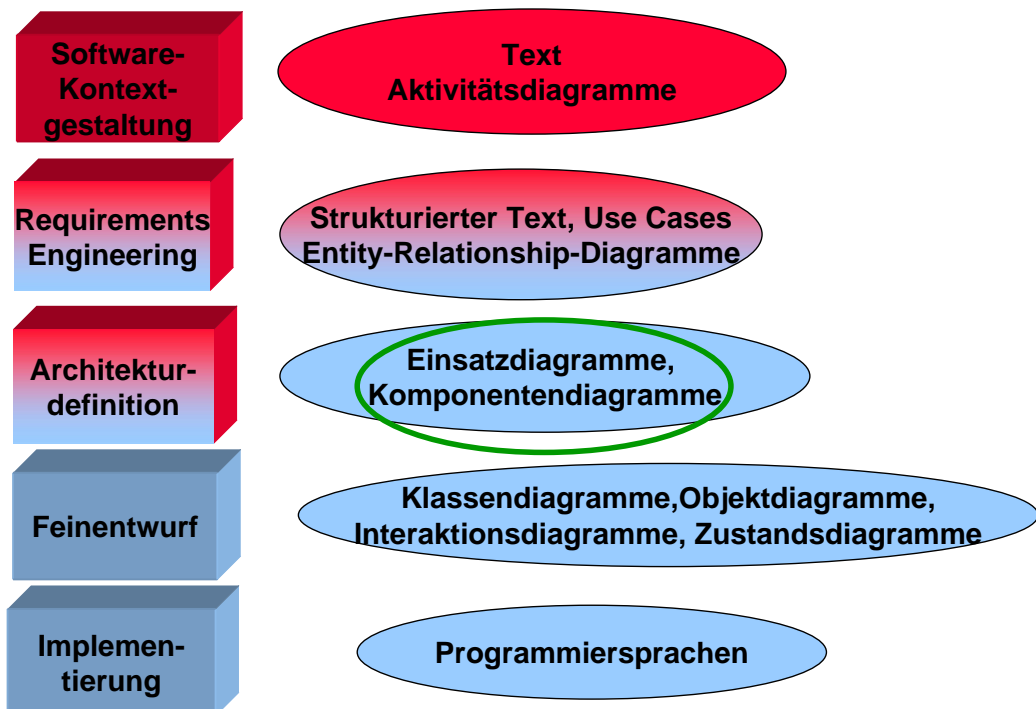
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



1.3.5. Beschreibungstechniken

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ 4.3.1. Inhalt und Aufgaben des Architekturentwurfs
- ◆ 4.3.2. Festlegung der Entwurfsziele
- ◆ 4.3.3. Architekturbeschreibung
- ◆ 4.3.4. Entwurfsprinzipien
- ◆ 4.3.5. Architekturmuster



4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.3.1. Definition: Architektur

IEEE-Std. 1471-2000:

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Beschreibt die **grundlegende Organisation eines Systems** verkörpert durch seine Komponenten, ihre Beziehungen zueinander und zur Umgebung
- ◆ Beschreibt die **Prinzipien des Entwurfs** (oft durch eine Referenzarchitektur)

4.3.1. Sichten auf die Architektur

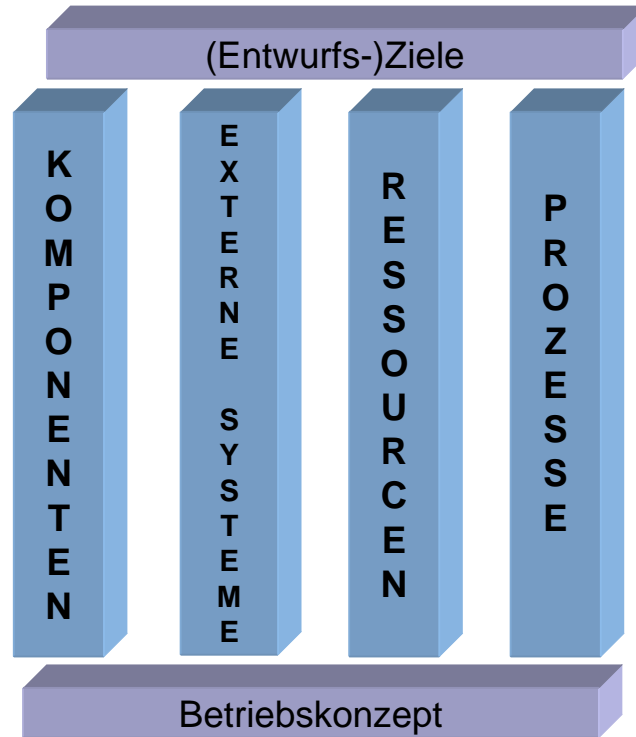
4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Konzeptionelle Sicht** für Auftraggeber, Nutzer
 - Grobüberblick (inkl. Externer Systeme), meist mit „Konfigurationsdiagrammen“
- ◆ **Komponentensicht** für Entwickler
 - Komponenten, Schnittstellen und innere Struktur (Bündelung und Konkretisierung der Analyseklassen), statische Qualitätsziele (z.B. Wartbarkeit)
- ◆ **Physische Sicht** für Auftraggeber, Implementierer
 - Ressourcen und Verteilung der Komponenten auf die Ressourcen, dynamische Qualitätsziele (z.B. Performanz)
- ◆ **Laufzeitsicht** für Integratoren, Implementierer
 - Prozesse, dynamische Qualitätsziele
- ◆ **Betriebssicht** für Auftraggeber (Operator)
 - Betriebskonzept

4.3.1. Welche Inhalte umfasst der Architekturentwurf?

- 4.Methode
- 4.1. Allgemeines
 - 4.2. Vorgehen RE
 - ▶ 4.3. Vorgehen Architektur
 - 4.4. Vorgehen Entwurf
 - 4.5. Vorgehen Test



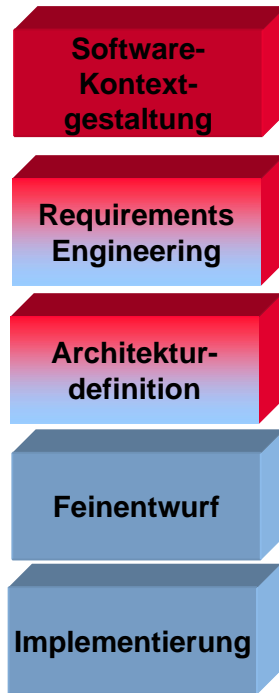
4.3.1. Entwurfsziele

- 4.Methode
- 4.1. Allgemeines
 - 4.2. Vorgehen RE
 - ▶ 4.3. Vorgehen Architektur
 - 4.4. Vorgehen Entwurf
 - 4.5. Vorgehen Test

- ◆ Entwurfsziele
 - Beschreiben die durch den Entwurf zu erreichenden Systemeigenschaften
 - Konkretisieren die Qualitätsanforderungen durch Betrachtung der Komponenten und Ressourcen sowie weiterer Rahmenbedingungen
- ◆ Beispiele für Entwurfsziele
 - Flexibilität (Konfigurierbarkeit, Wartbarkeit, Erweiterbarkeit)
 - Betriebssicherheit (Fehlertoleranz, Ausfallsicherheit, Verfügbarkeit, Sicherheit)

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Komponenten

- Erbringen eine bestimmte abgegrenzte **Dienstleistung** entsprechend ihrer **Schnittstelle**
- Sind eine sinnvolle **Liefer- und Entwicklungseinheit**, die möglichst **wiederverwendbar** sein sollte
- Können **wieder** aus Komponenten **bestehen**
- Werden durch Programmquellen, statische und dynamische Bibliotheken, Skripte und Makefiles repräsentiert

◆ Beispiele:

- **Fachliche Komponenten:** erbringen fachliche Funktionalität, z.B. Kundenverwaltung, Datumskomponente
- **Technische Komponenten:** erbringen technische Funktionalität, z.B. Datenbank, Batchsteuerung

- ◆ Oft noch Unterscheidung: **Subsystem** (für sich sinnvoll) und **Komponente** (Bausteine zur Wiederverwendung in größerem Kontext)

4.3.1. Komponentenschnittstellen

4.Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

◆ Komponentenschnittstellen

- Definieren die **Aufgaben des Dienstleisters** (Server) und den **Nutzen der Kunden** (Clients)
- Bestehen aus Angabe von **Syntax und Semantik der Operationen**, evtl. Parametertypen und Konstanten

◆ Beispiele für Beschreibungssyntax:

- Interface Definition Languages für COM oder CORBA
- Java-Interfaces
 - Achtung: bei Klassendefinition typischerweise mehr Informationen, z.B. durch private/protected

◆ **Semantik schwer zu beschreiben !**

Folie 17

4.3.1. Ressourcen

4.Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

◆ Ressourcen

- Umfassen die benötigte Software und Hardware

◆ Beispiele für Hardwareressourcen:

- Rechner (Prozessoren)
- Netzwerk (Kommunikation)
- Bildschirme (Ein/Ausgabe)
- Drucker (Ausgabe)

◆ Beispiele für Softwareressourcen:

- Middleware (z. B. Datenbankschicht),
- Systemsoftware (z.B. Betriebssystem)

Folie 18

4.3.1. Externe Systeme

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Externe Systeme** sind die Nachbarsysteme (d.h. existierende Anwendungssoftware), mit denen das System zusammenarbeitet
- ◆ **Achtung:** Abgrenzung zu Ressourcen manchmal nicht klar, wenn das andere System vor allem als Server arbeitet

4.3.1. Prozesse

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Prozesse**
 - Repräsentieren das System zur **Laufzeit**
 - Müssen **koordiniert** werden
- ◆ **Beispiele für Prozesse**
 - Dialogprozesse, reagieren auf Benutzeingaben
 - Batchprozesse

◆ Das Betriebskonzept

➤ Beschreibt die **Grenzfälle der Nutzung**, d.h.

- Installation, De-Installation
- Hochfahren, Herunterfahren
- Ressourcen-Ausfall
- Komponenten-Ausfall

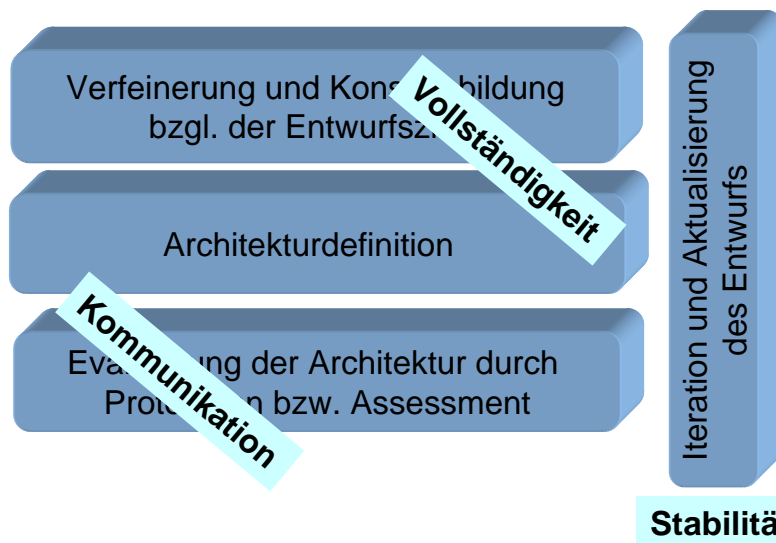
4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.1 Was ist beim Architekturentwurf zu tun?

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.3.1. Architekturdefinition

- ◆ Erzeugt Architekturdokumentation (4.3.3.)
- ◆ Verwendet Diagramme für die verschiedenen Sichten (4.3.3.)
- ◆ Beruht auf
 - Allgemeinen Entwurfsprinzipien (4.3.4)
 - Heuristiken, insbesondere Architekturmuster (4.3.5.)
- ◆ Setzt gute Technologiekenntnis voraus!

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.2. Festlegung von Entwurfszielen

- ◆ **Typische Ziele** [Starke 2002]
 - Allgemein: hohe Performanz, hohe Flexibilität, hohe Wartbarkeit
 - Persistenz der Daten
 - Integration von Altsystemen
 - Verteilung
 - Sicherheit (siehe Folie zu NFR Funktionalität)
 - Protokollierung
 - Internationalisierung

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Wird erreicht durch
 - Stärkere Hardware
 - Verringern der Kommunikation
 - Verringern der Flexibilität durch weniger Komponenten
 - Verringern der Verteilung durch Zentralisierung
 - Einführung von Redundanzen

- ◆ Verringert Flexibilität und Wartbarkeit

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ D.h. leichte Erweiterbarkeit des Systems

- ◆ Wird erreicht durch
 - Abstraktion (Schichten)
 - Viele Komponenten mit kleinen Aufgaben

- ◆ Verringert Performanz, erhöht Wartbarkeit

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ D.h. leichte Änderbarkeit
- ◆ von
 - Funktionalität
 - Datenstrukturen
 - Schnittstellen zu anderen Systemen
 - Benutzungsschnittstellen
 - Plattform
- ◆ Wird erreicht durch
 - Abstraktion
 - Kapselung (Verstecken interner Details)
- ◆ Verringert Performanz, erhöht Wartbarkeit

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Sicherstellen der Daten über die
Programmlaufzeit hinaus
- ◆ Persistenzmechanismus sollte gekapselt sein,
d.h. Zugriffe zu Datenbank oder Filesystem
- ◆ Oft verbunden mit Transaktionskonzept
- ◆ Beispiele:
 - Enterprise Java Beans

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Einbezug von Altsystemen und zugekauften Systemen
- ◆ Sollte möglichst wenig Änderungen an vorhandenen Systemen bewirken
- ◆ Insbesondere schwierig bei übergreifenden Transaktionen
- ◆ Typisches Beispiel:
 - Wrapper Konzept

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Ermöglicht
 - Einbindung von Altsystemen
 - Ausfallsicherheit
 - Lösung größerer Probleme
- ◆ Verursacht
 - Overhead in Kommunikation
 - Probleme bei der Entwicklung (z.B. Test)
- ◆ Benötigt Komplexe Basismechanismen
 - zur Bearbeitung verteilter Objekte
 - Für Datenkonvertierung und -transport
 - Für Fehlerbehandlung über Systemgrenzen hinweg

4.3.3. Architekturdokumentation

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

1. Einleitung (analog zu RE-Dokumenten)
 - Zweck, Umfang des Systems, Abkürzungen, Referenzen, Überblick, **insbesondere Entwurfsziele**
2. Soll-Architektur (ggf. aufbauend auf IST-Architektur) unter Angabe von
 1. Überblick
 2. Komponenten und Schnittstellen
 3. Externe Systeme (Umgebung)
 4. Ressourcen und Abbildung der Komponenten auf die Ressourcen
 5. Betriebskonzept
3. Hervorgehobene Aspekte
 1. Datenverwaltung (Persistenz)
 2. Zugangskontrolle und Sicherheit
 3. Allgemeiner Kontrollfluss
4. Komponentenbeschreibung
5. Annahmen (siehe RE-Dokumente)

[aufbauend auf
Bruegge, Dutoit 2004]

4.3.3. Architekturmodelle

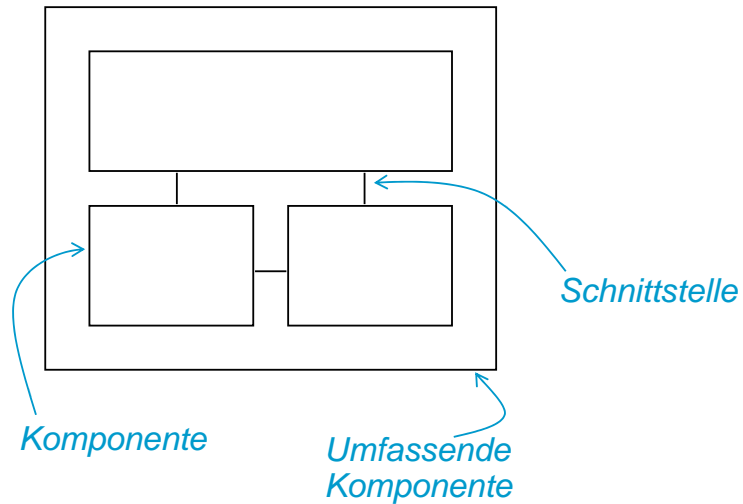
4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Komponentenstruktur**
 - Blockdiagramm
 - Kompositionsstrukturdiagramm (UML)
 - Komponentendiagramm (UML)
 - Paketdiagramm (UML)
- ◆ **Physische Struktur**
 - Konfigurationsdiagramm
 - Deploymentdiagramm (UML)

4.3.3. Blockdiagramme

- ◆ Blockdiagramme sind ein verbreitetes Hilfsmittel zur Skizzierung der logischen **Struktur** einer Systemarchitektur.



4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.3. Komponentenstruktur in UML

- ◆ UML bietet dafür mehrere Diagramme
 - **Paketdiagramm:** Überblick über Modell (rein logische Strukturierung)
 - **Kompositionsstrukturdiagramm (statisch) :** Wie ist mein System strukturiert und wie spielen (technische) Komponenten zusammen?
 - **Komponentendiagramm (dynamisch):** Welche (technischen) Komponenten entstehen zur Laufzeit und wie sind sie strukturiert?

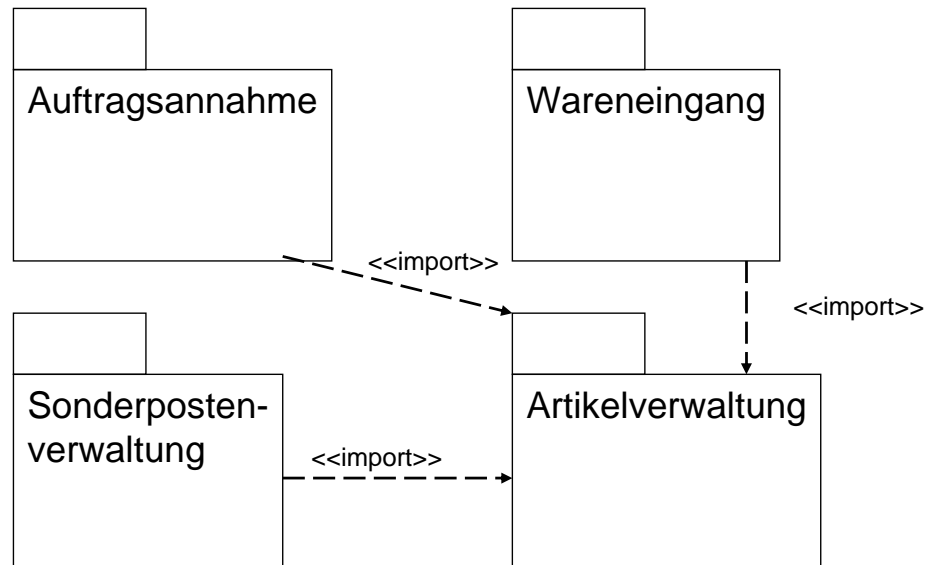
4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.3. UML Paketdiagramm

4.Methode

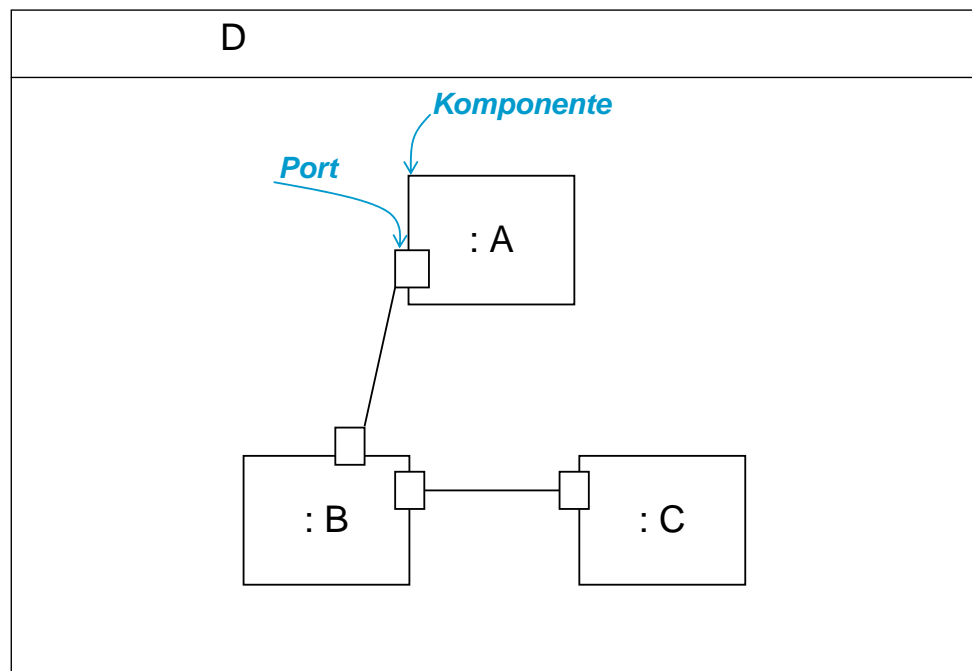
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.3.3. UML Kompositionsstrukturdiagramm

4.Methode

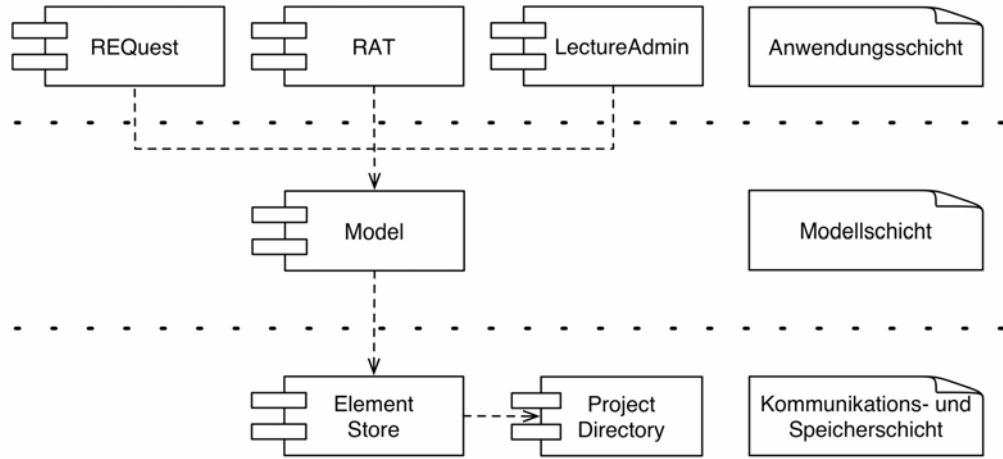
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.3.3. Beispiel (alte Notation)

4.Methode

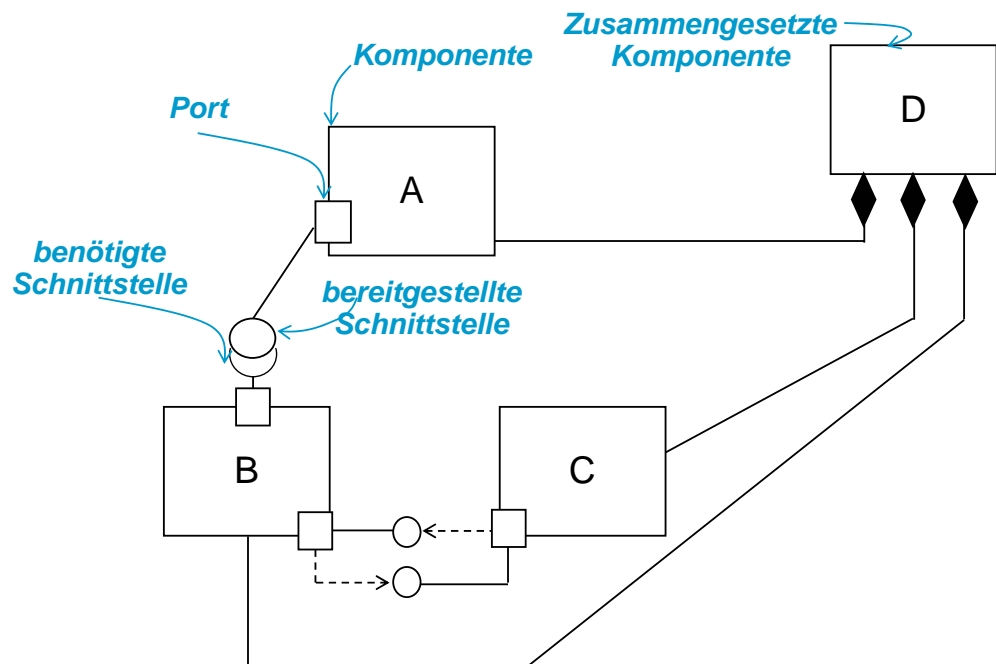
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



4.3.3. UML Komponentendiagramm

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

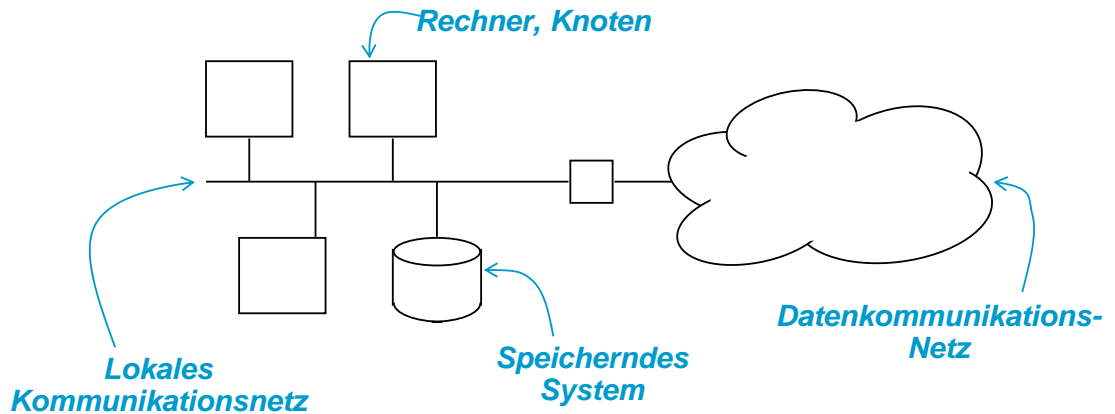


4.3.3. Konfigurationsdiagramme

- ◆ Konfigurationsdiagramme sind das meistverbreitete Hilfsmittel zur Beschreibung der physikalischen Verteilung von System-Komponenten.

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

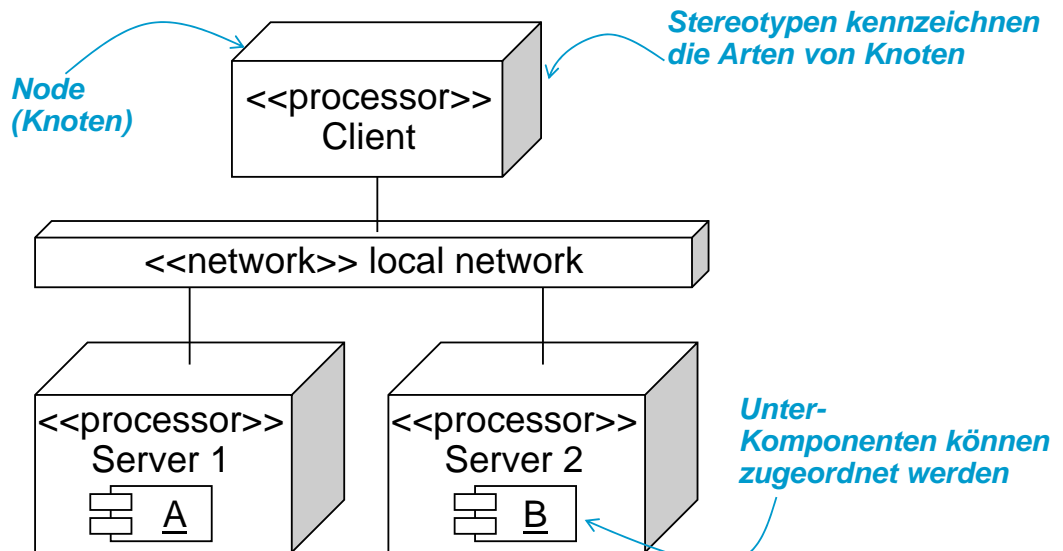


4.3.3. UML Verteilungsdiagramm

- ◆ engl.: *deployment diagram*
- ◆ zeigt die physische Verteilung von Systemen

4.Methode

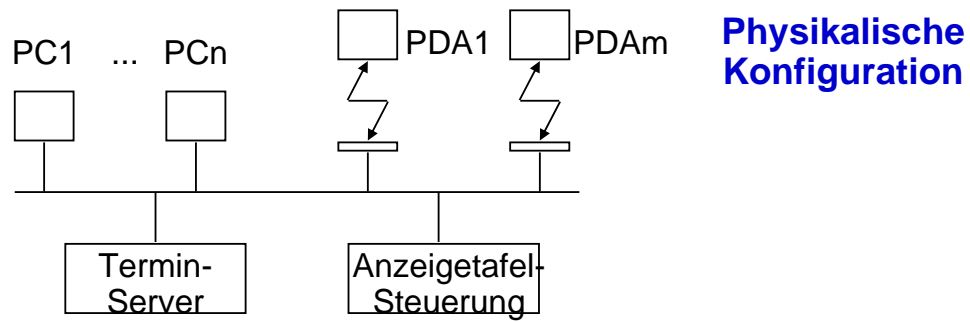
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



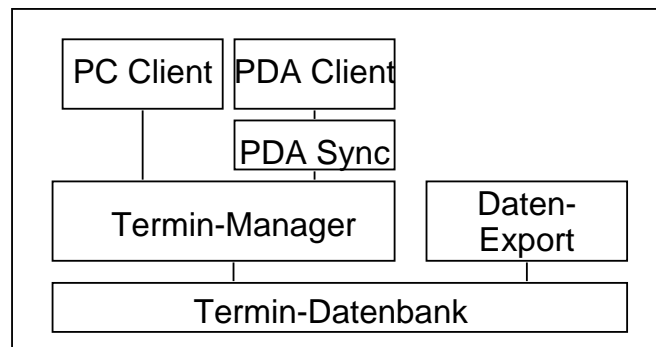
4.3.3. Beispiel Terminverwaltung

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



Blockdiagramm



Folie 41

4.3.4. Entwurfsprinzipien

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Hohe Kohäsion innerhalb einer Komponente**
 - erleichtert Verständnis, Wartung und Anpassung, da alle betroffenen Elemente (und keine anderen!) in der Komponente zu finden sind
- ◆ **Niedrige Kopplung**
 - erleichtert die Wartbarkeit und macht Systeme stabiler, da Änderungen nur bzgl. einer Komponente
- ◆ **Hierarchische Zerlegung, Schichtung**
 - Spezielle Form von hoher Kohäsion und niedriger Kopplung
- ◆ **Wiederverwendung**
 - Verringert Redundanz, erhöht Stabilität und Handhabbarkeit

Folie 42

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

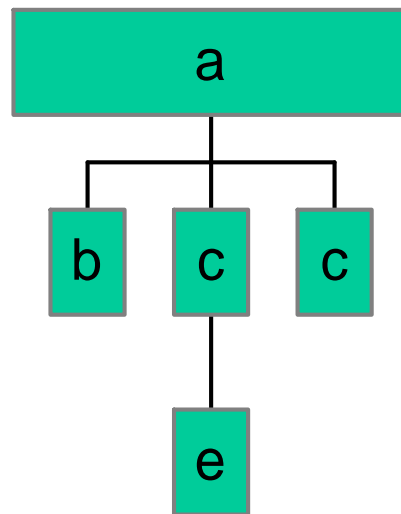
- ◆ ist ein Maß für die **Zusammengehörigkeit** der Bestandteile einer Komponente.
- ◆ **Hohe Kohäsion**: starke Abhängigkeit zwischen den Elementen einer Komponente,
 - Es gibt keine Zerlegung in Untergruppen von zusammengehörigen Elementen
- ◆ **Mechanismen zu Erreichung von Kohäsion**:
 - Früher: ähnliche Funktionalitäten zusammenfassen
 - Schlecht, wenn Daten verstreut
 - Prinzipien der Objektorientierung (Datenkapselung)
 - Verwendung geeigneter Muster zu Kopplung und Entkopplung

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ ist ein Maß für die **Abhängigkeiten** zwischen Komponenten.
- ◆ **Niedrige Kopplung**: geringe Abhängigkeiten zwischen Komponenten
- ◆ **Mechanismen zur Reduktion der Kopplung**:
 - **Schnittstellenkopplung**, d.h. Austausch nur über Schnittstellen
 - Z.B. get/set-Operationen statt Attributzugriff
 - ◆ **Datenkopplung**, d.h. gemeinsame Daten von Komponenten **vermeiden!**
 - ◆ **Strukturkopplung**, d.h. gemeinsame Strukturanteile **vermeiden!**
 - z.B. keine Vererbung über Komponentengrenzen hinweg

4.3.4. Hierarchische Zerlegung



Elemente jeder Ebene bilden eine Schicht:
Geschlossene Architektur:
Kommunikation nur innerhalb einer Schicht und zu angrenzenden Schichten

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.4. Interne Wiederverwendung

- ◆ ist ein Maß für die **Ausnutzung von Gemeinsamkeiten** zwischen Komponenten
- ◆ Hohe Wiederverwendung: Komponente wird in vielen Varianten oder in vielen Kontexten verwendet
- ◆ Mechanismen für Wiederverwendung:
 - Vererbung, Parametrisierung
 - Module/Objekte mit allgemeinen Schnittstellen
- ◆ **Aber: Wiederverwendung kann die Kopplung erhöhen:**
 - Schnittstellenkopplung und Strukturkopplung

4.Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- ▶ 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

4.3.4. Zusammenfassung Architektur

4.Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

- ◆ Architekturentwicklung basiert heute immer noch meist auf **Erfahrung**, es gibt wenig generelle Prinzipien
- ◆ Methoden zur klaren Definition von Entwurfszielen (insbesondere Lösung von Konflikten zwischen Zielen) und zur Evaluation von Architekturen sind noch Forschungsthemen

4.3.4. Literatur

4.Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

- ◆ B. Bruegge, A. Dutoit, Object-oriented Software engineering, Pearson, 2004
- ◆ A. Enders, D. Rombach, A Handbook of software and systems engineering, Pearson, 2003
- ◆ IEEE Std. 1471-2000, Recommended Practice for architectural description of software-intensive systems
- ◆ J. Siedersleben, Softwaretechnik, Hanser, 2003
- ◆ I. Sommerville, Software Engineering, Addison Wesley, 2001
- ◆ G. Starke, Effektive Software Architekturen, Hanser, 2002