
Enterprise Application Integration

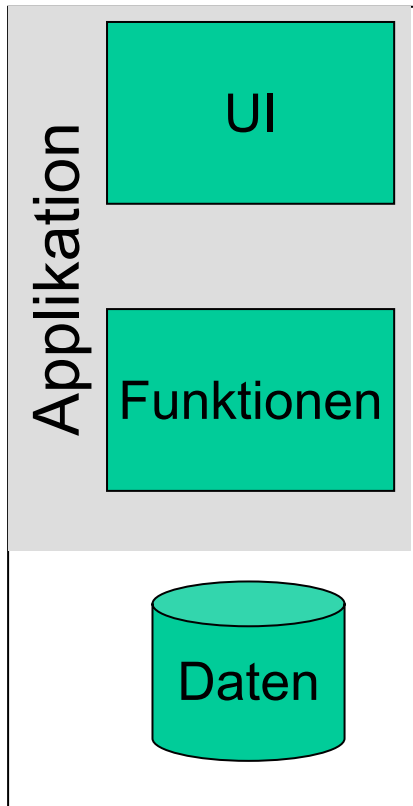
Ebenen der Integration

Prof. Dr. Christian Pape

Übersicht

- Aufbau mehrschichtiger Applikationen
 - Schichten
 - MVC
 - J2EE Entwurfsmuster (für Kommunikation Presentation Layer zu Business Layer)
- Ebenen der Integration
 - Benutzungsschnittstelle
 - Funktionsebene
 - Datenebene

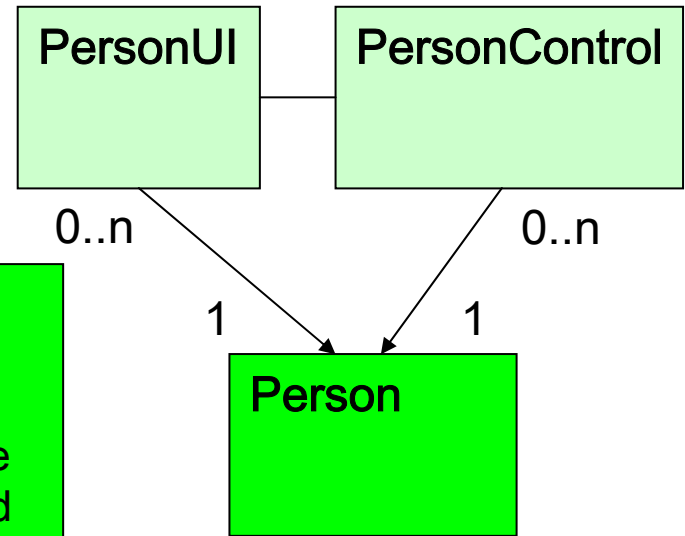
Aufbau mehrschichtiger Applikationen



User Interface / Presentation layer
Benutzerinteraktion, Dialogsteuerung, UI-abhängige Repräsentation von Daten („17.5.06“, „5/17/06“)

Geschäftsfunktionen / Business layer
Repräsentation von Daten unabhängig von deren Eingabe (java.util.Date), Funktionen und Workflow

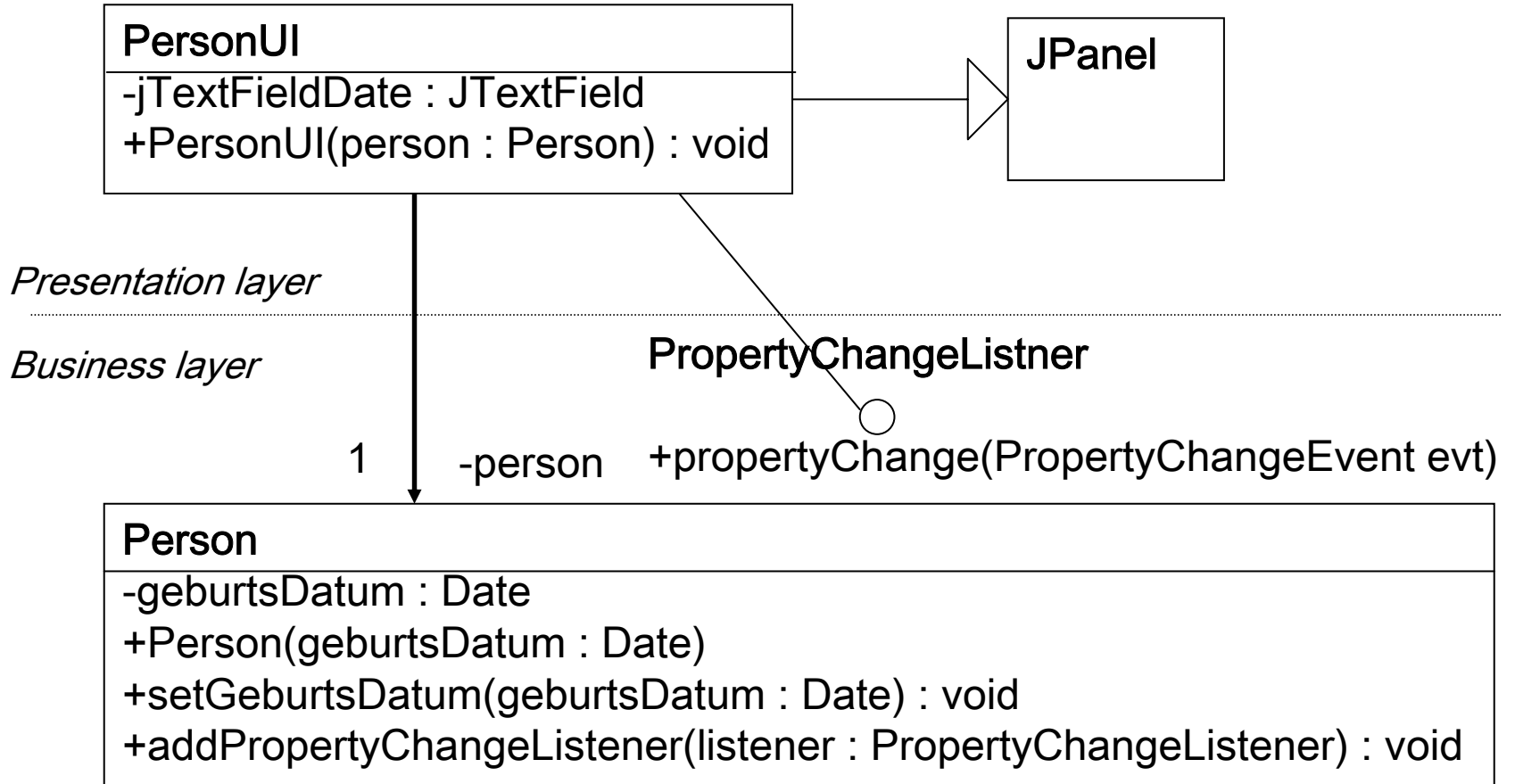
Datenhaltung
SQL Datenbanken
Datenintegrität unabhängig von Applikationen (date > 1.1.1900)



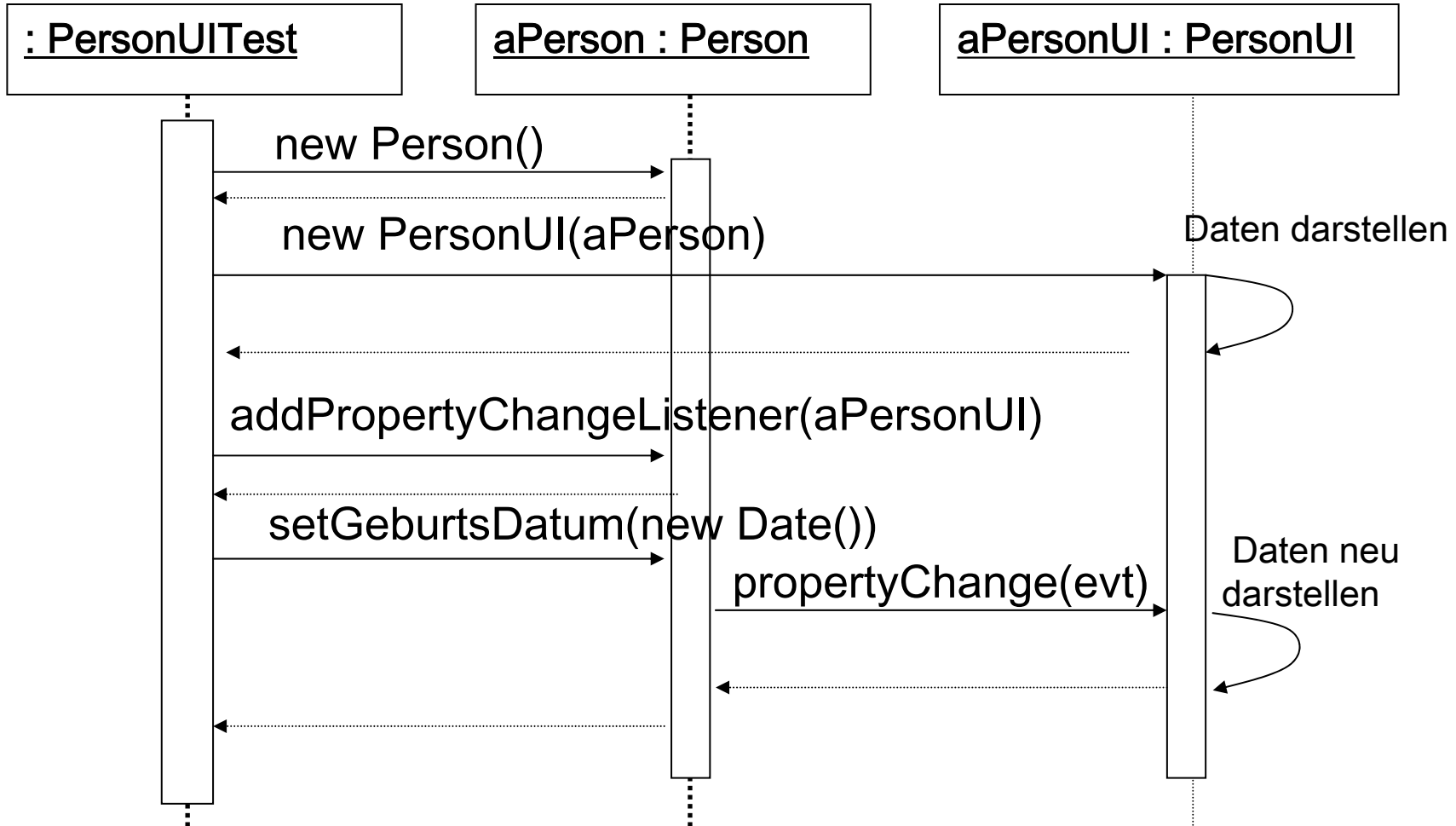
Model-View-Control (MVC)
Entwurfsmuster (Bspl. J2EE)

- View: Ein-/Ausgabe Daten (JSP)
- Control: Dialogsteuerung (Servlet)
- Model: Funktionen (EJB)

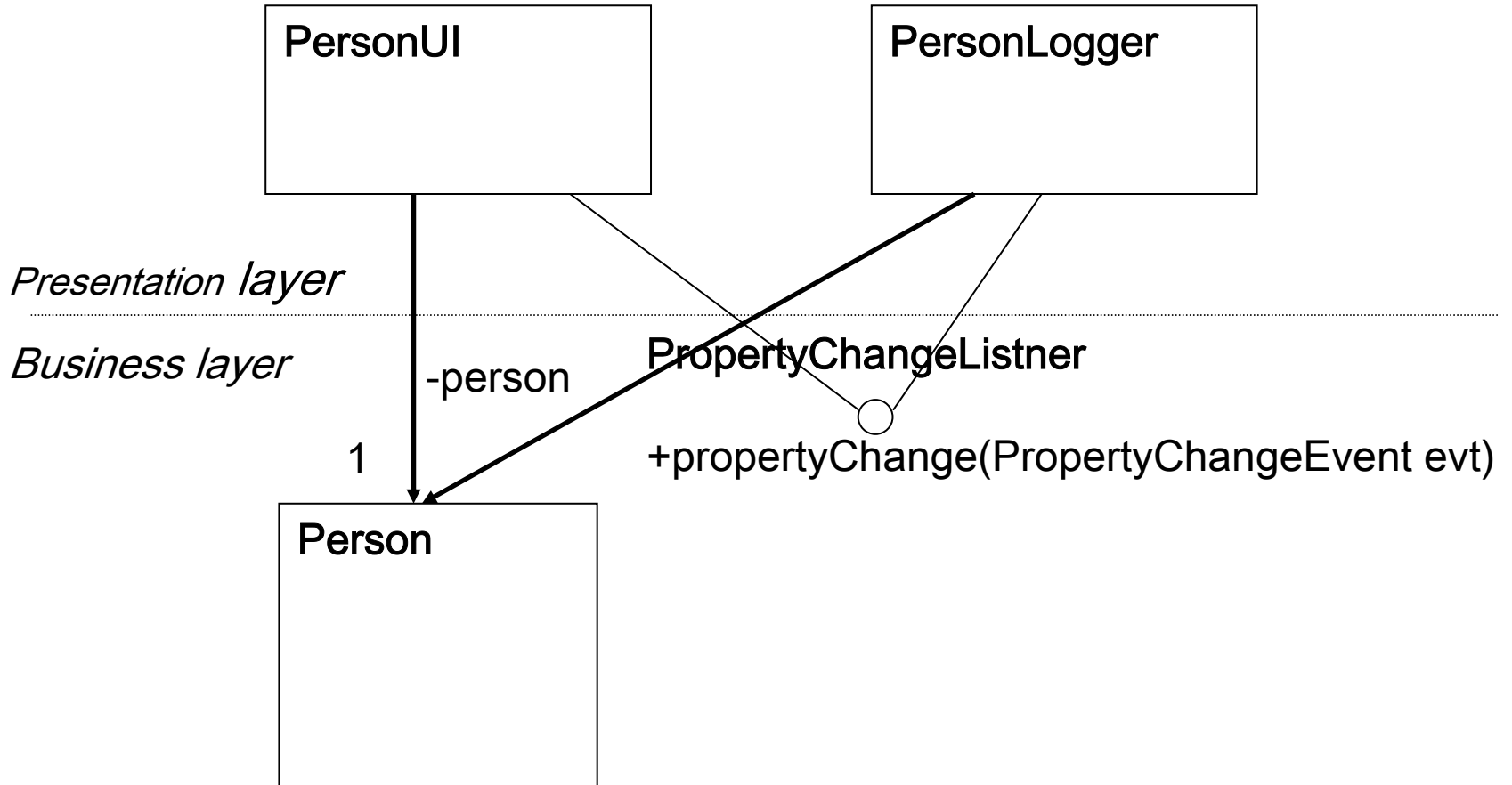
MVC / Java Swing



MVC / Java Swing



MVC / Java Swing



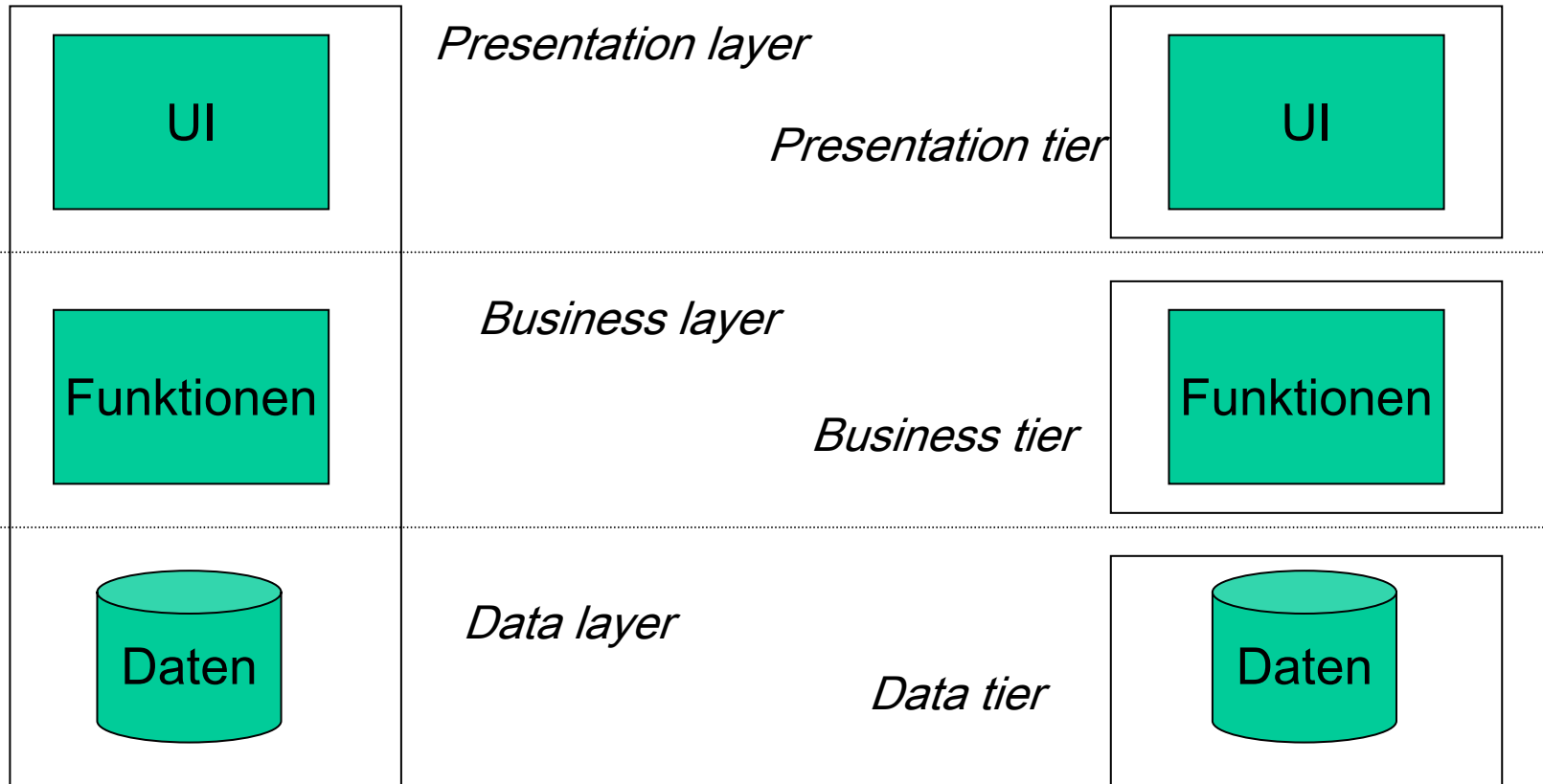
Übersicht

- Aufbau mehrschichtiger Applikationen
- Ebenen der Integration
 - Benutzungsschnittstelle
 - Funktionsebene
 - Datenebene

Ebenen der Integration

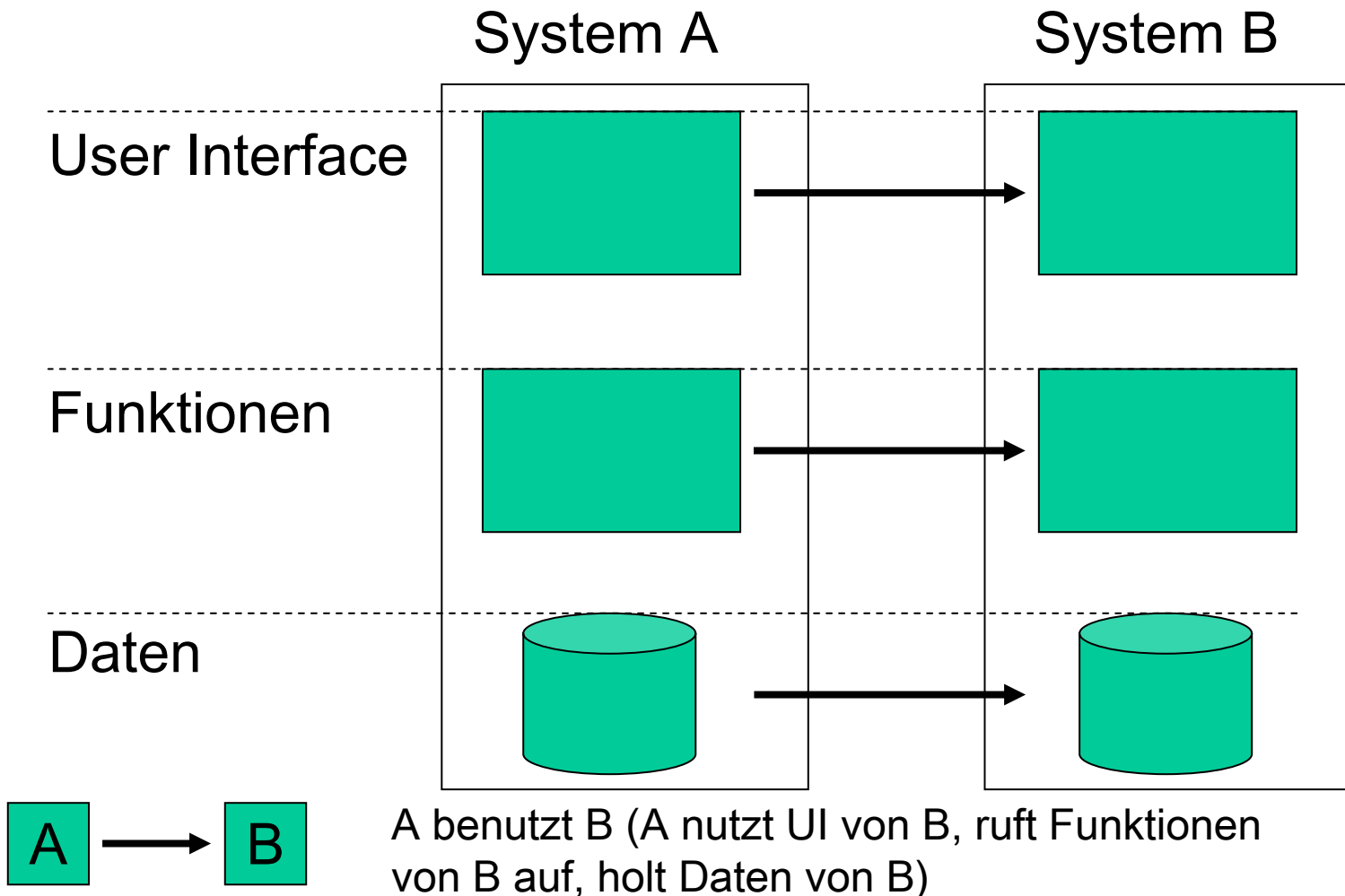
Layer: logische Schichten

Tier: physisch verteilbare Schichten

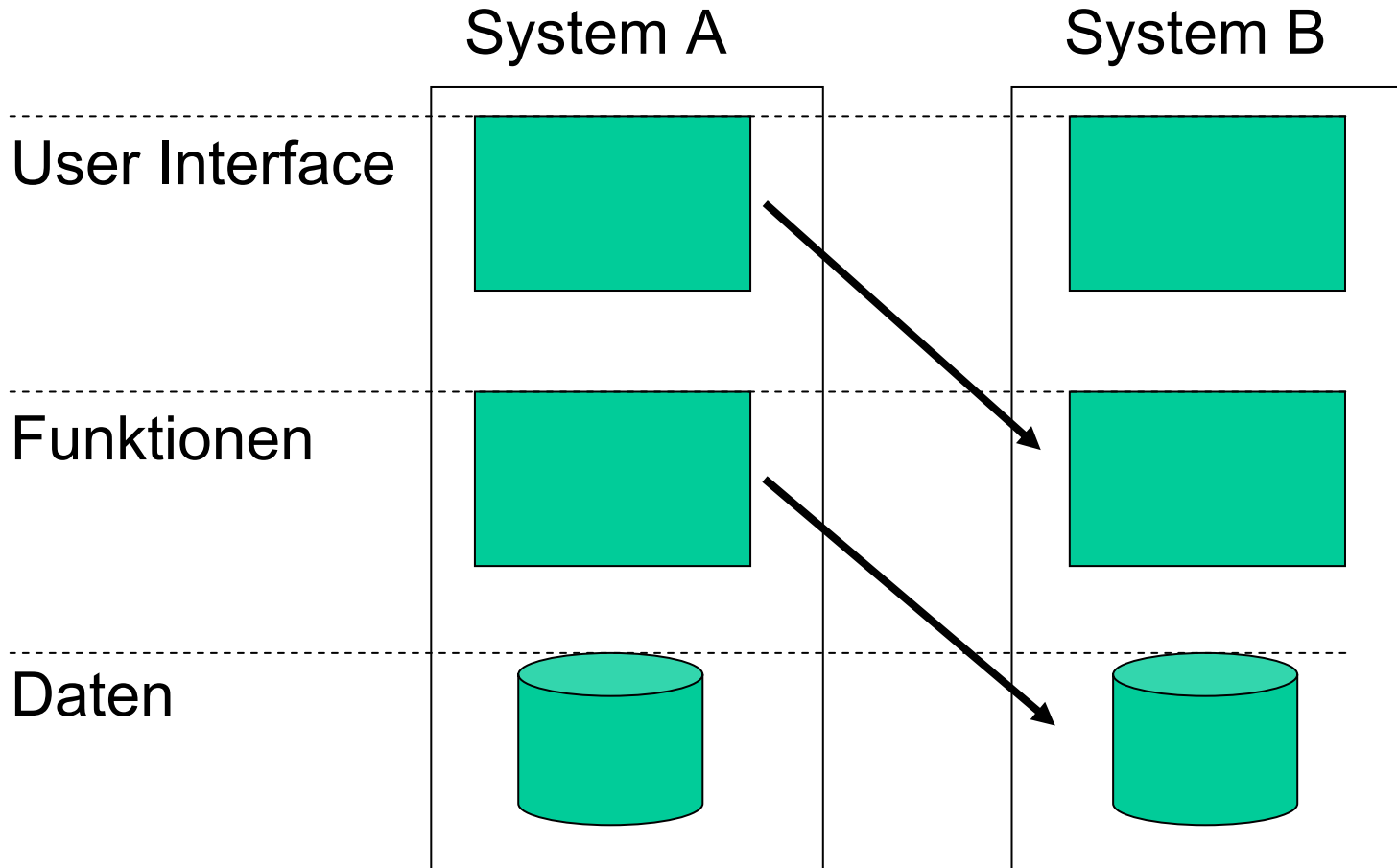


-
- Verschiedene Varianten zwischen A und B zu integrieren
 - Zwischen horizontalen Layer
 - Von oberen zu unteren Layer
 - Von unteren zu oberen Layer
 - Im Folgenden eine Integrationsrichtung
 - „A benutzt B“

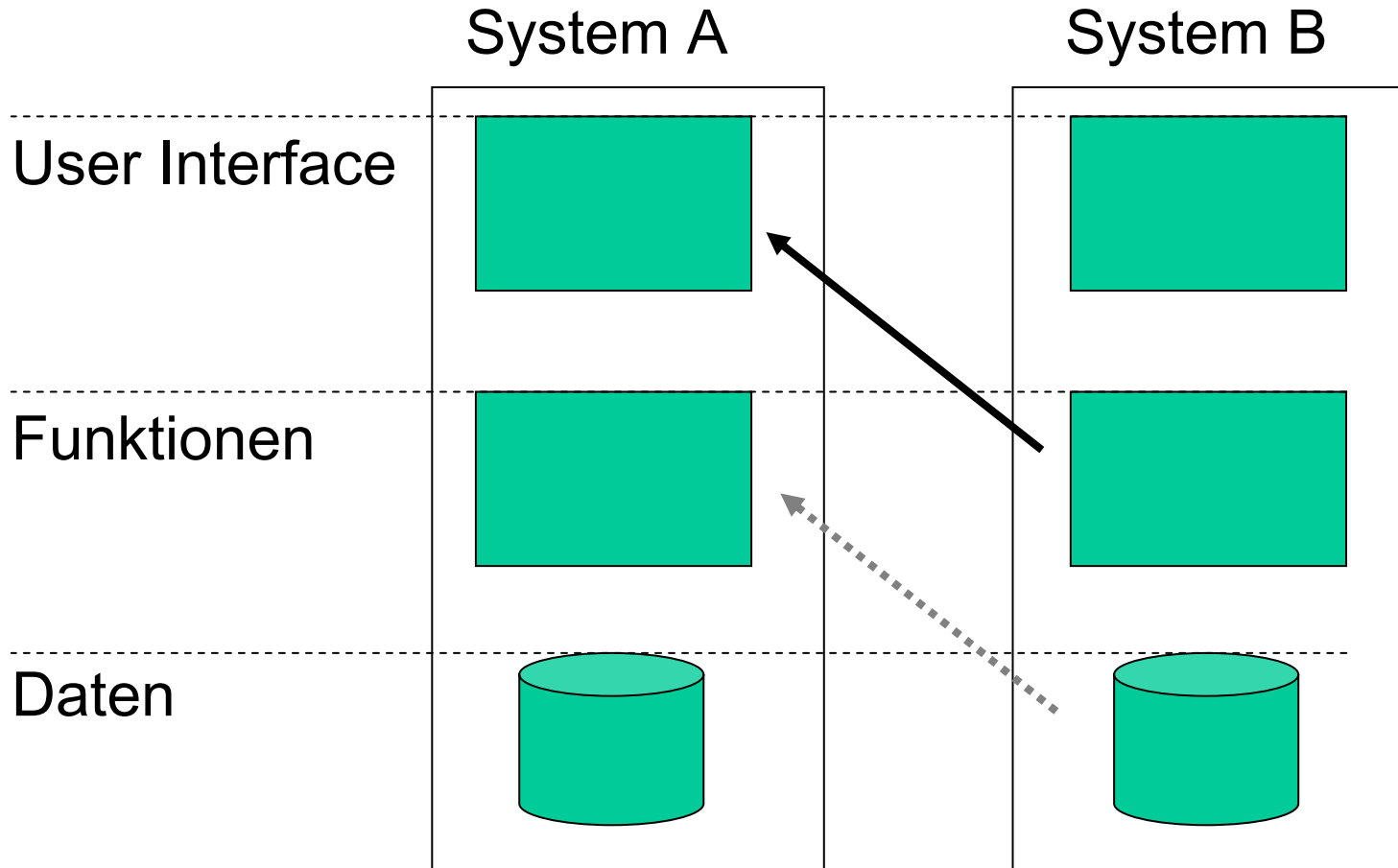
Ebenen der Integration / Horizontal



Ebenen der Integration / Vertikal A



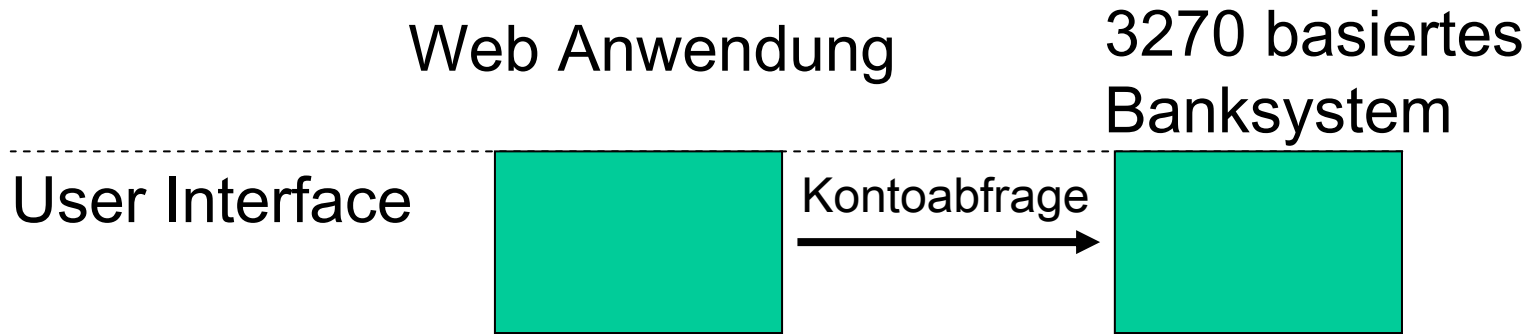
Ebenen der Integration / Vertikal B



Übersicht

- Aufbau mehrschichtiger Applikationen
- Ebenen der Integration: **Horizontal**
 - Benutzungsschnittstelle
 - Funktionsebene
 - Datenebene

Ebenen der Integration / Horizontal



- „Screen scraping“, „GUI-fizierung“
 - Benutzereingaben werden durch neues User-Interface simuliert
 - Ausgaben altes System werden als HTML durch neues System dargestellt
- Applet Lösungen: Applet ist eine 3270 Terminal Implementierung
- HTML Lösungen: Software setzte 3270 Bildschirm automatisch als HTML um
- Java API: Zugriff auf Felder des 3270 Bildschirm (80x24, Cursor)
- Produkte
 - IBM Host on demand
 - Open Connect
 - Viele andere

Ebenen der Integration / Horizontal

VT3270



The screenshot shows a VT3270 terminal window titled 'Session A'. The main application window displays a menu-driven interface for 'INTERROGATION DU FICHER ARTICLE' (Article File Inquiry) in the AGRSART/AGRSYS V5 system. The screen shows a list of articles with details for article 654123, including general information, short labels, and management codes. A 'Screen Light' dialog box is overlaid on the screen, providing a visual overview of the current screen's content and a set of function keys (F3-F21) for navigation and editing. The dialog box includes a 'Connexion' (Connection) status bar at the bottom left.

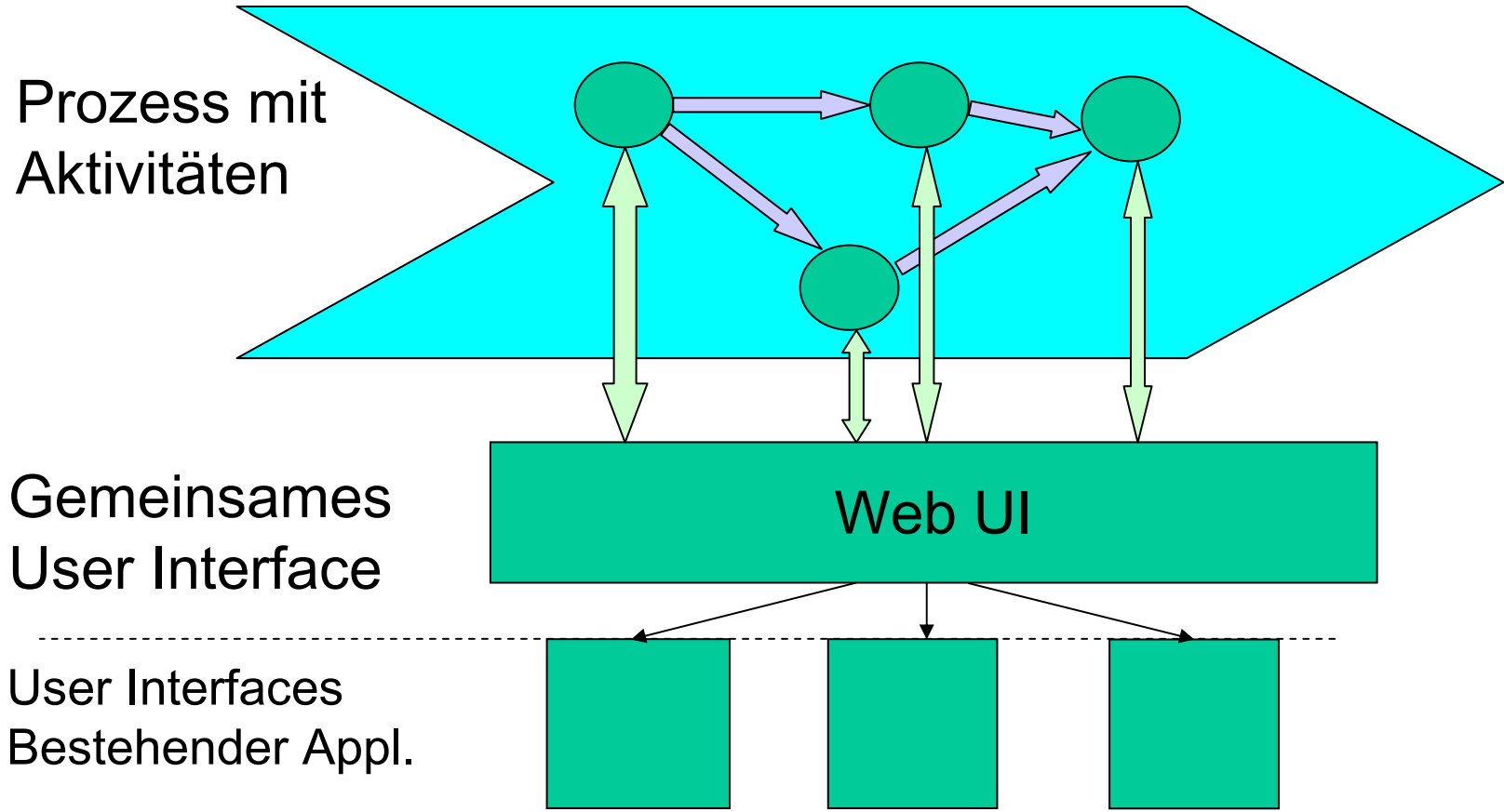
IBM HoD



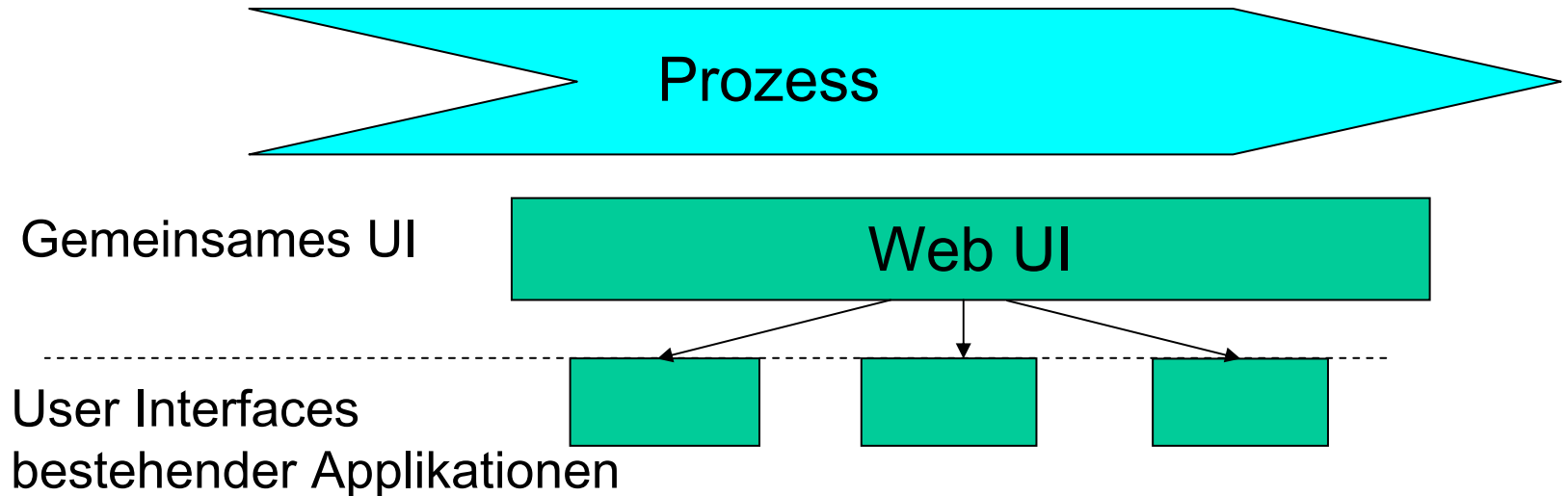
Ebenen der Integration / Screen scraping

Vorteile	Nachteile
<ul style="list-style-type: none">■ Vielzahl von Produkten für 3270 Host Anwendungen■ Relativ wenige Programmieraufwand■ Keine Änderungen an Host Anwendung■ Technische Neuerungen nutzbar (z.B. HTTPS)	<ul style="list-style-type: none">■ Herstellerabhängigkeit (Lizenzen teuer)■ Neue User Interfaces nah an 3270 Screens (Codes, Darstellung, Sprachwechsel)■ mehrerer 3270 Screens auf <i>ein</i> neues UI schwierig abzubilden■ Sicherheit (Benutzeranmeldung)■ Änderungen im UI der 3270 Anwendung haben immer Änderungen im integrierten System zur Folge

Ebenen der Integration / Horizontal



Ebenen der Integration / Horizontal



- Integration über HTML Frames (alles Webanwendungen)
 - Synchronisation Frames schwierig (z.B. via Java Script)
- Portalserver
 - Single sign login
 - Workflow
- Meist zusätzlich Integration auf Funktionsebene

Ebenen der Integration / Horizontal

The screenshot shows the Swisscom Fixnet website. The top navigation bar includes links for '+ Swisscom Gruppe', 'Online-Shop', 'Standorte', 'Offene Stellen', 'Seitenübersicht', and language options 'DE', 'FR', 'IT', 'EN', and 'Login'. A green arrow points to this bar. The main content area is titled 'Karriere' and features a banner for 'Ihre Chance: Offene Stellen bei Swisscom Fixnet AG'. Below this is a 'Jobs' section with a search form containing the following filters: 'Tätigkeitsbereich', 'Region', 'Organisationseinheit', 'Vollzeit / Teilzeit', 'Position', 'Aufschaltung', and 'Sprache'. A red box highlights this search form, and a red arrow points to it. At the bottom of the search form are buttons for 'Suchen', 'JobBasket*', 'JobBasket Login*', and 'JobAbo **'. Below the search form is a disclaimer: '*Mit der Login-Funktionalität werden die ausgewählten Inserate in einer Datenbank gespeichert und können beim nächsten Besuch auf der Website angesehen und bearbeitet werden. Mit der Funktionalität ohne Login stehen die Inserate nur während der aktuellen Browser-Session zur Verfügung.' and a checkbox: '**Möchten Sie regelmässig über neue offene Stellen per Mail informiert werden?'.

.Net /
W2K Server

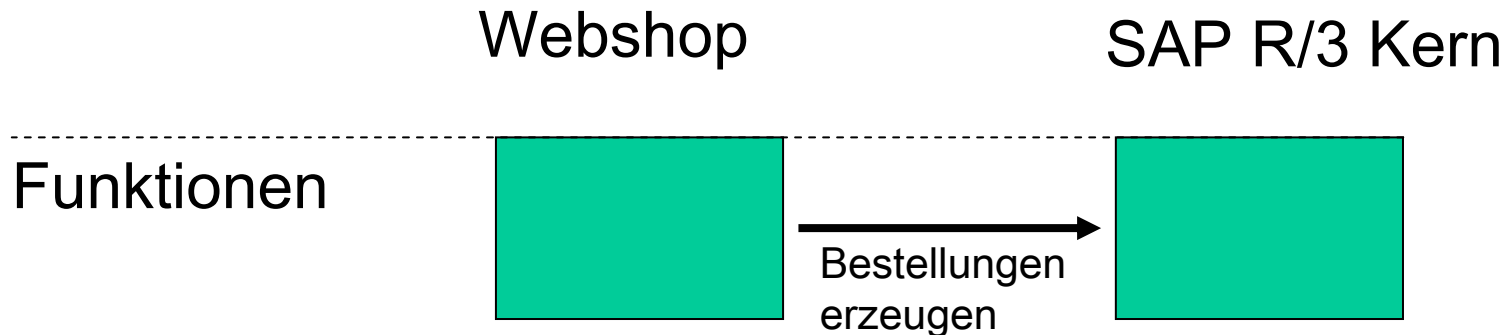
Einbindung
mit IFrame

J2EE /
Solaris

Ebenen der Integration / Gemeinsames UI

Vorteile	Nachteile
<ul style="list-style-type: none">■ Je nach Produkt Workflow konfigurierbar■ Nur eine Benutzeranmeldung■ Programmieraufwand moderat	<ul style="list-style-type: none">■ Herstellerabhängigkeit (Lizenzen teuer)■ Änderungen UI integrierte Appl. haben immer Änderungen im integrierten System zur Folge■ Nur bei Webanwendungen praktikabel

Ebenen der Integration / Horizontal



- Häufigste und wichtigste Integrationstyp
 - Technische Details: Verteilte Systeme
- Verschiedene Aufrufparadigmen
 - Remote Procedure Call (Web Services, DCE RPC)
 - Verteilte Objekte (CORBA, JRMI, COM+)
 - Message Queue meist mit XML Nachrichten
- Unterschiedliche Protokolle / Technologien
- Viele Standards / Produkte

Ebenen der Integration / Funktionsebene

Vorteile	Nachteile
<ul style="list-style-type: none">■ Teilweise Geringere Latenzzeiten (keine zusätzliche UI Schicht dazwischen)■ UI Änderungen im alten System wirken sich nicht auf neues System aus■ Transaktionelle Aufrufe auf Funktionsebenen, statt UI Ebene■ Neues UI frei gestaltbar■ Viele Standards und Technologien	<ul style="list-style-type: none">■ Meist Adapter für Geschäftslogik im neuen System nötig (Business Delegate)■ Überbrücken von Aufrufparadigmen (RPC zu OO, OO zu Nachrichten, ...)■ Überbrücken von Technologien (CORBA nach COM)■ Sauber definierte Schnittstelle nötig (ist auch ein Vorteil)■ Betriebszeiten Gesamtsystem nicht höher als geringste Verfügbarkeit jedes integr. Systems

J2EE Entwurfsmuster

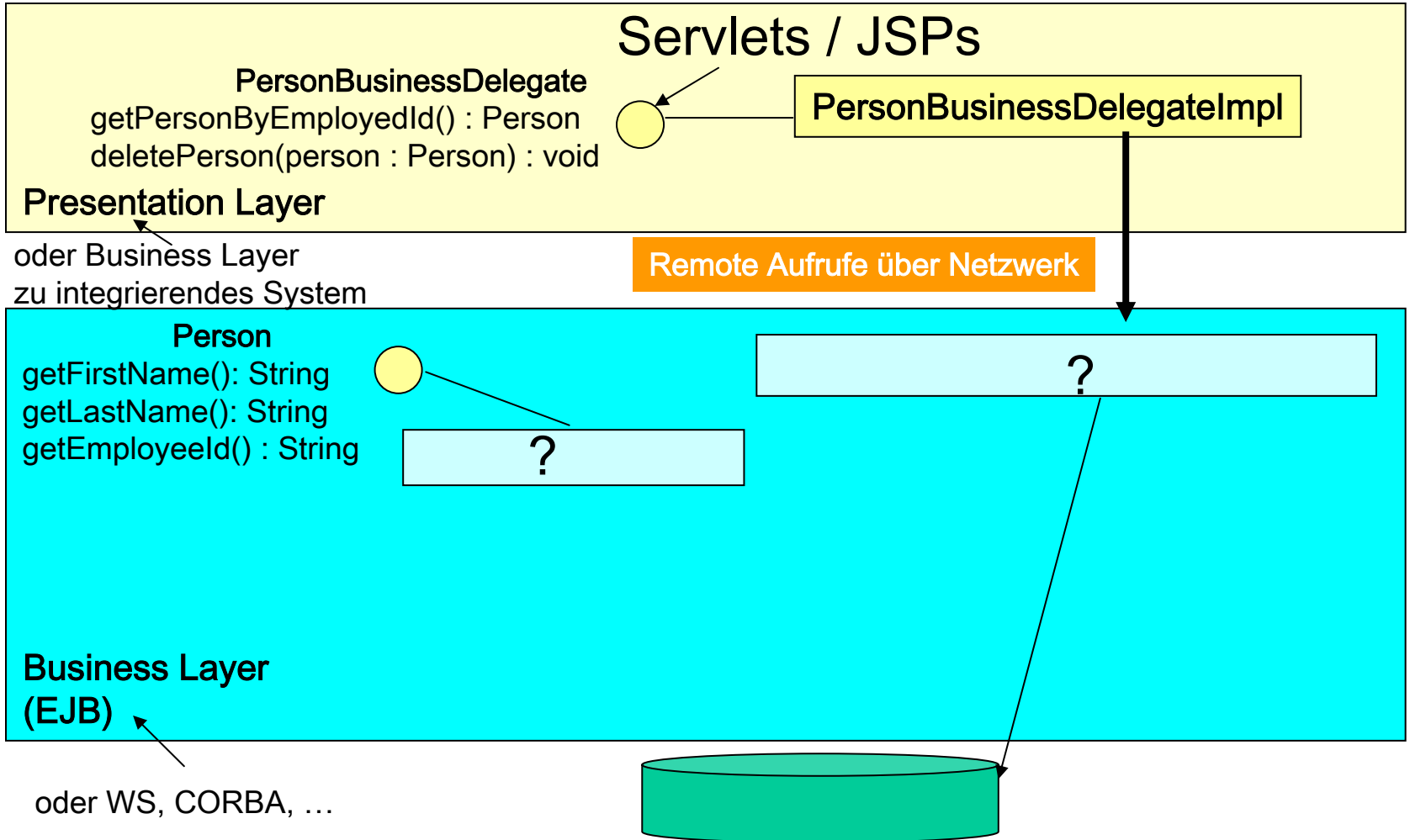
<http://java.sun.com/blueprints/patterns/catalog.html>

- Enterprise Java Beans (Auswahl von Mustern)
 - BusinessDelegate
 - Kapselt *Zugriff auf Business Layer* oder von anderen Systemen
 - Caching, Technische Exceptions fangen, Datenkonvertierung, ...
 - SessionFacade
 - Kapselt *Zugriff auf Entity Beans* (Persistence)
 - Entity Beans immer nur mit lokalen Schnittstellen impl.
 - Koordiniert Operationen der unteren Ebene (Entity Beans, ...)
 - Implementiert Funktionalität eines oder mehrere Anwendungsfälle
 - ServiceLocator
 - Kapselt Details für *Auffinden von Diensten* (URL, Portnummern, ...)
 - TransferObject
 - *Faßt komplexe Geschäftsdaten zusammen*, um Netzwerklast zu minimieren
 - TransferObject hat keine Geschäftslogik (nur getter/setter für Daten)

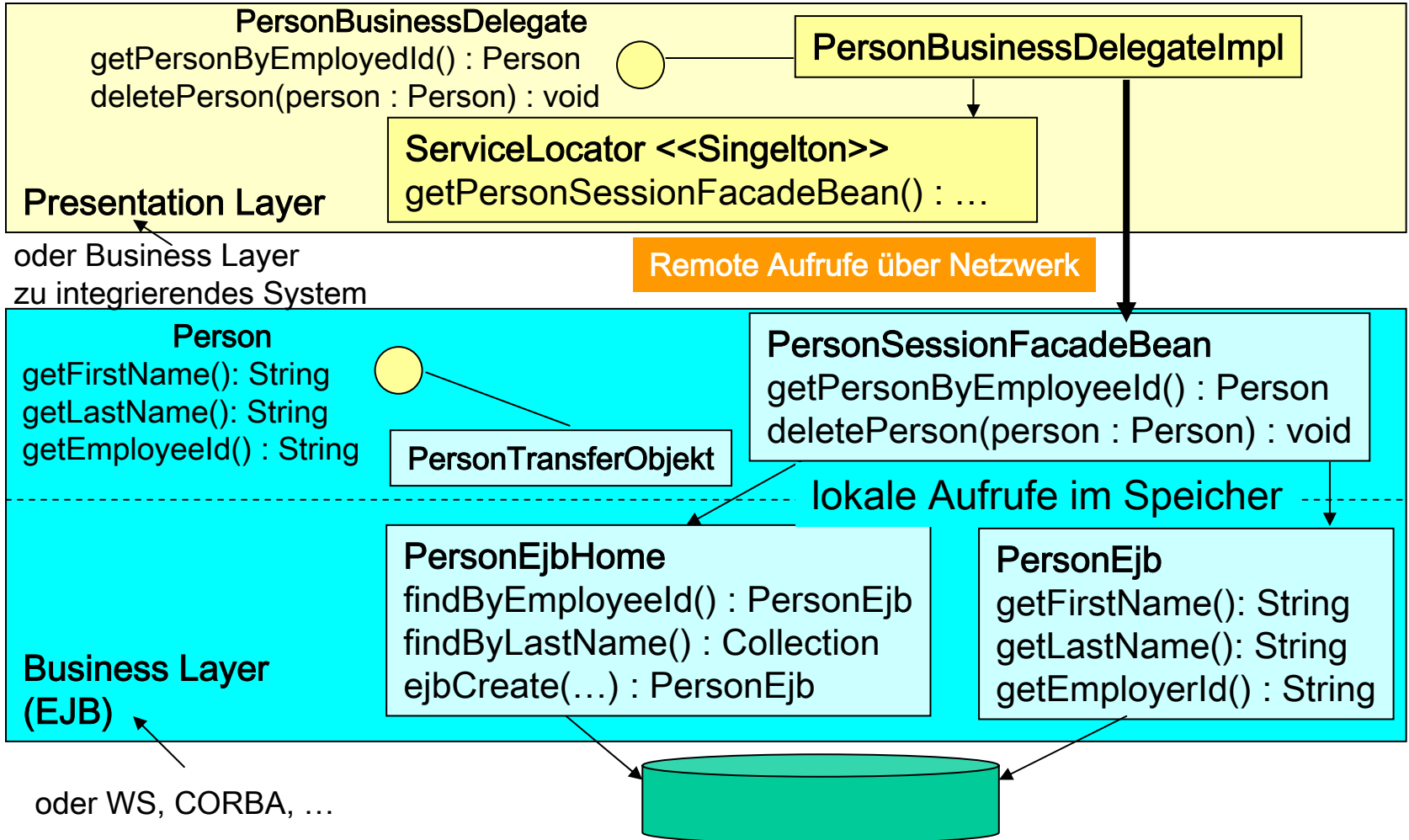
Person verwalten

- Personen
 - Person durch Personalnummer (Employee-ID) eindeutig identifiziert
 - Name, Vorname
- Funktionen
 - Person nach dieser Nummer suchen
 - Bestimmte Person Löschen
- 1. Schritt Entwurf Schnittstelle zum Business Layer (Java Interfaces für Business Delegate)
 - Funktionen
 - Daten
- 2. Schritt Implementierung mit einer bestimmten Technologie
 - EJB (Session Beans)
 - Alternativen: CORBA, Web Services, Java RMI
 - (stupid) Verwendung der J2EE Entwurfsmuster

Person verwalten



Person verwalten



Person verwalten

- Implementierung
 - Verwendung BusinessDelegate
 - Implementierung BusinessDelegate
 - Implementierung Session Bean
 - Implementierung Entity Beans

Person verwalten / Benutzung BusinessDelegate

(besser Factory Pattern verwenden)

```
PersonBusinessDelegate pbd = new PersonBusinessDelegateImpl();
Person person = null;
try {
    // Netzwerkaufruf oder Person ist im Cache des BusinessDelegate
    person = pbd.getPersonByEmployeeId("H876Z72");
} catch (BusinessException e) {
    // eigene Exception, kapselt RemoteException
    System.out.println("Person nicht gefunden. Ursache war "
        + e.getLocalizedMessage() );
}
```

- **Businessfunktionen entkoppelt von Servertechnologie**
 - Keine konkrete Implementierung sichtbar (nur Interfaces)
 - verwendeter Komponententechnologie verborgen
 - Datenformate für gesendete Daten (Objekte, XML, ...) verborgen

Person verwalten / Impl. Business Delegate

```
public class PersonBusinessDelegateImpl implements ... {
    private static PersonSessionFacadeBean psfb = null;

    public PersonBusinessDelegate() {
        psfb = ServiceLocator.getSingleton()
            .getPersonSessionFacadeBean();
        // fehlt: Exceptions behandeln
    }

    public Person getPersonByEmployerId(String employeeId)
        throws BusinessException {
        Person person = null;
        try {
            person = psfb.getPersonByEmployeeId(employeeId);
        } catch (Exception e) {
            throw new BusinessException(e);
        }
        return person;
    }
}
```

Person verwalten / Impl. Session Fassade

```
public class PersonSessionFacadeBean extends SessionBean {  
  
    private static PersonEjbHome personHome = null;  
  
    // Initialisierungen etc.  
    // remote interface  
    public Person getPersonByEmployerId(String employeeId)  
        throws RemoteException, FinderException {  
        PersonEjb personEjb = null;  
        Person person = null;  
  
        try {  
            personEjb = personHome.findByEmpoyeeId(employeeId);  
            person = personEjb.getPerson(); // Transfer Object erz.  
        } catch ( ... ) { ... }  
  
        return person;  
    }  
}
```

Person verwalten / Impl. Entity Bean

```
// nur lokale Interfaces, keine Remote Interfaces
public class PersonEjb extends EntityBean {
    //ejbCreate, ejbDelete, usw.
    // Annahme CMP, Accessor Methoden vom EJB Container impl.
    public abstract getFirstName();
    ...

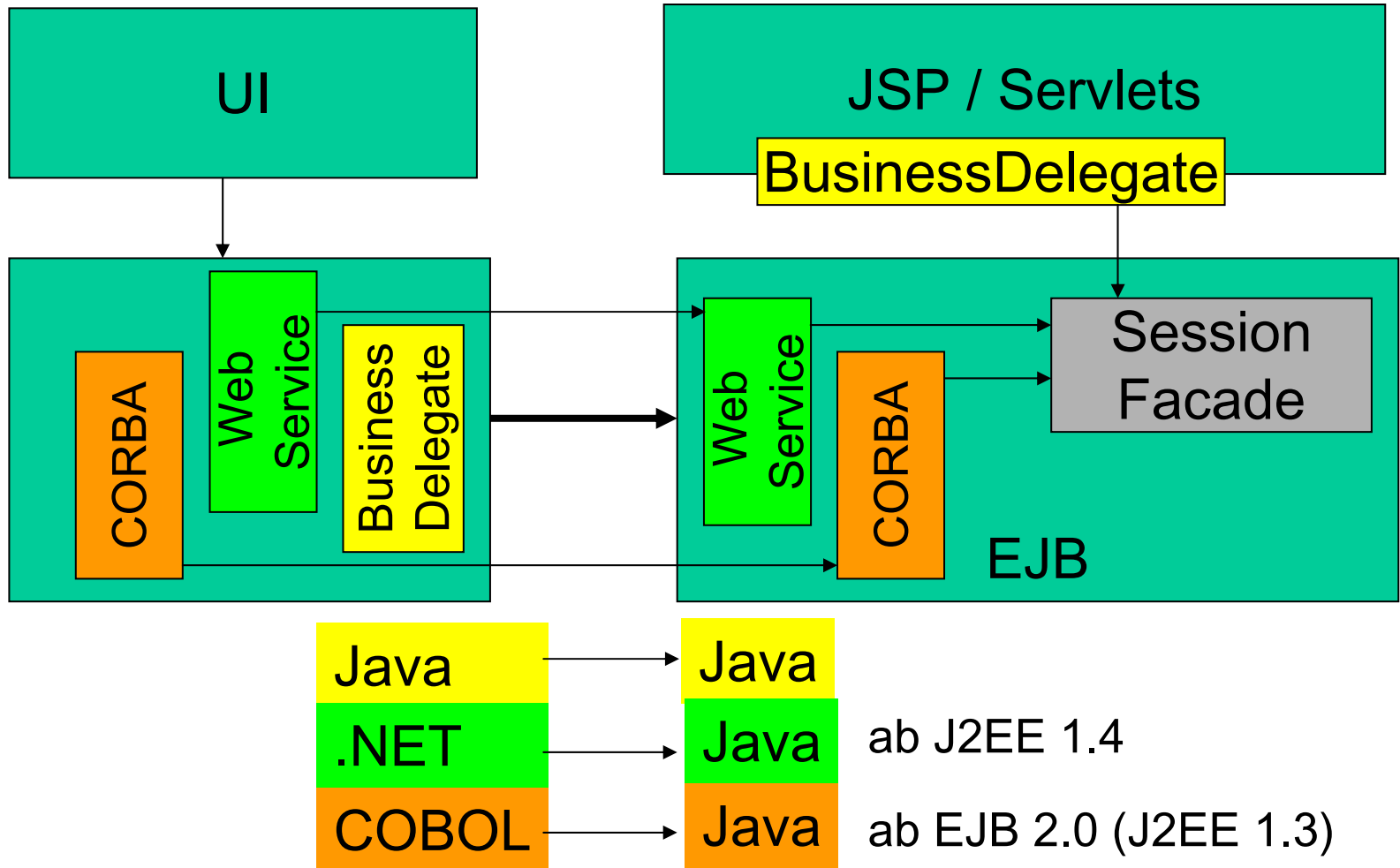
    public Person getPerson() {
        return new PersonTransferObjekt(
            getFirstName(),
            getLastName(),
            getEmployeeId() );
    }

    public void setPerson(Person person) {
        setFirstName(person.getFirstName());
        ...
    }
}
```

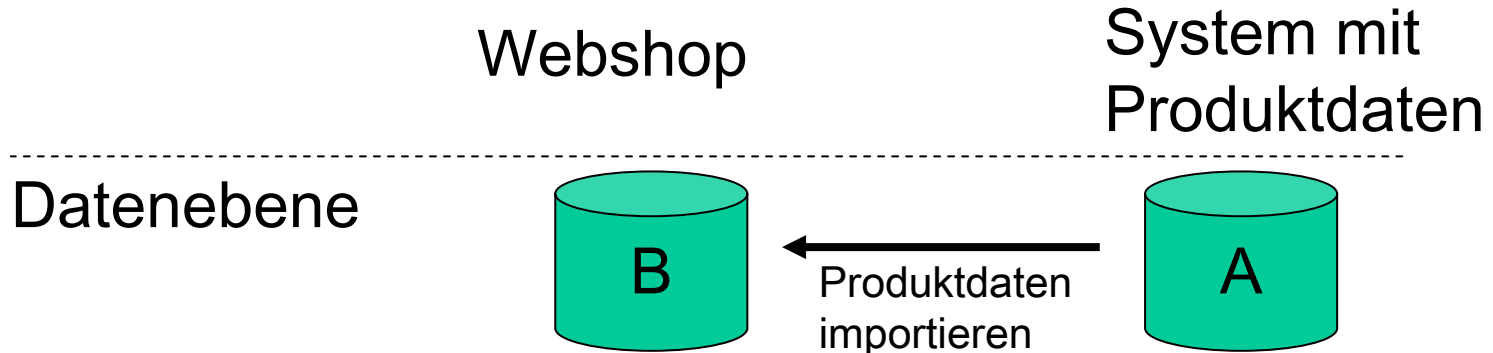
J2EE Entwurfsmuster

- Vorteile für Integration (mit Java)
- EJB 2.0 (J2EE 1.3)
 - CORBA IDL aus EJB erzeugbar
 - Session Facade dafür nutzen
- J2EE 1.4
 - Web Services aus (zustandslosem) Session Bean erzeugbar (EJB Container)
 - Session Facade direkt dafür nutzbar
 - Web Services aus Interface und Implementierung erzeugbar (Servlet Container)
 - Interface und Impl. Business Delegate nutzbar
- Einfache Verwendung von J2EE
Geschäftskomponenten für andere Technologien
 - .NET, C++, C, ...
 - Im günstigen Fall übers server-seitig Konfiguration

J2EE Entwurfsmuster - Integration



Ebenen der Integration / Horizontal



- Oft vorkommender Integrationstyp
- Datenreplikation
- CSV (comma separated values) Dateien
- SQL Skripte schreiben für:
 - zu exportieren Daten aus A (unload)
 - und zu importierende Daten in B (load)
- Periodische Imports: täglich, wöchentlich, monatlich
- Transport Daten z.B. via FTP

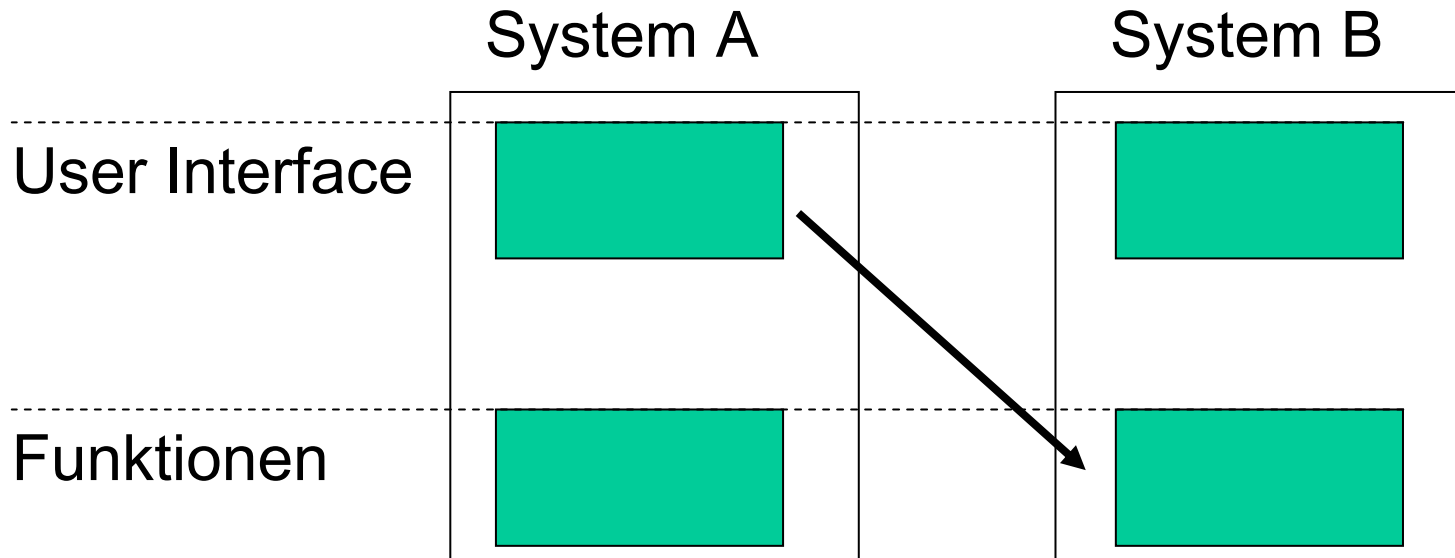
Ebenen der Integration / Datenebene

Vorteile	Nachteile
<ul style="list-style-type: none">▪ Freie Gestaltung des neuen Systems: geringe Latenzzeiten, hohe Leistung möglich▪ Neues System weitgehend von altem System entkoppelt: höhere Verfügbarkeit möglich▪ Import aus unterschiedlichen Quellen möglich	<ul style="list-style-type: none">▪ Nur für unveränderliche Daten sinnvoll (Stammdaten)▪ Datenqualität Quellsystem muss hoch sein▪ Unterschiedlicher Entwurf beider DB führt zu Problemen▪ Bei hohen Datenmengen Zielsystem für Zeitraum außer Betrieb▪ Nachbau von Geschäftsfunktionen des Quellsystems notwendig▪ Umwandlung Datentypen und Werte (Datum, Codes, ...)

Übersicht

- Aufbau mehrschichtiger Applikationen
- Ebenen der Integration: **Vertikal A**
 - Benutzungsschnittstelle
 - Funktionsebene
 - Datenebene

Ebenen der Integration / Vertikal A

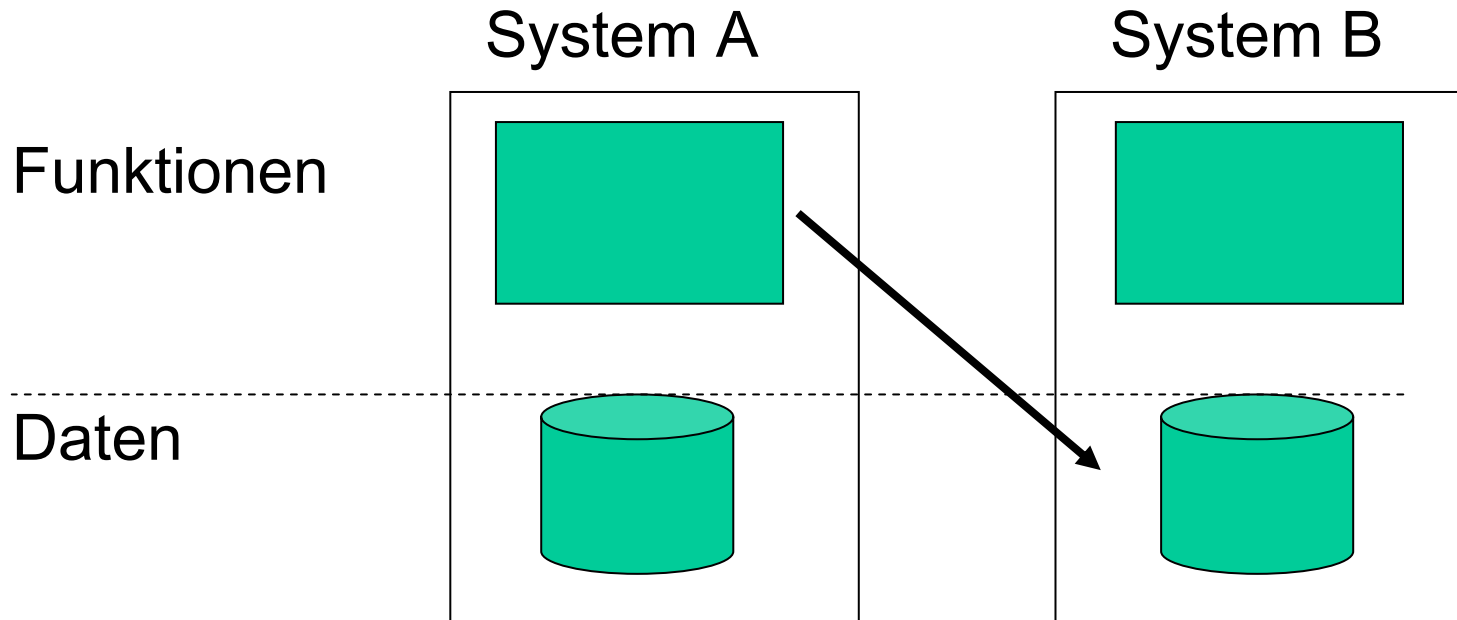


- Verteilte Objekte / Komponenten
- Meist nur möglich, falls Funktionen direkt aufrufbar
 - A und B im gleichen Adressraum, direkter Funktionsaufruf im gleichen Speicher
 - A und B auf unterschiedlichen Rechner: nur bei gleicher Technologie
- Falls nicht direkt aufrufbar: Horizontale Integration
 - Funktion zu Funktion

Ebenen der Integration / Datenebene

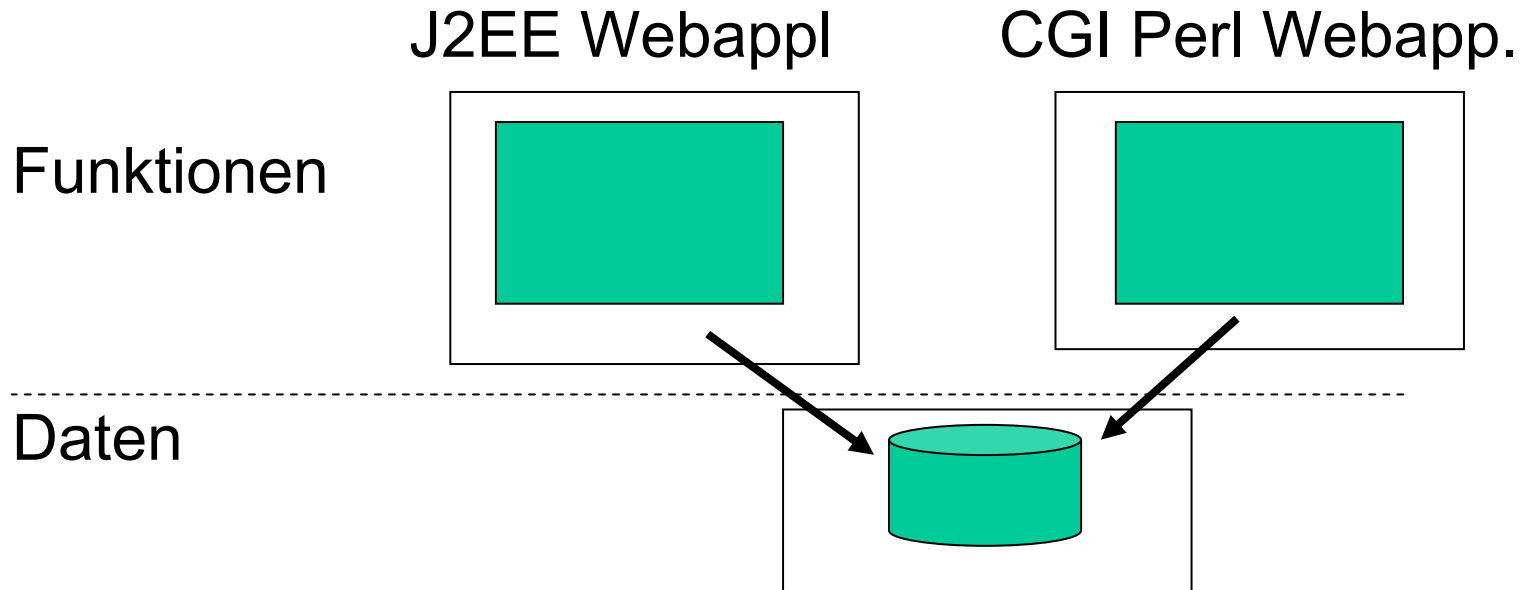
Vorteile	Nachteile
<ul style="list-style-type: none">■ Geringe Latenzzeiten■ Moderater Programmieraufwand	<ul style="list-style-type: none">■ Änderungen Funktionen haben direkte Änderungen im neuen UI zur Folge (starke Kopplung zwischen neuem UI und bestehenden Funktionen)■ Funktionen und Rückgabewerte beeinflussen neues UI (keine Adaption durch z.B. Business Delegate)

Ebenen der Integration / Vertikal A



1. Eine gemeinsam genutzte Datenbank
2. Verschiedene Datenbanken
3. Föderierte Datenbanken

Ebenen der Integration / Vertikal A

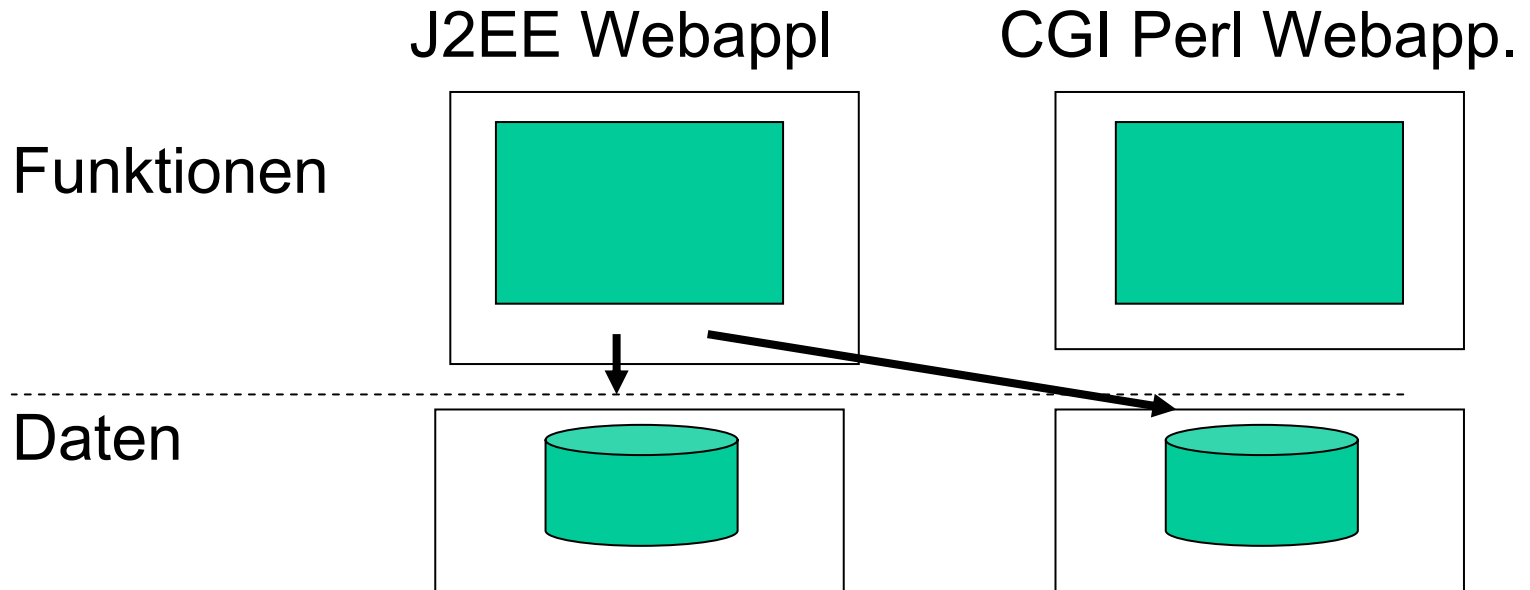


1. Eine gemeinsame genutzte Datenbank
 - Pro System eigenes Datenbankschema
 - Gemeinsam genutzte Relationen in einem Schema (Stammdaten wie Länder, Ortsnamen)

Ebenen der Integration / Vertikal A

Vorteile	Nachteile
<ul style="list-style-type: none">■ Teure Resource DB mehrfach genutzt■ Applikationen können Daten direkt austauschen und manipulieren■ Gemeinsame Daten immer aktuell	<ul style="list-style-type: none">■ Datenkonsistenz und –integrität ggf. schlecht, falls nicht vollständig in DB gelöst■ Änderungen gemeinsam genutzter Relationen haben Änderungen in vielen Applikationen zur Folge (starke Kopplung der Systeme)■ Redundante Geschäftslogik

Ebenen der Integration / Vertikal A

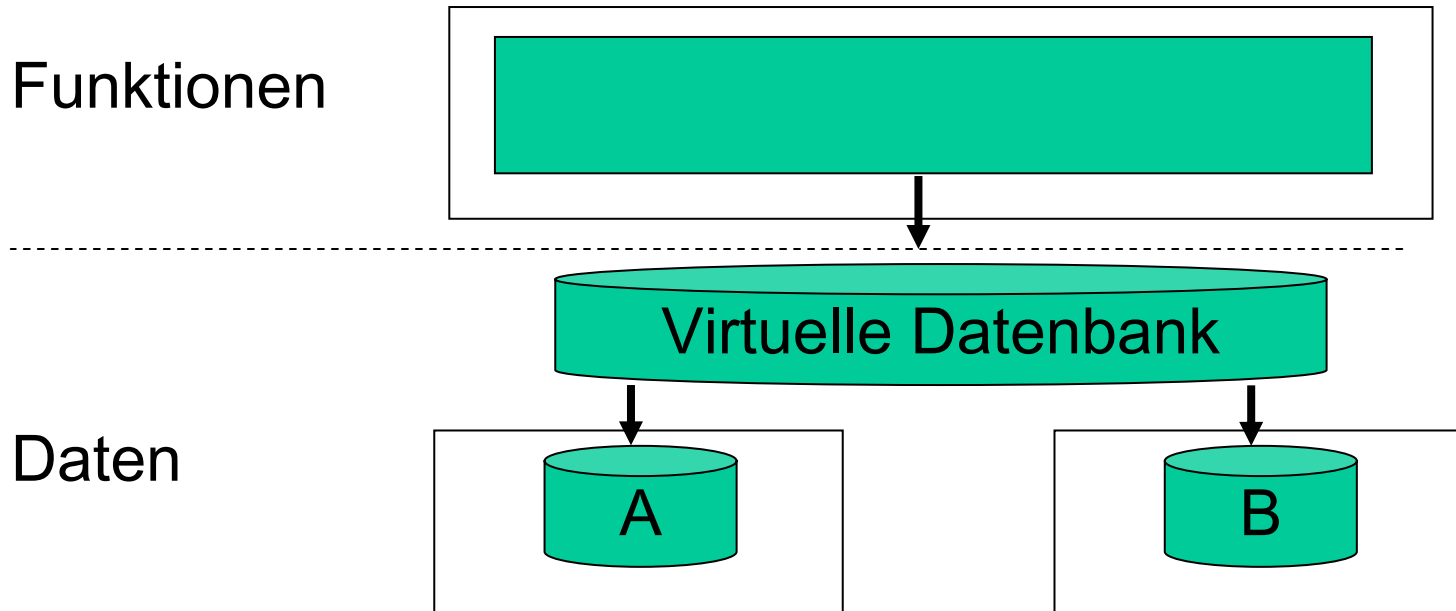


2. Verschiedene Datenbanken
- Auf verschiedenen Rechnern
 - Verschiedene Produkte (Oracle, DB2, MS SQL Server, ...)
 - Verschiedene Produktversionen
 - Verschiedene Entwurfsrichtlinien

Ebenen der Integration / Vertikal A

Vorteile	Nachteile
<ul style="list-style-type: none">▪ Applikationen können Daten direkt austauschen und manipulieren▪ Gemeinsame Daten immer aktuell	<ul style="list-style-type: none">▪ Redundante Geschäftslogik▪ DB übergreifende Transaktionen schwierig (2-phase commit)▪ Zusätzliche Kosten durch weitere DB▪ Ggf. unterschiedliche Treiber / Know-how für verschiedene DB Produkte

Ebenen der Integration / Vertikal A

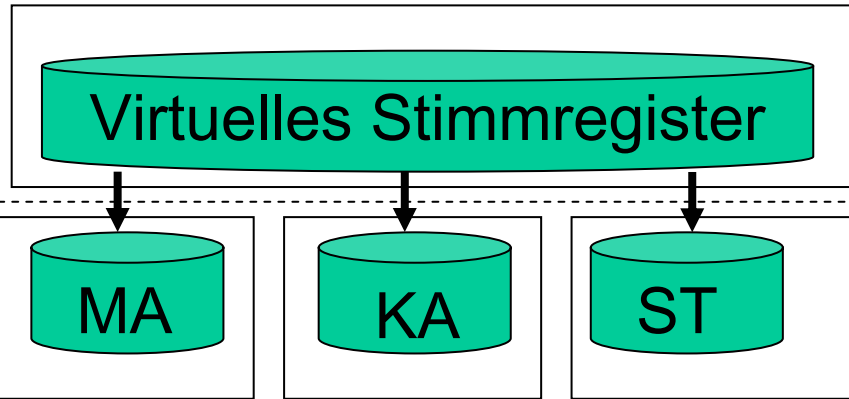


3. Konföderierte Datenbanken

- Ein Schema in virtueller Datenbank
- SQL Abfragen werden auf DBen A und B umgesetzt
- Produkte von IBM, Oracle

Ebenen der Integration / Vertikal A

Date für Wahlsystem
(Kanton / Länder)



- Gemeinden (DE, CH, ..) führt Stimmregistern
 - Enthält Wahlberechtigte Personen
 - Excel, MS Access Datei, Datenbank, unterschiedliche Systeme, verwoben mit Melderegister
- Land / Bund benötigt Daten für Wahlen
 - Keine Vorgaben an Datenbanken Gemeinden möglich (Gemeindeautonomie)

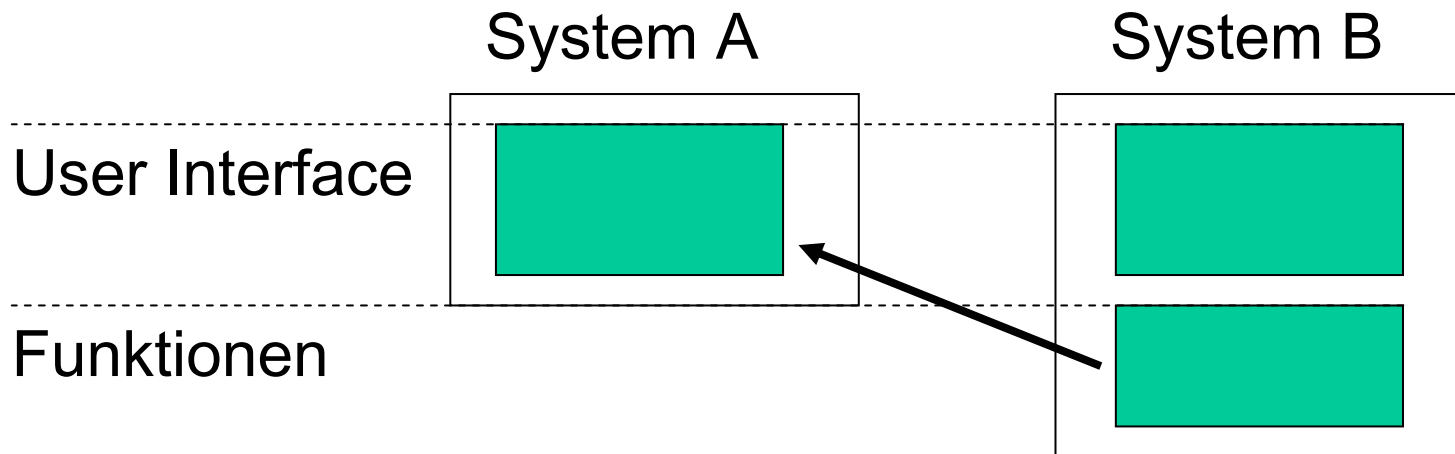
Ebenen der Integration / Vertikal A

Vorteile	Nachteile
<ul style="list-style-type: none">▪ Daten an einem Ort, keine Replikation nötig (theoretisch)	<ul style="list-style-type: none">▪ Keine Standards, einige Hersteller▪ Zusammenführen unterschiedlichste DB Schemata▪ Starke Kopplung an Schemata Quell-DB▪ Sicherheit / Netzwerk▪ Schreibzugriffe problematisch

Übersicht

- Aufbau mehrschichtiger Applikationen
- Ebenen der Integration: **Vertikal B**
 - Benutzungsschnittstelle
 - Funktionsebene
 - Datenebene

Ebenen der Integration / Vertikal B



- Zugriff auf Funktionen von A nicht möglich
- Funktionen werden von Serversystem B benötigt (evt. ohne Benutzerinteraktion)
- Z.B. System B soll Bestellungen von System A automatisch prüfen, überwachen, weiterverarbeiten, ...
- Funktionsaufruf muss Benutzerdialog simulieren
 - „Screen scraping“ nötig (TN3270, HTTP/HTML, Bildschirminhalt als Nachricht für Message Queue)
 - Konvertieren Bildschirmdaten in Geschäftsdaten

Ebenen der Integration / Vertikal A

Vorteile	Nachteile
<ul style="list-style-type: none">Keine (oder nur geringe) Änderungen an bestehender Anwendung	<ul style="list-style-type: none">wie bei horizontaler Integration UIHohe Latenzzeiten (fünf bis sechs Schichten beteiligt)Hoher Programmieraufwand

Zusammenfassung

■ Integration von Systemen

- Hängt vom Schichtenaufbau ab
- Verschiedenste Vor-/Nachteile (nicht erschöpfend)
- Wichtigsten Integrationsmethoden
 - Horizontal auf Funktionsebene
 - Horizontal auf Datenebene
 - Vertikal gemeinsame Datenbank
- Alle anderen nur in „Notfällen“

wünschenswert
↓
weniger
wünschenswert

■ Fokus Vorlesung

- Horizontal Funktionsebene
- Nachrichten-orientiert (MQ, XML, Entwurfsmuster)
- Etwas Komponenten-orientiert (CORBA, Entwurfsmuster)