

3.6 Physische Sicht

3.6.1 Implementations-Sicht

3.6.2 Verteilungs-Sicht

3.6.1 Implementations-Sicht (Implementation View)

- Zeigt die *Komponenten* und *Schnittstellen* eines Systems und deren Abhängigkeiten untereinander
- *Komponente* = Software-Baustein, der Teil der Implementierung des gesamten Systems ist

Komponenten (Forts.)

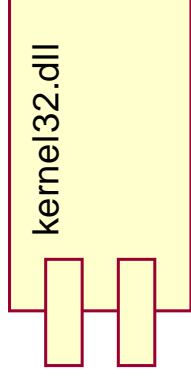
- Komponenten können sein:
 - Quell Code Dateien
 - Objekt Dateien und Libraries
 - ausführbare Programme (Executables)
 - Dokumente, Hilfe Dateien
 - Makefiles, Skripte
 - Dateien mit Daten, Datenbanktabellen
 - Initialisierungs- oder Konfigurationsfiles

Komponenten (Forts.)

- Komponenten Konzept findet man in den meisten Programmiersprachen, Betriebssystemen oder Programmiermodellen wieder:
 - Objekt Dateien, Libraries, class-Dateien
 - COM+ Komponenten, (Enterprise) Java Beans
 - CORBA, DCOM

Komponenten (Forts.)

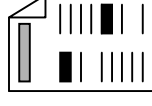
Graphische Darstellung:



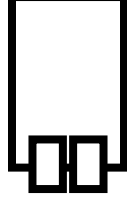
Komponenten - Stereotypen

- Stereotypen/Icons für Komponenten:

- <<document>>



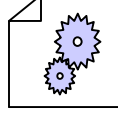
- <<executable>>



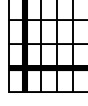
- <<file>> Source Code oder Daten



- <<library>>



- <<table>> Datenbanktabelle

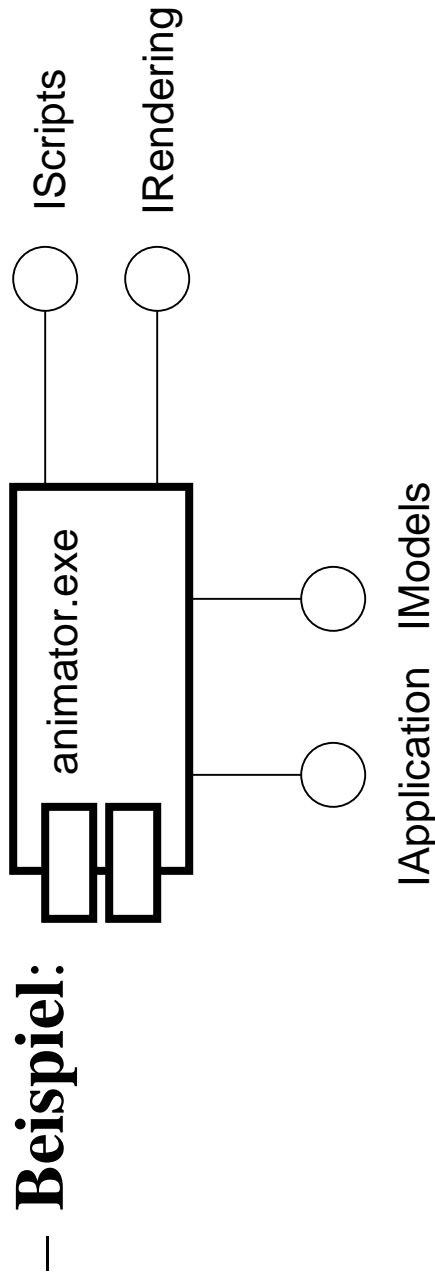


Komponentendiagramm

- *Komponentendiagramm*
 - zeigt Komponenten und deren Beziehungen untereinander
 - Fokussierung auf verschiedene Aspekte

Komponentendiagramm - API

- Modellierung eines **API** (Application Programming Interface)
 - *Schwerpunkt*: Darstellung der **Schnittstellen** und Operationen, die zur Verfügung gestellt werden

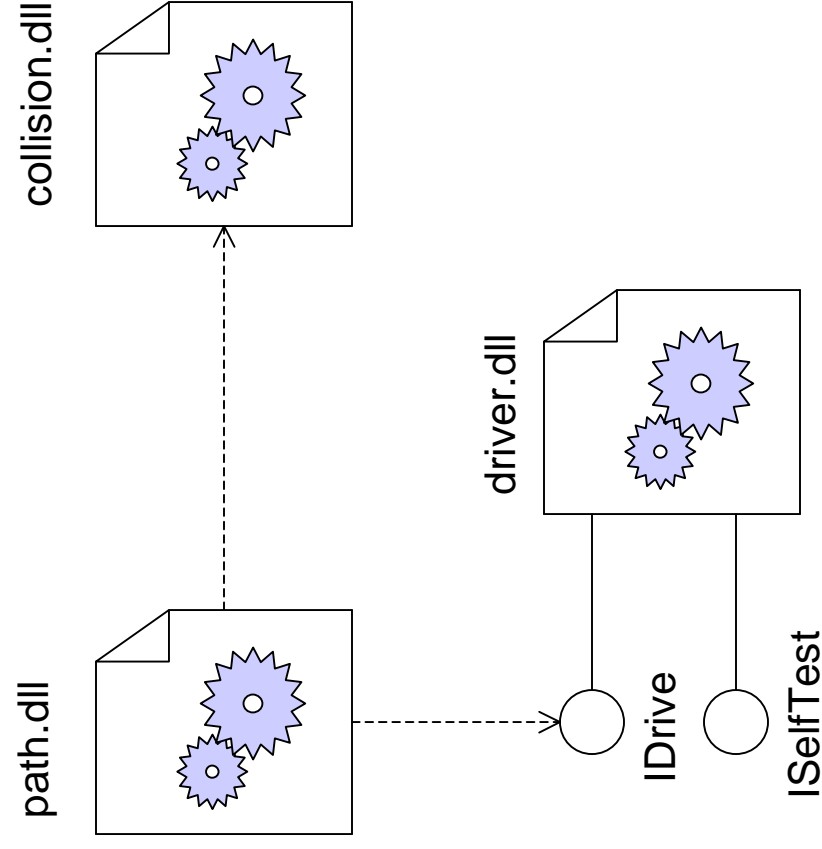


Komponentendiagramm - Release

- Modellierung einer kompletten **Release des Systems**
 - Nur sinnvoll für komplexe Systeme
 - *Schwerpunkt*: Darstellung der **Abhängigkeiten** zwischen **Libraries** und **Executables**
 - Explizite Darstellung zeigt zusätzlich **Schnittstellen**, die von den **Libraries benutzt** (importiert) und **realisiert** (exportiert) werden

Komponentendiagramm - Release (Forts.)

Beispiel:

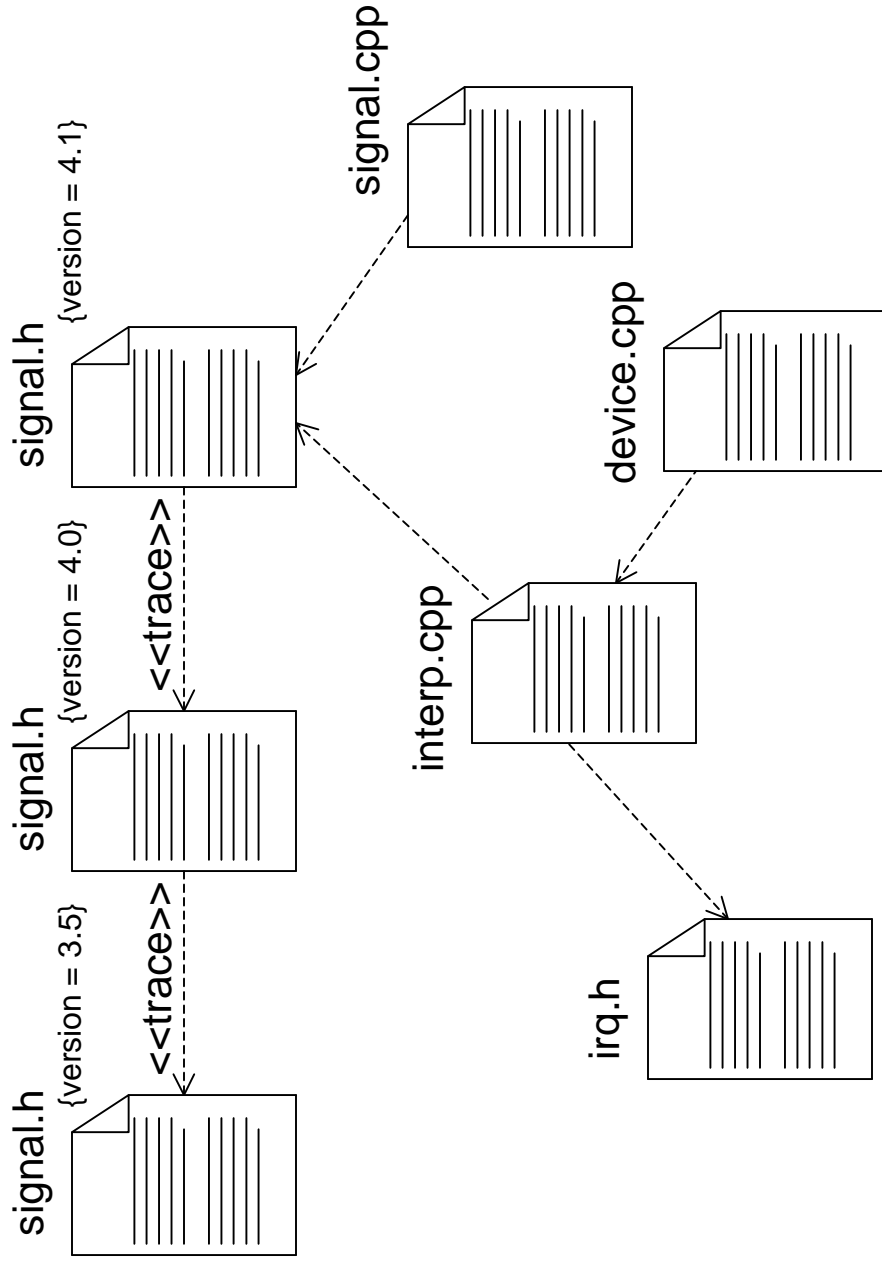


Komponentendiagramm - Source Code

- Modellierung von **Source Code**
 - *Schwerpunkt*: Darstellung von
 - **Versionsmanagement**
 - **Kompilationsabhängigkeiten**
 - Benutzung geeigneter Entwicklungswerkzeuge (z.B. cvs, makedepend) und UML für graphische Darstellung

Komponentendiagramm - Source Code (Forts.)

Beispiel:

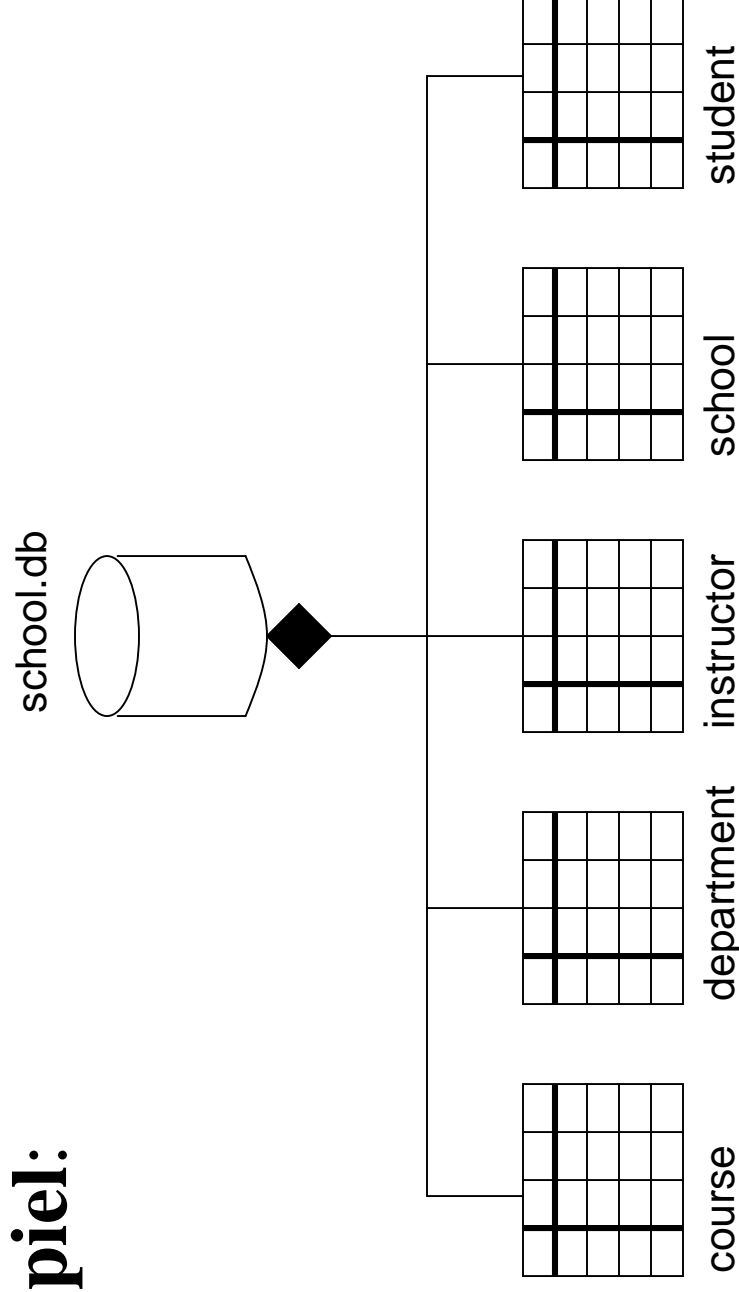


Komponentendiagramm - Datenbank

- Modellierung einer **Datenbank**:
 - *Schwerpunkt*: Darstellung der verschiedenen **Tabellen** des Datenbank Schemas sowie deren **Abhängigkeiten**

Komponentendiagramm - Datenbank (Forts.)

Beispiel:



Entwurf von Komponenten

- **Ziel:**
 - möglichst **wenig Abhängigkeiten** zu anderen Komponenten
 - **Ersetzbarkeit**
 - keine *direkten* Abhängigkeiten zu bestimmten Komponenten, sondern nur **Benutzung von Schnittstellen**
 - **Bereitstellung** von Funktionalität über **Schnittstellen**

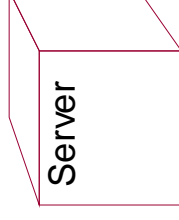
3.6.2 Verteilungs-Sicht (Deployment View)

- Zeigt die **Topologie** der Hardware Knoten eines Systems und die **Verteilung von Komponenten** auf den Knoten zur Laufzeit
- Darstellung in **Verteilungsdiagramm**
- **Knoten** (Node) = zur Laufzeit physisch vorhandenes Gerät, das über Speicher und (meistens auch) Rechenleistung verfügt

Knoten

- **Beispiele:**
 - PC
 - Server
 - RAID System
 - Modem
 - Internet

- **Graphische Darstellung:**

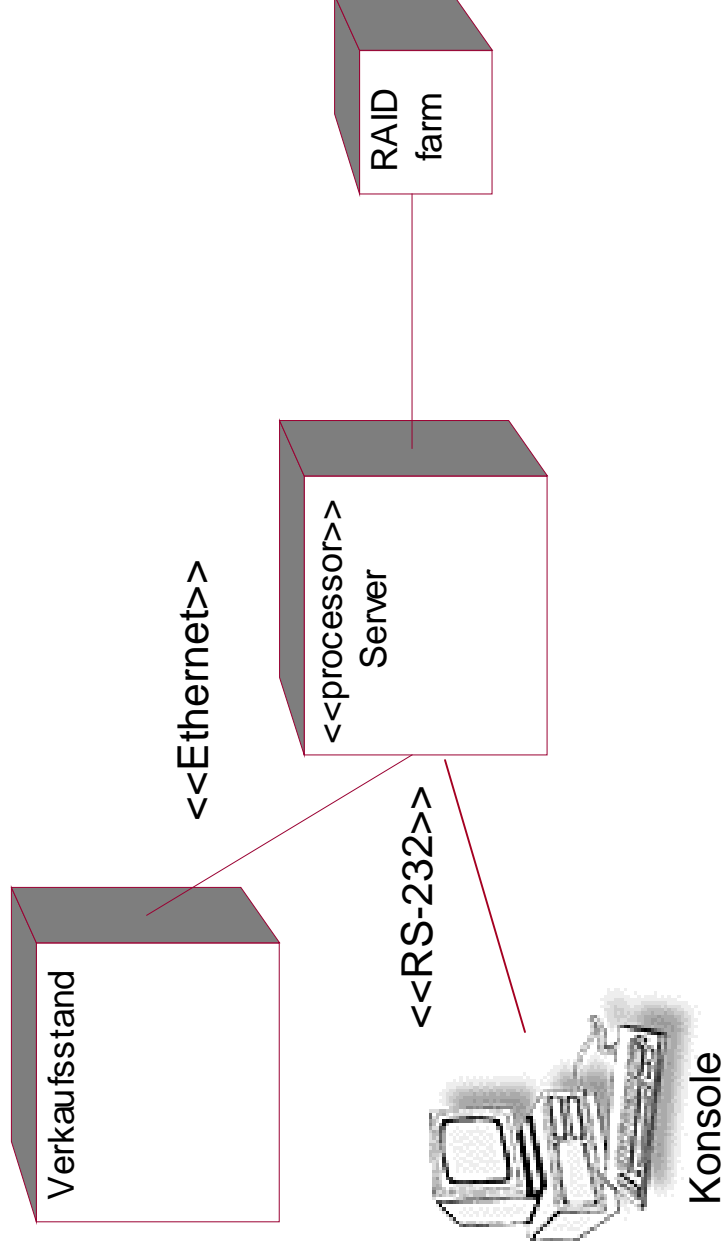


Assoziation/Verbindung

- **Assoziation** bzw. **Verbindung** (Link) zwischen Knoten repräsentiert **physikalische Verbindung**
- **Beispiele:**
 - Ethernet
 - ISDN
 - RS-232
- Nutzung von Stereotypen/Icons zur näheren Beschreibung von Knoten oder Assoziationen

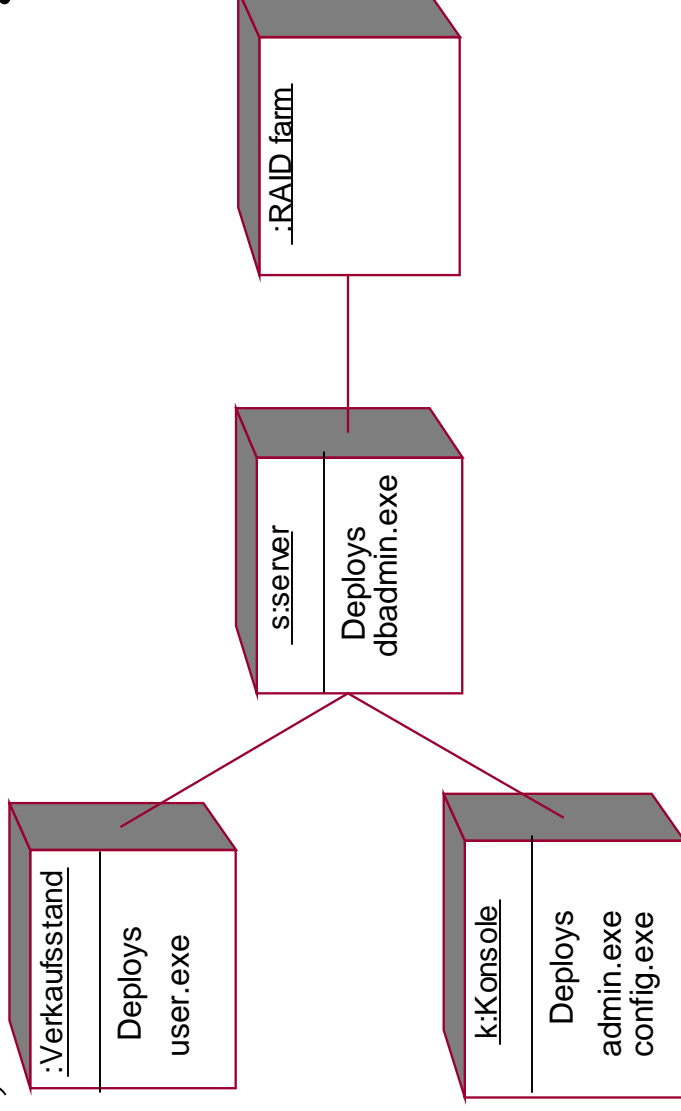
Assoziation/Verbindung (Forts.)

Beispiel:



Komponentenverteilung

- Komponenten, die auf Knoten ausgeführt werden, werden in zusätzlichem Fach aufgelistet

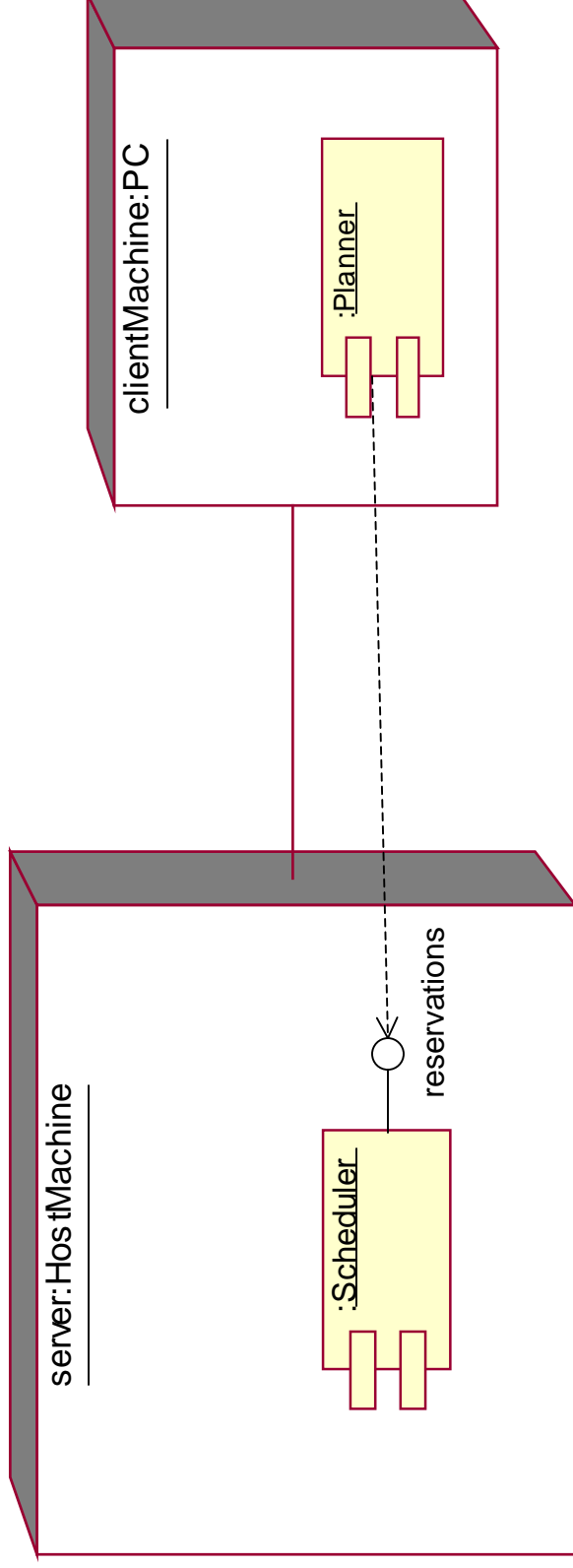


Verteilungsdiagramm - Inhalte

- Modellierung der *Topologie* von
 - Client/Server Systemen
 - verteilten Systemen
 - eingebetteten Systemen
- Modellierung der *Verteilung von Komponenten* auf Knoten
 - Mapping des Komponentendiagramms auf Topologie der Knoten

Komponentenverteilung

Beispiel:



Physische Sicht - Zusammenfassung

- *Implementations-Sicht* für statische Struktur und Abhängigkeiten der implementierten Elemente
- *Verteilungs-Sicht* für Hardware Topologie und Verteilung von Komponenten auf Knoten
 - sinnvoll für verteilte Systeme
 - Darstellung von Objektmigrationen