Software Engineering III (SE3)

Softwarearchitektur und Verteilte Systeme

Prof. Dr. Arne Koschel

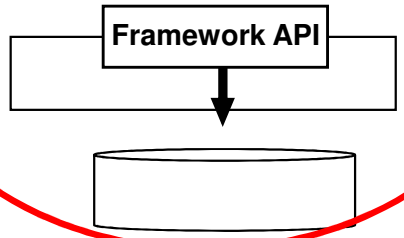
Hochschule Hannover

Fakultät IV, Abteilung Informatik

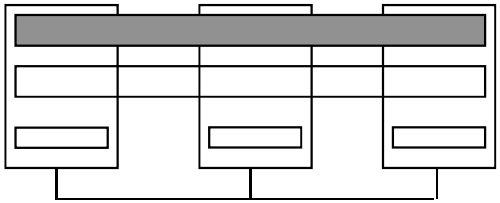
Software Engineering III – Wo sind wir?

Softwarearchitekturen

Softwarearchitekturen: Begriffe & Fallbeispiel Persistenz-Framework

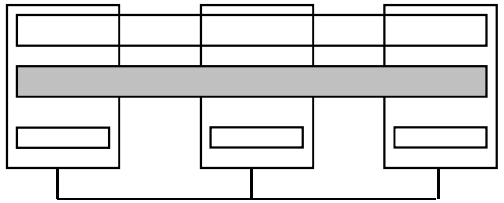


Verteilte Softwarearchitekturen



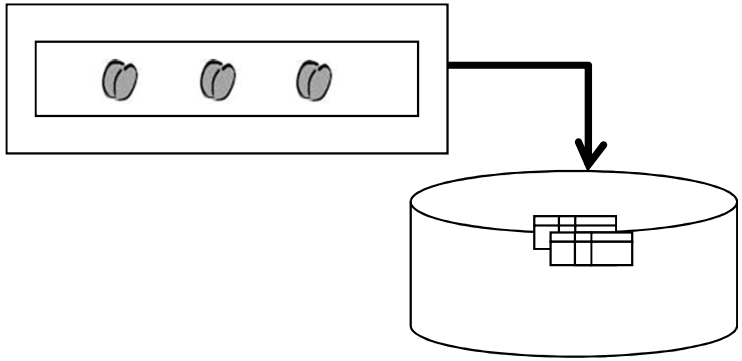
Middleware/Verteilung

Sockets, RMI, MoM/JMS Web Services



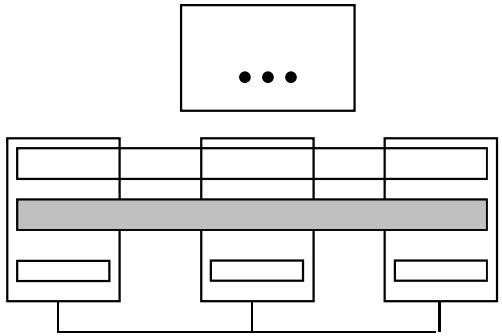
Enterprise Java

Java EE (EJB, …)



Verteilung – weitere Konzepte

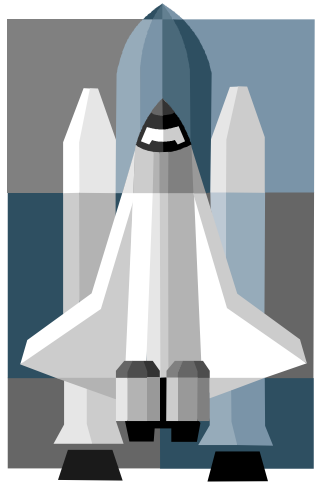
Weitere Middleware



Gliederung

Einstiegsfrage

- **Was bedeutet für mich „Softwarearchitektur“?**



  - 1 Min. – Eigene Erklärung
  - 2 Min. – Diskussion mit dem/r direkten Nachbarn/in
    Ziel: Ein „zusammenfassendes Gruppenergebnis“
  - 2 Min. – Diskussion in 5-7er-Gruppen
    Ziel: Ein „zusammenfassendes Gruppenergebnis“
  - 5 Minuten – Zusammentragen aller Gruppenergebnisse

„Was bedeutet für mich „Softwarearchitektur?“ - Gruppenergebnisse

- Auf abstrakterer Ebene die Teile beschreiben aus denen Software besteht.
- Masterplan für alle Personen eines Projektes
- Sicht auf außen auf die Software
- „Schichten“ als Beispiel für ein Softwarearchitekturmuster
- Persistenz als Teil einer Softwarearchitektur

Motivation für eine Softwarearchitektur

- die **Entwicklung komplexe Softwaresysteme** erfordert
  - **Bauplan des Systems**
  - Spezifikation der wesentlichen **Strukturen**
  - **deutlich höhere Abstraktionsebene als Source Code**

Definition **aus L. Bass: Software Architecture in Practice, Addison Wesley, 2010:**

*„The **software architecture** of a program or computing system is the **structure** of the system, which comprises **software components**, the **externally** visible **properties** of those components and the **relationships between them** “*

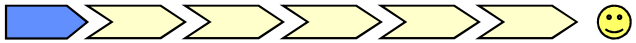
Softwarearchitektur

- ist die **Struktur des Softwaresystems**
  - übergeordnete **Organisation** des Systems
  - mit den **globalen Kontrollstrukturen**

- definiert **Softwarekomponenten**
  - wesentliche Bausteine der Software
  - **Realisierungsdetails** sind **verborgen**
  - Komponenten **interagieren und kooperieren**

Architekturtypen

- welche Systemaspekte werden in der Architektur beschrieben?
  - **Fachliche Architektur**: fachliche Komponenten
  - **Technische Architektur**: anwendungsübergreifende Komponenten



Fachliche Architektur

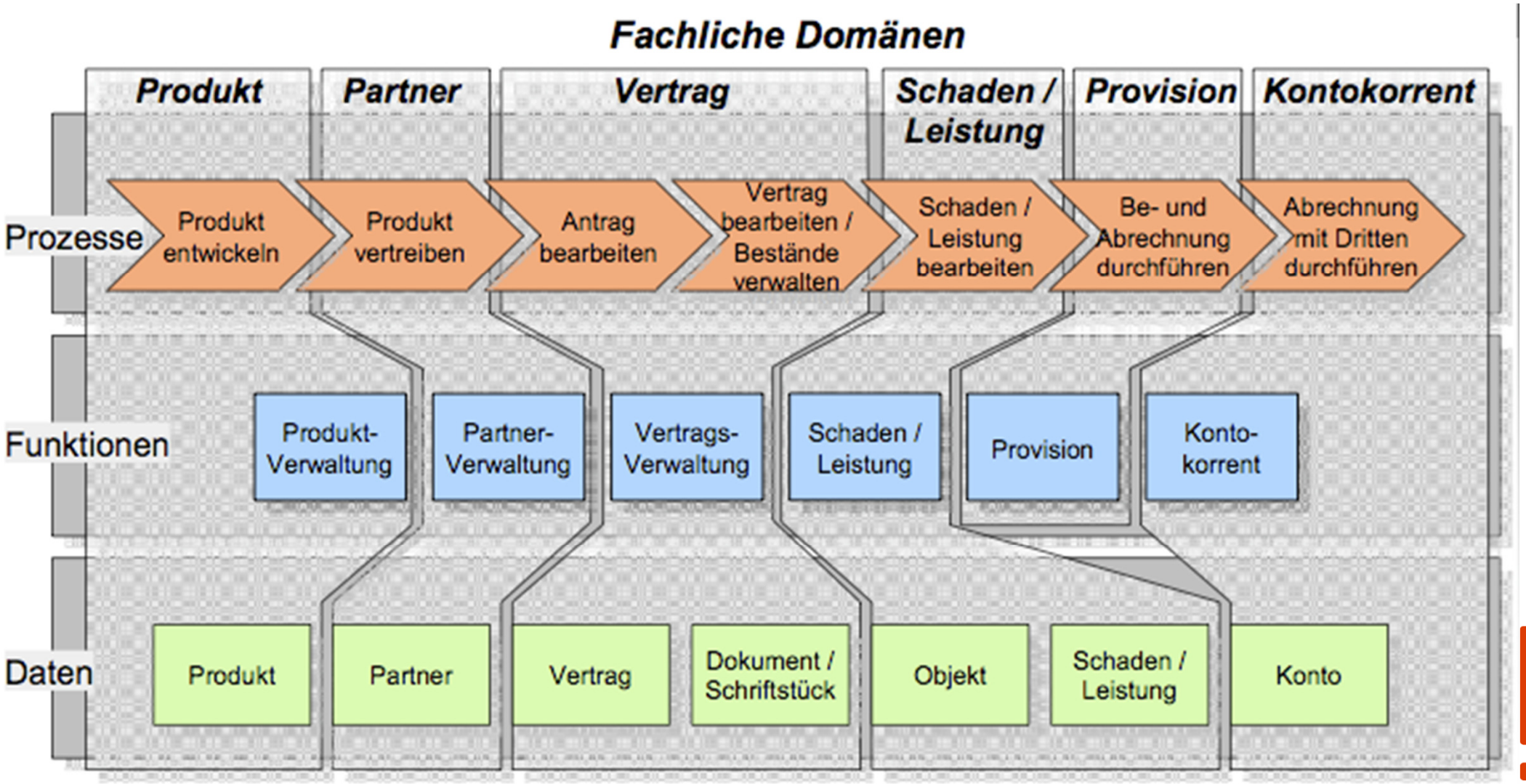
- Software aus **Sicht der Anwender**
  - an den fachlichen Zielen ausgerichtet
  - getrieben durch die Domäne und die funktionalen Anforderungen

- **fachliche Komponenten**: Geschäftsobjekte, Geschäftsregeln, Geschäftsprozesse

- **Eigenschaften:**
  - oft **singulär** (einzigartig): es gibt wenige Referenz- oder Standardarchitekturen
  - oft fachlich anspruchsvoll und deshalb schwierig
  - **hoher Abstraktionsgrad**
  - **unabhängig von technischen Plattformen**

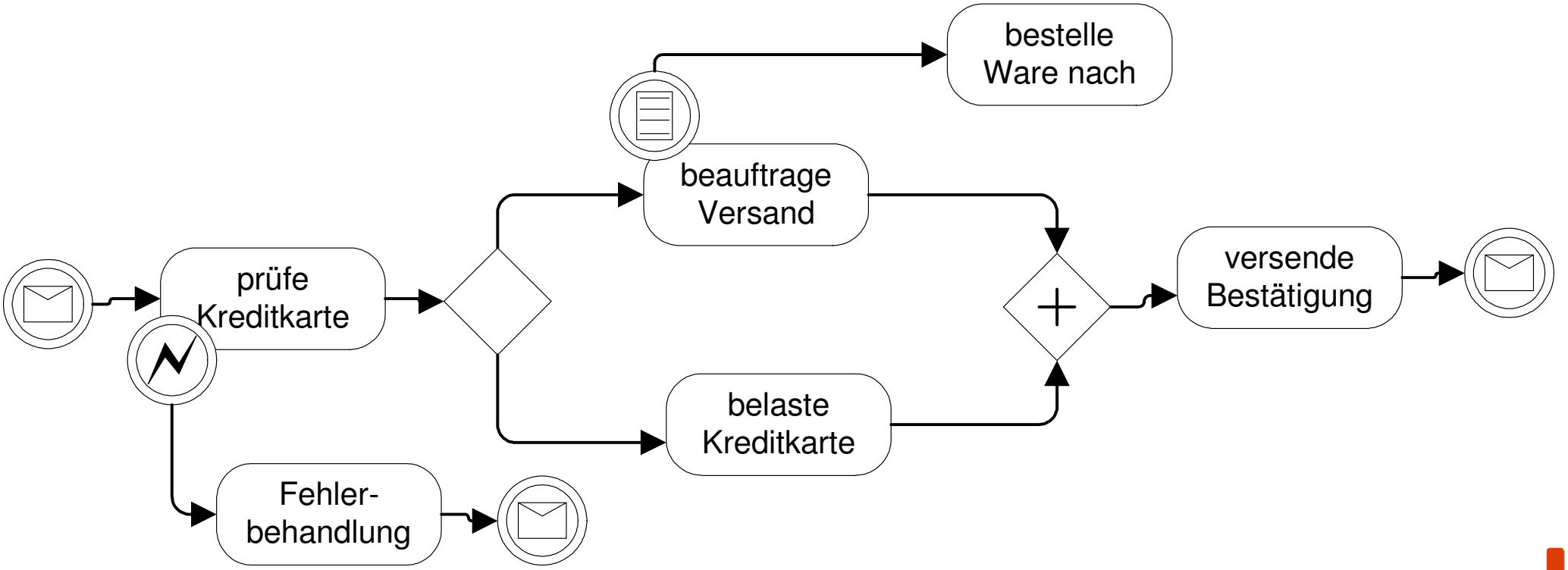
Beispiel (1): Fachliche Architektur

- Anwendungsarchitektur der deutschen Versicherungswirtschaft VAA



Beispiel (2): Fachliche Architektur

- Prozessbeschreibung in BPMN (Business Process Modeling Notation)



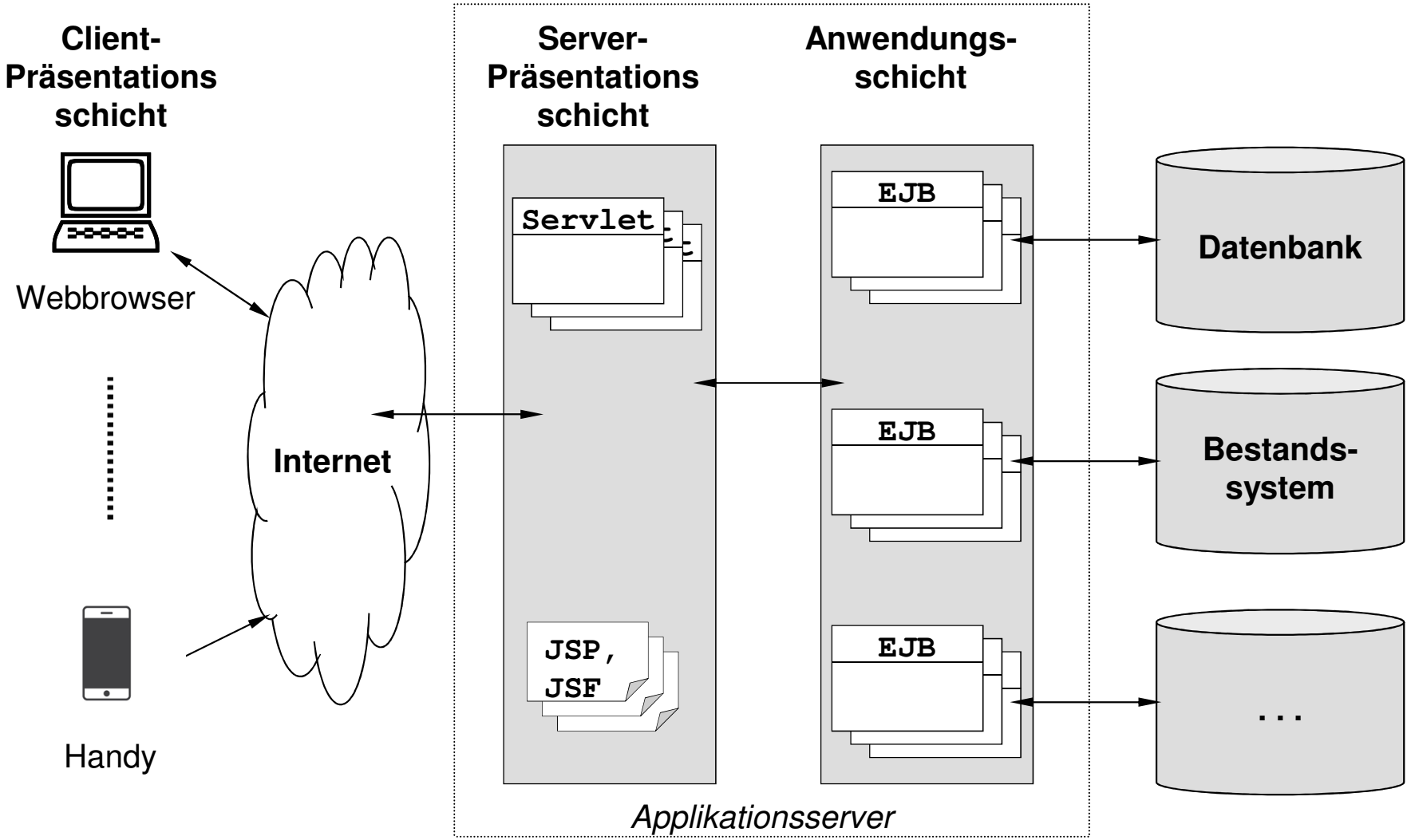
Technische Architektur

- Bausteine für **infrastrukturelle Basisdienste**
  - **Präsentation** (für Fenster- oder **Websysteme**)
  - Backend-Access (**Persistenz**, Legacy-System-Access,..)
  - **Kommunikation**, Sicherheit, Zugriffsrechte, Workflow-Mgmt., Logging, …

- **Eigenschaften**:
  - wird von jeder Anwendung benötigt
  - **niedrigerer Abstraktionsgrad** (= implementierungsnah)
  - **domänenneutral**
  - **abhängig von der technischen Plattform**

- **Referenzarchitekturen:** Java EE, CORBA, Web Services, Grid, P2P, Cloud, …
- **Produkte /Technologien:** Java EE App. Svr., Struts, JSF, Frameworks (Spring, Hibernate, JDO...)

Beispiel: Technische Architektur – Java EE



aus J.Dunkel, A. Holitschke: Softwarearchitektur, Springer, 2003

Architektursichten

- es gibt **unterschiedliche Sichten** (oder Perspektiven) auf das selbe System

Kontext-Sicht (auch: Context View) – „Blick auf Nachbarsysteme“

- Welche Nachbarsysteme gibt es außerhalb des Systems (mit welchen Schnittstellen)?

Struktur-Sicht (auch: Component View) – statisch

- beschreibt Architektur-Bausteine mit deren Schnittstellen und Kommunikations-Beziehungen

Verhaltens-Sicht (auch: Behaviour / Interaction / Dynamic View) – dynamisch

- Interaktionen und Zustandsbeschreibungen von Architektur-Bausteinen

Abbildungs-Sicht (auch: Deployment View)

- Abbildung der Architektur-Bausteine auf
  - Hardware
  - Softwaremodule (=Dateien und Pakete)

➔ Beispiele

Detaillierungsgrad der Beschreibung

- mehrere **zweck- und zielgruppenorientierte** Dokumente
  - was will der potentielle Leser erfahren, bzw. was kann er verstehen?

- **Design-Rationale**
  - Rahmenbedingungen und Gründe für eine Architektur (Architekturentscheidung)
- **Architektur-Steckbrief**
  - kurzer Überblick für alle Stakeholder
- **Architektur-Dokument**
  - beschreibt alle Strukturen, Komponenten und Abläufe
  - Einsatz von Modellen (UML)
- **Entwicklerhandbuch**
  - kochbuchartige Anleitung für Entwickler: wie wird in der Architektur implementiert
  - Quelltextbeispiele

Eigenschaften von Komponenten

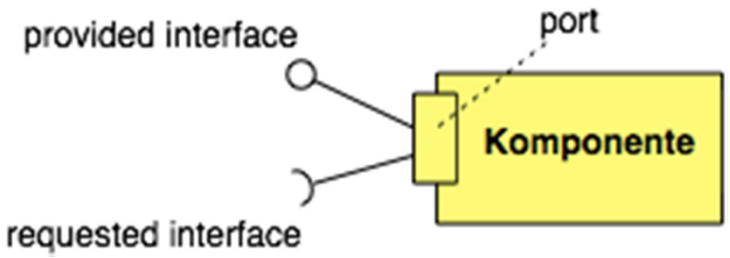
- **zerlegen das Gesamtsystem** in seine Bausteine
  - schaffen damit neue Abstraktionsebene **(divide and conquer)**

- sind durch **Schnittstellen** beschrieben
  - **exportierte Schnittstellen** (provided interfaces): bereitgestellte Dienste
  - **importierte Schnittstellen** (requested interfaces): benötigte Dienste

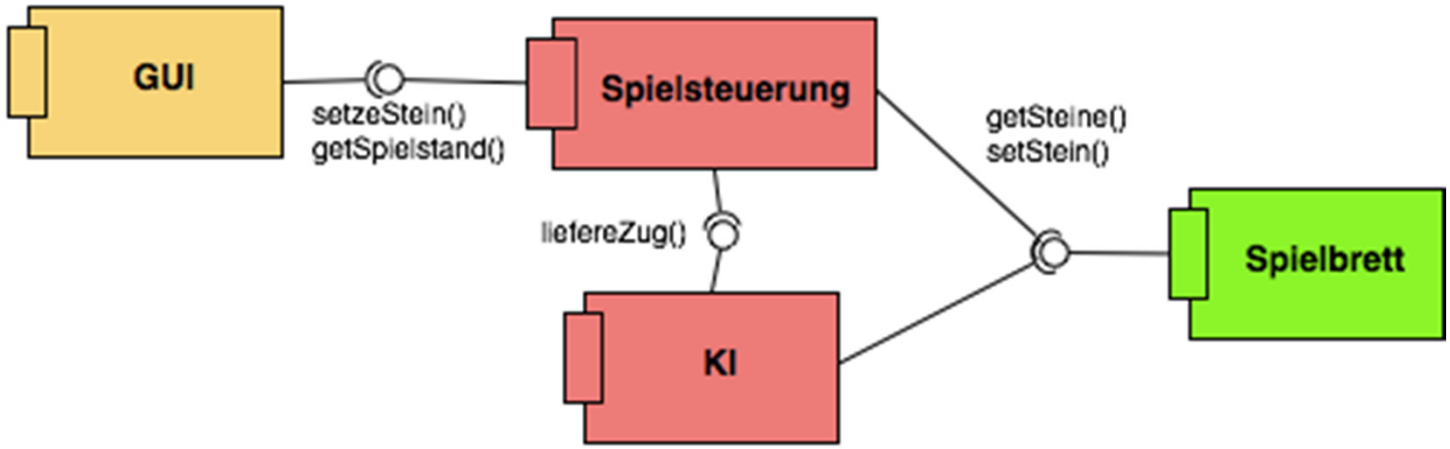
- gewährleisten **information hiding**
  - verstecken die Implementierung
  - kapseln Realisierungsdetails (→ sind eine Abstraktion)
  - enthalten ggf. weitere Komponenten

UML2 – Komponentendiagramm

- neues Diagramm zu Beschreibung von Architekturbausteinen

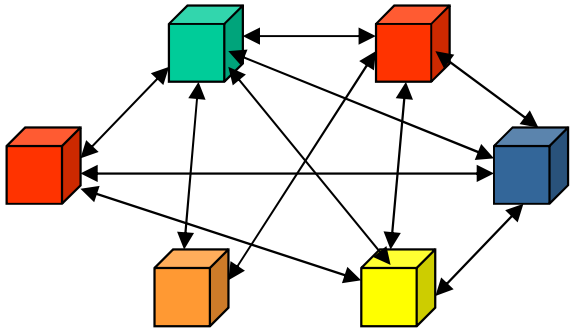


Beispiel: Komponentendiagramm für ein Brettspiel

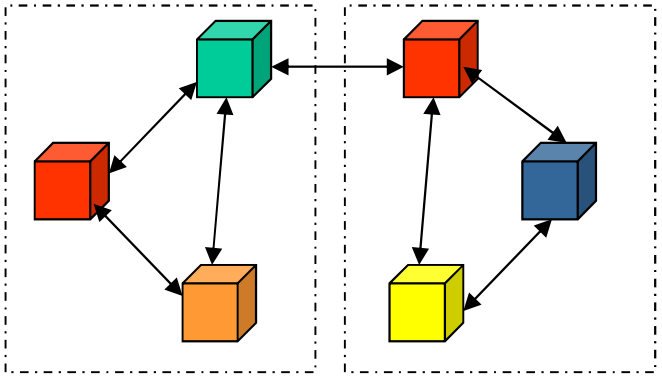


Leitgedanken bei der Strukturierung von Software-Systemen

- Leitgedanke 1: **Starke Kohärenz**
  - Teilsystem besitzt zusammenhängende, klar umrissene Aufgabe
  - Einfachheit, Verständlichkeit, Redundanzfreiheit, bessere Teambildung
- Leitgedanke 2: **Lose Kopplung**
  - schmale Schnittstellen zwischen Teilsystemen (kleine Menge von Methoden (API))
  - Teilsysteme kennen nur wenige andere Teilsysteme
  - Robustheit gegenüber Änderungen



stark gekoppelte Teilsysteme



lose gekoppelte Teilsysteme

Architekturmuster

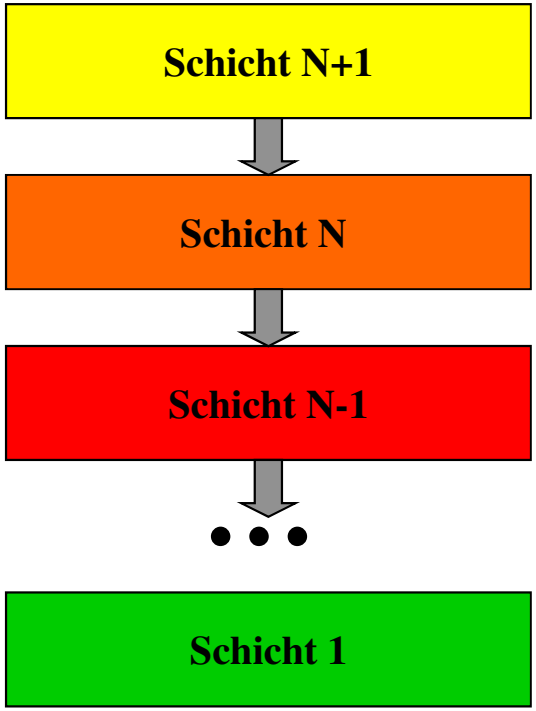
- legen den **Stil der Gesamtarchitektur** fest (grobe Granularität)
- **beschreiben das gesamte System**
- Beispiele:
  - Schichten (Layers), Event-Driven Architecture (EDA),
    Service-based Architectures (Service-oriented Architecture (SOA), Microservices),
    Hexagonale Architekturen, Peer-to-Peer, Blackboard, …

Entwurfsmuster

- beschreiben einen **bestimmten Aspekt** des Systems
- besitzen mittlere Granularität
- Beispiele:
  - Data Access Objects, Factory, Enterprise Integration Pattern, Analyse-Pattern, . . .

Schichtenarchitektur

- Zerlegung des Softwaresystems in verschiedene Schichten
- Heutzutage sehr verbreitetes Architekturmuster



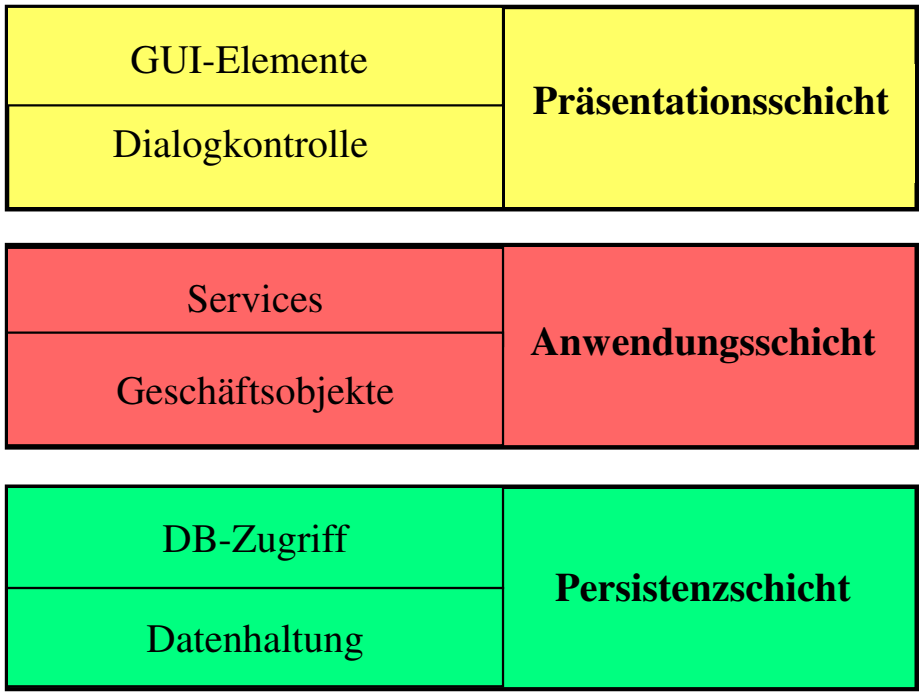
Eigenschaften:

- eine Schicht *N* stellt Dienste zur Verfügung, die nur von der darüber liegenden Schicht *N+1* genutzt werden können/dürfen
- eine Schicht *N* nutzt ausschließlich Dienste der Schicht *N-1*

Beispiel: 3 typische Softwareschichten

1. Präsentationsschicht

- **Präsentation fachlicher Daten**
  - mittels GUI-Elementen (Textfelder, Radiobuttons ,...)
- **Dialogkontrolle**
  - sendet Daten an die Anwendungsschicht (ausgelöst durch Benutzerereignis)
  - empfängt Daten aus der Anwendungs-schicht und bereitet sie auf
  - wählt das nächste Fenster aus

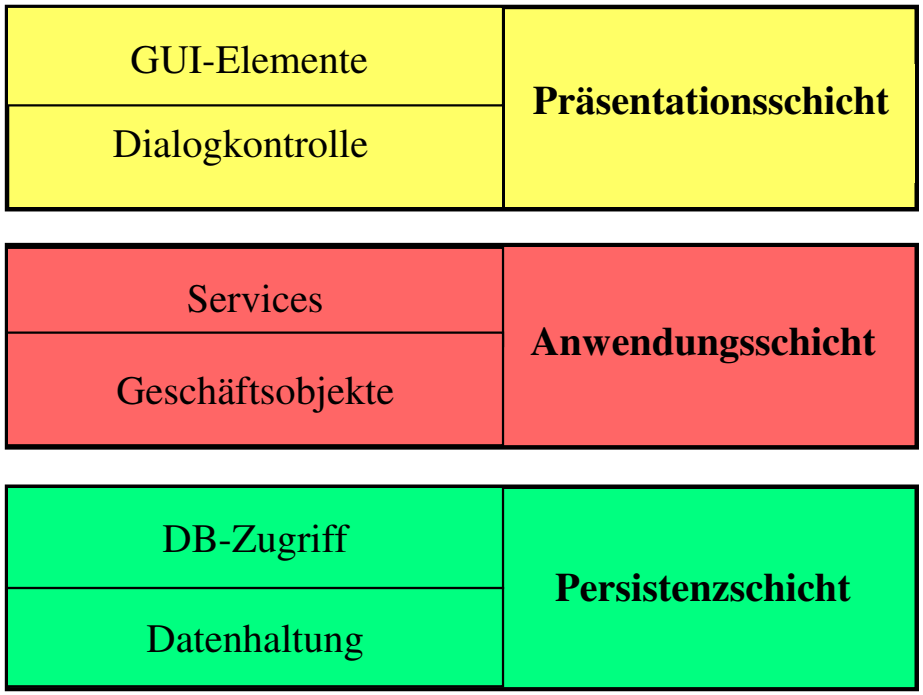


aus J.Dunkel, A. Eberhart, S. Fischer, C. Keiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

- Ziel: Präsentationsschicht weiß möglichst wenig von der Geschäftslogik

2. Anwendungsschicht

- **Services**
  - realisieren Methoden /Dienste für die Geschäftsprozesse der Anwendung
  - können z.B. durch GUI-Ereignisse ausgelöst werden
- **Geschäftsobjekte**
  - realisieren Konzepte der Domäne
  - datentragende Klassen



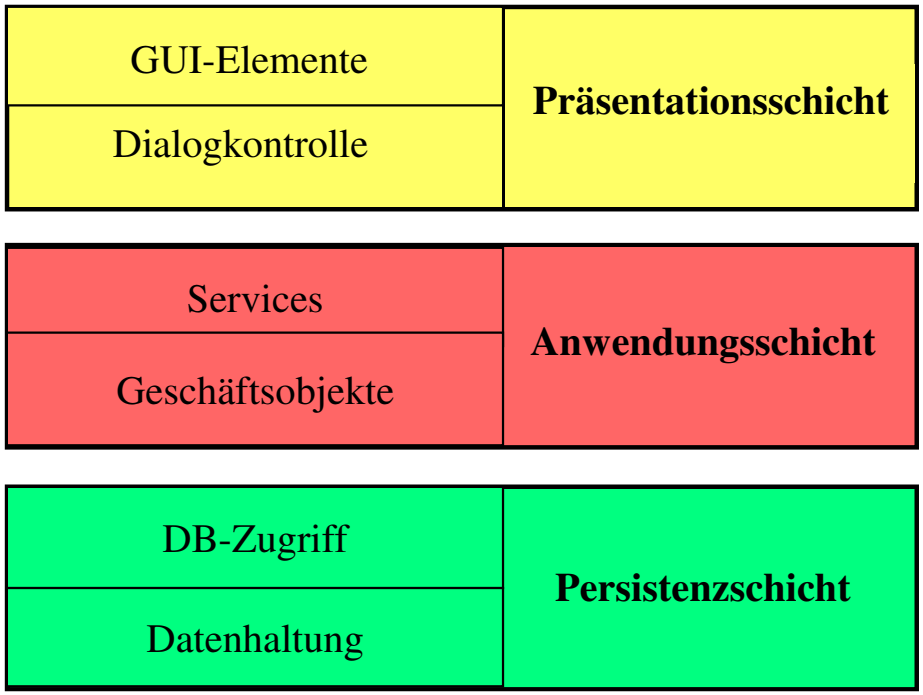
Quelle: J.Dunkel, A. Eberhart, S. Fischer, C. Keiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

- Ziele: Anwendungsschicht
  - kennt keine Präsentationsobjekte
  - nutzt die Persistenzschicht über schmale Schnittstelle

3. Persistenzsschicht

- **Datenhaltung**
  - Verwaltung fachlicher Daten in einem Datenbanksystem (RDBMS; XML-Dateien, Datei,..)

- **DB-Zugriff**
  - Code für den Zugriff auf die Datenbank (Suchen, Speichern von Objekten, Transaktionen ...)
  - bei RDBMS: Abbildung von Objekten auf DB-Tabellen (O/R-Mapping – object to relational mapping)
  - realisiert durch Persistenzframework



aus J.Dunkel, A. Eberhart, S. Fischer, C. Kleiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

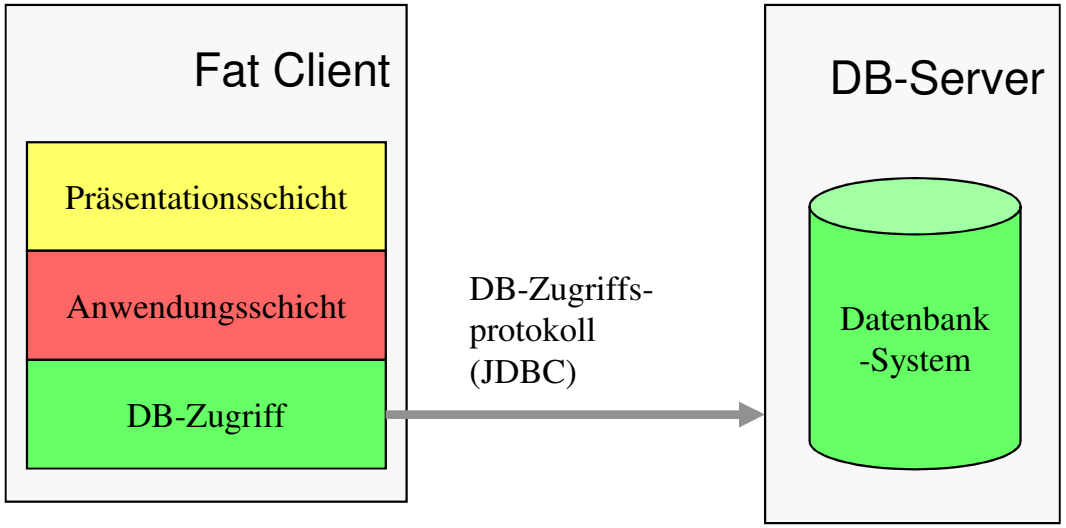
- Ziel: Persistenzschicht kennt nur wenig Strukturen der Anwendungsschicht (meist die Geschäftsobjekte)

Physikalische Verteilung der Schichten

- **Schichten** werden auf verschiedene Rechnersysteme **verteilt**
  - Zweischichten-
  - Dreischichten-
  - Mehrschichten (N-Tier)-Architektur

- **Motivation**
  - **mehrere Benutzer** sollen gleichzeitig die selben Daten verwenden
  - **Skalierbarkeit** höhere Anforderungen (Zahl der Dienstaufrufe, größere Datenmengen)

Zweischichtenarchitektur

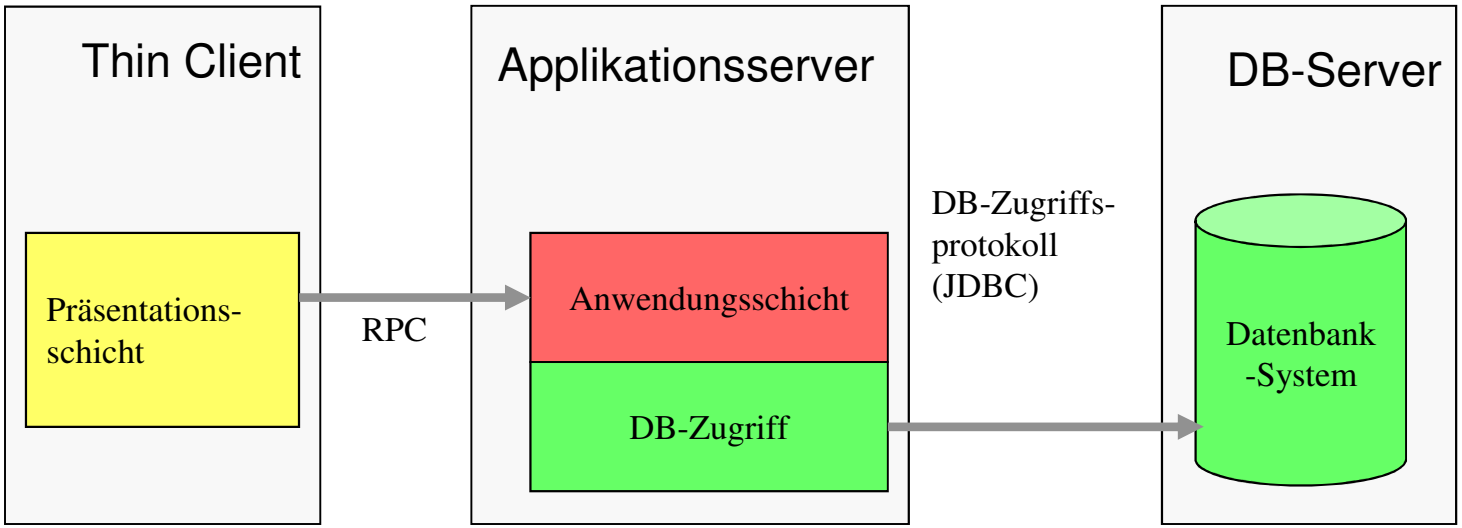


aus J.Dunkel, A. Eberhart, S. Fischer, C. Keiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

- **Fat-Client**: enthält alle drei logischen Schichten der Anwendung
- **DB-Server**: verwaltet die Daten

- **Protokoll**: DB-Zugriff (über JDBC)

Dreischichtenarchitektur

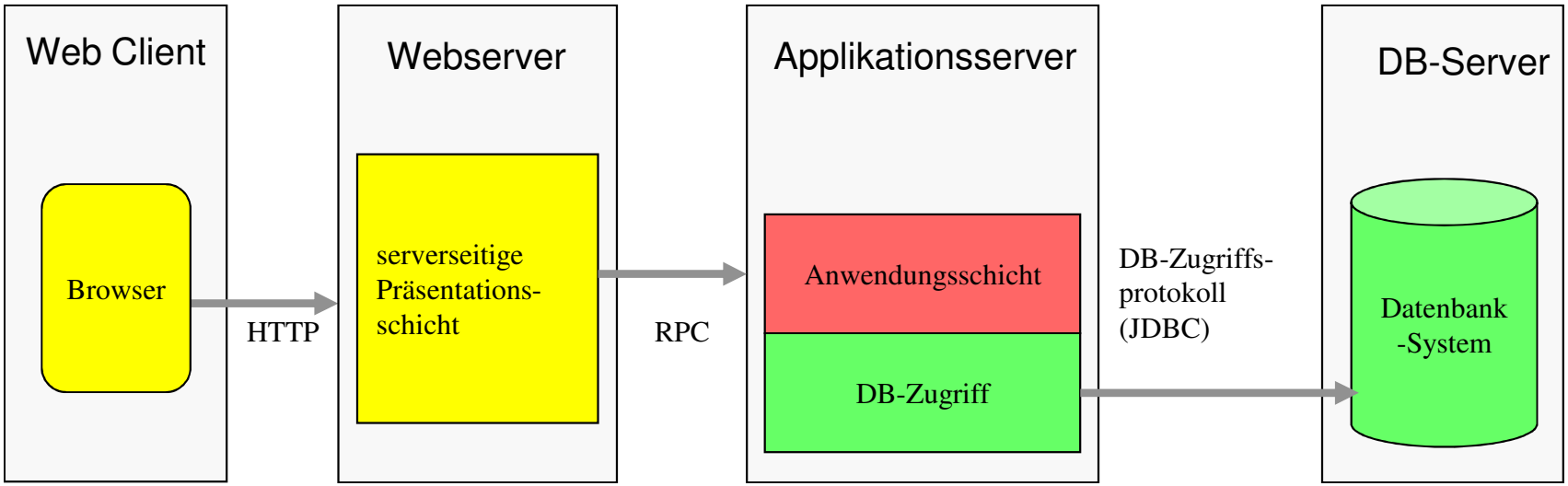


aus J.Dunkel, A. Eberhart, S. Fischer, C. Keiner, A. Koschel:
Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

- **Thin Client**: enthält nur die Präsentationsschicht (bspw. Java-Swing)
- **Applikationsserver**: enthält Geschäftslogik und DB-Zugriffsschicht
- **DB-Server**: verwaltet die Daten

- **Protokolle**: DB-Zugriff (über JDBC) und
  Zugriff auf die Anwendungsschicht über RPC (remote procedure call)

Mehrschichtenarchitektur



aus J.Dunkel, A. Eberhart, S. Fischer, C. Keiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008

- **Web-Client**: enthält nur Browser zum Anzeigen von Webseiten
- **Webserver**: serverseitige Präsentationsschicht zur dynamischen Erzeugung von Webseiten (JSPs, Servlets,...)
- **Applikationsserver**: enthält Geschäftslogik und DB-Zugriffsschicht
- **DB-Server**: verwaltet die Daten
- **Protokolle**: DB-Zugriff (über JDBC); Zugriff auf die Anwendungsschicht über RPC (remote procedure call) und HTTP

Zusammenfassung

- **Definition von Softwarearchitektur**
  - definiert Bauplan des Systems über zusammenwirkende Komponenten
  - fachliche Architektur: beschreibt die fachlichen Funktionalitäten
  - technische Architektur: beschreibt infrastrukturelle Basisdienste
- **Architektur-Beschreibung**
  - verschiedene Sichten und Detaillierungsgrade
- **Eigenschaften**
  - starke Kohäsion und lose Kopplung von Komponenten
- **Architekturmuster**
  - Schichtenarchitektur
  - die drei klassischen Softwareschichten
- **Physikalische Verteilung der Schichten**
  - Zwei-, Drei- und Mehrschichtenarchitektur

Literatur „Softwarearchitektur“

- Verwendete Quellen
  - [DEFKK08]: J.Dunkel, A. Eberhart, S. Fischer, C. Kleiner, A. Koschel: Systemarchitekturen für verteilte Anwendungen, Hanser, 2008
  - [CoHaKoTr05]: Conrad, S., Hasselbring, W., Koschel, A., Tritsch, R.: Enterprise Application Integration, Spektrum Wiss. Verlag, 2005
  - [GhKoRaSt17]: M. Gharbi, A. Koschel, A. Rausch, G. Starke Basiswissen für Softwarearchitekten, dpunkt.verlag, neueste Auflage.
  - [DunHol03]: J. Dunkel, A. Holitschke: Softwarearchitektur für die Praxis, Springer, 2003
  - [Bass03]: L. Bass, P. Clements, K. Bass: Software Architecture in Practice: 3rd Edition. Addison-Wesley, 2010

- Weitere Quellen (ggf. neuere Auflagen)
  - [ReuHas06]: R. Reussner, W. Hasselbring (Hrsg.): Handbuch der Software-Architektur, dpunkt.verlag, 2006
  - [VACIMNVZ05]: O. Vogel, I. Arnold, A. Chughtai, E. Ihler, U. Mehlig, Th. Neumann, M. Völter, U. Zdun: Software-Architektur – Grundlagen – Konzepte – Praxis, Spektrum, 2005
  - [So12]: I. Sommerville: Software Engineering, 9. Auflage, Pearson