Informationsintegration und mobile Web-Anwendungen

Übung bzw. „Der rote Faden“
Dr. Asarnusch Rashid

Klick-And-Bau

Informationsintegration und Webportale

Nr	Dozent	Thema
1	Rashid	Einführung
2	Walter	Web-Technologien im Überblick
3	Mülle	Enterprise Application Integration
4	Rivera-Pelayo	Mobile WebApps: Frameworks und Architekturen
5	Braun	Mehrschichtenarchitekturen und EJB
6	Walter	Datenbankabstraktion und Hibernate
7	Braun	Informationsintegration: Architekturen und Systeme
8	Walter	Dienstorientierte Integration und Web Services (+Mashups)
9	Zander	Semantische Integration: Grundlagen, RDF & OWL
10	Rashid	Übung
11	Stühmer, Riemer	Ereignisgesteuerte Architekturen und Echtzeitdatenintegration
12	Rashid	Kontextmanagement
13	Happel (audriga)	Praxisbeitrag
14	von Stackelberg	Forschungsbeitrag: Crowd-basierte Meinungssammlung
15	Rashid	Abschlussveranstaltung

Ziele der Vorlesung

Informationsintegration und mobile Web-Anwendungen

- **Resultierende Hauptprobleme**
  - Technische Heterogenität
  - Inhaltliche Heterogenität
  - Skalierbarkeit
  - Verteilung
  - Wiederverwendung

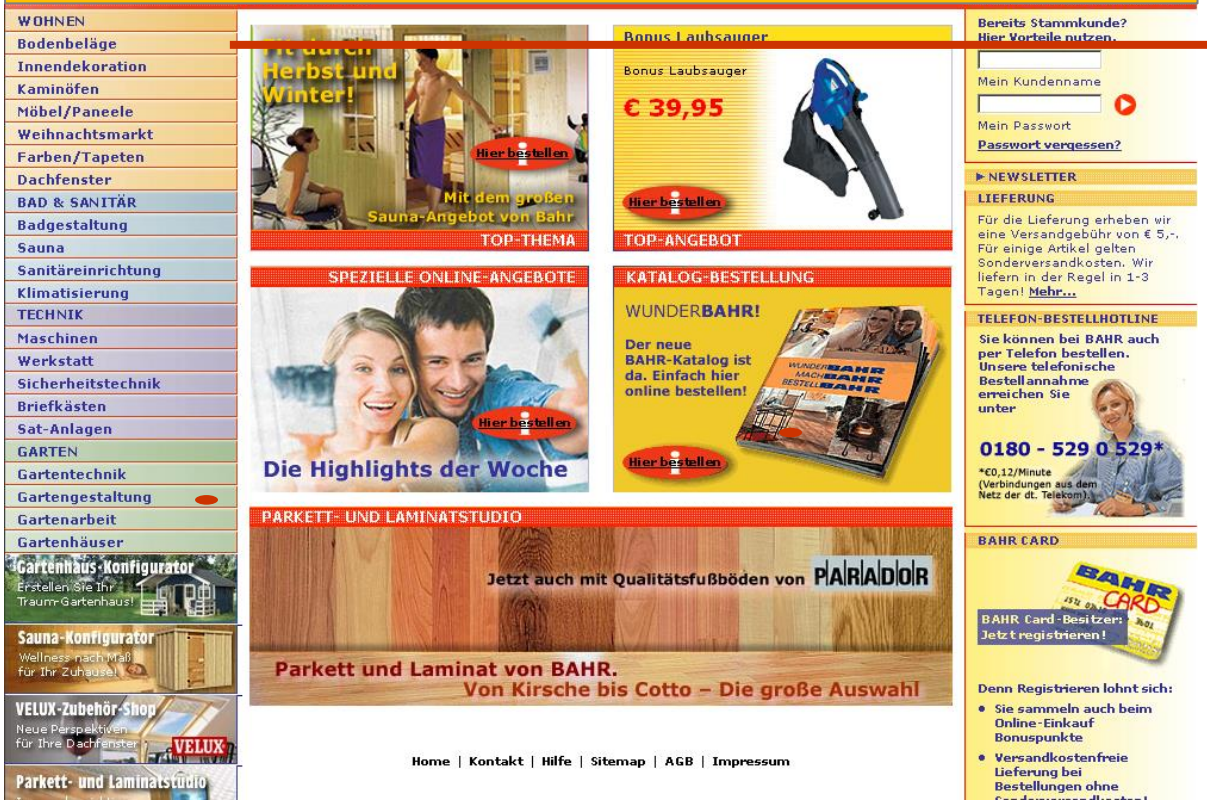
Ziel: Konzeption und Analyse

- Welche Techniken und Verfahren existieren?
- Fähigkeit zur Auswahl geeigneter Techniken

BEISPIELAUFBAU EINES WEBPORTALES

Produktkatalog

Klick-und-bau.com



Produkte mehrerer Anbieter

- Einheitliche Integration der Artikeldaten
- Geeignete Zusammenführung von Daten: Duplikate erkennnen, einordnen in Produktkatalog
- Einheitliche Such- und Filterfunktion

Der Weg zum Webportal

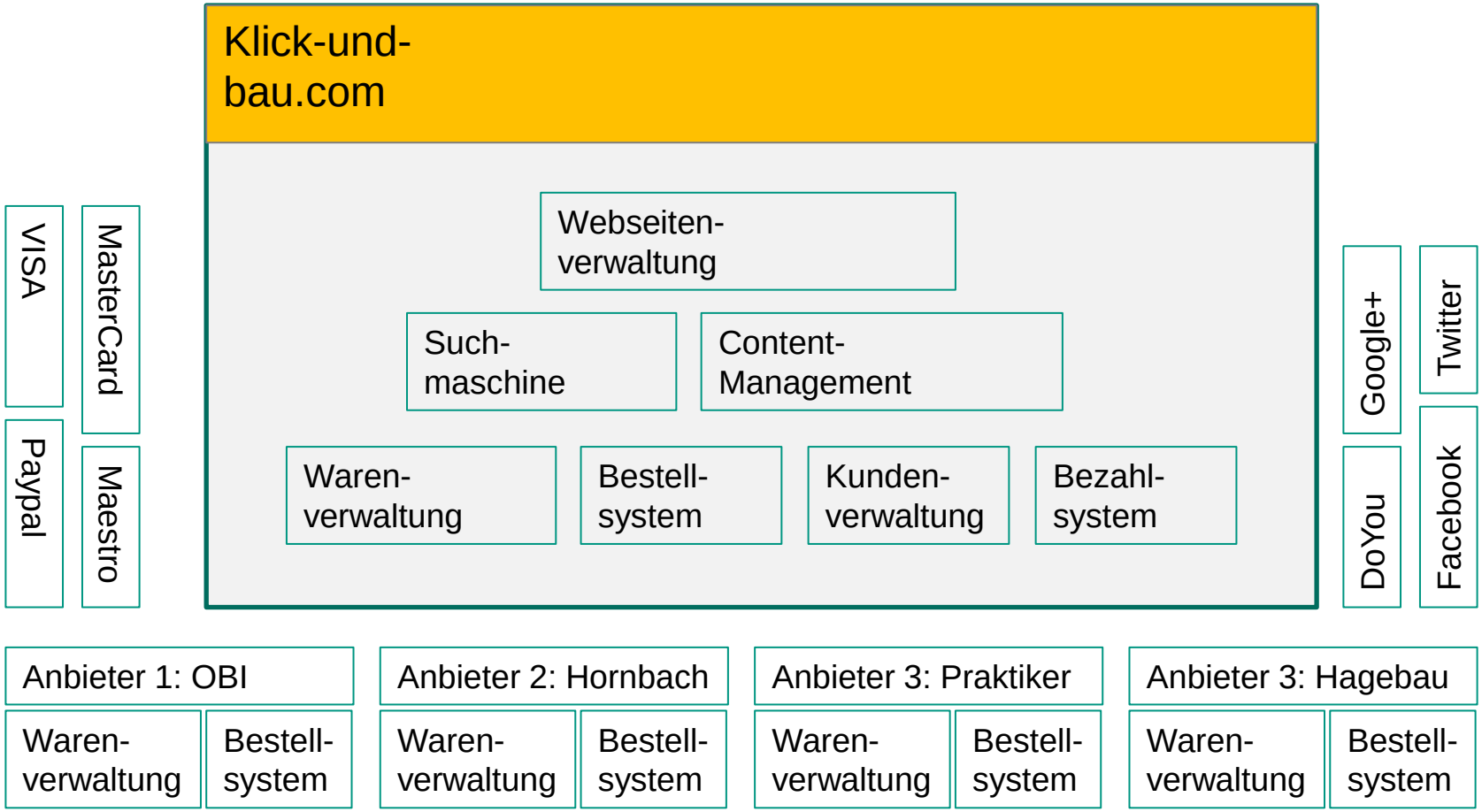
- **Regelfall**: *nachträgliches* Aufsetzen eines Portals auf Basis vorhandener Informationen und Dienste

- Wesentliche Schritte:
  1. Identifikation der Inhalte und ihrer Quellen
  2. Festlegung einer Integrationsarchitektur
  3. Technische Integration
  4. Inhaltliche Integration
  5. Visuelle Aufbereitung und Personalisierung
  6. Bereitstellung und Betrieb der Plattform

- Auch bekannt als *Enterprise Application Integration (EAI)*

Schritt 1: Identifikation der Inhalte und ihrer Quellen

Eine Analyse der Inhalte und Quellen führte zur folgenden Übersicht



Schritt 2: Festlegung der Integrationsarchitektur

- Vielzahl verschiedener Quellen
  - Warenwirtschaftssysteme
  - Bestellsysteme
  - Buchhaltungssysteme
  - Kundendatenbank (Customer-Relationship)

- Unterschiedliche Technologien

⇒ Technische Heterogenität,

⇒ ebenso für Inhalte (verschiedene Schemen, etc)

Leitfrage 1

Welche Infrastruktur benötige ich, um meine unternehmensinternen (sowie externen) Anwendungen zusammenzubringen?

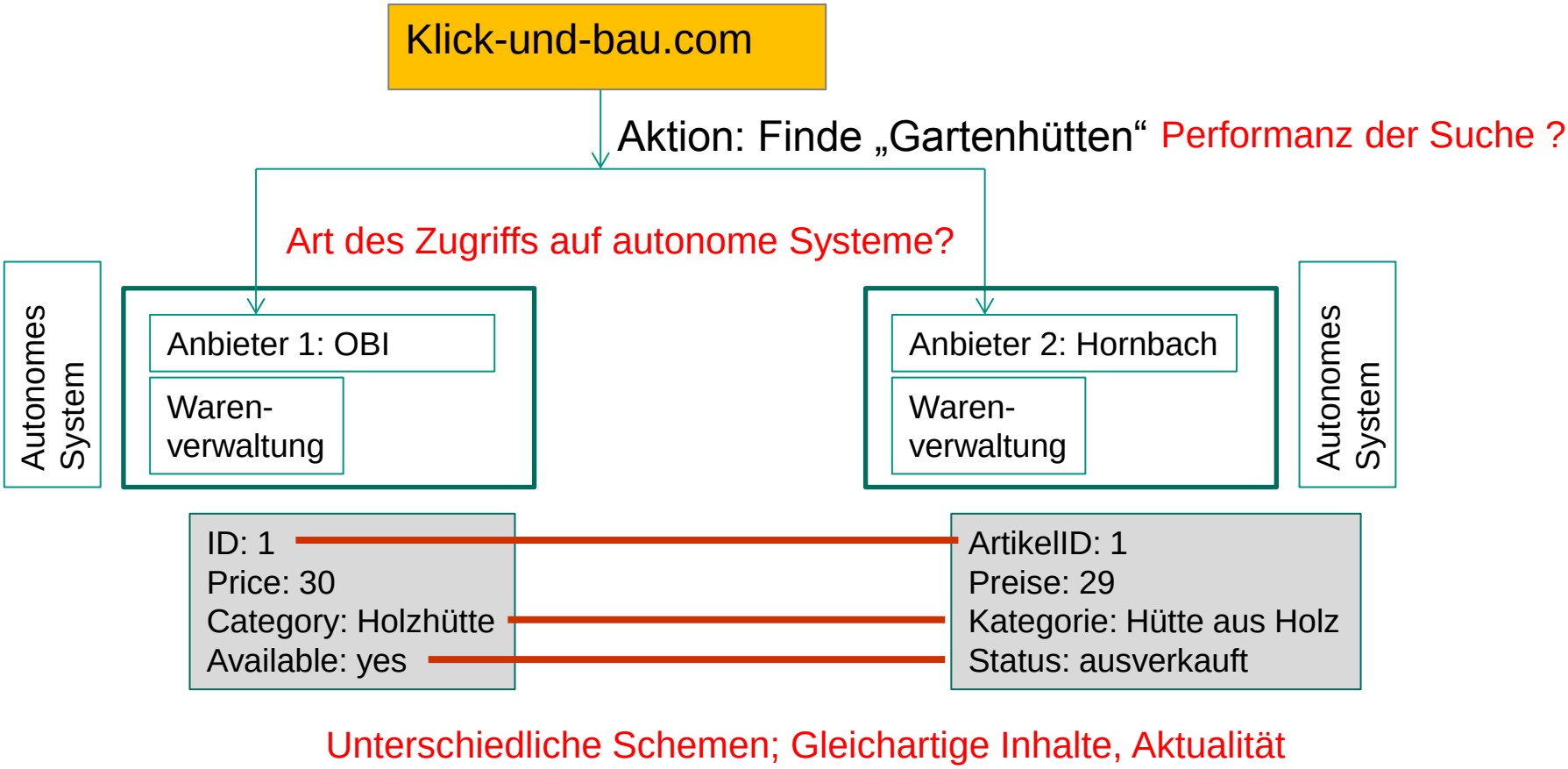
Schritt 3: Technische Integration

- Wie baut man skalierbare, wartbare und flexible Anwendungen auf?
  - Mehrschichtenarchitekturen und Application Server

  [out of scope: skalierbare verteilte Architekturen, z.b. Cloud Computing]
- Welche Technologien eignen sich zur Integration heterogener Datenbanken?
  - Datenbankabstraktion / Extract-Transform-Load (ETL)
- Welche Technologien eignen sich zum Aufbau umfangreicher Webanwendungen?
  - Auswahl geeigneter Rahmenwerke
- Best Practice Konzepte und Produktbeispiele für Enterprise Application Integration.
  - Fokus auf Skalierbarkeits- und Wartbarkeitsanforderungen

Schritt 4: Inhaltliche Integration (1)

Informationsintegration über mehrere Quellen, Beispiel „Artikelsuche“



Schritt 4: Inhaltliche Integration (2)

- Probleme
  - lose Kopplung
  - starke Heterogenität (technisch wie semantisch)
  - eingeschränkte Gestaltungsmöglichkeiten für die Infrastruktur
  - Beherrschbarkeit bei großen Systemen

Leitfrage 2:

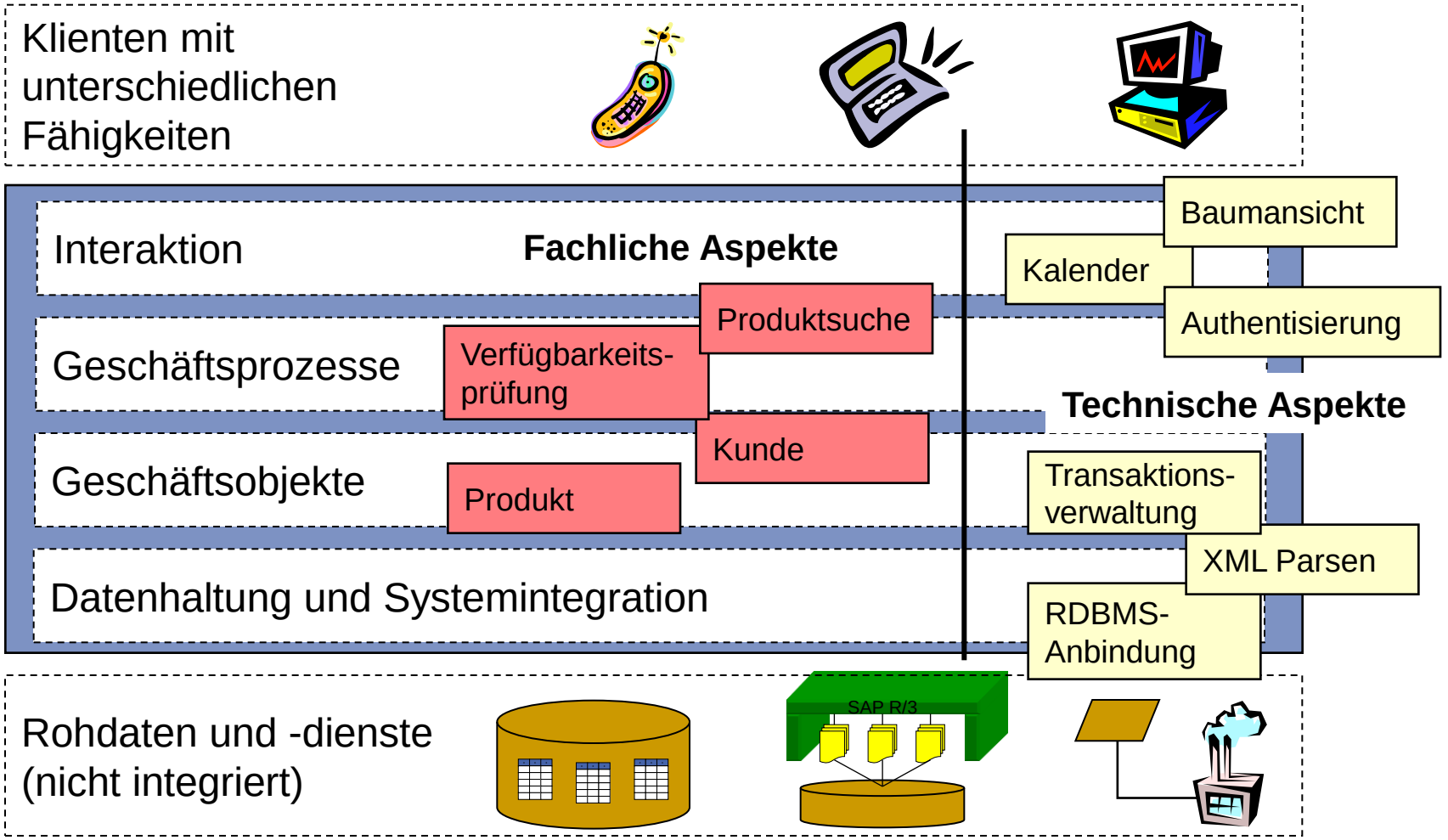
Welche Auswirkungen hat die **starke Autonomie der zu integrierenden Teilsysteme?**

Schritt 4: Inhaltliche Integration (3)

- Architekturen und Systeme zur Integration autonomer Informationsquellen
  - Auf welchen Ebenen werden Informationen integriert?
  - Virtuelle vs. Materielle Integration von Daten
- Verständnis für die erhaltenen Daten
  - Semantische Integration mittels RDF
  - Einsatz von Ontologien
- Alternativer Zugriff auf Informationsquellen (statt auf DB direkt)
  - Dienstorientierte Integration, Web-Services, Orchestration, Echtzeitereignisse

MEHRSCHICHTEN-ARCHITEKTUREN

„Separation of Concerns“



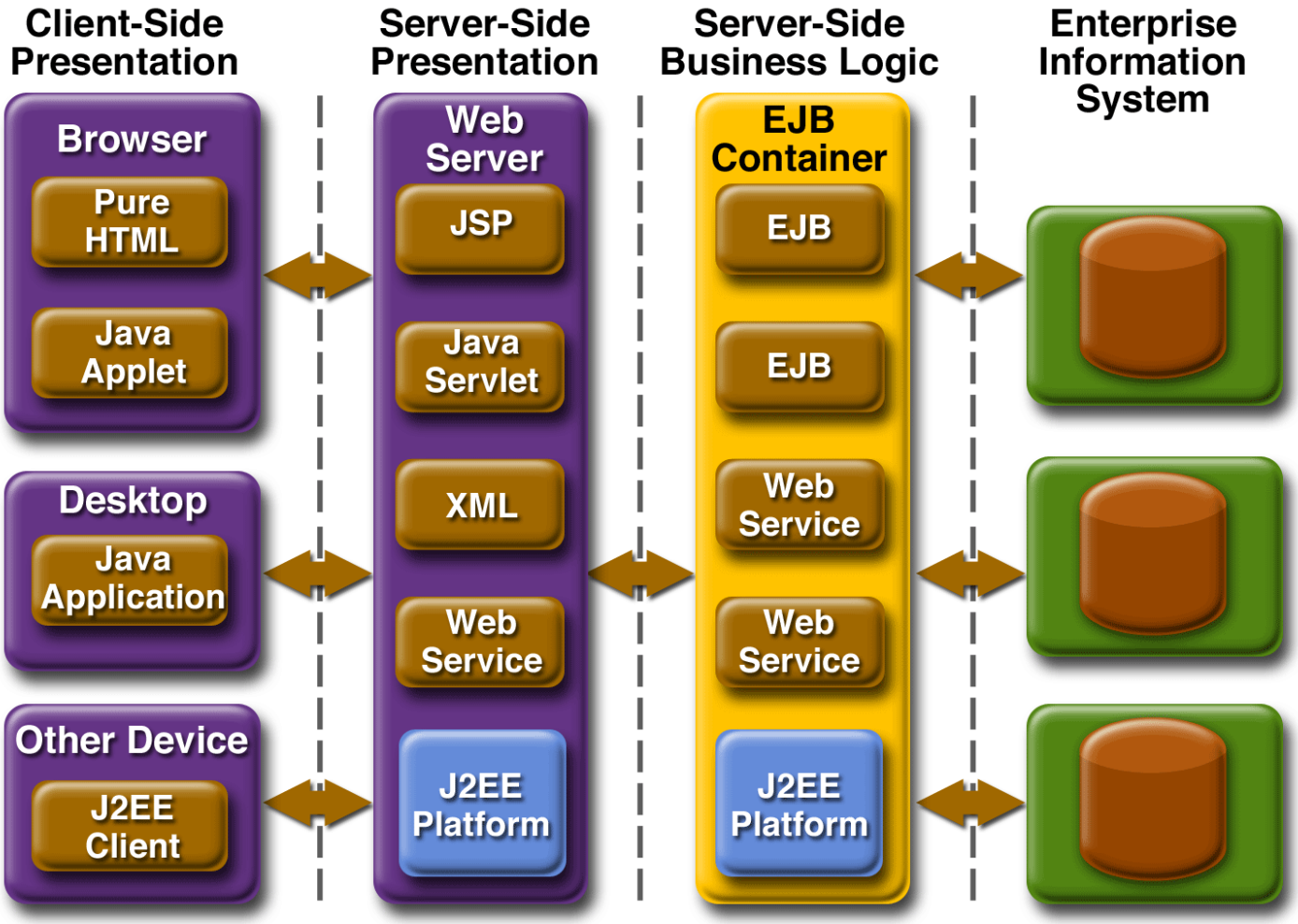
Die Vorteile auf einen Blick

- Mehrschichtenarchitektur
  - Austauschbarkeit einzelner Schichten
  - 1:N-Lösungen (z.B. ein Prozess – mehrere Oberflächen)
- Modularisierung
  - Detaillierte Strukturierung des Entwicklungsprozesses
  - Eine gewisse Wiederverwendbarkeit von Teillösungen
- „Separation of Concerns“
  - Hohe Wiederverwendbarkeit und Austauschbarkeit technischer Module (hier noch unternehmensintern!)

Typische Schichten (tiers)

- Präsentation
- Anwendungslogik / Business / Geschäftsprozesse
- Geschäftsobjekte / Data Access Layer
- Daten / Datenbank
- Integration

Anatomie einer J2EE-Anwendung (2)



EJB vs. Spring

- Lange Zeit umstritten, ob EJB oder POJO (Plain Old Java Object) mit Frameworks wie Spring der bessere Weg ist

- Inzwischen haben sich die beiden Lösungen angenähert
  - EJB leichter zu benutzen: weniger Konfigurationsdateien, mehr Annotationen, Dependency Injection, Nutzung von JPA, …
  - Spring hat einen kompletten Stack an Funktionalität entwickelt, der ähnliche Aufgabenstellungen wie EJB abdeckt

- Hauptunterschied
  - EJB lokalisiert das Framework im Application Server (Container)
    - Dadurch schwergewichtigere AS, z.B. JBoss, Websphere, etc.
  - Spring ist ein Framework oberhalb des Application Servers
    - Dadurch lassen sich auch leichtgewichtige Application Server einsetzen, wie z.B. Tomcat, Jetty oder Resin

DATENINTEGRATION

Zusammenfassung (2)

- **Verwaltung von Daten**
  - Speicherung der Daten : Interaktion mit Datenbank(en)
    - Datenbankzugriff: Abstraktion von Datenbanksystemen durch Apis, Tools
    - Leichtgewichtige ORM Verfahren: Laden von Daten direkt aus Datenbank
  - Integration der Daten in die Anwendung : Daten -> JavaObjekte
    - Schwergewichtige ORM Verfahren: Verwaltung der Persiszenzobjekte, z.b. Mittels Hibernate

- **Integration von Daten**
  - Erhalt der Daten: Pull vs. Push
    - Pull: ebenfalls mittels ORM, ideal: Verfügbarkeit von Web Services
    - Push: Daten werden in verschiedensten Formaten übermittelt.
  - Formate der Daten
  - Erkennung von Duplikaten etc.
    - Einsatz von ETL Verfahren

Arten des Objektrelationalen Mappings

- **Leichtgewichtige Rahmenwerke für ORM**
  - Unterstützen die Anfrageausführung, z.B. Die Verwaltung von Transaktionen
  - Bieten Methoden zum einfachen Iterieren über Tabellen an, um von Tabelleninhalt in Objektinhalte zu transformieren ; Bieten das vereinfachte Erstellen von SQL Anfragen an
  - Verdecken nicht von spezifischen Eigenschaften von SQL und deren Erweiterungen des Datenbanksystems.
- **Schwergewichtige Rahmenwerke für ORM**
  - Verwalten selbständig alle Anfragen an die Datenbank, inkl. Transaktionen
  - Verwalten selbständig alle Transformationen von den Tabellen der Datenbank zu den Objekten der Anwendung.
  - Verdecken vollständig von allen spezifischen Eigenschaften von SQL des Datenbanksystems
  - Erstellen und verwalten selbständig alle notwendigen Tabellen im Datenbanksystem.

Einsatz leichtgewichtiger Rahmenwerke für ORM

Bewertung leichtgewichtiger Rahmenwerke für ORM

Nachteil

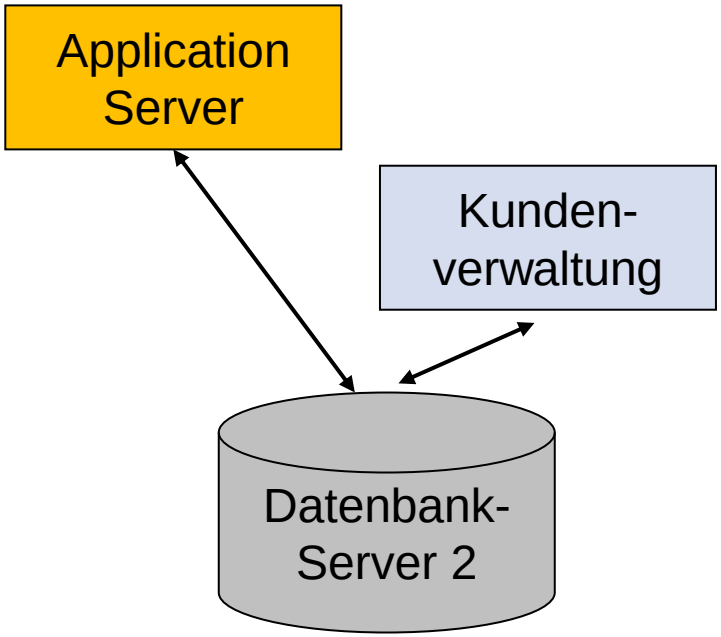
- Entwickler müssen SQL Syntax des Datenbanksystems beherrschen

Vorteil

- Einfluss auf Tabellenschema
- Umfangreiche Optimierung möglich

Übliches Einsatzgebiet

Direkter Datenbankzugriff auf bestehende Datenbanken



Selbstkonfiguriertes ORM mit Spring

Anforderung: Zugriff auf Daten in Drittsystem (z.b. Artikelverwaltung von Anbieter A) soll ermöglicht werden.

Erfordert:

1. Konfiguration der Klassen und Interface für Daten in der eigenen Anwendung
2. Definition des zu verwendenden SQL Statements
3. Implementierung des ORM
4. Konfiguration der Spring Beans (XML)

Einsatz schwergewichtiger Rahmenwerke für ORM

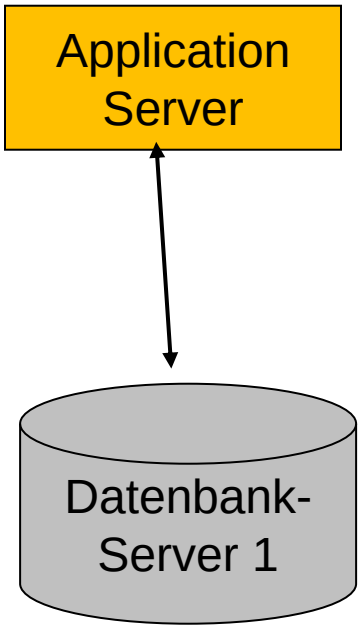
Bewertung schwergewichtiger Rahmenwerke für ORM

Nachteil

- Kein (bzw. geringer) Einfluss auf resultierendes Datenbankschema

Vorteil

- *„Programmierung“* von Datenbankanfragen
- Komplette Abstraktion von der Datenbank



Übliches Einsatzgebiet

Objekte der eigenen Applikation

Beispiel: Hibernate (JPA)

ORM mit Hibernate

Anforderung: Zugriff auf Datenobjekte aus der Anwendung heraus, ORM soll verwaltet sein. Beispiel: Bestellsystem.

Verwendung von Hibernate für das Bestellsystem erfordert:

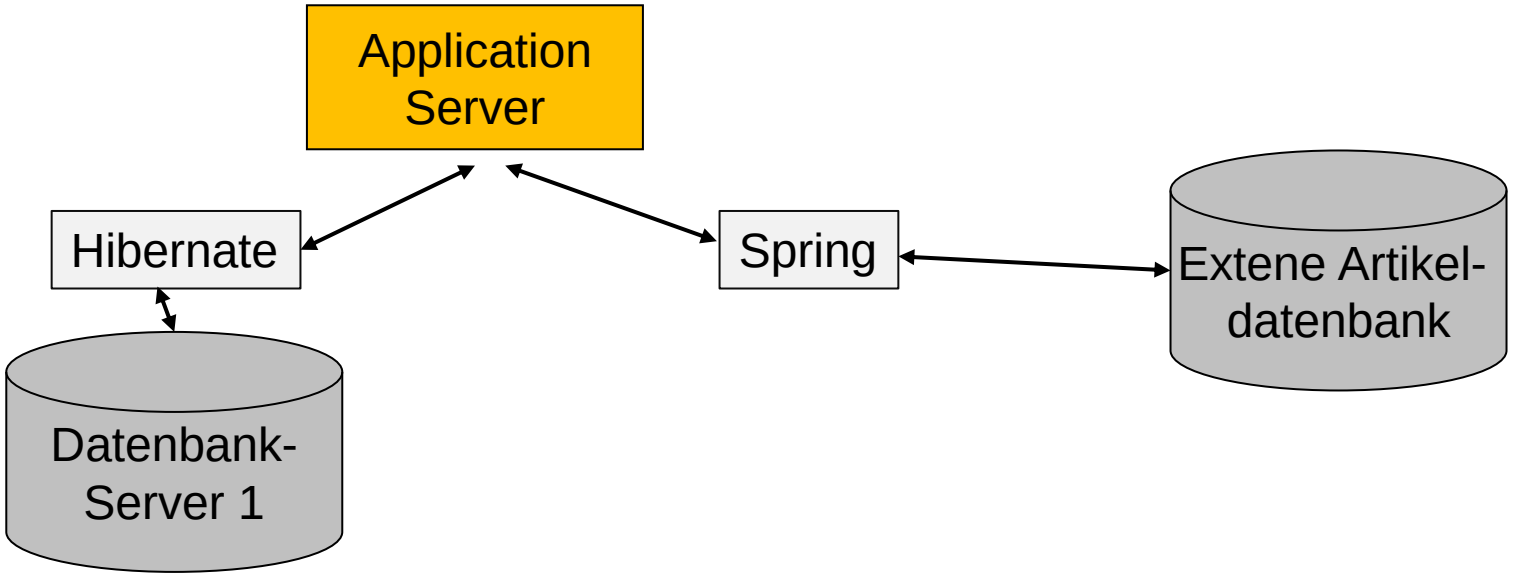
1. Erstellung von Persistenz - Objekten
2. Annotation der Klasse und Attribute
3. Definition von Datenbankabfragen mit Hibernate Query Language (HQL)
4. Definition und Implementierung einer Abfrageschnittstelle.
5. Konfiguration des Entity-Managers

ORM Verfahren für Portal kombinierbar

Beispiel: Integration von Artikeldaten von Anbieter

- Spring: lädt Daten von externer Datenbank, verwendet Article Klasse
- Ruft in Hibernate addArticle auf => Daten geladen in eigener DB

classOrder() class Customer() **class Article()** ...



INFORMATIONSINTEGRATION

Anbindung: Virtuell vs. materialisiert

- Aufbau einer zentralen Datenbasis im Portal, in die die Inhalte der angeschlossenen Anbieter in einem Vorverarbeitungsschritt importiert werden. Diese Datenbasis stellt die integrierte Sicht dar

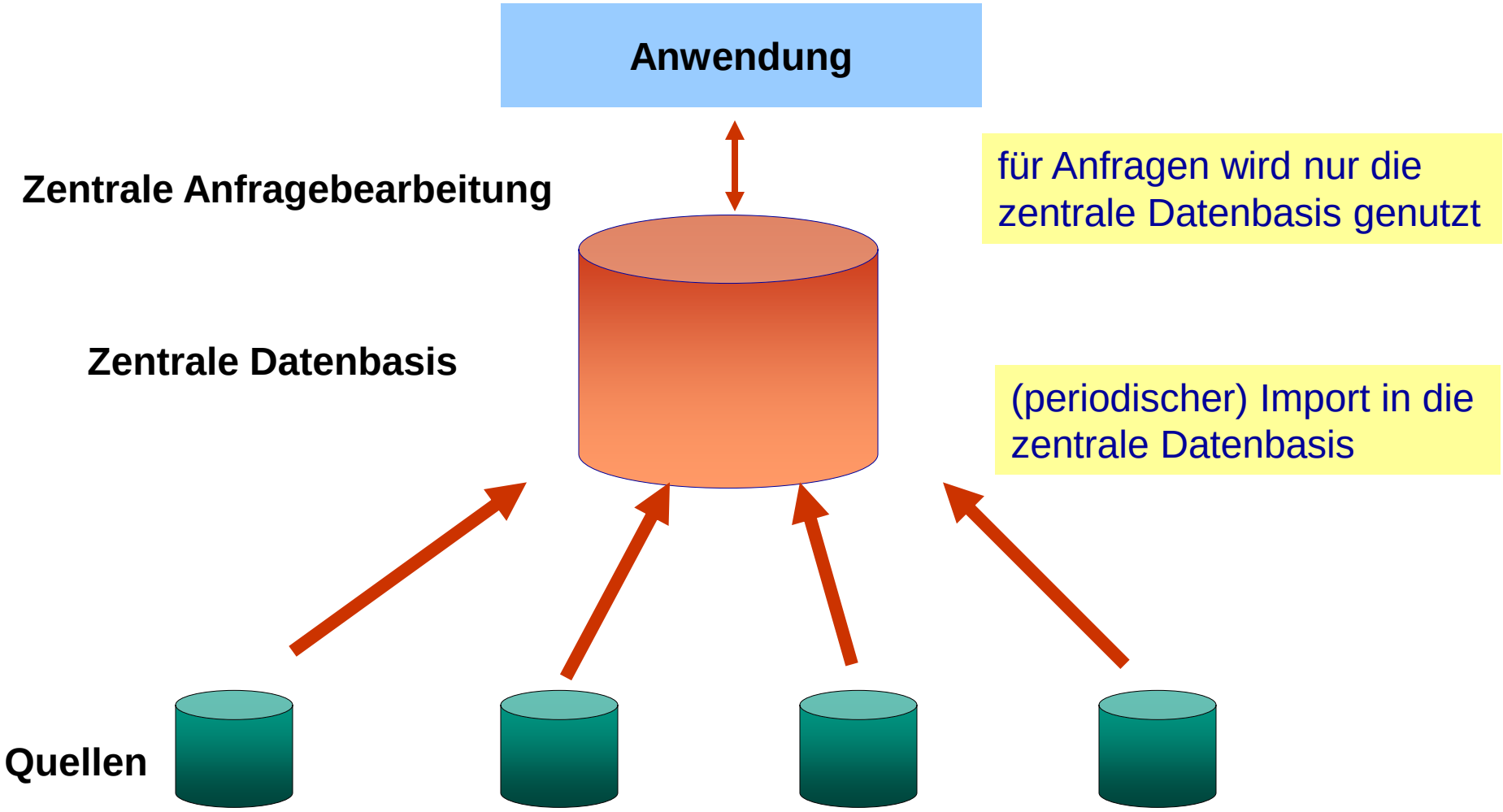
  materialisierter (physischer) Ansatz
  a priori Integration

- Nutzung der Systeme der Fremdanbieter für die Anfrageauswertung (Durchreichen von Anfragen). Die integrierte Sicht ist physisch nicht vorhanden; sie wird von Mediatoren dynamisch bereitgestellt:

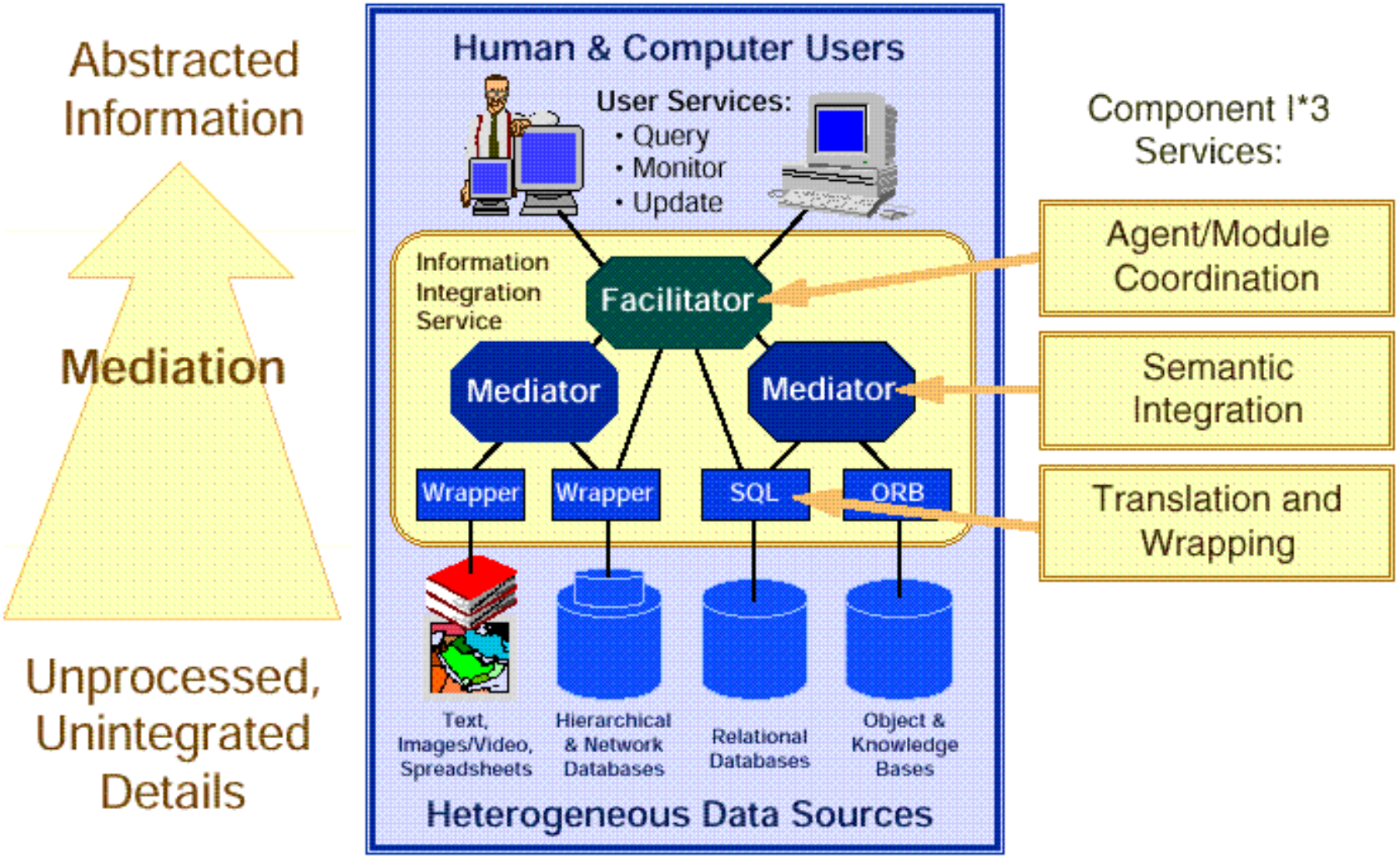
  

  virtueller (logischer) Ansatz
  a posteriori Integration bei Bedarf

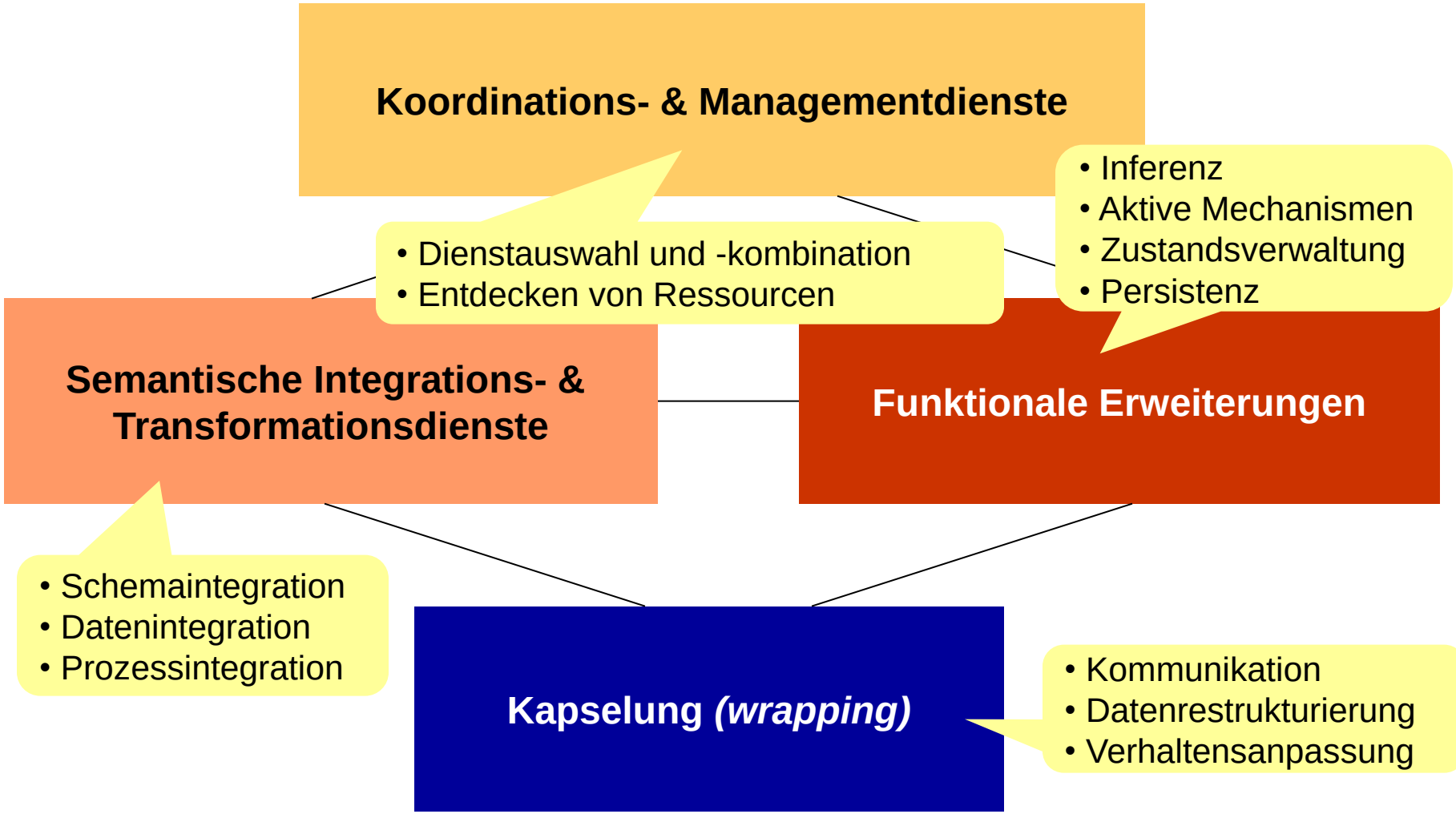
Materialisierter Ansatz



I³-Referenzarchitektur nach [AHK95]



Intelligent Integration of Information (I^3)



Problembereiche

Auf folgende Problembereiche soll im folgenden näher eingegangen werden:

- Facilitator
  - Quellenauswahl
- Mediator
  - einheitliches Informationsmodell
  - Anfragezerlegung, Anfrageübersetzung
  - semantische Integration (s. nächste Vorlesung)
  - Objektverschmelzung
- Wrapper
  - Informationsextraktion

Virtuell vs. Materialisiert – Fazit

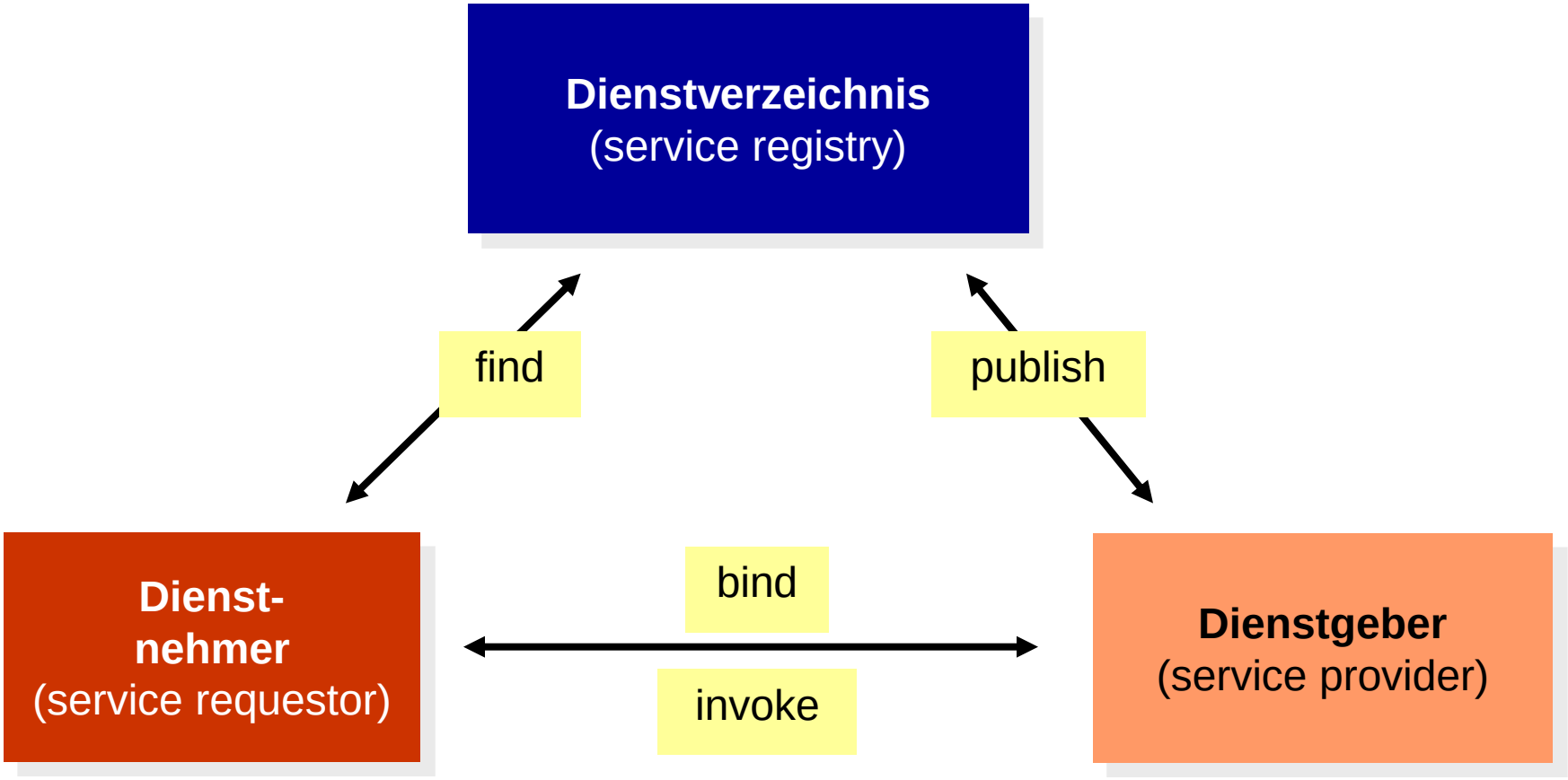
- Virtueller Ansatz ist zu bevorzugen bei Datenquellen
  - mit hoher Änderungsrate
  - mit großem Datenvolumen
  - mit geringer Anfragehäufigkeit/geringem Anfragevolumen
- Um den Implementierungsaufwand für ein virtuelles Integrationssystem in Grenzen zu halten:
  - Wie heterogen sind die anzubindenden Systeme?
  - Welche Anfragefunktionalität wird benötigt? (Joins?, boolesche Verknüpfungen, ...)
- Oft bietet sich auch ein hybrider Ansatz an
  - kleinere, weniger mächtige Quellen werden repliziert
  - die restlichen werden virtuell angebunden

Virtuell vs. Materialisiert – Fazit (2)

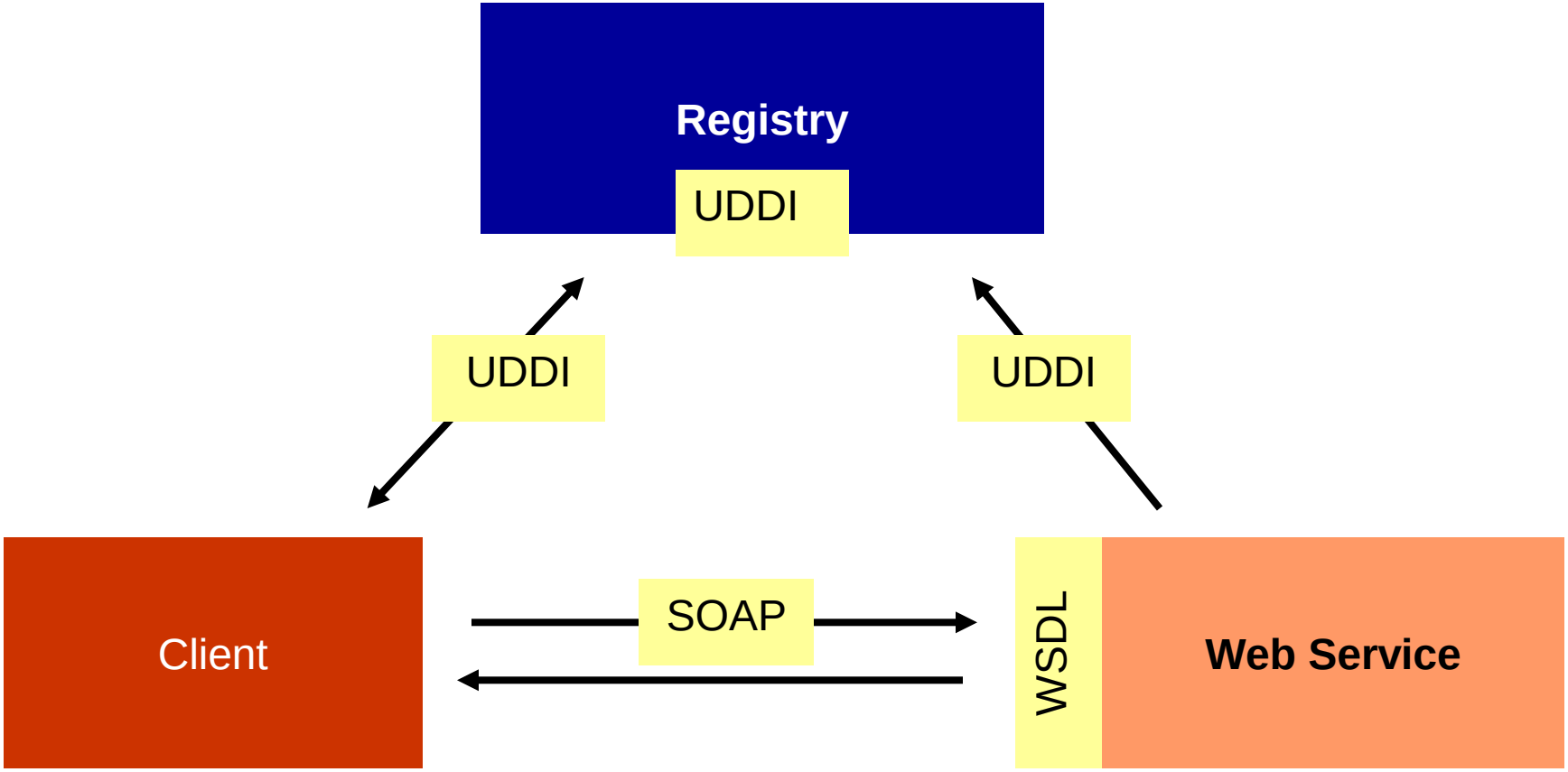
- Was die Erfahrung zeigt:
  - virtuelle Integrationsansätze ohne wohldefinierte Schnittstellen an der Grenze des eigenen Einflussbereiches taugen nur für Ad-hoc-Integration oder "feindliche Nutzung"
  - Ist eine Kooperation mit dem Anbieter möglich, so erhöht dies deutlich die Stabilität
  - Zur Erhöhung der Dienstgüte setzt sich bei überschaubaren Datenvolumina der materialisierte Ansatz durch
    - Preissuchmaschine.de: pull-Ansatz
    - froogle.de: push-Ansatz
- Fazit
  - materialisierten Ansatz sollte man als Ausgangspunkt sehen
  - wenn das nicht geht (organisatorisch/technisch), dann virtuell

DIENST-ORIENTIERTE ARCHITEKTUREN

Dienstorientierte Architekturen



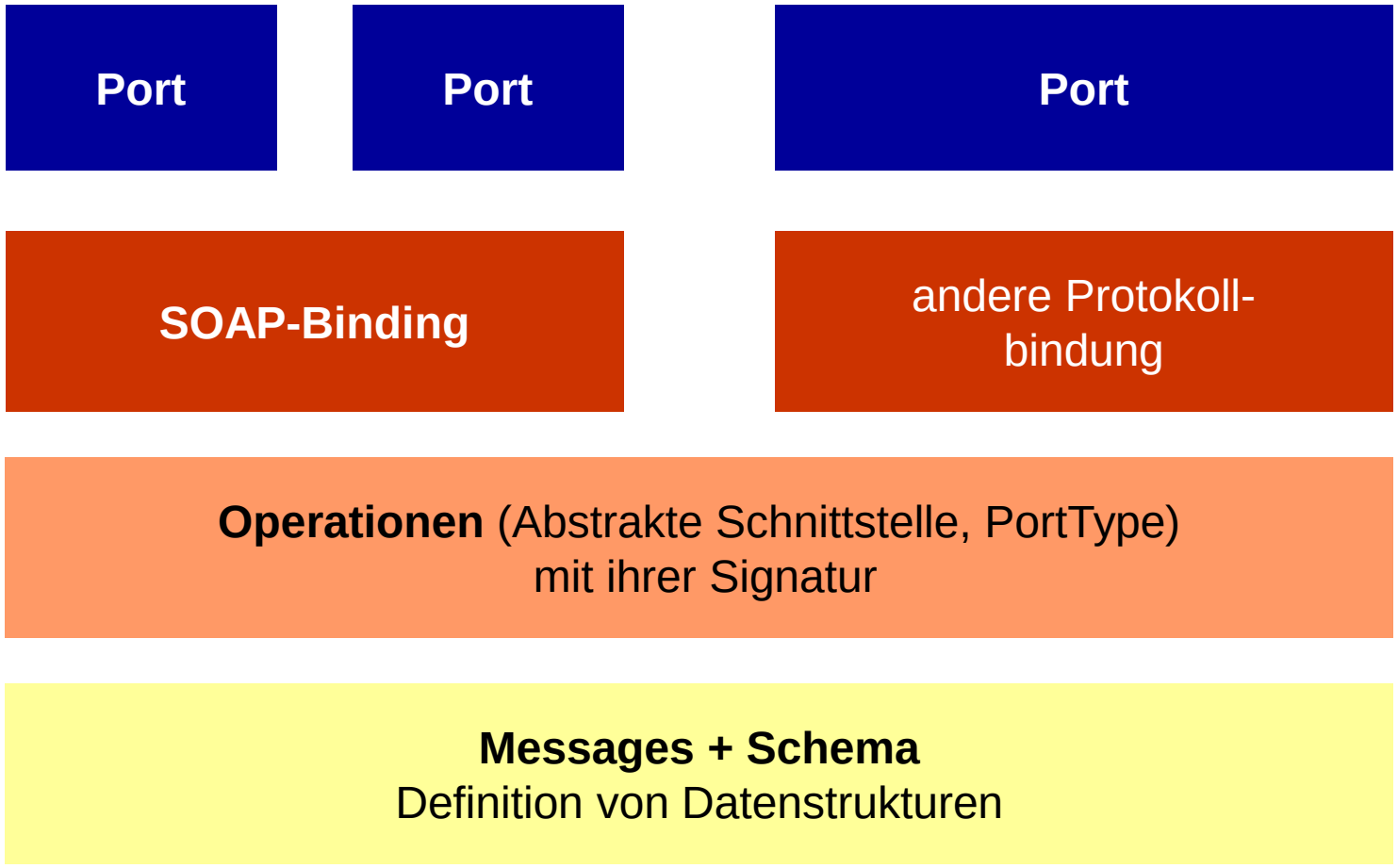
SOAP, UDDI, WSDL



WSDL: Überblick

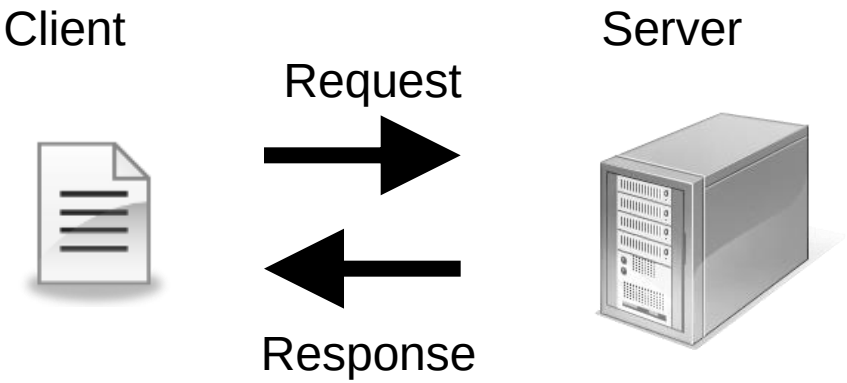
- Welche Operationen bietet der Dienst an?
- Welche Parameter haben die Operationen
  - Struktur der Aufrufnachricht
  - Struktur des Ergebnisses
  - Rückgriff auf XML Schema
- Wo und wie kann ich einen Dienst erreichen?
  - Adresse
  - Informationen über das Protokoll
    - z.B. SOAP, auch mehrere Protokolle!
    - verwendetes Transportprotokoll: HTTP, SMTP etc.
- aber: keine semantische Dienstbeschreibung

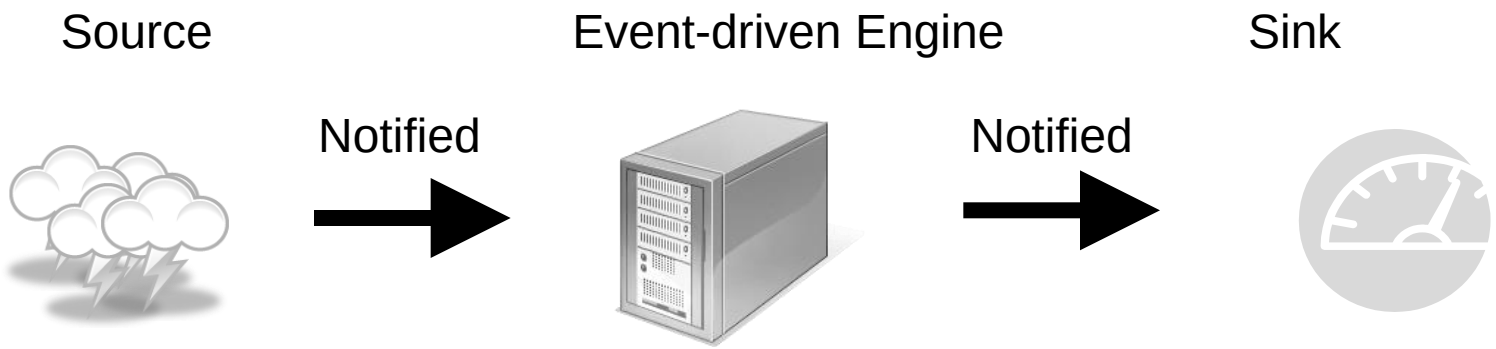
WSDL: Aufbau


Port
Port
Port
SOAP-Binding
andere Protokoll-bindung
Operationen (Abstrakte Schnittstelle, PortType) mit ihrer Signatur
Messages + Schema
Definition von Datenstrukturen

EREIGNISGESTEUERTE ARCHITEKTUREN

Time- & Request-driven vs. Event-driven





EDA Application Characteristics

- Sense and Respond
  - Timely response when reality deviates from expectation
- Asynchrony
  - Timing of events are not controlled
  - communication is usually unidirectional
- Global situational awareness
  - Awareness by correlating multiple sources of data
  - from outside the enterprise with enterprise data

EDA Principles

- Report current events as they happen
- Push notifications
- Producer decides when to send
- Respond immediately
- One-way communication; fire-and-forget
- Free of commands; not prescriptive
- Decoupling of sender and receiver

Chandy, Schulte (2009)

Comparison of SOA and EDA

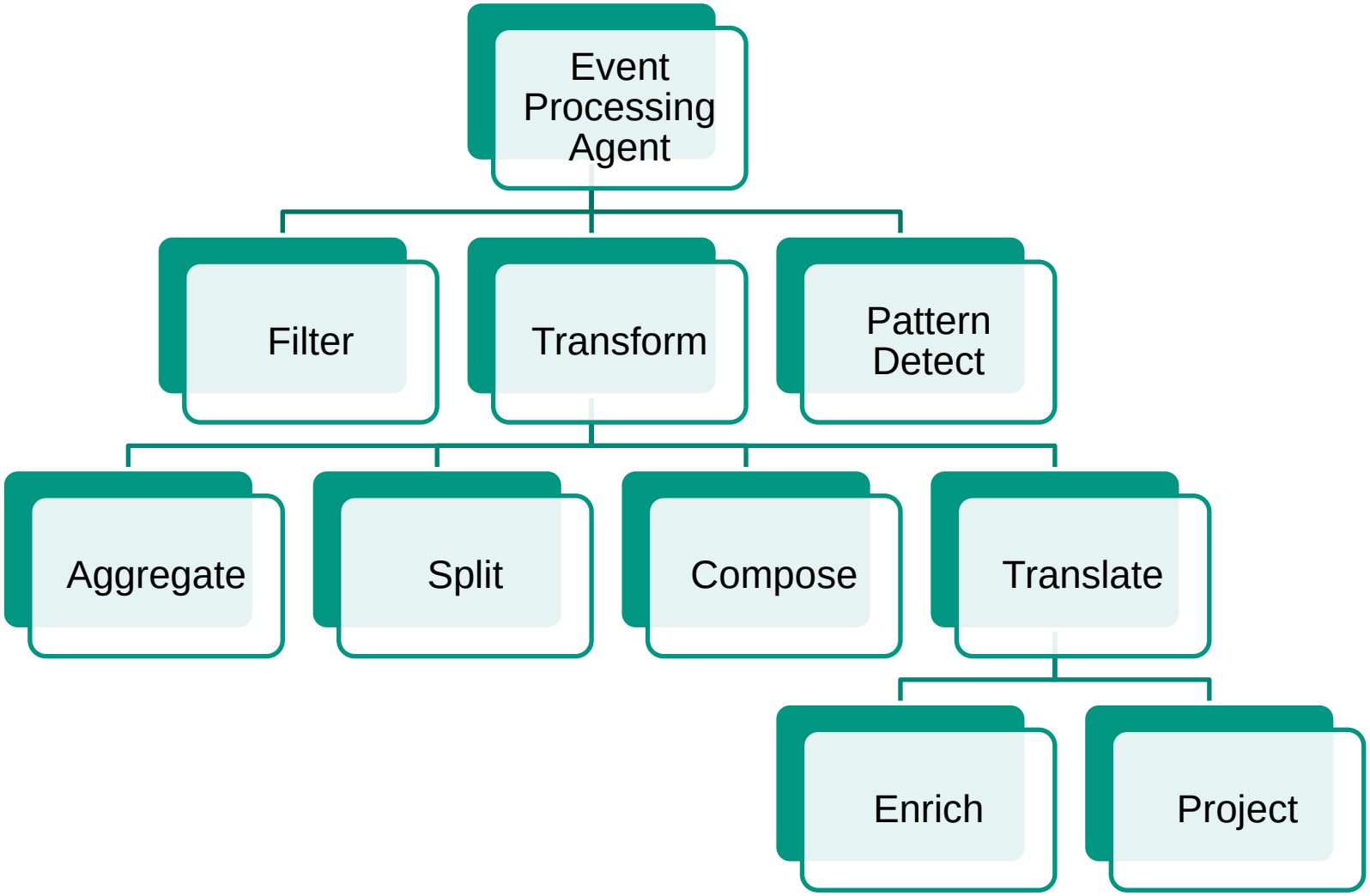
Attribute	SOA	EDA
Design focus	Services and service interfaces	Events and event message
Processing model	Client invokes functionality	Source sends message to middleware, sink receives message
Modularity	Clients/ services	Event source/ event sink
Communication pattern	Request/ response	Publish/ subscribe
Transaction control	Client	Independent
Relation between components	Client knows task, name, address; service has to be available; only service knows implementation	Source and sink do not know each other; source does not know if a sink exists
Dependencies between components	Interfaces, versions, SLA	Event schema, version
Degree of coupling	Loose coupling	Extremely loose coupling

Bruns, Dunkel (2010)

Event Processing Agent

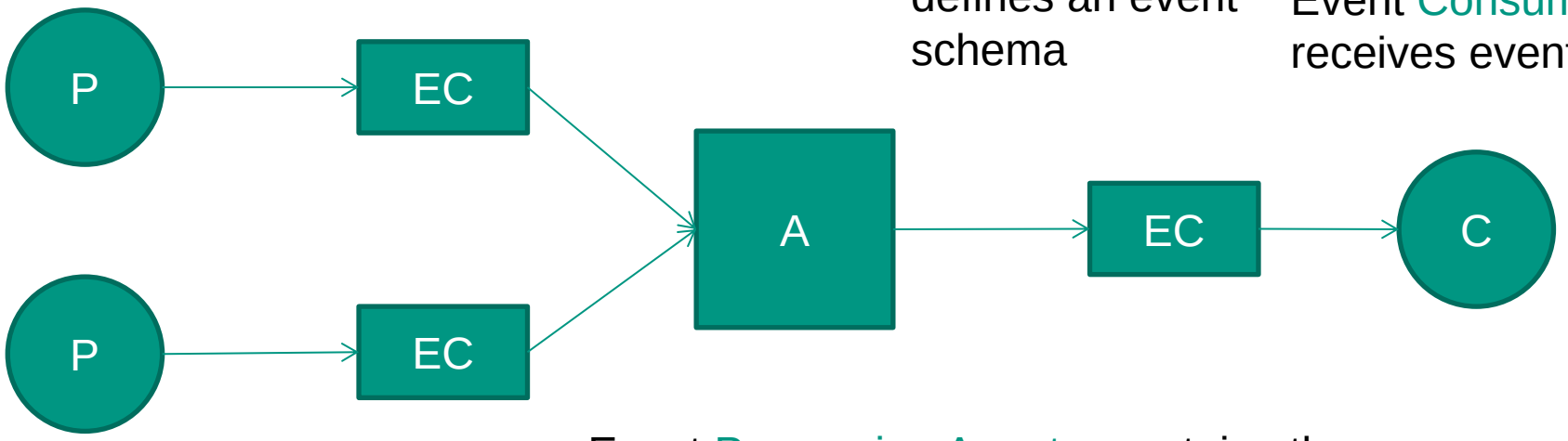
- Input Adapter
  - transforms events into an internal format and puts events into input event stream
- Event Processing Network (EPN)
  - is composed of Event Processing Agents (EPA)
  - EPA monitor events streams to detect and act on events
  - EPA filter, match, and derive (translate, aggregate, split,…)
- Output Adapter
  - translates events into metrics, messages or function calls

Event Processing Agent Operations



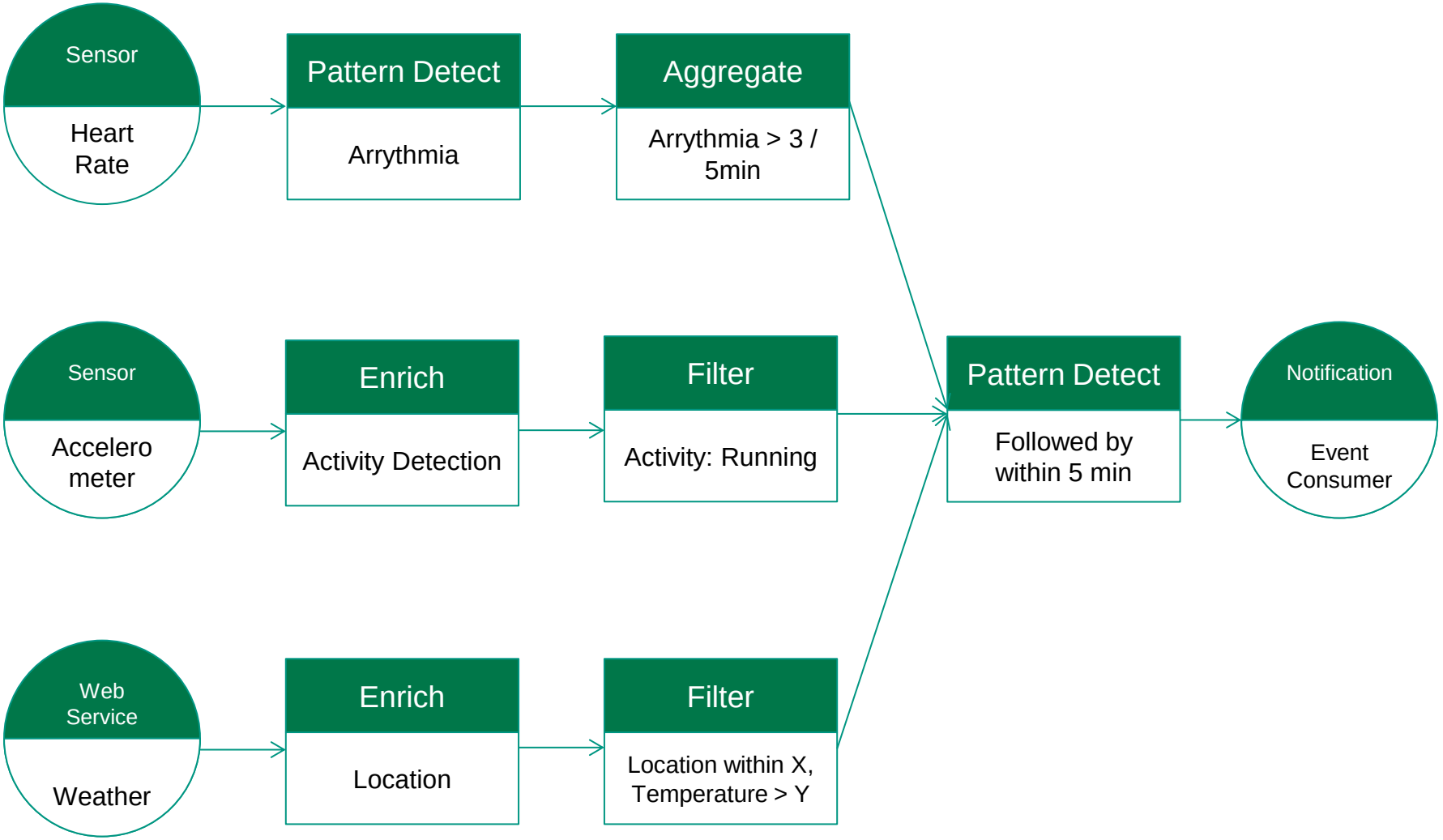
Etzion, Niblett (2011)

Event Processing Network



Etzion, Nibblett (2011)

Example: Event Processing Network

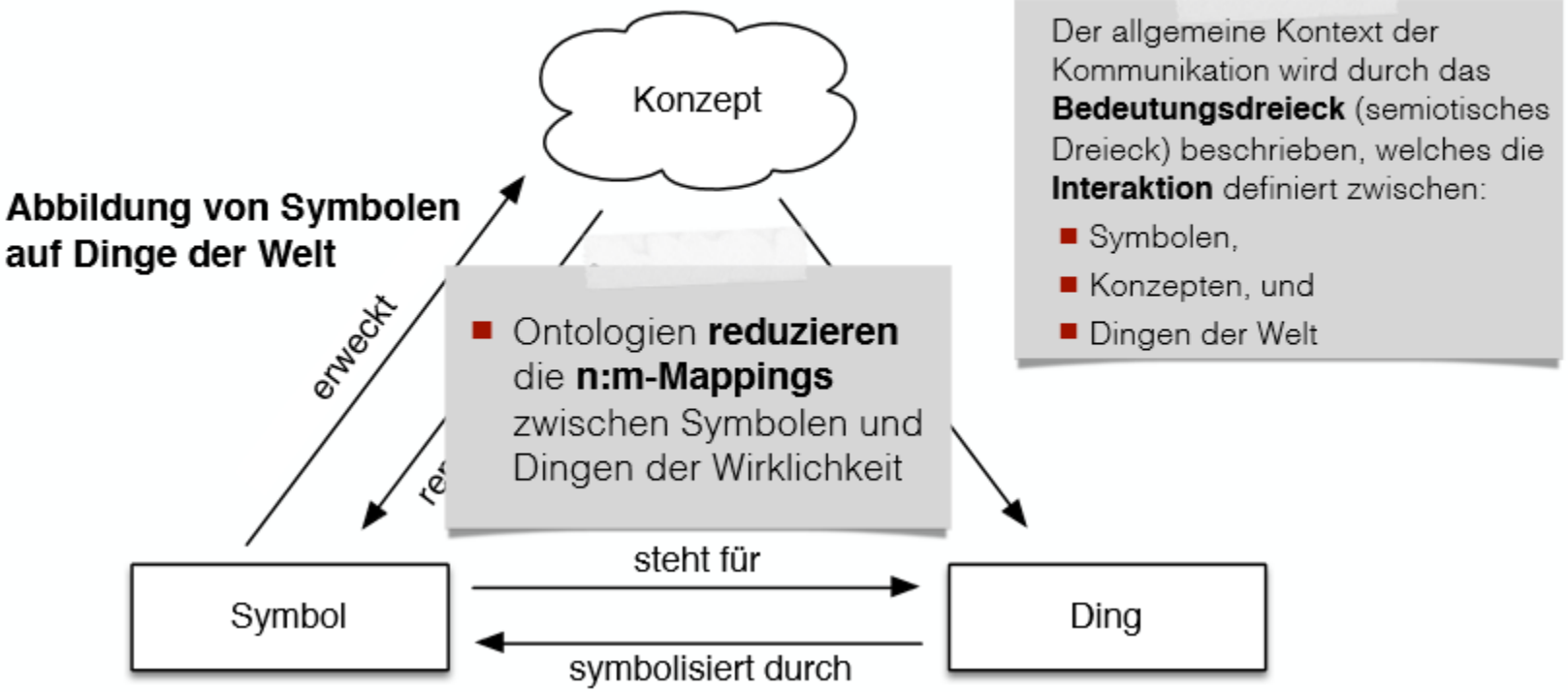


SEMANTISCHE INTEGRATION

Zusammenfassung

- Ontologien
  - Definitionen
  - Eigenschaften
  - Rolle in der Informationsintegration
- Ontologie-Sprachen
  - Übersicht und Anforderungen
  - RDF/S
  - OWL

Semiotische Dreieck



Eigenschaften

- Abstrakte Eigenschaften
  - Spiegelt das gemeinsame Verständnis einer Domäne wider
    - Wird gemeinschaftlich erstellt oder erarbeitet
  - Explizites Modell der Domäne
    - Enthält wenig verstecktes Wissen
  - Formale Spezifikation
    - Das modellierte Modell kann von einem Computer interpretiert werden
- Konkrete Eigenschaften
  - Klassen (Konzepte/Begriffe), Instanzen von Klassen (Individuals/Objekte), Eigenschaften von Instanzen einer Klasse (Properties/Attribute) sind häufig Elemente von Ontologien
  - Zusammenhänge von Klassen, Eigenschaften und Instanzen werden häufig über Axiome postuliert
  - Die Bedeutung einer Ontologie basiert oft auf Prädikatenlogik
  - In der Interpretation wird oft eine „offene Welt“ unterstellt (Open World Assumption)

Schritt 5: Visuelle Aufbereitung und Personalisierung

- Erstellung der Webseite mit geeignetem Webrahmenwerk

  Tutorials zeigen hierzu die geeigneten Vorgehen, z.B.

  - Erstellung von umfangreichen Webseiten mittels Apache Wicket
  - Grundlagen von Google Web Toolkit
- Mobile Web Apps
  - Personalisierung der Seite auf Devices, Best practices
- Leichtgewichtige Integrationsansätze
  - Integration von Drittanbieterdaten auf Präsentationsebene

WEB-TECHNOLOGIEN

Zusammenfassung Grundlagen

- **http**: Protokoll im Internet, Adressierung mit URLs
- **WebServer**: Serverseite, erweiterbar (z.B. Gateways)
- **Browser**: Clientseite, dient zur Darstellung von Inhalten
- **(X)HTML**: dient zur Formatierung von Webseiten
- **Kommunikation**: zustandslos, Request – Response Prinzip
- **CSS**: dient zur Trennung von Inhalt und Formatierung
- **JavaScript**: ermöglicht Ausführung von Skripten im Browser

Zusammenfassung: kleine Webanwendungen

- **Für kleine Anwendungen**
  - Skalierbarkeit, Wartbarkeit und Wiederverwendbarkeit stellen keine Anforderung dar
  - Plugins ermöglichen Spezialanwendungen
  - Alle gezeigten Skriptansätze ermöglichen die Generierung von dynamischen HTML-Seiten.
  - Erstellung von Anwendungen mit Desktop-Charakter hier nicht nötig
  - Generierung von statischen Seiten ermöglicht auch umfangreiche Anwendungen bei geringen Ressourcen

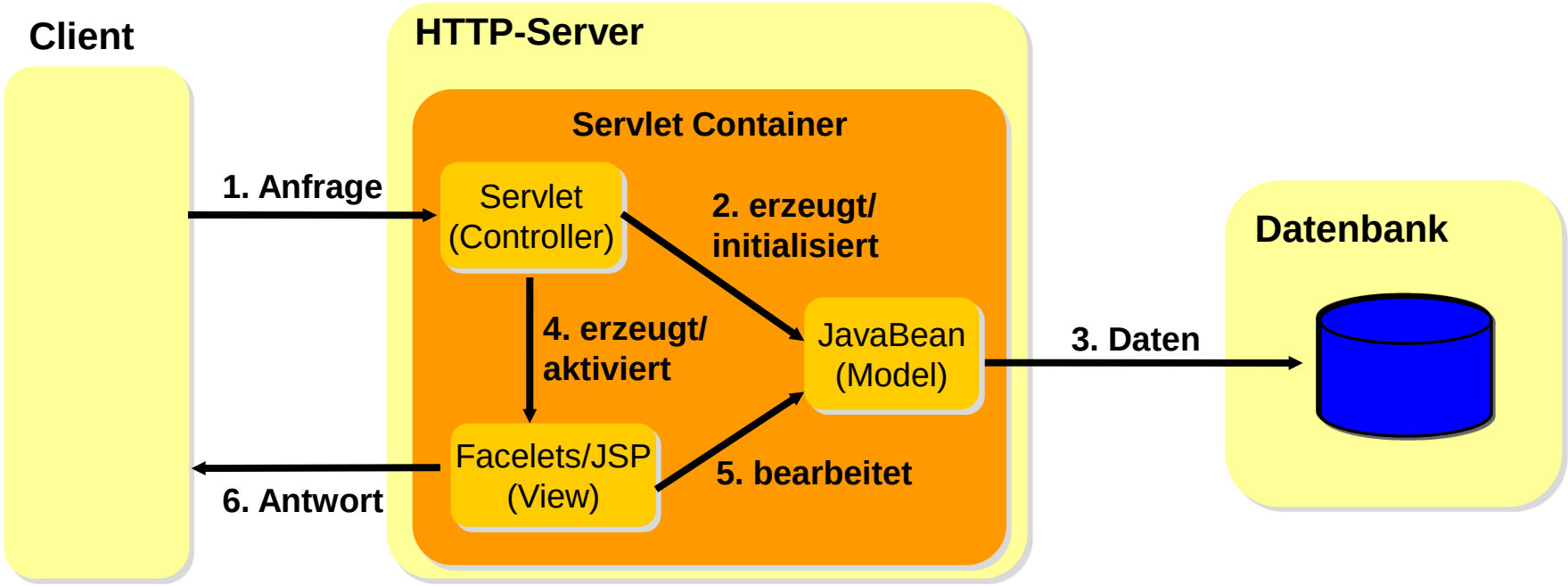
    Umsetzung der Anwendung also entscheidend von Vorkenntnissen der Programmierer und den installierten Interpretern / Gateways auf einem Web-Space

Zusammenfassung: komplexe Webanwendungen

- **Zusammenfassung Hauptkriterien**
  - Erfüllung der genannten Anforderungen, Fokus auf
    - AJAX-Support
    - Erweiterbarkeit
    - **Trennung von Anwendung und Darstellung**
  - Lizenzierung: Proprietär vs. open-source
  - Programmiersprache (im Folgenden: Focus auf JAVA)
  - Zeitaufwand zur Erstellung von Web-Anwendungen
  - Support: Qualität der Mailinglisten-Antwortzeit

JSF: Java Server Faces

- **Grundidee**: Komponentenverhalten wie in SWING
  - Java Beans als Model; JSP Seiten / Facelets und Komponentenbaum als View; Action Listener und Servlet als Controller

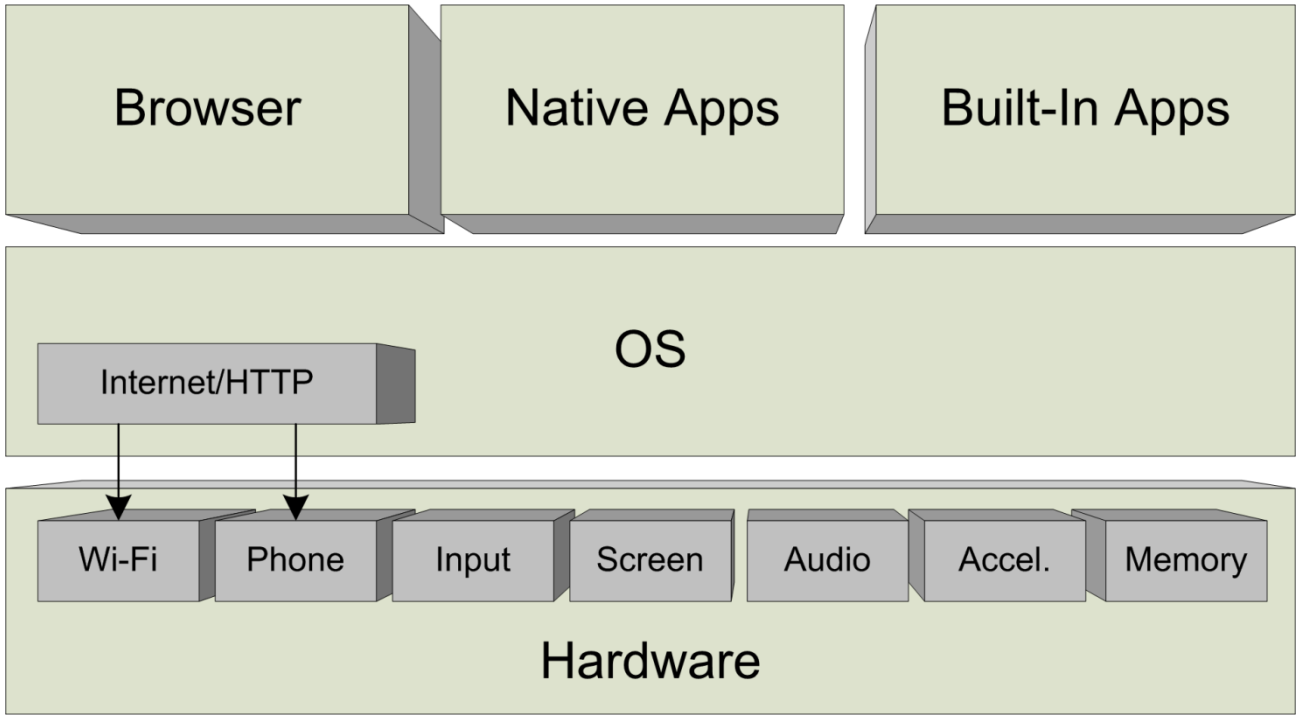


Zusammenfassung

- Umfangreiche Webanwendungen erfordern Web-Rahmenwerke
  - Ermöglichen langfristige Wartung einer Anwendung
  - Trennung von Darstellung und Anwendung wichtig
  - Verdecken Komplexität der zugrunde liegenden Webtechnologien
    - Umfangreiche Anforderungen an ein Web-Rahmenwerk
  - Integration von AJAX in ein Rahmenwerk ermöglicht die Erstellung von Anwendungen mit Desktop-Charakter
- Auswahl des „richtigen“ Rahmenwerks
  - Bei neuen Projekten: Prüfen, ob aktuelle Rahmenwerke die Anforderungen besser/direkter/schneller erfüllen können

MOBILE WEB APPS & GWT

Zusammenfassung



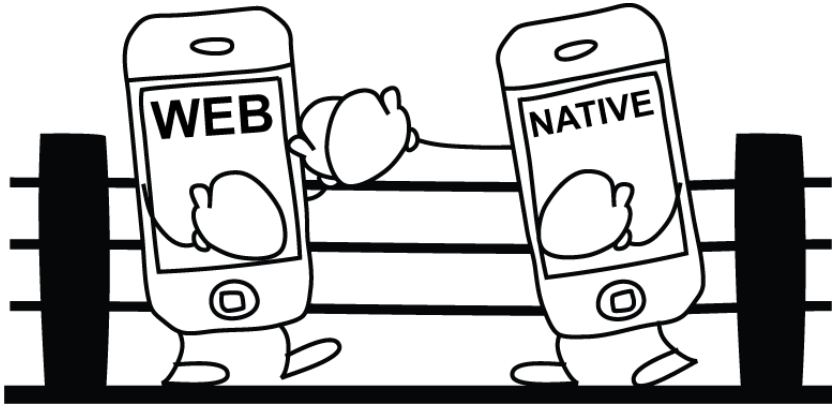
Samsung
Apple
Nokia
Blackberry
Sony
Motorola

Android
iOs
J2ME
JAVA ...

- Native App Entwicklung wünschenswert aber
  - Vielzahl an unterschiedlicher Betriebssystem
  - Vielzahl an unterschiedlichen Geräteeigenschaften

Native vs. Web App

- Zu berücksichtigende Faktoren:
  - Benutzer Oberfläche (look & feel)
  - Entwicklung (Plattform, Sprache…)
  - Fähigkeiten (Gerät eigene Eigenschaften)
  - Monetarisierung (Zahlung, Werbung…)
  - Übertragunsverfahren (Download, Installieren, Updates…)
  - Versionierung der App (Nutzer Updates vs. Web Update)
  - Stärke vs. Schwäche (Performance, Aufwand…)



Google Web Toolkit Bewertung

(+) Trennung von Anwendung und Darstellung

(+) Einfach erlernbar und schnell anwendbar – kein JavaScript

(+) Stetig wachsende Community – guter Support und Dokumentation

(+) Gewohnte IDE kann genutzt werden

(+) Entwicklung der GUI entspricht grundlegend der Entwicklung durch AWT bzw. Swing

(+) Unit-Tests sind möglich

(+) Anwendung kann im Hosted- sowie Web-Mode getestet werden

(+) Durch RPC-Schnittstelle muss sich der Entwickler nicht mehr direkt um die Details der asynchronen Kommunikation mittels *HttpXMLRequest* oder der Serialisierung von Objekten kümmern

Google Web Toolkit Bewertung

(-) Getrennte Betrachtung von Client und Serverseite (Bereitstellung von Daten durch SOAP basierte RPC Calls)

(-) Proprietärer Compiler

(-) Generiertes JavaScript ist kryptisch (proprietär), Debugging nur in Java

(-) JavaScript-Compiler und *Hosted Mode Browser* sind nicht Open-Source und dürfen nicht weitergegeben werden bzw. nur mit Genehmigung von Google

(-) Google behält sich vor, die Lizenzbedingungen in der Zukunft jederzeit zu ändern

Offline Nutzung - Lokales Speichern

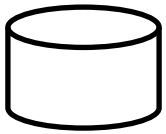
Für die Offline Nutzung einer Mobile Web App ist folgendes zu beachten:

- Lokales Speichern der benötigten Daten über localStorage
- Definition der lokal zu cachenden Dateien über ein manifest file
- Managen der Verbindungsänderungen mit online und offline events
- Definition einer Synchronisationsstrategie

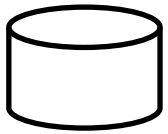
4 Möglichkeiten zur lokalen Speicherung von Daten:

- Web Storage mit localStorage und sessionStorage
- WebSQL
- IndexedDB
- Dateisystemzugriff

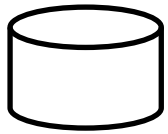
ÜBUNGSAUFGABEN



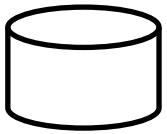
Amazon.de



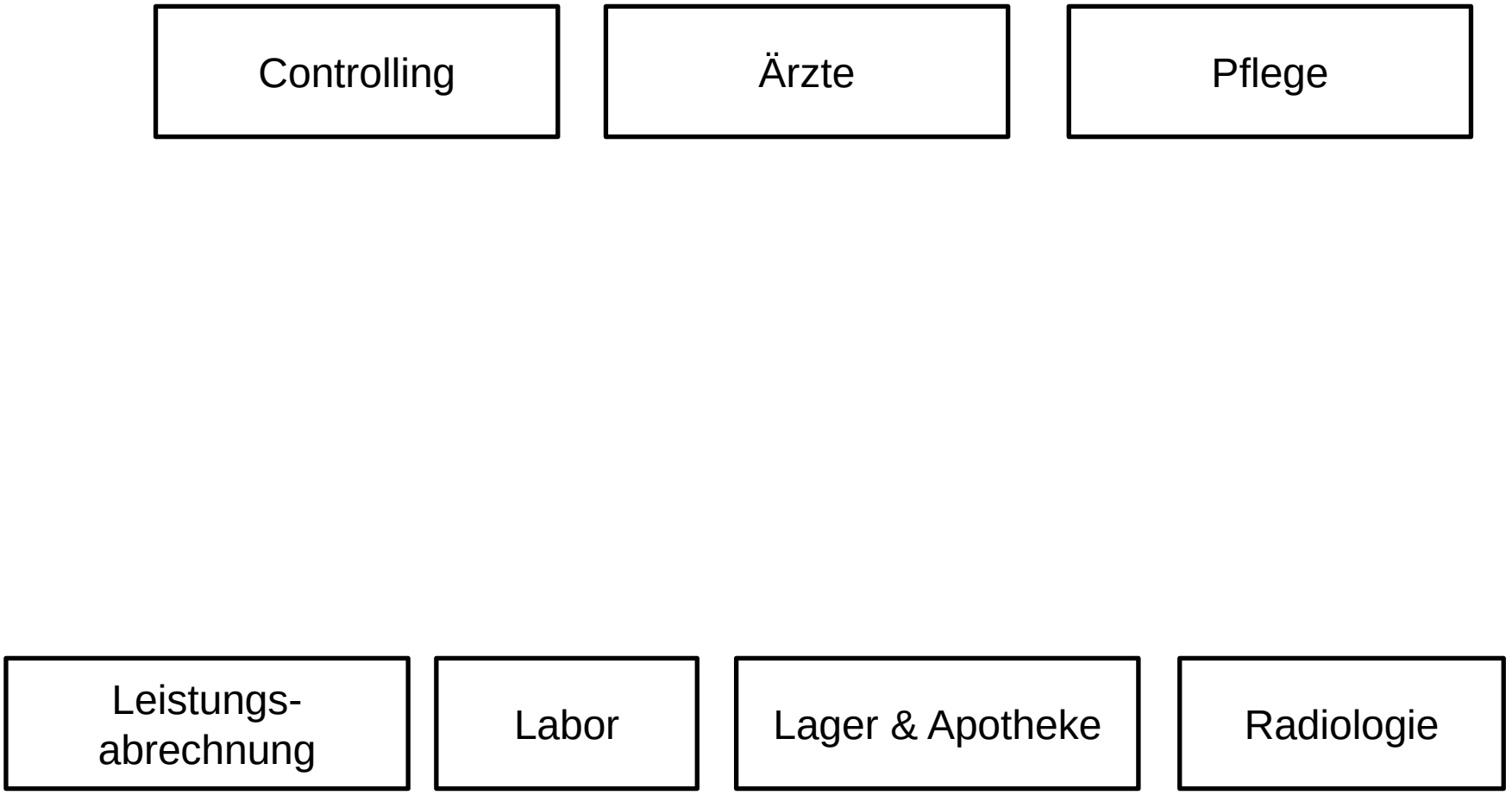
UB Karlsruhe

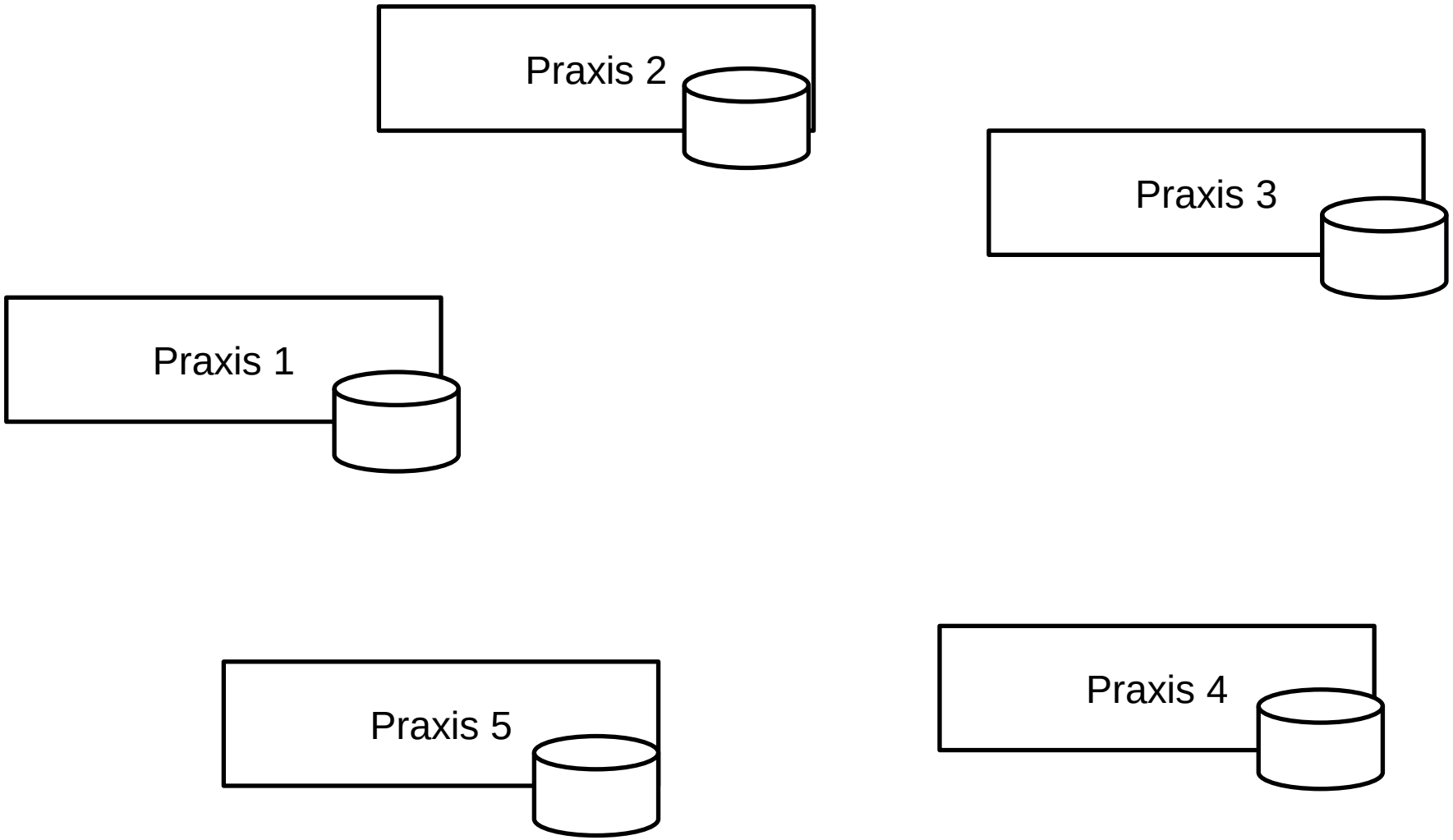


Zeitschriften-
inhaltsdienst

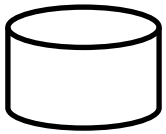


Google
Scholar

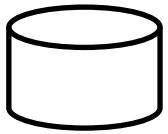

Controlling
Ärzte
Pflege
Leistungs-
abrechnung
Labor
Lager & Apotheke
Radiologie


Praxis 2
Praxis 3
Praxis 1
Praxis 5
Praxis 4

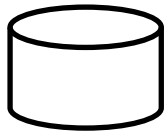
QM System Autohersteller: Verwaltung von Reklamationen



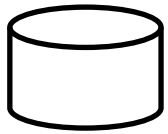
Zulieferer A



Zulieferer B



Zulieferer C



Zulieferer D

Übungsaufgaben 1

- Was sind Herausforderungen an Informationsintegration?
- Was sind Herausforderungen an mobile Web-Anwendungen?
- Auf welche Ebenen kann Integration erfolgen?
  - Datenebene/ Informationsebene, Applikationsebene / Dienstebene/ Ereignisebene, Präsentationsebene
  - Vor- und Nachteile?
- Wie sind Datenquellen zu integrieren?
  - Virtuell/ Materialisiert
  - Vor- und Nachteile?
  - Anzahl Mediatoren, Wrapper, Facilitator?
- Wie ist die Datenbank anzubinden?
  - Relational vs ORM (Leichtgewichtig vs. Schwergewichtig)
  - Vor- und Nachteile?

Übungsaufgaben 2

- Wie ist die Anwendungsarchitektur zu wählen?
  - Mehrschichtenerachitektur / EAI
- Wie erfolgt Anbindung mobiler Anwendungen?
  - Native vs. cross-platform vs. Web
  - Vor- und Nachteile?
- Wie können Ontologien hier helfen?
  - Ontologie vs. RDF vs. OWL
  - Semantisches Mapping, Domänenmodell, Maschinenlesbar/ Reasoning
- Welche Vorteile bieten Ergeignisgesteuerte Architekturen?
- Was sind Ergeignisgesteuerte Architekturen?
  - Event Processing Network, Input Adapter, Output Adapter
  - Vor- und Nachteile?
  - Event Processing Agents (EPA): filter, match, and derive (translate, aggregate, split,…)